Enhancing Spectrum Utilization through Cooperation and Cognition in Wireless Systems

Hariharan Shankar Rahul

Enhancing Spectrum Utilization through Cooperation and Cognition in Wireless Systems

by

Hariharan Shankar Rahul

S.M., Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1999
B.Tech., Computer Science and Engineering, Indian Institute of Technology, Madras, 1997

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

February 2013

Author . . . .
Department of Electrical Engineering and Computer Science
September 28, 2012

Certified by . . . .
Dina Katabi
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Enhancing Spectrum Utilization through Cooperation and Cognition in Wireless Systems

by

Hariharan Shankar Rahul

Submitted to the Department of Electrical Engineering and Computer Science
on September 28, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

We have seen a proliferation of wireless technologies and devices in recent years. The resulting explosion of wireless demand has put immense pressure on available spectrum. Improving spectrum utilization is therefore necessary to enable wireless networks to keep up with burgeoning demand.

This dissertation presents a cognitive and cooperative wireless architecture that significantly enhances spectrum utilization. Specifically, it introduces four new systems that embody a cross-layer design for cognition and cooperation. The first system, SWIFT, is a cognitive cross technology solution that enables wideband devices to exploit higher layer network semantics to adaptively sense which portions of the spectrum are occupied by unknown narrowband devices, and weave the remaining unoccupied spectrum bands into a single high-throughput wideband link. Second, FARA is a cooperative system that enables multi-channel wireless solutions like 802.11 to dynamically use all available channels for all devices in a performance-aware manner by using information from the physical layer and allocating to each link the frequency bands that show the highest performance for that link. SourceSync, the third system, enables wireless nodes in last-hop and wireless mesh networks to cooperatively transmit synchronously in order to exploit channel diversity and increase reliability. Finally, MegaMIMO enables wireless throughput to scale linearly with the number of transmitters by enabling multiple wireless transmitters to transmit simultaneously in the same frequency bands to multiple wireless receivers without interfering with each other.

The systems in this dissertation demonstrate the practicality of cognitive and cooperative wireless systems to enable spectrum sharing. Further, as part of these systems, we design several novel primitives - adaptive spectrum sensing, time alignment, frequency synchronization, and distributed phase-coherent transmission, that can serve as fundamental building blocks for wireless cognition and cooperation. Finally, we have implemented all four systems described in this dissertation, and evaluated them in wireless testbeds, demonstrating large gains in practice.

Thesis Supervisor:	Dina Katabi
Title:	Professor of Electrical Engineering and Computer Science

To my parents, for their unconditional love, support, and guidance.

Academic Ancestors

Dina Katabi, Ph.D., Massachusetts Institute of Technology, 2003.
Dissertation: *Decoupling Congestion Control and Bandwidth Allocation Policy With Application to High Bandwidth-Delay Product Networks*

David Dana Clark, Ph.D., Massachusetts Institute of Technology, 1973.
Dissertation: *An Input/Output Architecture for Virtual Memory Computer Systems*

Jerome Howard Saltzer, Sc.D., Massachusetts Institute of Technology, 1966.
Dissertation: *Traffic Control in a Multiplexed Computer System*

Fernando José Corbató, Ph.D., Massachusetts Institute of Technology, 1956.
Dissertation: *A Calculation of the Energy Bands of the Graphite Crystal by Means of the Tight-Binding Method*

John Clarke Slater, Ph.D., Harvard University, 1923.
Dissertation: *The Compressibility of the Alkali-Halides*

Percy Williams Bridgman, Ph.D., Harvard University, 1908.
Dissertation: *Mercury Resistance as a Pressure Gauge*

Wallace Clement Sabine, A.M., Harvard University, 1888.
Regarded as the founder of the field of architectural acoustics, Sabine was the acoustical architect of Boston Symphony Hall, widely considered among the top three concert halls in the world for its acoustics.
Dissertation: *Various titled works*

John Trowbridge, S.D., Harvard University, 1873.
Dissertation: *I. A New Form of Galvanometer. II. On the Electromotive Actions of Liquids Separated by Membranes. III. On the Electrical Condition of Gas Flames, IV. Ohms Law Considered from a Geometrical Point of View. V. Induced Currents and Derived Circuits*

Joseph Lovering, A.B., Harvard University, ca. 1833.
Dissertation: *The Application of the Principle of Fluxions to the Solution of Different Problems*

Benjamin Peirce, B.A., Harvard University, 1829.
”The Founder of High Mathematics in America”, Lord Kelvin.
Dissertation: *Various titled works*

Nathaniel Bowditch, M.A., Harvard University, ca. 1802.
Considered the founder of modern maritime navigation, Bowditch invented the eponymous Bowditch Curve, and was America’s first insurance actuary.

Bowditch was a self-taught mathematician and did not have a formal advisor.

Acknowledgments

I'm grateful to my advisor, Prof. Dina Katabi, for her support and mentorship. Dina's commitment to research is truly an inspiration. Some of the most fun times I had during my Ph.D. were when Dina used to walk into the office, and we'd spend hours discussing a whole ton of research ideas, turning what initially seemed to be wacky and out-there to something potentially useful and feasible. I'm especially grateful to her for pointing me in the direction of the wideband wireless project in Prof. Sodini's group - it got me started on physical layer wireless research and forms the backbone of my dissertation. It wasn't just about the ideas, though; Dina's enthusiasm about diving into the details, her willingness to brainstorm or serve as a sounding board, anytime of the day or night, weekday or weekend, and her high standards have been critical to improving the quality and completeness of my research. And, through all this, her unfailing ability to keep the forest firmly in sight in the midst of all these trees were instrumental in ensuring that our papers and presentations always distilled out the core contributions of our work. It has been a privilege to work with Dina and be a part of her vibrant and intellectually exciting group. I hope that I will be able to apply a fraction of what I have learnt from her work ethic and writing skills in my professional career.

I'd also like to thank the members of my committee, Prof. Charles Sodini, Prof. Hari Balakrishnan, and Dr. Victor Bahl. Charlie's WiGLAN radios with their unique programmable digital baseband hardware and wideband transmission capabilities helped me not just to learn about hardware design and physical layer wireless, but also served as the platform on which three of the four pieces of work in this dissertation were developed. Thank you, Charlie, for your collaboration, and your careful reading and comments on my papers.

Hari took me under his wings when I was but an inexperienced graduate student, and I learnt a lot about research working with him for my S. M. during my first go-around in grad school. Victor graciously agreed to serve on my committee on short notice and has always been a great mentor, dispensing friendly advice and encouragement, and providing me with exposure to industry research through my internship at MSR.

This dissertation builds on several papers [121, 119, 120, 122], and I'm indebted to my collaborators on these papers. Nate Kushman was clutch in ensuring that we completed SWIFT in time for SIGCOMM. Haitham Hassanieh patiently tuned our system and performed thousands of measurements for SourceSync, doing battle with unwieldy power cords and bulky carts through the passages of the lab. Swarun Kumar spent many sleepless months with me wrestling with USRPs, staring at frequency plots, and working on our wireless testbed for MegaMIMO. I'd also like to thank Farinaz Edalat and Jit Ken Tan for their work on developing the basic OFDM transceiver and PCI interface for the WiGLAN radios.

My work has been significantly improved by conversations with and comments from the following people at MIT and beyond: Fadel Adib, Victor Bahl, Ranveer Chandra, Shyamnath Gollakota, Haitham Hassanieh, Szymon Jakubczak, Nate Kushman, Srikanth Kandula, Sachin Katti, Swarun Kumar, Thomas Moscibroda, Asfandyar Qureishi, Lixin Shi, and Jue Wang. A special shout out to Manish Bharadwaj who was very patient with me as I muddled through learning about wireless, signal processing and hardware and spent many hours sharpening my understanding of the subject.

Thank you to Mary McDavitt, our administrative assistant, for always paying attention to make sure we were well fed during paper deadlines, and for accommodating all our crazy bureaucracy-defying last minute equipment purchases. And thanks again to Dina for being immensely generous with the one thing that probably trumps all in a graduate student's life. High quality free food, be it lovely cakes from Finale, Flour, and the like for birthdays, group outings and celebrations at excellent restaurants in the Boston area, and night upon night of non-pizza dinners for a whole month before paper deadlines!

Much like the Beatles, I too got by with a little help from my friends. Clandestine night time whiffleball in the lab with Jen, Sachin, Srikanth, and Asfand. Bike rides to Walden with Karen. Thanksgiving parties with Jen and James. Pub trivia with Hennessy, Fan, Fiorenza, Spaeth, Ward, Joyce, and the rest. Kicking back with Michael, Manish, Matt, Gillian, Nate, Deb, Zewei, Robin, Amrith, and Swati. The Web, without which this dissertation would either not exist or would've been completed in half the time.

My uncle and aunt helped me to get settled when I first came to the US, and provided me with a home away from home in Montreal that helped me get through my first couple of winters in Boston.

Namrata has been a great source of good cheer these past years. A special thanks to her for putting up with my spending almost all my waking hours and some of my sleeping hours in the lab. For working in the lab so we could spend more of our downtime together. For being a sounding board for pitches and presentations. And, above all, for her patience!

Finally, and most importantly, I am deeply, deeply grateful to my parents. Without the sacrifices they have made for me over the years, the value they have always placed on education, their encouragement when I was down, and their pushing me to excel, I wouldn't be where I am today. This thesis is dedicated to them.

Previously Published Material

Chapter 2 revises a previous publication [121]: H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to Share: Narrowband-Friendly Wideband Networks, *Proc. ACM SIGCOMM*, August 2008.

Chapter 3 revises a previous publication [119]: H. Rahul, F. Edalat, D. Katabi, and C. Sodini. Frequency-Aware Rate Adaptation and MAC Protocols, *Proc. ACM MOBICOM*, September 2009.

Chapter 4 revises a previous publication [120]: H. Rahul, H. Hassanieh, and D. Katabi. SourceSync: a distributed wireless architecture for exploiting sender diversity, *Proc. ACM SIGCOMM*, August 2010.

Chapter 5 revises a previous publication [122]: H. Rahul, S. Kumar, and D. Katabi. MegaMIMO: scaling wireless capacity with user demands, *Proc. ACM SIGCOMM*, August 2012.

Contents

Acknowledgments **9**

Previously Published Material **13**

List of Figures **27**

List of Tables **29**

1 Introduction **31**

1.1 SWIFT 34

1.2 FARA 36

1.3 SourceSync 38

1.4 MegaMIMO 41

1.5 Contributions 44

2 Learning to Share: Narrowband-Friendly Wideband Networks **49**

2.1 Overview 49

2.2 Related Work 52

2.3 Problem Domain 54

2.4 OFDM Background 55

2.5 SWIFT 56

2.6 Implementing SWIFT 71

2.7 Performance Evaluation 73

2.8 Discussion 84

3 Frequency-Aware Rate Adaptation and MAC Protocols 87

3.1 Overview . . . 87
3.2 Related Work . . . 90
3.3 Frequency Diversity . . . 92
3.4 FARA . . . 93
3.5 Per-Subband SNR Estimation . . . 94
3.6 Frequency-Aware Rate Adaptation . . . 96
3.7 Frequency-Aware MAC . . . 100
3.8 Load-Aware Contention . . . 103
3.9 Performance . . . 104
3.10 Discussion . . . 116

4 SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity 117

4.1 Overview . . . 117
4.2 Related Work . . . 123
4.3 SourceSync . . . 124
4.4 Symbol Level Synchronization . . . 125
4.5 Joint Channel Estimation . . . 135
4.6 Smart Combiner . . . 136
4.7 Using SourceSync to harness sender diversity . . . 138
4.8 Performance . . . 142
4.9 Discussion . . . 152

5 MegaMIMO: Scaling Wireless Capacity with User Demands 153

5.1 Overview . . . 153
5.2 Related Work . . . 157
5.3 MegaMIMO . . . 159
5.4 Distributed Phase Synchronization . . . 159
5.5 MegaMIMO Protocol . . . 163
5.6 Compatibility with 802.11 . . . 169
5.7 Decoupling Measurements to Different Receivers . . . 172
5.8 Diversity . . . 174
5.9 MegaMIMO's Link Layer . . . 174

5.10 Testbed and Implementation 176
5.11 Results 178
5.12 Discussion 194

6 Discussion and Concluding Remarks **195**
6.1 A Wireless Architecture for Cooperation and Cognition 195
6.2 Future Work 196

References **201**

List of Figures

1-1 **United States Spectrum Allocation.** The figure shows that all radio spectrum is allocated to existing technologies, with the current static spectrum sharing model, there is little to no spectrum available for use by new technologies. 35

1-2 **Frequency diversity across 100 MHz of 802.11a spectrum as observed by two receivers for transmissions from the same sender.** The figure shows that the SNRs of different frequencies can differ by as much as 20 dB on a single link. Further, different receivers prefer different frequencies. 37

1-3 **Opportunistic routing with sender diversity.** SourceSync enables multiple forwarders to transmit jointly to the destination. 40

1-4 **Last-hop with sender diversity.** SourceSync enables multiple APs to transmit jointly on the downlink. 41

1-5 **Traditional vs. Joint Multi-User Beamforming.** In a traditional multi-user beamforming system with multiple 2 antenna APs, only 1 AP can transmit on a given channel at any given time. This leads to a maximum of 2 simultaneous packet transmissions regardless of the total number of APs. In contrast, MegaMIMO enables all APs to transmit on the same channel, allowing up to $2N$ simultaneous packet transmissions if there are N 2-antenna APs. 42

2-1 **Schematic of an OFDM System.** 55

2-2 **Cognitive Aggregation.** While narrowband devices exist (e.g., 802.11 laptop), SWIFT still uses the remaining non-contiguous chunks of spectrum as if they were one wireless link. 56

2-3 **802.11a Power Profile.** The observed power of an 802.11a transmitter at different SWIFT locations is very different, highlighting the difficulty in picking a power threshold that works at all locations. 58

2-4 **Control Flow for Adaptive Sensing Algorithm** 61

2-5 **Conversion of bits into OFDM symbols:** Values in individual frequency bins are combined in each time sample, and can be recovered only by computing appropriately aligned FFTs. 63

2-6 **Variation of the ratio of energy in A to the energy in B as the window positions vary relative to the packet.** 64

2-7 **Packet detection uses a filtered version of the time-domain signal to produce an estimate of the packet start.** The aligned signal can then be processed in the frequency domain for the rest of the chain. 67

2-8 **Bin Disagreement Causes Communication Failure.** If the transmitter sends in bins 1, 3, 4, and 5 while the receiver listens in 1, 2, and 5, then the receiver will decode noise in bin 2 as data, and miss data in bins 3 and 4. These insertions and deletions will cause a misalignment in the demodulated data stream, creating an error pattern than cannot be rectified by standard error-correcting codes. 68

2-9 **Wideband Radio Used in SWIFT** 70

2-10 **Testbed Map:** Node Locations are Highlighted. 72

2-11 **Approaches to Narrowband-friendliness:** Presents the throughput-range tradeoff, and shows that SWIFT, illustrated in (d), is as friendly to 802.11a as LOW, while attaining dramatically higher throughput and operating range. 73

2-12 **No Single Threshold Works Across Locations:** This figure plots the ideal threshold that ensures safe narrowband operation while maximizing bins usable by wideband. 802.11a nodes at locations 7-10 are not affected by wideband, and hence the ideal threshold for these locations is infinity. . . . 76

2-13 **No Threshold is Safe and Efficient in All Locations** 77

2-14 **Adaptive Sensing is Robust:** At each location, SWIFT finds the correct unusable bins, *i.e.* those that interfere with 802.11a. 78

2-15 **Responsiveness of Adaptive Sensing:** The top graph shows that 802.11a throughput is not hindered for longer than 0.5 seconds by SWIFT. The bottom graph shows that, when 802.11a first appears, SWIFT backs off to a conservative bin choice within 120 ms, but quickly converges to a maximal set of safe bins. 79

2-16 **Robustness to disagreement:** The figure shows the probability of a transmission succeeding as a function of the number of disagreeing bins. It shows that SWIFT is robust to as much as 40% disagreement between the set of transmitter and receiver bins. 80

2-17 **SWIFT reaction to TCP web downloads:** (a) and (b) show that, even in the face of intermittent 802.11a traffic, SWIFT avoids affecting 802.11a transmissions, while (c) shows that it does this while still achieving 90% of its original throughput. 81

2-18 **Throughputs in a Network:** (a) and (b) show the throughputs of the four 802.11a pairs, with and without SWIFT. SWIFT has no impact on 802.11a, while, still getting good throughput as seen in (d). In contrast, (c) shows that non-adaptive wideband transmitters reduce 802.11a throughput by around 50%. 83

3-1 **Frequency diversity across 100 MHz of 802.11a spectrum as observed by two receivers for transmissions from the same sender.** The figure shows that the SNRs of different frequencies can differ by as much as 20 dB on a single link. Further, different receivers prefer different frequencies. 88

3-2 **Multipath Effect Causes Frequency Diversity.** Signals from different paths combine at the receiver constructively or destructively depending on their phases. Since the phase is a linear function of the frequency, the destructive and constructive patterns differ across frequency bands causing different frequencies to have different SNRs. 93

3-3 **OFDM PHY semantics with and without FARA.** In FARA-enabled devices, the choice of modulation and FEC code rate is done independently for each OFDM subband. 97

3-4 **Simplified FARA frame with three concurrent packets.** A FARA sender transmits concurrently to multiple receivers, by allocating to each receiver a subset of the OFDM subbands. To ensure fairness, *i.e.*, to ensure that all clients obtain the same average rate, distant receivers that experience lower per-frequency SNRs are allocated more frequencies. Note that the frequencies used for one receiver are not necessarily consecutive. They are made consecutive in the figure to simplify the drawing. The frame header includes metadata about the intended nexthops as well as their allocated subbands. . 100

3-5 **The WiGLAN radio platform used in FARA's evaluation.** 104

3-6 **Testbed topology showing node locations.** The node marked `tx` is used as a transmitter/AP. The other 17 locations are used for receivers. 106

3-7 **Frequency Diversity in Non-Line-of-Sight (NLOS):** The top graph presents the SNR in each 1-MHz wide subband across the 100 MHz band of our radio for a typical NLOS channel in our testbed. The graph shows that the subband SNRs can differ by more than 20 dB. The bottom graph shows the SNRs of two subbands in the top graph as a function of time. It reveals that the subband SNRs are stable over a multiple-second time period, thereby allowing an adaptive scheme to harness the frequency diversity. 107

3-8 **Frequency Diversity in Line-of-Sight (LOS):** The top graph presents the subband SNR for a typical LOS channel in our testbed. It shows that the subband SNRs can differ by more than 5-10 dB. While the variation is smaller than in NLOS channels, it is still significant. The bottom graph shows the SNRs of two subbands in the top graph as a function of time. It reveals that the subband SNRs are stable over a multiple-second time period, thereby allowing an adaptive scheme to harness the frequency diversity. 108

3-9 **Mapping SNRs to a bitrate (i.e., a modulation and code rate):** Plots the throughput per 1-MHz subbband as a function of SNR for each choice of modulation and code rate. It shows that for any SNR, the optimal choice of modulation and code rate is fairly clear. 109

3-10 **Stability of the relation between SNR and the optimal modulation and code rate across time and space.** The figure shows the envelope of the functions in Fig. 3-9, for measurement collected at receivers `A1` and `D4` at different times. The two envelopes match closely showing that the SNR dictates the best modulation and code rate. 110

3-11 **FARA Rate Adaptation:** FARA's frequency-aware rate adaptation achieves higher throughput than SampleRate's frequency-oblivious rate adaptation at all locations, with gains varying from $1.4\times$ to $3.6\times$ for a 100 MHz wide channel, and $1.1\times$ to $1.5\times$ for the 20 MHz channel. 111

3-12 **Gains from a Frequency-aware Architecture:** The figure plots two CDFs. The dashed line is the CDF of the ratio of client throughput under FARA to its throughput in traditional 802.11 networks which use SampleRate and CSMA MAC. The solid line is the CDF of the ratio of client throughput under FARA with a CSMA MAC and traditional 802.11 with SampleRate and CSMA. The CDFs show that FARA provides on average $3\times$ throughput gain. 70% of the gain comes from FARA's frequency-aware rate adaptation, and 30% is due to its frequency-aware MAC protocol. 113

3-13 **Load balancing with FARA:** The figure plots the Jain's Fairness Index as a function of the ratio of the number of clients on AP1 to those on AP2. Note that fairness is optimal when the index is 1, and is worst when the index is $1/n$, where n is the total number of clients. 115

4-1 **Opportunistic routing with sender diversity.** SourceSync enables multiple forwarders to transmit jointly to the destination. 121

4-2 **Last-hop with sender diversity.** SourceSync enables multiple APs to transmit jointly on the downlink. 122

4-3 **FFT windows at a receiver for a single transmitter.** Any FFT window within the slack is valid. Any other FFT window would include energy from the previous symbol and hence is invalid. 125

4-4 **FFT windows at a receiver for two transmitters.** In order to decode both transmissions, the symbols from the transmitters must arrive at the receiver aligned within the slack of the CP. 127

4-5 **Unwrapped channel phase of OFDM subcarriers in a flat fading channel.** The slope is a function of the detection delay. 128

4-6 **Joint frame from the perspective of the senders.** Symbols in solid blue are transmitted by the lead sender, symbols in dotted red by the co-sender, and symbols in white reflect silence periods. The co-sender hears the lead sender's transmission after a delay of d_1, waits for SIFS$-(d_1 + \Delta_1 + h_1)$ after processing the synchronization header, followed by a wait of w_1, and then begins its transmission. 132

4-7 **Format of joint frame seen by the receiver.** 132

4-8 **Synchronization at two receivers.** One-way delays are shown. No choice of wait time allows perfect alignment at both receivers. 134

4-9 **SourceSync for the last hop.** SourceSync can harness sender diversity using concurrent transmissions from many APs. 138

4-10 **SourceSync with Opportunistic Routing.** SourceSync exploits the fact that many relays hear a packet to improve throughput. 140

4-11 **Testbed map.** Node locations are highlighted. 143

4-12 95^{th} **percentile synchronization error.** SourceSync ensures that the synchronization error is less than 20 *ns* across the operational range of 802.11 SNRs. 145

4-13 **CP reduction with SourceSync.** SourceSync enables concurrent transmissions to achieve high SNR with a significantly lower CP than an unsynchronized baseline that does not compensate for delay differences. 146

4-14 **Delay spread of a single sender.** The OFDM channel in the time domain has 15 significant taps, which corresponds to the CP length required with synchronization. 147

4-15 **Power gains.** SourceSync achieves a 2–3 dB gain over a single sender across the range of SNRs. 148

4-16 **Frequency diversity gains.** SourceSync improves the SNR in each subcarrier and creates a flatter SNR profile. 149

4-17 **SourceSync at the last hop.** The red dotted line is the CDF of throughput using selective diversity (*i.e.* single best AP). The blue solid line is the CDF of throughput using sender diversity across both APs with SourceSync. The CDFs show that sender diversity produces a median gain of 1.57× over selective diversity. 151

4-18 **SourceSync with opportunistic routing.** SourceSync together with ExOR provides gains both over ExOR alone, and over traditional single path routing. The median gains are 1.26-1.4× over single path routing, and 1.35-1.45× over ExOR, depending on the bitrate. 151

5-1 **Traditional vs. Joint Multi-User Beamforming.** In a traditional multi-user beamforming system with multiple 2 antenna APs, only 1 AP can transmit on a given channel at any given time. This leads to a maximum of 2 simultaneous packet transmissions regardless of the total number of APs. In contrast, MegaMIMO enables all APs to transmit on the same channel, allowing up to $2N$ simultaneous packet transmissions if there are N 2-antenna APs. 155

5-2 **Channel matrix with 2 APs transmitting to 2 clients.** 160

5-3 **Packet Structure from the perspective of APs and the receiver.** Symbols in blue are transmitted by the lead AP, symbols in red by the slave AP, and symbols in white reflect silence periods. 164

5-4 **802.11 Channel Measurement.** MegaMIMO measures channels with 802.11 clients by sending a series of two-stream transmissions. Every transmission includes the reference antenna, L_1, as well as one other antenna (either L_2 or S_1 in our example). Note that for clarity, the figure does not show the transmissions to/from R_2 and S_2, but MegaMIMO naturally measures the channels to R_2 simultaneously. 171

5-5 **Testbed Topology.** Client locations are marked by red circles, and AP locations by blue squares. *Note that the figure shows the set of possible locations for clients and APs, and different subsets of locations are picked for different experiments.* 177

5-6 **Degradation of SNR due to phase misalignment.** Even with only 2 transmitters, a misalignment of just 0.35 radians can reduce the SNR by almost 8 dB at the receivers due to interference. 179

5-7 **CDF of observed phase misalignment.** The median misalignment is 0.017 radians, and the 95^{th} percentile misalignment is 0.05 radians. 180

5-8 **Accuracy of Phase Alignment.** Average INR at receivers for various numbers of receivers across different SNR ranges. INR stays below 1.5dB across SNRs even with 10 receivers. 181

5-9 **Linear Array Topology.** This is the MegaMIMO equivalent of a traditional MIMO transmit and receive array. This topology has N APs spaced uniformly along one edge of a rectangular room, and N clients similarly spaced along the opposite edge. The length of the AP array is L meters, and the AP-client separation is H meters. 183

5-10 **Condition number of the channel matrix as a function of inter-AP separation in line-of-sight.** The channel matrix becomes better conditioned as the inter-AP separation becomes larger. When the inter-AP separations are very small, the system is poorly conditioned similar to traditional MIMO. At large inter-AP separations, the system is well conditioned even in a pure line-of-sight scenario. 184

5-11 **Throughput gain as a function of number of APs for different AP-client separations.** MegaMIMO provides degrees-of-freedom gain and linear throughput scaling even in a line-of-sight scenario because it can take advantage of large inter-AP spacing. Further, as the AP-client distance decreases, the minimum required inter-AP spacing to obtain linear scaling decreases allowing MegaMIMO to pack more APs in the same space. In both cases, as the number of APs becomes large (and the inter-AP spacing small), the gains of a naive MegaMIMO system that uses all available APs reduce and eventually become zero. However, as described in Section 5.9, an actual MegaMIMO system would not use all APs, but adaptively use only as many APs as necessary to achieve the maximum possible gain, as indicated by the green horizontal lines. 186

5-12 **Scaling of throughput with the number of APs.** In this experiment, the number of APs equals the number of receivers. At all SNRs, MegaMIMO's network throughput increases linearly with the number of APs while total 802.11 throughput remains constant. 188

5-13 **Fairness.** CDFs of per-client throughput gain. Across all SNRs, MegaMIMO provides all clients with very similar gains. 191

5-14 **Diversity Throughput.** Throughput of a MegaMIMO client when using diversity with 2, 4, 6, 8 and 10 APs. MegaMIMO can achieve close to maximum rate even to a client unable to receive any packets with 802.11. 192

5-15 **Throughput achieved using MegaMIMO on off-the-shelf 802.11n cards.** MegaMIMO significantly improves the performance of off-the-shelf 802.11n cards at high (>18 dB), medium (12-18 dB) and low (6-12 dB) SNRs. 193

5-16 **Fairness Results.** For all nodes in our testbed, MegaMIMO delivers a throughput gain between 1.65-2×, with a median gain of $1.8\times$ across SNRs. This shows that MegaMIMO provides similar throughput gains for every node in the network. 193

List of Tables

3-1 **Minimum required SNR for a particular modulation and code rate (i.e., bitrate).** Table is generated offline using the WiGLAN radio platform by running all possible bit rates for the whole operational SNR range. The SNR field refers to the minimum SNR required to maintain the packet loss rate below 1% (see Section 3.9 for experimental setup). 98

i

Chapter 1
Introduction

Wireless connectivity has become a *sine qua non* for a wide variety of systems. It enables a variety of applications such as mobility, distributed sensing and monitoring, peripheral connectivity, smart grids, and so on. The proliferation of wireless applications has led to the wide deployment of a large number of technologies such as 802.11 (and its various flavors 802.11b, 802.11a, 802.11g, 802.11n, 802.11ac *etc.*) and LTE for data communication, Bluetooth for peripherals, ZigBee for sensors, DECT for cordless phones *etc.* Simultaneously, the proliferation of devices such as notebooks, netbooks, smartphones, tablets *etc.* has led to a dramatic increase in the demand for wireless data.

This explosion in wireless demand has put immense pressure on available spectrum. For example, the recent move of television bands from analog to digital was driven by the need to free up about 200 MHz of additional spectrum for data communication in white spaces, public emergency networks *etc.* Major cellular providers have been aggressively acquiring additional spectrum, and upgrading their networks to support higher data rates. Despite all this, however, the FCC predicts that given the current trends in wireless demand growth, and based on current technologies, there will be a spectrum shortfall in 2013 [43]. Similarly, the Rysavy Research Group predicts that the average demand per user will exceed the average capacity per user sometime in 2013 [126].

The key challenge is that since wireless is inherently a broadcast medium, and wireless spectrum is a fundamentally limited resource, wireless technologies and devices need to share the available spectrum. Today's wireless networks adopt a static approach to spectrum sharing. The approach attempts to ensure that only one wireless link operates in any portion of the spectrum in a region at any given time. Different technologies are typically statically assigned different parts of the spectrum to operate in, and devices operating in the same part of the spectrum orchestrate their transmissions so that only one device transmits at a time.

However, static sharing makes inefficient use of wireless spectrum. For instance, not all wireless technologies are present in all locations, and a system that can dynamically detect and use spectrum unoccupied by other technologies at any given location can increase the efficiency of spectrum usage. Similarly, different sender-receiver pairs experience different performance from the same spectrum, and a system that dynamically divides spectrum between different links based on the performance they observe can improve wireless throughput and reliability.

Motivated by these observations, this dissertation proposes an *agile* spectrum sharing architecture. In contrast to a statically configured division of spectrum across wireless nodes, our architecture enables wireless nodes to dynamically adapt the spectrum they operate in based on their surrounding wireless environment. We demonstrate that such dynamic spectrum sharing increases the throughput and reliability of wireless networks, and ultimately improves spectrum utilization.

The work in this dissertation builds on the ideas of cognitive and cooperative communication. Cognitive radios were first introduced by Mitola *et. al.* in [67], which presented a broad vision for “wireless personal digital assistants (PDAs) and the related networks that are sufficiently computationally intelligent about radio resources and related computer-to-computer communications to detect user communications needs as a function of use context, and to provide radio resources and wireless services most appropriate to those needs.” This initial vision has been followed by work on detection of devices in licensed frequency bands [9], economic models for spectrum sharing [22, 21], and centralized and distributed spectrum sharing protocols [2, 162, 163, 14, 93].

The performance and reliability benefits of simultaneous transmission of data from multiple devices were first recognized in early work by van der Meulen [101], where he explored a simple three node wireless system. Since then, there has been much work on a variety of fronts, including formally characterizing the reliability gains in this small system [33], evaluating a variety of cooperation schemes [89], and expanding the results to a larger network [81, 84]. In addition to reliability, there has also been a wealth of work on the benefits of joint transmission of data from multiple nodes to enhance throughput using coding [157, 84] as well as MIMO [112, 113] techniques.

However, prior work on cognitive radios largely focuses on licensed bands, where the goal is to ensure that a cognitive secondary user does not interfere with a licensed primary user whose signal format (including modulation, occupied frequencies, pilots, packet format *etc.*) is well known. The large body of theoretical work on cooperative transmission assumes transmitters that are tightly coupled and synchronized in time, frequency, and phase, and systems that implement this theoretical work largely try to replicate this assumption by connecting transmitters or receivers with high bandwidth low attenuation cables, or highly accurate clocks.

As a result, existing techniques are not amenable to a wide variety of practical scenarios. For instance, in unlicensed frequency bands, there is no single primary user; instead, multiple different technologies, whose signal formats and occupied frequencies are unknown, contend with each other. Further, it is often not economical or feasible, for instance, in indoor deployments, to assume that transmitting devices can share the same clock and be tightly synchronized with each other. These challenges have proved to be major barriers to adoption of cognitive and cooperative technologies in practice.

This dissertation provides a simple and practical approach to agile spectrum sharing networks. It relaxes the restrictions of prior work requiring that devices have knowledge of contending signal patterns, or can tightly synchronize their signals in time, frequency, or phase using shared clocks. Instead, the systems presented in this dissertation adopt a cross-layer approach to cognition and cooperation. In particular, they exploit higher layer network semantics, and devise digital algorithms that automatically identify, calibrate and correct for the variations in channel characteristics across different wireless devices. Thus, the algorithms in this dissertation can address the practical scenarios of widely deployed wireless devices in unlicensed frequency bands, and enable agile spectrum sharing both across technologies, and within a single technology.

This dissertation introduce four new systems: SWIFT, FARA, SourceSync, and MegaMIMO, that embody such cross-layer design for cognition and cooperation.

- SWIFT is a cognitive cross-technology solution that enables wideband devices to adaptively sense which portions of the spectrum are occupied by narrowband devices. SWIFT exploits the fact that the link and transport layers of narrowband devices typically respond when faced with interference to accurately identify occupied portions of the spectrum, and devises physical layer algorithms that can communicate using the remaining unoccupied portions of the spectrum.

- FARA is a cooperative system that enables multi-channel wireless solutions like 802.11 to share spectrum among wireless devices. FARA's MAC uses information from the physical layer to share spectrum across devices in a performance-aware manner to increase performance and utilization as compared to today's static division of channels.
- SourceSync enables wireless nodes in last-hop and wireless mesh networks to cooperatively transmit jointly in order to exploit channel diversity. SourceSync's link layer can leverage physical layer coding and synchronization mechanisms to exploit channel diversity and increase the reliability of wireless transmissions.
- MegaMIMO scales wireless throughput with the number of transmitters by enabling multiple wireless transmitters to transmit simultaneously in the same frequency bands to multiple wireless receivers without interfering with each other. MegaMIMO's protocol works across the physical, link and network layers to enable such joint transmission, thereby providing dramatic improvements in network throughput and reliability.

The systems in this dissertation demonstrate the practicality of cognitive and cooperative wireless systems to enable agile spectrum sharing. Further, as part of these systems, the dissertation presents several novel primitives - adaptive spectrum sensing, time alignment, frequency synchronization, and distributed phase-coherent transmission, that have applicability to a wide variety of wireless systems. All four systems described in this dissertation have been implemented and evaluated them in wireless testbeds, demonstrating large gains in practice.

■ 1.1 SWIFT

The first system in this dissertation, presented in Chapter 2, is SWIFT. SWIFT addresses the challenge of designing high throughput wideband systems that can efficiently utilize

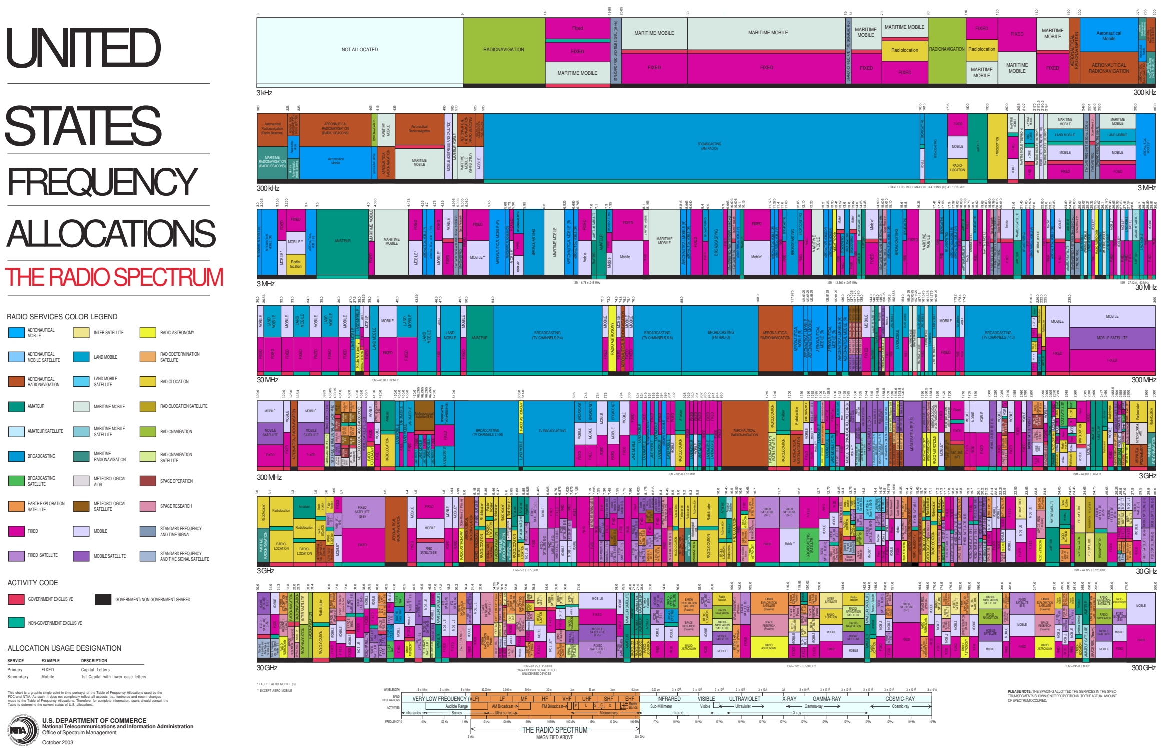


Figure 1-1: United States Spectrum Allocation. The figure shows that all radio spectrum is allocated to existing technologies, with the current static spectrum sharing model, there is little to no spectrum available for use by new technologies.

wireless spectrum. Wideband technologies have the potential to satisfy the demand of bandwidth hungry rich media applications.

With the traditional static spectrum sharing architecture, wideband systems would need to be assigned a large contiguous chunk of spectrum for their use. However, the proliferation of technologies in recent decades, coupled with the allocation of different chunks of spectrum to different technologies, makes it challenging to find contiguous unoccupied swaths of spectrum. Fig. 1-1, which depicts the allocation of spectrum to various wireless technologies, illustrates this problem. Even if one could find a large contiguous chunk of unoccupied spectrum, allocating it statically to a wideband technology would be inefficient because this spectrum would go unused in locations where the wideband technology is currently not being used.

SWIFT is designed for wideband systems that can opportunistically share spectrum with existing narrowband technologies. Its design is motivated by measurement studies [32, 100] showing that, while various wireless technologies exist throughout the spectrum, only a few such technologies are usually operational in a house or small geographic area,[1] and hence a large number of non-contiguous frequency bands are likely to be unused. SWIFT is designed for a dynamic spectrum usage model that can accommodate a diversity of unknown narrowband technologies. It detects exactly which bands are occupied by these narrowband devices, and composes the remaining unoccupied bands to build a single wireless link. Such a system allows wideband devices to operate at normal power without affecting narrowband devices, and delivers on the promise of simultaneously achieving high throughput, range, and coexistence.

In order to achieve its goals, SWIFT needs to address three challenges: (a) how does SWIFT detect the frequency bands that it must avoid, to allow narrowband devices to operate normally?, (b) how does the PHY layer operate across chunks of non-contiguous frequencies?, and (c) Given that different nodes might perceive different usable frequencies, how do SWIFT nodes communicate?.

SWIFT detects frequency bands occupied by narrowband devices by exploiting common network semantics of these devices, in particular, the fact that many narrowband devices react when faced with interference, either at the lower layers [13, 40], or at higher layers [70]. This observation allows SWIFT to directly identify frequency bands which, if used, would interfere with narrowband devices. Thus, SWIFT probes frequencies whose power profile indicates that they might be in use by narrowband devices, monitors the change in the narrowband power profile when probed, and backs off if it perceives narrowband reaction. It then designs a cognitive PHY that incorporates cross-layer information from the adaptive sensing subsystem.

Chapter 2 describes the design of each of the components of SWIFT, the prototype hardware and software implementation on a custom wideband radio, and a testbed evaluation.

■ 1.2 FARA

The next system in this dissertation is FARA, presented in Chapter 3. FARA addresses the challenge of rate adaptation and spectrum sharing in wideband wireless systems. Wireless systems are increasingly moving to wider frequency bands in order to achieve high throughput and efficient spectrum utilization. For instance, the state-of-the-art Wi-Fi stan-

[1]The measured average spectrum occupancy is 5.2% [32].

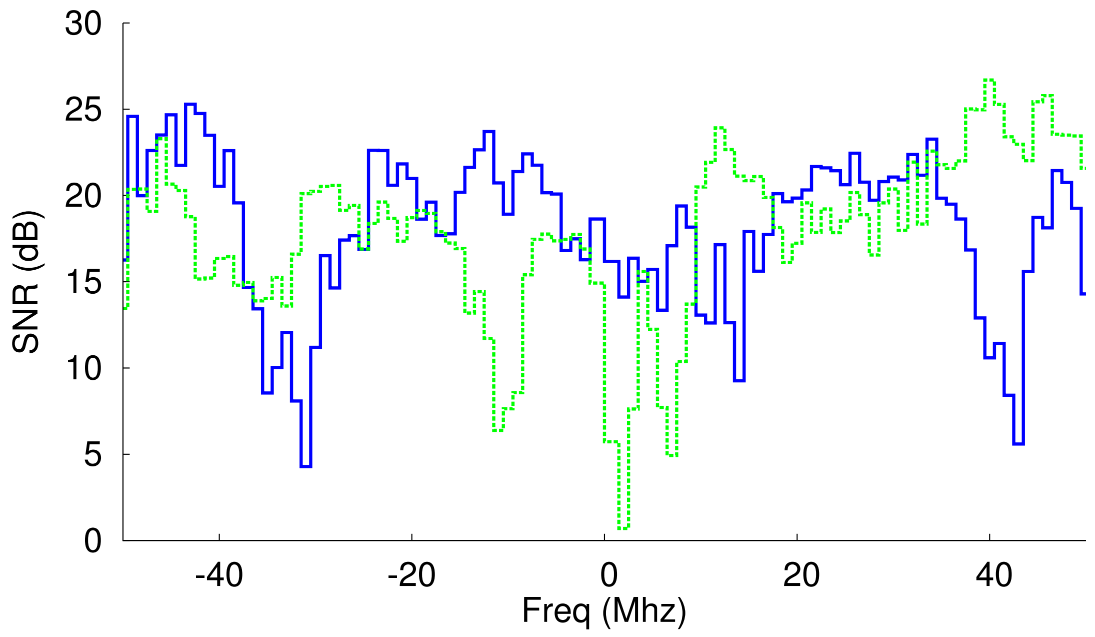


Figure 1-2: Frequency diversity across 100 MHz of 802.11a spectrum as observed by two receivers for transmissions from the same sender. The figure shows that the SNRs of different frequencies can differ by as much as 20 dB on a single link. Further, different receivers prefer different frequencies.

dard, 802.11n [6], has a 40 MHz high throughput mode which bonds together two 20 MHz bands, and the upcoming 802.11ac standard [116] is capable of operating in even wider bands of 80 MHz. Cognitive wideband solutions such as SWIFT, which is described in Chapter 2, and ultra-wideband technologies in the 2.4 GHz and 60 GHz spectrum operate over hundreds of MHz, or even a few GHz, in order to support the bandwidth requirements of high definition video and other rich media. The FCC has recently permitted unlicensed use of about 100-250 MHz of white spaces vacated by the move of television bands from analog to digital. Further, several empirical studies [58, 108] also show that the 802.11 multi-channel model, which divides the spectrum in 20 or 40 MHz channels among access points leads to inefficient spectrum utilization due to imbalance in the load handled by different access points.

FARA is an agile spectrum management scheme for such wideband networks. The key challenge addressed by FARA is the frequency diversity experienced in these wide frequency bands. This frequency diversity can either be due to interference from narrowband devices operating in the wideband as in SWIFT, or due to multipath effects [144]. Multipath propagation causes multiple delayed and attenuated copies of the signal to combine with each other at a receiver. This combination can result in differing levels of constructive or destructive interference, and thereby different fading, at the receiver. The exact fading behavior and resulting performance (SNR) of a frequency for a particular sender-receiver pair naturally depends both on the specific frequency as well as the paths traversed by the signal between the sender and the receiver. Frequencies that are far apart from each other can experience significantly different SNRs for a single sender-receiver pair, and this effect becomes increasingly important with wider frequency bands. Additionally, different sender-receiver pairs experience different SNRs in the same frequency, and hence the frequencies that show good performance for a particular pair can be significantly different than those that show good performance for another pair. Fig. 3-1 shows empirical measurements of the SNRs across 100 MHz of the 802.11a spectrum, as observed by 2 clients for transmissions from the same AP (see Section 3.9 for experimental setup). The figure reveals that different frequencies show a difference in SNR of over 20 dB both for a single link and across links.

Today's rate adaptation and medium access control protocols however are spectrum-oblivious. They assign the same bitrate to all frequencies for a particular sender-receiver pair, and divide the medium across different sender-receiver pairs in a performance-oblivious manner. As a result, they cannot deal with the challenges, or exploit the opportunities, presented by frequency diversity in wideband channels.

FARA's spectrum aware architecture, in contrast, can improve network throughput and spectrum utilization in emerging wideband Wi-Fi, cognitive, and white space networks. FARA has four key components: (a) fine-grained spectrum performance estimation, (b) a spectrum-aware link layer, (c) a performance-aware spectrum sharing and MAC layer, and (d) load aware contention to improve spectrum utilization.

Chapter 3 describes the detailed architecture of each of the components of FARA, and provides results of the implementation and testbed evaluation.

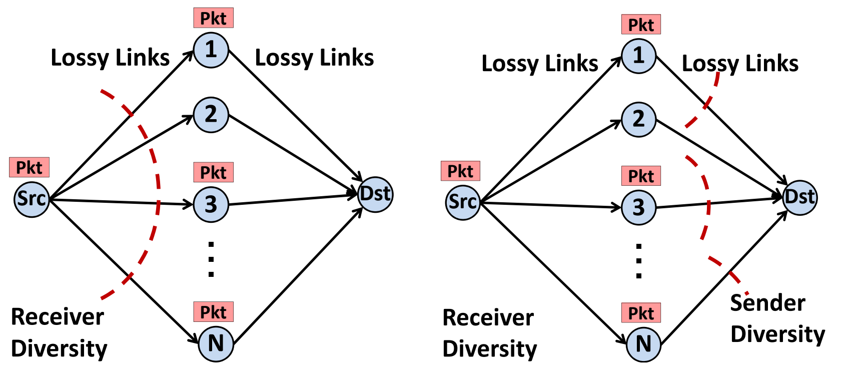
■ 1.3 SourceSync

The third system in this dissertation, presented in Chapter 4 is SourceSync. SourceSync is a cooperative wireless system that enables multiple senders to transmit a packet simultaneously in the same spectrum to exploit channel diversity. As described in the previous section on FARA, channel diversity is a fundamental property of wireless networks. This diversity manifests itself both in channels from a sender to multiple receivers (*i.e.*, when a sender transmits a packet, it traverses different channels to different receivers, and is unlikely to experience fading simultaneously at all receivers), and from multiple senders to a receiver (*i.e.*, when multiple senders transmit packets simultaneously to a receiver, the packet is unlikely to experience fading from all senders at the same time).

The ability to harness diversity across senders in order to ensure that all frequencies have good performance is particularly important in wideband channels, which experience frequency diversity. Transmitting a packet simultaneously from multiple senders to a receiver ensures that no frequency is deeply faded at the receiver, since it is unlikely that a frequency simultaneously experiences fading from all senders. However, while transmit diversity has been analyzed in theory [84], Wi-Fi networks do not use it in practice, and instead use only one transmitter at a time, say, the one with the strongest channel [107]. This contrasts starkly with the variety of mechanisms [19, 26, 125, 155, 79] that exploit receive diversity, *i.e.*, the simultaneous reception of a transmitted packet across multiple receivers.

Harnessing simultaneous transmission from multiple senders is challenging for multiple reasons. First, it requires that the transmitters are aligned at the symbol level so that their signals do not interfere with each other. Such alignment requires fine grained synchronization across transmitters, which is difficult to achieve [36, 78, 53, 119], since the different transmitters need to time their transmissions without the assistance of a shared clock or central controller. SourceSync instead uses shared reception of a wireless packet to synchronize all transmitters. The challenge is to make such shared reception robust to the typical hardware and channel dependent variations in packet detection delay, which can be on the order of hundreds of *ns* [151]. Second, even when transmitters are aligned, they need to orchestrate their signals so that they combine constructively and increase the resulting signal strength at the receiver. Finally, receivers of such simultaneous transmissions need to be able to decode these transmissions while dealing with the fact that different transmitters have different hardware impediments (*e..g*, different oscillators with slightly differing operating frequencies).

SourceSync addresses these challenges by devising a new mechanism for fine-grained time synchronization of wireless transmitters. SourceSync's Symbol Level Synchronizer uses shared packet reception as a time reference across transmitters, computes robust es-

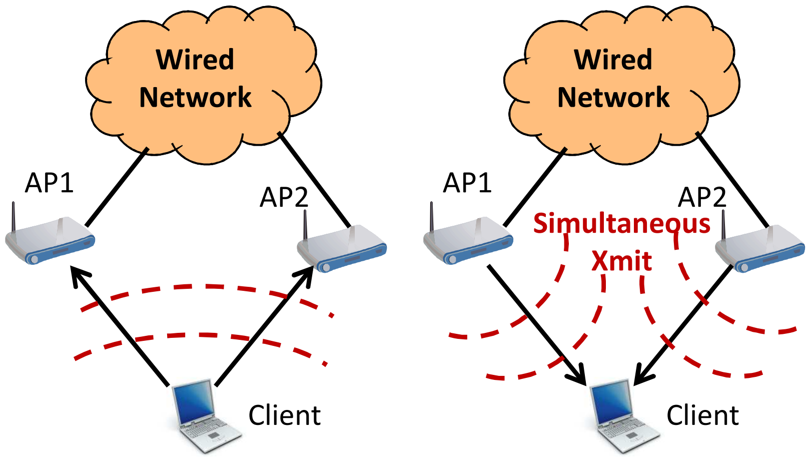


(a) First-hop receiver diversity (b) Second hop sender diversity

Figure 1-3: Opportunistic routing with sender diversity. SourceSync enables multiple forwarders to transmit jointly to the destination.

timates of signal propagation delays between transmitters and receivers, as well as hardware turnaround times at transmitters, and compensates for these delays prior to transmission. SourceSync's robust estimation can accurately eliminate the inherent variability in packet detection delays, and also passively leverage data packets to track changes in channel delays over time. In addition, SourceSync transmitters use space time block codes in a distributed manner to ensure that transmitted signals combine constructively at receivers independent of transient channel conditions. Finally, SourceSync receivers can independently estimate the channels from multiple senders, track the variations of these channels across the duration of a packet, and compose them to compute the aggregate channel as a result of joint transmission. Taken together, these mechanisms enable distributed synchronized transmission to harness sender diversity.

The dissertation shows how SourceSync is used to develop protocols for exploiting sender diversity with opportunistic routing as well as last-hop routing. Existing opportunistic routing protocols such as ExOR, MORE, SOAR, and MIXIT leverage receive diversity, *i.e.*, the property that it is unlikely that all nodes close to the destination do not receive a transmitted packet, as shown in Fig. 1-3(a). SourceSync complements the opportunistic receptions exploited by existing protocols with opportunistic synchronous transmissions from multiple forwarders as shown in Fig. 1-3(b), *i.e.*, since it is likely that multiple intermediate nodes receive a packet, they can transmit it simultaneously thereby improving network reliability and throughput. Similarly, with lasthop routing, existing protocols like MRD, SOFT, and Link-Alike exploit the fact that a transmitted packet on the uplink is unlikely to be lost at all access points (APs). This allows them to exploit receive diver-



(a) Uplink receiver diversity (b) Downlink sender diversity

Figure 1-4: Last-hop with sender diversity. SourceSync enables multiple APs to transmit jointly on the downlink.

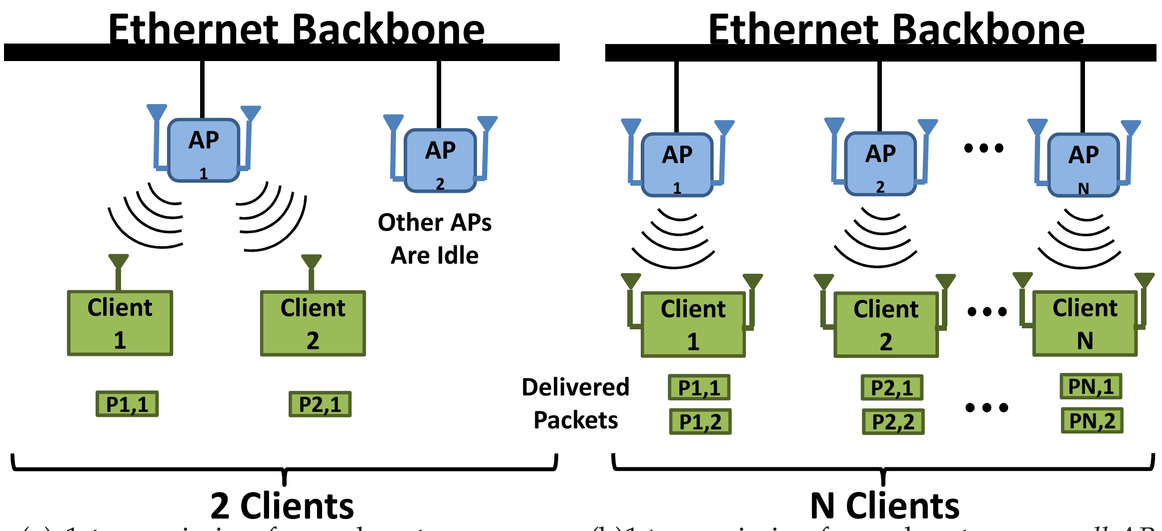
sity in the case of lossy uplinks without needing retransmissions, as shown in Fig. 1-4(a). However, these protocols cannot exploit sender diversity across APs on the downlink. SourceSync enables multiple APs to harness downlink sender diversity by transmitting a packet simultaneously to a client, as shown in Fig. 1-4(b).

Chapter 4 describes the design of SourceSync's components, and how they are used to exploit sender diversity in wireless mesh and last-hop networks. It also presents detailed results from an implementation and testbed evaluation.

■ 1.4 MegaMIMO

The final system, MegaMIMO, is described in Chapter 5. MegaMIMO is a cooperative wireless system that addresses the fundamental wireless scaling problem: in today's wireless networks, as the number of users increases, the throughput available to each user decreases.

Dense Wi-Fi networks today, for instance, in convention centers, hotels, stadiums, enterprises *etc.*, are unable to keep up with user demands [145, 66]. This has led to high profile failures in recent times like the collapse of the Wi-Fi network during Steve Jobs's iPhone 4 keynote. Cellular networks too face a similar challenge, with demands forecast to exceed capacity within the next few years [126]. This capacity crunch is despite dramatic improvements in individual link capacity and wireless device performance in recent years.



(a) 1 transmission for each antenna on a *single* AP

(b)1 transmission for each antenna on *all APs*

Figure 1-5: Traditional vs. Joint Multi-User Beamforming. In a traditional multi-user beamforming system with multiple 2 antenna APs, only 1 AP can transmit on a given channel at any given time. This leads to a maximum of 2 simultaneous packet transmissions regardless of the total number of APs. In contrast, MegaMIMO enables all APs to transmit on the same channel, allowing up to $2N$ simultaneous packet transmissions if there are N 2-antenna APs.

The problem, rather, is that user demands scale up with the number of wireless devices in the network, but total network throughput does not. MegaMIMO is a system that enables wireless networks to scale their throughput with the number of transmitting devices.

MegaMIMO allows multiple wireless transmitters to simultaneously transmit different packets to multiple independent wireless receivers. The key idea behind MegaMIMO is joint multi-user beamforming. Multi-user beamforming is a known technique that enables a MIMO transmitter to deliver multiple independent packets concurrently to receivers that each have fewer antennas. Fig. 1-5(a) shows a single 2-antenna access point deliver two independent packets to two single antenna receivers. In contrast, as shown in Fig. 1-5(b), MegaMIMO enables multiple access points operating in the same frequency bands to deliver packets concurrently to multiple independent receivers without interfering with each other. Such a system scales wireless throughput with the number of transmitting devices, and delivers as many concurrent packets as the total number of antennas across all APs. Furthermore, it can leverage continuing performance and reliability improvements of individual links and devices (*e.g.*, more antennas per device, lower receiver noise figure, better codes *etc.*).

The main challenge in implementing MegaMIMO is the need for phase-coherent transmission across distributed transmitters. Specifically, in order that each client can decode its intended signal without interference from other signals, beamforming attempts to ensure that, at each client, the signals intended for all other clients cancel each other out. Transmitters therefore need to control the relative phases of their signals in order to achieve this cancellation. In the case of a single transmitter, this requirement is naturally satisfied as all the transmitted signals are modulated by a single oscillator. However, in the case of MegaMIMO, since different transmitters have different oscillators with slightly different frequencies, the signals from the different transmitters rotate relative to each other. This causes their phases to diverge, thereby preventing beamforming from working.

It might seem that transmitters can keep their phases coherent by estimating their relative frequency offsets, say $\Delta\omega$, and compensating for the accumulated phase error over a time t as $\Delta\omega t$. However, such an approach is susceptible to even small errors in estimation of frequency offsets. For instance, even an error of 10 Hz (which is 4 parts per *billion* of typical Wi-Fi carrier frequencies, say 2.4 Ghz, several orders of magnitude smaller than the typical 802.11 tolerance of 20 parts per *million*, or cellular tolerance of 1-2 parts per *million*) can lead an accumulated error of 20 degrees (0.35 radians), within a short time interval of 5.5 ms. Such a high phase error will lead to significant interference during beamforming, and prevent receivers from decoding their packets. Thus, it is not practical to rely on frequency offset estimation to keep transmitters phase-coherent across several milliseconds.

MegaMIMO avoids this problem by directly measuring phase offsets at the time of interest, and correcting for them, rather than inferring them from frequency offsets. The key idea is to elect one of the APs as a lead, and use its phase as a reference for the system. Other APs act as slaves, measure the change in phase of the lead AP, and change the phases of the signals they transmit so that they maintain a desired alignment with the signals of the lead AP. In particular, the lead AP begins each MegaMIMO joint data transmission with a couple of symbols. All slave APs use these symbols to estimate their phase offset relative to the lead AP, and correct their subsequent signals during the joint data transmission. During the data transmission itself, slave APs use an estimate of the frequency offset to correct for changes in phase through the packet. This limits the phase error accumulation to within a packet, and slave APs use a long term average of the frequency offset to ensure that the phase error during a packet is bounded.

Chapter 5 expands on this idea, describes its implementation, and demonstrates that it can enable distributed transmitters to accurately beamform to multiple receivers. Further, it extends MegaMIMO to work with off-the-shelf 802.11 cards, so that MegaMIMO can be implemented simply by an upgrade to the AP infrastructure, without any changes to clients. Finally, it details results of MegaMIMO implementation and evaluation both with programmable radios and off-the-shelf 802.11 cards.

■ 1.5 Contributions

The high level contribution of this dissertation is to demonstrate a practical agile spectrum sharing architecture based on novel network primitives that enable cognition and cooperation in wireless networks, and prototype implementations and experimental evaluations that demonstrate large practical performance gains. The specific contributions of this dissertation that derive from this high level contribution are highlighted below.

■ 1.5.1 An Agile Spectrum Sharing Architecture

This dissertation establishes agile spectrum sharing as a practical architecture for wireless networks. The systems in this dissertation show how wireless devices can dynamically adapt the spectrum they use, as well as the signals they transmit to improve network performance and enhance spectrum utilization. SWIFT and FARA are architectures that divide spectrum among wireless devices in a usage- and performance-aware manner, while SourceSync and MegaMIMO enable multiple wireless devices to operate simultaneously in the same spectrum to leverage spatial diversity. The ability of all these wireless systems to dynamically share spectrum has been demonstrated using practical implementations and evaluations showing large performance gains.

■ 1.5.2 Wireless Coordination Primitives

The systems in this dissertation present several novel coordination and synchronization primitives that can enable a variety of cooperative wireless architectures. The key primitives presented are:

Adaptive Spectrum Sensing: SWIFT's spectrum sensing algorithm allows it to determine what bands are occupied by other wireless devices, and how they are affected by

SWIFT's transmissions. It shows how wireless systems can detect what frequencies are used by other wireless devices by leveraging the typical response of a wide variety of narrowband devices to interference, either at lower layers, for example, carrier-sense to abstain from transmission when other devices are using the medium, or autorate, where interference can lead to a device picking lower rates/less dense modulation schemes, or at higher layers, for example TCP back off on packet loss. In addition to spectrum sensing, SWIFT also demonstrates how a wireless sender-receiver pair can mutually agree on which spectrum bands to use for communication even in the absence of any control channel to achieve consensus.

Frequency Synchronization: SourceSync demonstrates how multiple transmitters, each with their own oscillator and differing carrier frequencies, can perform coarse frequency offset correction to ensure that adjacent subbands in the spectrum do not interfere with each other. It also shows how receivers can independently track the frequency offset of each of the independent transmitters and correct for them to enable the decoding of a joint transmission.

Fine-grained symbol level alignment: SourceSync shows how multiple transmitters can use shared packet reception as a robust, low-overhead synchronization mechanism to trigger joint transmission. In particular, it shows how wireless transmitters can overcome the inherent variability in packet detection that is induced by noise and receiver hardware variations, and achieve tight symbol alignment on the order of 20 *ns*.

Distributed Phase-Coherent Transmission: MegaMIMO builds on the frequency and symbol alignment in SourceSync to provide phase-coherent transmission across distributed transmitters. MegaMIMO is the first system that enables different transmitters that are powered by different oscillators and do not share a clock to emulate a single multi-antenna transmitter where all antennas are driven by the same oscillator.

In addition to the systems described in this dissertation, these primitives can enable a wide variety of existing information-theoretic schemes that rely on synchronization across wireless transmitters or receivers.

■ 1.5.3 Cross-layer Algorithms for Cognition and Cooperation

This dissertation demonstrates the power of a cross-layer approach to building cognitive and cooperative wireless networks. The main instances of such a cross-layer approach are:

- SWIFT's adaptive spectrum sensing leverages wireless response to interference both at lower (PHY and MAC) and higher (network) layers to determine usable spectrum bands for wideband transmission. Further, it works in conjunction with SWIFT's cognitive PHY to dynamically adapt the spectrum bands used by SWIFT both for sensing as well as for communication.

- FARA's cross-layer rate adaptation algorithm is PHY-aware, and uses fine-grained information from the PHY about performance in individual spectrum subbands to determine the optimal transmission rate for each subband. Additionally, FARA's medium access control (MAC) layer leverages this information to perform dynamic spectrum allocation across wireless links. This allows FARA to achieve significantly higher network performance than traditional wireless MACs that are PHY-unaware and divide spectrum across wireless links without regard to performance.

- SourceSync's physical layer provides coding and synchronization mechanisms that enable cross-layer sender diversity schemes at the link layer for both wireless mesh and last-hop wireless networks.

- MegaMIMO integrates the PHY, MAC, and network layers to enable multiple wireless transmitter-receiver links to operate at the same time. In particular, MegaMIMO's PHY layer modifies transmitted signals based on MAC layer decisions about which links will operate jointly at any time. MegaMIMO's MAC, in turn, uses PHY information about the channels characterizing different links to determine what combination of links can maximize network layer performance.

■ 1.5.4 Practical Evaluations Demonstrating Large Gains

All four systems presented in this dissertation have been deployed and evaluated in actual testbeds. The evaluations reveal that each system provides large performance and reliability gains. The actual gains depend on the particular system and are described in detail in the individual chapters, but the main evaluation results are highlighted below:

- SWIFT was evaluated in an indoor testbed the WiGLAN radio platform [37]. SWIFT safely coexists with narrowband devices while simultaneously providing high throughput and good range. In comparison to a baseline system that coexists with narrowband devices by operating below their noise level, SWIFT is as narrowband-friendly, but its throughput is $3.6 - 10.5\times$ higher, and its range is $6\times$ greater.

- FARA was implemented on the WiGLAN radio hardware, and evaluated in an indoor testbed. FARA is effective at harnessing frequency diversity, and delivers a median throughput gain of $3.1\times$ over traditional frequency-oblivious rate-adaption and MAC layers in our testbed.

- SourceSync was implemented using the WiGLAN radio platform, and prototypes of both last-hop diversity and opportunistic routing with sender diversity were evaluated in an indoor testbed. SourceSync achieves a median throughput gain of 57% over traditional last-hop schemes that pick the best AP. Additionally, the combination of SourceSync and ExOR [19] achieves a median throughput gain of up to 45% over ExOR alone, and up to 2× over single-path routing.

- MegaMIMO was implemented both in a testbed with Universal Software Radio Peripherals (USRP2) [41] nodes acting as APs and clients, and in a testbed with USRP2 APs and off-the-shelf 802.11n clients. MegaMIMO's throughput increases linearly with the number of APs. In particular, in an experimental testbed with 10 USRP2 APs and 10 USRP2 clients, MegaMIMO can achieve a median throughput gain of $8.1 - 9.4\times$ over traditional 802.11 unicast, across the range of 802.11 SNRs. MegaMIMO's ability to linearly scale the network throughput with the number of transmitters applies to off-the-shelf 802.11 clients. Specifically, MegaMIMO can transmit simultaneously from two 2-antenna APs to two 2-antenna 802.11n clients to deliver a median throughput gain of $1.8\times$ compared to traditional 802.11n.

Chapter 2
Learning to Share: Narrowband-Friendly Wideband Networks

Wideband technologies in the unlicensed spectrum can satisfy the ever-increasing demands for wireless bandwidth created by emerging rich media applications. The key challenge for such systems, however, is to allow narrowband technologies that share these bands (say, 802.11 a/b/g/n, Zigbee) to achieve their normal performance, without compromising the throughput or range of the wideband network. This chapter presents SWIFT, a system that allows wideband devices to dynamically detect which parts of the spectrum are occupied by narrowband devices, and weave together the remaining unoccupied parts of the spectrum to create a high throughput cognitive wideband system.

2.1 Overview

Users' desires to share high definition audio and video around the home are driving the need for ever-increasing wireless bandwidth [31, 69], and wideband radios, whose frequency bandwidth spans hundreds of MHz to many GHz, have been proposed as a solution [69, 152, 37]. These radios mainly operate in the unlicensed spectrum, which is populated by a variety of legacy narrowband devices (e.g., 802.11a/b/g, Zigbee), as well as a slew of emerging technologies (e.g., 802.11n). The key problem in operating these wideband systems is to ensure that they neither hinder the performance of narrowband devices in these bands, nor sacrifice their own throughput or operating range. Overcoming this problem requires a network design that achieves high throughput even when interferers continuously exist, a fundamental departure from traditional wireless networks, which are crippled by interference.

This chapter presents SWIFT, a **S**plit **W**ideband **I**nterferer **F**riendly **T**echnology that safely coexists with narrowband devices operating in the same frequencies. SWIFT's key feature is cognitive aggregation: the ability to create high-throughput wireless links by weaving together non-contiguous unused frequency bands that change as narrowband devices enter or leave the environment. Our design is motivated by measurement studies [32, 100] showing that, while various wireless technologies exist throughout the spectrum, only a few such technologies are usually operational in a house or small geographic area,[1] and hence a large number of non-contiguous frequency bands are likely to be unused. SWIFT's ability to detect and utilize exactly these unoccupied bands, and compose them to build a single wireless link, allows wideband networks to operate at normal power without affecting narrowband, and delivers on the promise of simultaneously achieving high throughput, operating range, and coexistence.

SWIFT bridges two areas in wireless communications: cognitive radios, and wideband and ultra-wideband design. While there has been a lot of interest in cognitive communication, most proposals have focused on the licensed spectrum [14, 9, 23], where the primary users of the band are known *a priori*, and hence this knowledge may be incorporated into detecting if the band is occupied by the known signal pattern. In contrast, SWIFT focuses on the unlicensed band, where narrowband devices are many, and their signal patterns are unlikely to be known. Further, cognitive proposals attempt to find a single unused band which they may opportunistically use, while SWIFT aggregates the bandwidth of many such bands to maximize throughput. Similarly to cognitive radios, Wideband (WB) and Ultra-wideband (UWB) technologies have to cooperate with existing users of the spectrum. They have, however, tried to bypass the coexistence problem by reducing their transmission power below the noise floor of narrowband devices [152, 105, 3], and limiting themselves to a single contiguous band. While this allows narrowband devices to operate unhindered, it sacrifices the WB device's throughput, operating distance, or both.

[1]The measured average spectrum occupancy is 5.2% [32].

To achieve its goal of high throughput, range, and narrowband-friendliness, SWIFT has to address three key challenges:

- *How does SWIFT detect the frequency bands that it must avoid, to allow narrowband devices to operate normally?* In the absence of any information about the narrowband signal, traditional solutions avoid frequency bands that show high narrowband power [9]. This approach uses observed power (or the lack of it) in a band as a proxy for whether interference in this band is detrimental (or irrelevant) to operation of the narrowband device, and is known to have both false positives and false negatives [141]. Instead, SWIFT has a novel *adaptive sensing* technique that exploits common network semantics, by observing that many unlicensed devices react when faced with interference, either at the lower layers [13, 40], or at higher layers [70]. This observation allows SWIFT to directly address the key goal of cognition: identifying frequency bands whose use could interfere with narrowband devices. Thus, SWIFT probes ambiguous frequencies, monitors the change in narrowband power profile, and backs away if it perceives narrowband reaction.

- *How does the PHY layer operate across chunks of non-contiguous frequencies?* The current PHY layer of high-throughput wireless systems assumes a known and contiguous communication band, and breaks down in the presence of narrowband devices. For example, even basic primitives like packet detection can be triggered incorrectly by power from narrowband transmissions. SWIFT introduces a *cognitive PHY* that incorporates cross-layer information from the adaptive sensing subsystem into the basic signal processing algorithms.

- *Given that different nodes might perceive different usable frequencies, how do SWIFT nodes communicate?* Varying proximity to narrowband devices between SWIFT transmitter-receiver pairs may lead to differences in their choice of usable frequency bands. Since state of the art high-throughput wireless systems (e.g. OFDM) communicate across a frequency band by striping the data bits sequentially across sub-frequencies in the band, disagreement in the set of usable sub-frequencies between a sender-receiver pair leads to unknown insertions and deletions in the data stream, which cannot be dealt with by typical error-correcting codes. SWIFT's *in-band consensus scheme* transforms these insertions and deletions into bit errors, which can be dealt with using standard error-correcting techniques, and hence enables communication despite uncertainty in the environment.

We have built SWIFT in a custom wideband radio hardware [37]. Our implementation addresses the major details of computational complexity, storage, and pipelining inherent in building a wideband wireless transceiver and apparent only at the hardware level. We evaluate our design in a testbed of wideband nodes and 802.11 narrowband devices. Our results reveal the following findings.

- SWIFT safely coexists with narrowband devices while simultaneously providing high throughput and good range. In comparison to a baseline system that coexists with narrowband devices by operating below their noise level, SWIFT is as narrowband-friendly, but its throughput is $3.6 - 10.5\times$ higher, and its range is $6\times$ greater.

- Adaptive sensing is effective. As compared to a threshold based approach, which is neither efficient for wideband nor safe for narrowband across all locations, adaptive sensing accurately identifies interfered frequency bands, and provides efficiency while still being safe for narrowband.

- SWIFT nodes can communicate despite disagreement over narrowband spectrum usage and tolerate up to 40% disagreement about the usable frequency bands.

To the best of our knowledge, SWIFT is the first system where wideband nodes are shown in a working deployment to coexist safely with unknown narrowband devices, while forming a network of their own.

2.2 Related Work

SWIFT brings together research in two threads of wireless communications: wideband systems, and cognitive radios.

(a) Wideband Systems. The last couple of years have seen tremendous successes in the implementation of WB and UWB radios [27, 37, 69, 152]. This work falls in two major categories: low power consumption, low-rate radios for precision location and tracking systems, and high throughput radios for personal area networks and wire replacement in homes and offices [69, 31].

An intrinsic problem for high-throughput wideband radios, however, is coexistence with narrowband devices with which they share the unlicensed bands. Prior work tries to avoid interfering with narrowband devices by transmitting below their noise level [152, 105]. This approach inherently limits the throughput and operating range of the WB radio [152]. Further, in many cases, it fails to achieve its goal of protecting narrowband devices [105, 3]. Mishra *et al.* [104] propose to detect and avoid WiMax operating in the same band as an ultra-wideband device. Their work however is specific to WiMax, and can deal neither with general narrowband devices nor with a dynamic environment. Also, their implementation considers only a wideband sender and does not include a wideband receiver.

While most prior work is focused on a single link and the PHY layer, SWIFT's components span multiple areas, including signal processing, coding, and network protocols, which together successfully address the issue of coexistence with dynamic and unknown narrowband devices.

(b) Cognitive Radios. The realization of the congested spectrum allocation and its inefficient utilization [32, 100] has led to a surge of interest in cognitive communications. Work here has largely focused on detecting unused bands (spectrum sensing) and providing methods for sharing these bands among cognitive radios (spectrum sharing).

Prior work on spectrum sensing focuses on the licensed band, where it is crucial that cognitive secondary users do not interfere with the licensed primary user. The most basic approach involves measuring the energy level in a band. Energy detection is cheap, fast, and requires no knowledge of the characteristics of the signal. However, choosing energy thresholds is not robust across a wide range of SNRs [9]. Though more sophisticated mechanisms such as matched filter detection [9] are more accurate, they require knowledge of the transmitted signal (modulation, packet format, pilots, bandwidth, etc.) and thus work only for known technologies.

Architectures for spectrum sharing fall in two categories: centralized and distributed [9]. Centralized approaches [2, 22, 21] require a controller, such as a base station or spectrum broker, to allocate spectrum to all cognitive users. Distributed approaches [162, 163, 14, 23, 93] have MAC protocols that rely on one or more control channels to coordinate spectrum access.

While our work builds on these prior foundations, it makes three major departures.

First, cognitive radios focus on finding a single contiguous unoccupied band, whereas SWIFT weaves together multiple non-contiguous unoccupied bands to create a high-throughput wideband link. Second, SWIFT introduces new spectrum sensing mechanisms that exploit network semantics to strengthen traditional energy based techniques for unknown signals. Third, SWIFT allows communicating nodes to agree on usable frequencies using a fully distributed consensus scheme that requires no control channels.

■ 2.3 Problem Domain

SWIFT is designed to provide high throughput wireless connectivity for rich media appliances in a home scenario. It operates in the unlicensed spectrum, and is intended to function in the presence of narrowband devices that utilize the same part of the spectrum, and which might persist for long periods, or arrive and depart within minutes or hours, e.g., a laptop utilizing an 802.11 wireless connection.

SWIFT is a cognitive architecture for OFDM wideband radios. We focus on Orthogonal Frequency Division Multiplexing (OFDM) because it has emerged as the technique of choice for the majority of wireless technologies, such as wideband digital communication [37], ultra-wideband [5], 802.11 a/g/n [13, 56] and WiMAX [11]. The rest of our description focuses on single antenna radios, but our ideas are also applicable to wideband MIMO radios, as they too use OFDM [20].

Robust detection of narrowband devices without any knowledge of their signal patterns or other characteristics is impossible [141]. Since it is impractical to assume known signal patterns in the unlicensed band, SWIFT focuses its design on the practical scenarios that could arise in the environment of interest. Specifically, SWIFT addresses situations in which the following constraints apply:

1. It is acceptable to treat narrowband traffic as best effort. Specifically, narrowband devices should continue to experience the same average throughput and loss rate in the presence of wideband nodes as without them, but their requirements are not any more stringent than what is expected from today's wireless LANs.

2. The capacity of the wideband network exceeds its peak traffic. This implies that the medium exhibits frequent idle intervals such that narrowband devices that perform

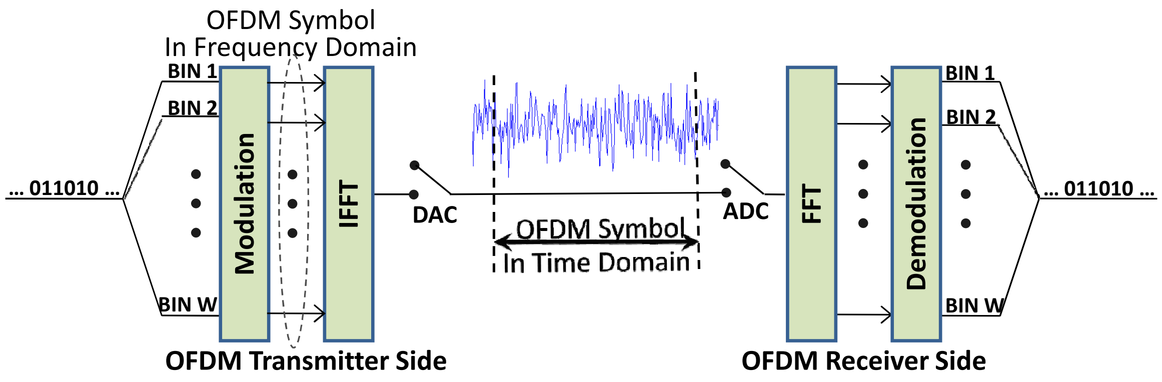


Figure 2-1: Schematic of an OFDM System.

carrier sense are not completely locked out. Sufficient capacity can arguably be obtained by increasing the spectrum width spanned by the wideband radio.

3. Narrowband technologies of interest in this chapter react to interference. This reaction can be at lower layers, for example, carrier-sense abstaining from using the medium, or autorate changing modulation schemes, or at the higher layer, for example, TCP backing off on sustained packet loss. Further, these devices are expected to operate at reasonable SNRs (a few dB above the noise floor, e.g. 802.11a/b/g/n). Narrowband devices that operate below or around the noise floor are expected to have their own mechanisms to combat interference, as they need them in such a regime.

■ 2.4 OFDM Background

This section provides a simplified description of OFDM focused only on issues related to this chapter. OFDM divides the used RF bandwidth into many narrow sub-channels, called OFDM bins. Each OFDM bin can be treated independently from other bins, and may use a different modulation (e.g., BPSK, 4-QAM) or transmission power. A data stream is striped into bits, with different numbers of bits assigned to each bin based on its modulation scheme. An assignment of modulated bits to each of the OFDM bins is called an OFDM symbol, see Fig. 2-1. The frequency representation of the OFDM symbol is converted to a time domain OFDM symbol by using an Inverse Fast Fourier Transform (IFFT) and sent on the medium by the transmitter.

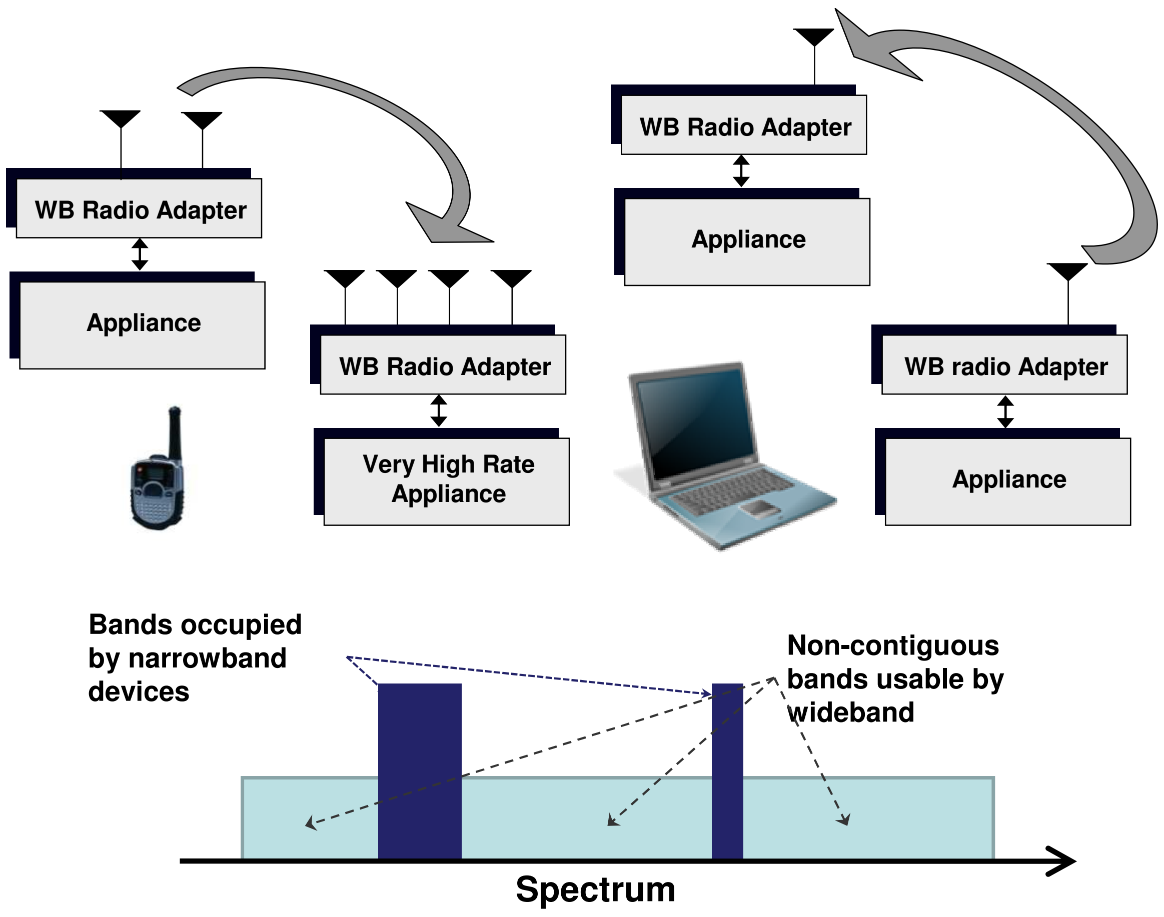


Figure 2-2: Cognitive Aggregation. While narrowband devices exist (e.g., 802.11 laptop), SWIFT still uses the remaining non-contiguous chunks of spectrum as if they were one wireless link.

The receiver first determines the exact sample at which the packet starts. It then aligns the time samples on OFDM symbol boundaries, and performs a few basic signal processing tasks like Carrier Frequency Offset (CFO) and channel estimation. Next, the aligned time signal is passed to a Fast Fourier Transform (FFT) module to produce the frequency representation. The data symbols are then converted to their frequency representation, corrected for the channel, and demodulated to retrieve the transmitted data bits.

■ 2.5 SWIFT

SWIFT is designed around the concept of *cognitive aggregation*. Similar to the cognitive radio vision, cognitive aggregation is based on detecting narrowband systems and avoiding their frequency bands. Unlike prior cognitive systems, which use only a single contiguous band, cognitive aggregation merges many non-contiguous bands into a single high-throughput communication channel, as shown in Fig. 2-2. Such a design is critical when using a wide band in the unlicensed spectrum since a wide contiguous unused band typically does not exist. SWIFT implements a cognitive aggregation design by utilizing three key components: (a) a spectrum sensing mechanism based on determining how SWIFT's selection of frequency bands impacts narrowband transmissions, rather than just how the narrowband transmissions look to SWIFT, (b) a cognitive PHY layer that can operate over non-contiguous spectrum bands, and (c) a consensus protocol that allows SWIFT nodes to agree on usable frequency bands despite uncertainty about which bands are occupied by narrowband devices. Below, we explain each of these components in detail.

■ 2.5.1 Adaptive Spectrum Sensing

SWIFT senders must learn the set of OFDM bins in which they can send while being narrowband-friendly.

How do we detect bins that interfere with narrowband?

Ideally, SWIFT could directly measure how its choice of transmit bins affects a narrowband device. Since this is typically not possible, and given that one does not know the signal details for arbitrary unlicensed narrowband devices, prior cognitive devices passively listen for narrowband devices, and avoid all frequency bins in which they see power above some threshold [9]. This approach essentially uses information about how SWIFT observes the narrowband transmissions to guess how a SWIFT transmission would be observed by the narrowband device. Such an approach is problematic for two reasons.

First, it is difficult to pick a power threshold [141] to precisely identify occupied bins, because the correct value varies with time and proximity to the narrowband device. Fig. 2-3 illustrates this issue. It shows the power profile of an 802.11a narrowband device operating on channel 52, as observed by two SWIFT nodes at different distances from the 802.11a transmitter. In this scenario, the narrowband device uses bins 3 through 23. Clearly, no single fixed threshold would eliminate exactly the correct set of bins used by the narrowband device at both locations. This problem becomes even worse in the presence of variable power levels among narrowband devices. For example, portable 802.11 devices such as laptops and handheld devices often transmit at power levels well below the maximum in

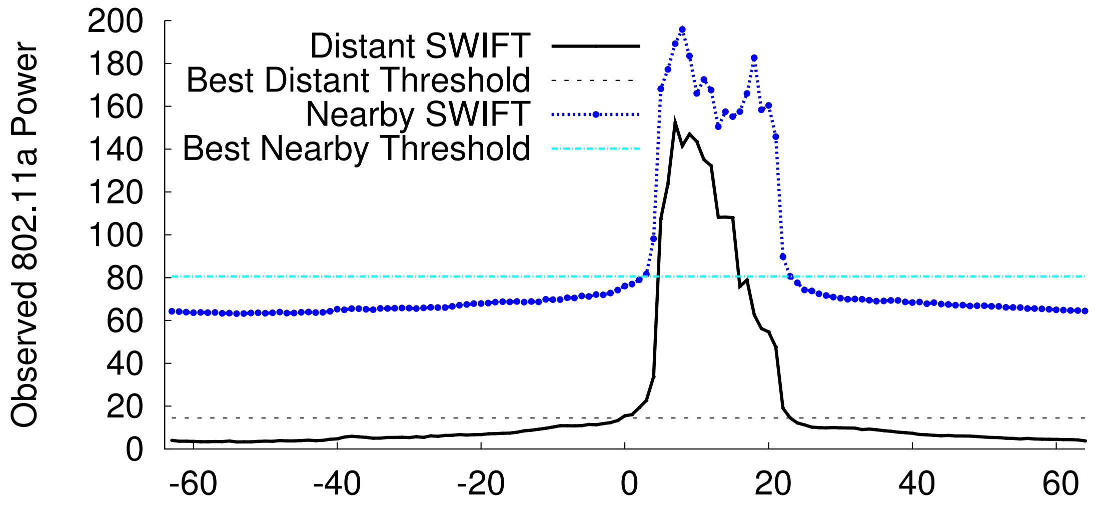


Figure 2-3: 802.11a Power Profile. The observed power of an 802.11a transmitter at different SWIFT locations is very different, highlighting the difficulty in picking a power threshold that works at all locations.

order to conserve their battery, meaning that even though SWIFT's effect on two devices in different locations might be very different, the transmissions from those two devices might be indistinguishable from SWIFT's perspective. Accounting for all this variability requires using a very conservative threshold that wastes many bins.

Second, even if one could identify the exact bins the narrowband device uses for its transmissions, this may not be the correct set of bins to avoid. Since transmitters leak power into bins adjacent to the ones they use, a wideband transmitter might need to avoid bins that are unused by the narrowband device if using them would leak significant power into the narrowband bins. Conversely, a wideband device might be able to use bins that are used by the narrowband device without affecting narrowband operation. This might happen if the narrowband device is far away from the wideband transmitter, or uses highly redundant coding schemes (e.g., Zigbee [164]). Because these effects depend on the distance and receive sensitivity of the narrowband device, it is impossible to account for them without being extremely conservative in the choice of threshold.

The key problem with current solutions is that they use the wideband device's view of the narrowband transmissions in an open loop, as a proxy for how the narrowband device will observe the wideband transmissions. Asymmetric links, and varying transmission powers and receive sensitivities, make this a poor proxy. SWIFT instead uses a technique we call *adaptive sensing*, which closes the loop by taking advantage of the observation that many narrowband devices react in some perceivable way if wideband transmissions disrupt their transmissions. In particular, a large class of narrowband technologies in the unlicensed spectrum reacts to interference, either at lower layers (e.g., carrier-sense and autorate) or higher layers (e.g., TCP or end-user backoff). Intuitively, SWIFT pokes the narrowband device by putting power in ambiguous bins, notes any changes in the narrowband power profile, and backs away if such a reaction is observed.

Note that our goal with adaptive sensing is not to use narrowband bins during short gaps in narrowband transmissions; rather, we design it to immediately relinquish bins that it suspects of being used by narrowband devices, and reuse them only when confident that the narrowband devices have disappeared for several minutes.

Detecting Narrowband Reaction

SWIFT continuously senses the medium whenever it is not sending or receiving a packet. It converts the incoming time signal to the frequency domain using an FFT, and then calculates the current power in each bin. These power measurements are used both to detect the existence of a narrowband device, and to identify whether the narrowband device has reacted to the wideband device.

SWIFT detects the presence of a narrowband device in a bin, by comparing the power in that bin to the *noise floor*. SWIFT computes the noise floor by taking advantage of its wide band. Since it is highly unlikely that narrowband devices are simultaneously present in all bins, SWIFT just computes the minimum power across all bins and averages it over time to estimate the noise floor. Before SWIFT runs its adaptive sensing algorithm to choose the correct set of bins, it uses a *conservative threshold* that declares a bin *narrowband-occupied* if the power in that bin exceeds the noise floor by 3 dB in any sample, and *narrowband-free* otherwise. A sample is considered narrowband-occupied if any bin in that sample is narrowband-occupied.

SWIFT also uses its power measurements to compute four metrics that capture the most common responses to interference.

- *Inter-transmission time* captures the behavior of narrowband devices that react to interference by backing off (e.g. 802.11 or TCP backoff). It is computed by counting the number of consecutive narrowband-free samples.

- *Transmission duration* captures the behavior of devices that fall back to more robust, lower rate modulation schemes, thereby taking a longer amount of time for each transmission (e.g. autorate in 802.11). It is computed by counting the number of consecutive narrowband-occupied samples.

- *Average narrowband power* allows SWIFT to deal with multiple narrowband devices in the same band (e.g., two 802.11 devices). If SWIFT interferes with a nearby device causing it to backoff, but a more distant device fills in the freed bandwidth such that none of the other metrics changes, the average power will significantly decrease, allowing SWIFT to detect the change. This metric is computed by averaging the power in narrowband-occupied samples over a window.

- *Probability of transmission immediately after SWIFT* captures whether SWIFT triggers the carrier-sense reaction of narrowband. If SWIFT triggers narrowband carrier-sense, the narrowband device will not transmit immediately after a SWIFT packet, because it waits to ensure that the medium is free (In 802.11, this translates to the DIFS, followed by a random contention window). The metric is computed by looking at the power immediately after SWIFT finishes transmitting a packet, and setting a flag to 0 if the sample is narrowband-free, and 1 otherwise. The probability is computed as the average of these flags over a recent window.

SWIFT maintains sufficient statistics to compute the mean and variance of each metric. To achieve high confidence in the value of a particular metric, SWIFT needs to collect multiple measurements of that metric. Note that for the first three metrics, SWIFT gets one measurement every time it sees a narrowband transmission. The last metric is different, however, in that it can be measured independent of whether the narrowband device transmits or not. If the narrowband device has nothing to send though, the fact that no narrowband transmission is observed provides no information. Hence, SWIFT only includes samples of this metric when it senses a narrowband transmission within some maximum time after a SWIFT packet (1 ms in our implementation). Thus, the confidence of our estimates of all four metrics depends only on how many samples are obtained, and is independent of how sporadically the narrowband device transmits.

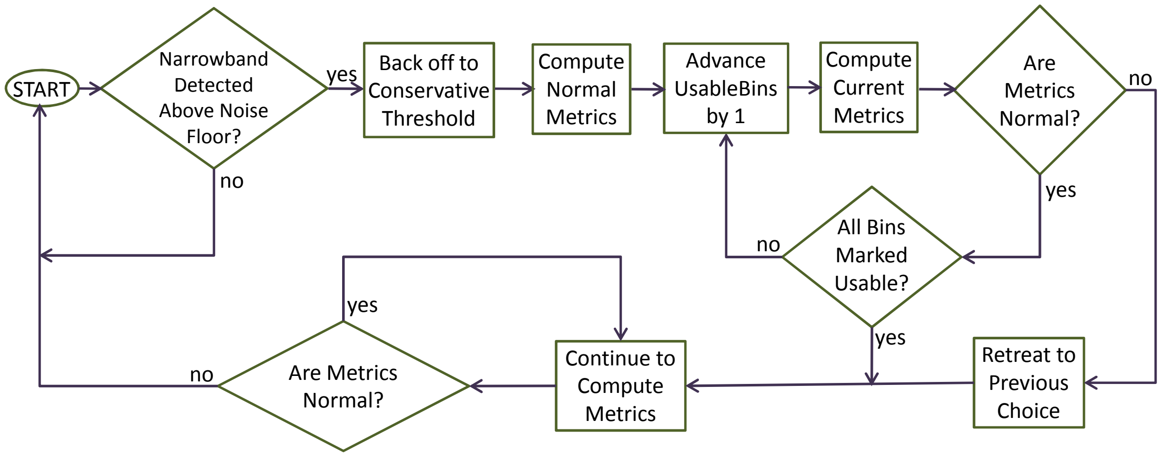


Figure 2-4: Control Flow for Adaptive Sensing Algorithm

Adaptive Sensing Algorithm

We define a bitvector `UsableBins`, which identifies the set of bins that SWIFT currently uses. The adaptive sensing algorithm starts with a conservative choice of `UsableBins` that does not interfere with the narrowband device, and iteratively tightens the setting of `UsableBins` to converge on the maximal set of usable bins that does not affect the narrowband device. Fig. 2-4 shows the control flow of our algorithm, which we describe in detail below.

Whenever SWIFT first detects narrowband power in a bin (using the conservative threshold), it immediately backs away from that bin, and updates `UsableBins` accordingly. This conservative choice of `UsableBins` allows SWIFT to be confident that observations made in this state represent normal narrowband behavior.

After gathering enough data at this normal setting, SWIFT begins the process of determining a choice of `UsableBins` that does not affect the narrowband device, but provides a maximal number of available bins. It starts by grouping contiguous sets of narrowband-occupied bins into a single *narrowband group*. Each narrowband group is then assigned a top and bottom bin which bound, for this narrowband group, the range of bins which must be left unused.

Next, SWIFT will try to grow `UsableBins` by using the top and bottom bins in each narrowband group and observing whether the narrowband device reacts. At each step, SWIFT alternates between reducing the top bin by one and increasing the bottom bin by one. For each choice of `UsableBins`, SWIFT waits to gather data measuring the effect of this new choice. It continuously monitors the incoming data by comparing the metrics with this bin choice to those observed under normal behavior with the conservative bin choice. If, at any point, SWIFT determines that it has impacted any of the metrics, it immediately moves back one step, and resets `UsableBins` to the previous decision. If, however, after gathering enough data, SWIFT determines that none of the metrics are impacted, it moves on to the next step, and tightens its choice further by one bin.

For each narrowband group, SWIFT independently continues this process until it either reaches a bin choice for which it notices the narrowband device reacting, in which case it retreats to the previous `UsableBins` setting, or it marks as usable all bins in this narrowband group and still notices no reaction. At this point, SWIFT continues to monitor the metrics and compare them to normal. If it notices a change at any point, SWIFT retreats to the conservative choice of `UsableBins`, recomputes normal metrics, and repeats the probing process, as shown in Fig. 2-4.

Note that this algorithm inherently deals with dynamics. For example, if the narrowband device moves closer or farther after SWIFT has finalized a bin choice, the average narrowband power metric will change from normal, and cause SWIFT to reinitiate the entire probing process. Furthermore, if all narrowband devices in a group depart, SWIFT will stop seeing any transmissions in the narrowband group, time out the entire group after a predefined interval, and reclaim these bins. Also, as articulated in Section 2.3, a narrowband device appearing in a new band currently occupied by SWIFT will always have the opportunity to transmit during SWIFT's idle intervals, and hence be quickly detected, allowing SWIFT to immediately back away and trigger the adaptive sensing algorithm for this new narrowband group.

Measuring Statistically Significant Changes

When should SWIFT decide that changes in some metric are not due to statistical aberrations, but reflect a real change in the performance of the narrowband device?

SWIFT uses a statistical test called a *t-test*, typically used to decide whether a drug has had a statistically significant effect on the population studied [25]. A t-test takes the means, variances, and number of samples of the two compared sets: normal and current. It computes the following *t-value* where the $\bar{x}$'s and σ's represent the means and standard deviations, respectively, of the two sets, and n_{norm} and n_{curr} refer to the number of samples

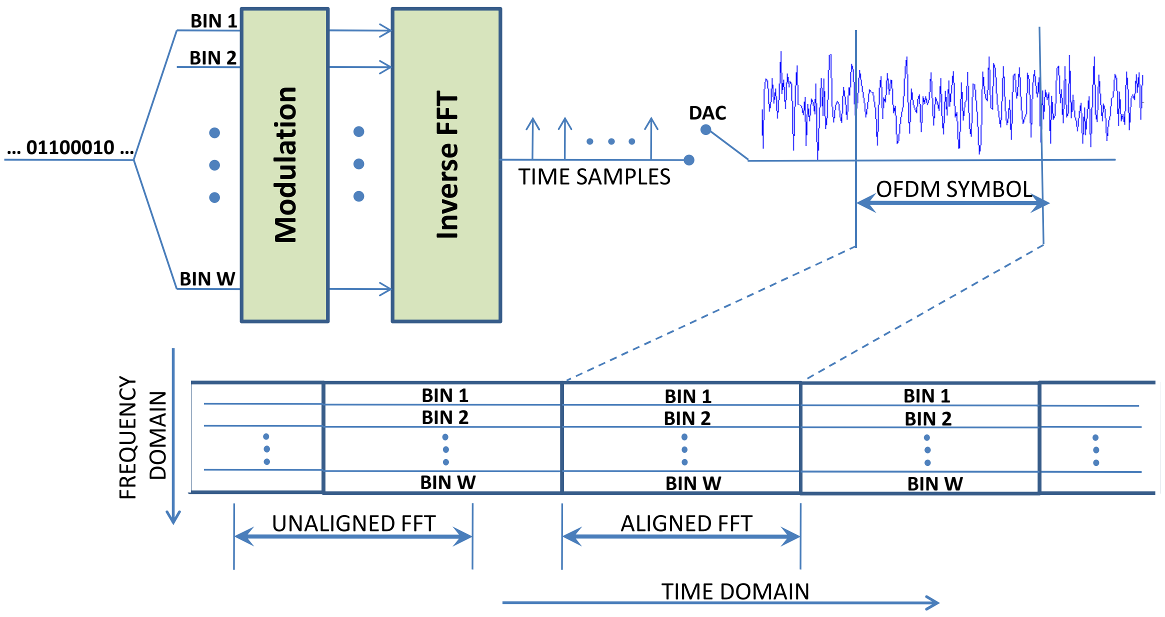


Figure 2-5: Conversion of bits into OFDM symbols: Values in individual frequency bins are combined in each time sample, and can be recovered only by computing appropriately aligned FFTs.

in the normal and current set:

$$t = \frac{\bar{x}_{norm} - \bar{x}_{cur}}{\sqrt{\frac{\sigma_{norm}}{n_{norm}} + \frac{\sigma_{cur}}{n_{cur}}}}.$$

To determine whether any difference between the means is statistically significant, the t-value must be combined with an alpha level, which represents the acceptable probability of being wrong. In our case, this value represents the probability that the t-test will tell us that SWIFT is interfering even if it is not. This is a parameter which effectively sets the aggressiveness of SWIFT. We use an alpha level of 0.05, typical for scientific and medical studies. The t-value combined with the alpha level and the total number of samples is then used in a table look-up to determine whether the t-test passes, i.e., whether SWIFT has had a statistically significant impact on narrowband.

■ 2.5.2 Cognitive PHY

The cognitive PHY uses the output of adaptive sensing to provide a single high-throughput link over the set of usable bins.

On the transmitter, this means ensuring that no power is used in bins marked as narrowband-occupied by the adaptive sensing module. This is straightforward with OFDM since it naturally allows different power assignments for each frequency bin.

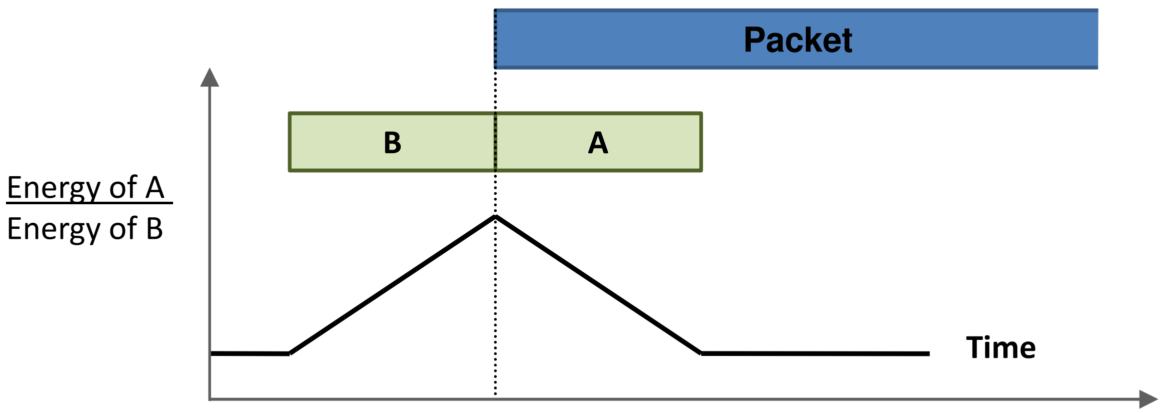


Figure 2-6: Variation of the ratio of energy in A to the energy in B as the window positions vary relative to the packet.

On the receiver side, the cognitive PHY has to ensure that the receiver can receive in non-contiguous bins even when narrowband devices are using the other bins. At first, it might seem that this can be done analogous to the transmitter by taking the FFT of the incoming signal, and just using values from the bins of interest. However, this is impractical. To understand why, consider the frequency-time diagram in Fig. 2-5 which illustrates how the N OFDM frequency bins are converted to N time samples that together represent an OFDM symbol. As can be seen, the correct frequency domain values can be retrieved from the time samples only when the FFTs are aligned correctly on OFDM symbol boundaries. But the receiver can align the FFT correctly on symbol boundaries only if it knows the starting sample of a packet in the first place!

Hence, we need to modify a few basic receiver algorithms to cope with non-contiguous bands.

(a) Receiver Packet Detection: In order to perform any processing on a packet, the receiver first needs to determine the start of the packet within a few time samples. Typically, this is done using the double sliding window algorithm [63], which uses energy ratios to determine the time sample where a burst of energy is received on the medium. This algorithm calculates the ratio of received energy over two consecutive sliding time windows as shown in Fig. 2-6. When only noise is received, the ratio is flat since both windows essentially contain only the same energy. When the packet edge starts to cover the A window, the energy in the A window gets higher, while the B window still contains only noise energy. The ratio keeps rising and ideally peaks at the point when the edge between A and B is aligned exactly at the start of the packet.

Since packet detection happens in the time domain, it cannot distinguish between energy from narrowband devices and wideband transmitters, and can be spuriously triggered by narrowband transmissions. Recall that SWIFT concurrently transmits with narrowband devices by using separate frequencies. Hence, if the receiver is kept busy with false packet detections, it is very likely to miss desired wideband transmissions.[2]

The solution is to actively filter the narrowband devices, allowing the receiver to perform packet detection on the clean signal consisting primarily of power from wideband transmitters. The choice of the bins to filter is driven by the adaptive sensing module. However, the receiver may not be able to use a filter per narrowband group since filters are resource-intensive in hardware. Hence, SWIFT is designed to use a small fixed number of bandstop filters, whose widths and center frequencies are dynamically configured. Note that since these filters are purely on the receiver side, by definition, they do not affect narrowband devices. A particular filter choice that is not perfectly aligned with the desired set of bins to be filtered only affects packet detection to the extent of the amount of narrowband energy that it lets in, or the amount of wideband transmitted energy it filters out.

We formulate the filter computation problem as a dynamic program that eliminates as many narrowband bins as possible, while maximizing the amount of received wideband energy. The module first compresses the input bitmask into a sequence of runs of consecutive 1's and 0's. Since a single narrowband interferer occupies a continuous portion of the spectrum, this transforms the mask, which was originally as large as the total number of bins of the wideband system, into a much smaller number, L, proportional to the number of active narrowband interferers.

Let K be the maximum allowable number of filters. For a given bitmask and a filter assignment, for each run r, define

$$\mathbf{X_r} = \begin{cases} \text{Length of run } r & \text{if } r \text{ should be filtered but is not} \\ & \text{included in the filter} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{Y_r} = \begin{cases} \text{Length of run } r & \text{if } r \text{ should not be filtered but is} \\ & \text{included in the filter} \\ 0 & \text{otherwise} \end{cases}$$

[2]Due to the hardware pipelining typical to receivers [63], they cannot receive packets while they are still working on the spuriously detected packet and have not rejected it.

The cost, $C_{i,j,k}$, of using at most $k, 1 \leq k \leq K$ filters in the interval $[i,j], i < j, 1 \leq i,j \leq L$, is calculated as $\alpha X_r + \beta Y_r$ where α is the weight assigned to bins that are not filtered, and β is the weight assigned to bins spuriously filtered. α and β might be set to 1 in the simple case, but more generally, can be set proportional to the average interference power and transmission power perceived by the receiver. We now wish to minimize $C_{1,L,K}$. This problem exhibits the "optimal substructure" property, *i.e.* the cost of a combined run can be derived from the cost of two sub-runs, and hence can be solved by dynamic programming. Specifically, we compute:

$$C_{i,j,k} = \min\left\{C_{i,j,k-1}, \min_{i<t<j, 0\leq f\leq k}\{C_{i,t,f} + C_{t,j,k-f}\}\right\}$$

Intuitively, the first component propagates the cost of using $k-1$ filters up, while the second splits the interval $[i,j]$ into two sub intervals, one of which contains some or all of the filters, and the other the remaining intervals. It only remains to define the initial conditions: $C_{i,j,0} = \alpha X_r$, and $C_{i,j,1}$ can be computed in $O(L^2)$ time by trying all possible single filter assignments in the interval $[i,j]$.

We compute the dynamic program top-down using a table to memoize the costs. The table has at most $O(L^2K)$ entries, and each entry is computed in $O(LK)$ time for a total computational complexity of $O(L^3K^2)$ time. Note that, in practice, the running time is insignificant for several reasons: (a) The total number of filters, K, is small, usually less than 5, (b) the total number of active narrowband interferers, L, is negligible compared to the size of the band, and (c) many of the memoized costs can be used in future runs as the set of used bins usually changes only marginally between iterations of the system.

(b) Receiver Packet Processing: Now that the start of the packet has been detected accurately, the receiver has the right alignment for the symbols and the rest of the packet processing can be done in the frequency domain over the actual bins used by the wideband system, as shown in Fig. 2-7. Specifically, carrier frequency offset estimation, which is traditionally done in the time domain, is instead performed in the frequency domain after zeroing out the contributions of bins occupied by narrowband, as determined by adaptive sensing. This permits a more precise estimate than an application of the time domain estimation algorithms on the noisy filtered signal used for packet detection.

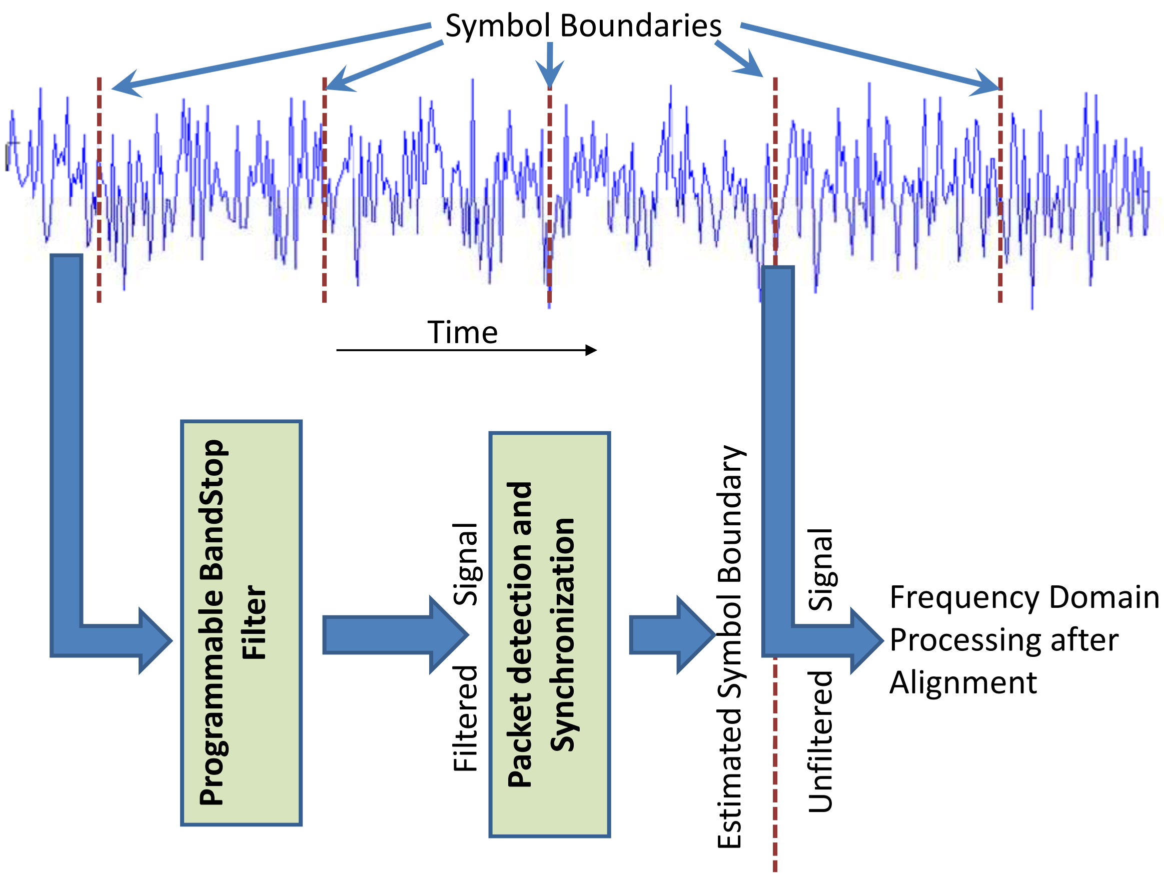

Symbol Boundaries
Time
Programmable BandStop Filter
Filtered Signal
Packet detection and Synchronization
Estimated Symbol Boundary
Unfiltered Signal
Frequency Domain Processing after Alignment

Figure 2-7: Packet detection uses a filtered version of the time-domain signal to produce an estimate of the packet start. The aligned signal can then be processed in the frequency domain for the rest of the chain.

(c) Data reception: Recall that the transmitter, while assigning data to bins, zeros out all bins that are deemed unusable by adaptive sensing, and stripes data only across the remaining bins. Similarly, when the receiver collects the received data, it only utilizes bits from bins that are deemed unoccupied by narrowband devices. Again, we note that since data reception happens after the alignment provided by packet detection, it can work on the unfiltered signal and hence can precisely remove bins susceptible to narrowband interference.

■ 2.5.3 Communication Over Uncertain Bands

Since each node in a SWIFT network independently decides the bands that it can use for transmission and reception, differences in proximity to narrowband devices and variations in time make it likely that a transmitter and receiver identify different bins as usable. For example, a wideband sender and receiver that are just a few meters apart may differ in

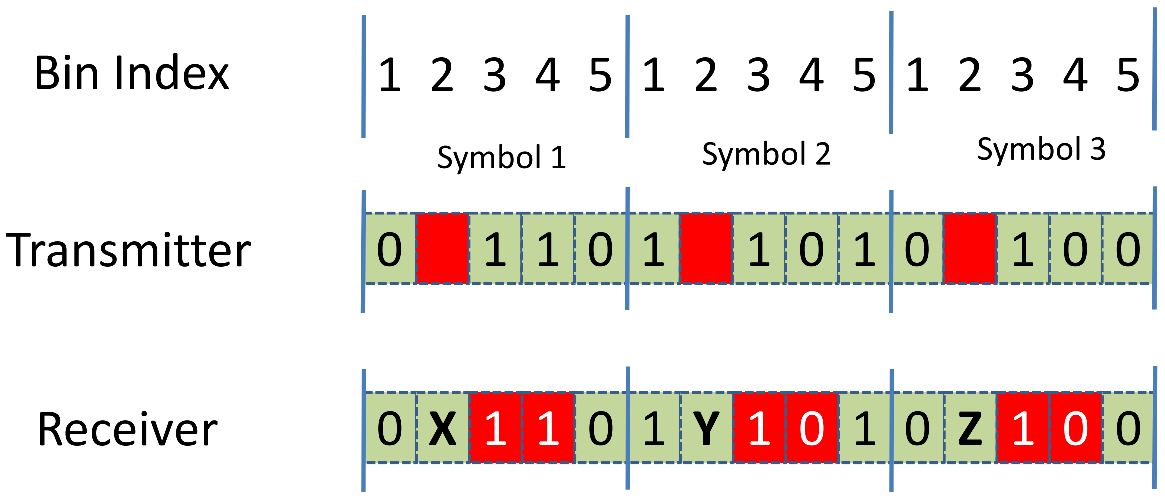


Figure 2-8: Bin Disagreement Causes Communication Failure. If the transmitter sends in bins 1, 3, 4, and 5 while the receiver listens in 1, 2, and 5, then the receiver will decode noise in bin 2 as data, and miss data in bins 3 and 4. These insertions and deletions will cause a misalignment in the demodulated data stream, creating an error pattern than cannot be rectified by standard error-correcting codes.

their perspectives of narrowband-occupied bins by as much as 10-20 MHz as we show in Section 2.7.2.

This disagreement between a transmitter and its receiver can be a fatal obstacle to establishing an OFDM communication link. To understand why, recall that an OFDM transmitter stripes data across all usable OFDM bins. A receiver reconstructs the original data by extracting bits from the individual bins. Thus, as shown in Fig. 2-8, if the receiver expects data in a bin that the transmitter did not send in, it will result in insertion of bits into the data stream. Conversely, if the transmitter sends data in a bin that the receiver does not expect data in, it will manifest itself as deletions of bits from the data stream. Thus, disagreements about bins result in alignment and framing errors, and produce a wireless channel that has unknown insertions and deletions, which conventional error correcting codes cannot deal with.

We solve this problem using two mechanisms: (a) an infrequent synchronization phase when the communicating wideband pair has a drastic disagreement, say, when a wideband node boots up, or when many narrowband devices in different bands appear simultaneously, and (b) a low overhead handshake, which is used when nodes that have previously agreed experience a limited disagreement, say, because a single narrowband device was turned on or moved closer.

SWIFT nodes are equipped with a robust initial synchronization mechanism. Each SWIFT node divides the whole transmission band into chunks of 16 bins, checksums and codes the value of its `UsableBins`, and sends it simultaneously in all chunks. Assuming that the bandwidth of the wideband node is large enough, and has enough bins that are not interfered with narrowband, at least one of these chunks in this *sync packet* will be received correctly, allowing the nodes to establish connectivity. Note that the sync packet uses all OFDM bins, and hence does not suffer from an alignment problem.

Even after a SWIFT node pair is synchronized, they can still suffer from occasional disagreements, for example, when adaptive sensing changes the set of usable bins on a node. We leverage the existing agreement to transform the potential disagreements into bit errors, *i.e.*, we transform the hard problem of unknown insertions and deletions into the simpler problem of bit errors, a problem that all wireless links know how to deal with by adding practical error correcting codes.

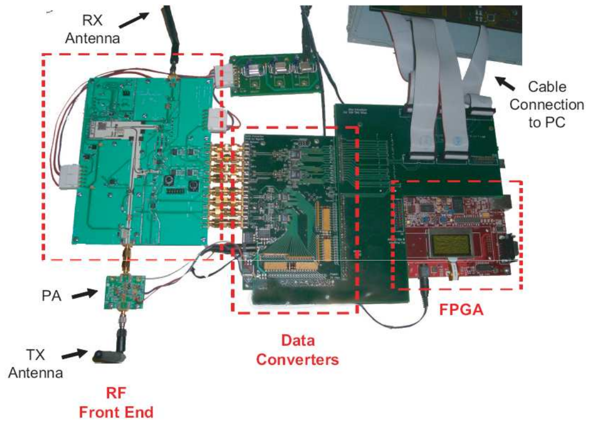
To do so, SWIFT exploits the following key observation. If the transmitter stripes the data across the previously agreed bins, there will be no deletions or insertions. The problem, however, is that, by transmitting in the old bins, some of which may no longer be free, the transmitter might hinder a narrowband device. To address this problem, SWIFT stripes the data across the previously agreed bins, but transmits only in the subset that is still usable. The receiver, which still expects to receive data across the old agreement, receives data in the intersection of the old and new bins correctly, but sees errors in the other bins. However, this can be easily fixed by using a simple error correcting code with sufficient redundancy to cover the expected extent of disagreement between old and new bins.

SWIFT uses a low-overhead handshake to quickly resolve disagreements. The data in the handshake is the new set of usable bins, and the striping technique is as described above. Once the handshake terminates, the nodes resume normal data exchange.

2.5.4 Network Issues

This section briefly describes how we compose multiple SWIFT links to build a network.

(a) The MAC: We use a carrier sense based MAC similar to 802.11 [49]. A node senses the medium and transmits if the medium is not busy. However, a direct application of the carrier sense technique of narrowband radios, which just checks for the total received power



Parameter	*Value*
Carrier Freq.	5.247 GHz
Data BW	100 MHz
Number of Bins	100 ($\times$1 MHz)
Symbol Period	1.4 μs
Uncoded BER	10^{-3}
Bin Modulation	BPSK, 4- 16-, 64-QAM
Max Link Len	10 m
Average Output Power	7.5 dBm

Figure 2-9: Wideband Radio Used in SWIFT

in the band to exceed a threshold, will unnecessarily reduce the transmission opportunities of SWIFT nodes since narrowband transmitters are always likely to be using some part of the band and hence preventing the wideband radio from transmitting. Instead, SWIFT's carrier sense focuses only on the bins declared usable by adaptive sensing. Specifically, when a node wants to send, it computes an FFT of the observed power, and proceeds with its transmission only if a large fraction of its usable bins are below the wideband carrier sense threshold.[3] The use of frequency-domain carrier sense ensures that SWIFT does not interfere with ongoing narrowband transmissions in the frequency bands used by SWIFT, even if those narrowband devices had not been previously detected by adaptive sensing.

Further, while wideband nodes can use an 802.11-like MAC, they need to wait for a relatively longer period to check that the medium is idle, *i.e.*, they should use a longer DIFS interval than typical values picked by narrowband devices. This ensures that a narrowband device that has just arrived into the environment can quickly access the medium and trigger adaptive sensing.

The SWIFT MAC randomly jitters the start of a probing epoch to ensure that different SWIFT nodes perform adaptive sensing independently. Further, a node uses control packets analogous to RTS/CTS to notify other SWIFT nodes of the start and end of a probing epoch in order to avoid simultaneous probing by multiple nodes. While this solution works for small wideband networks, extensions to larger networks may require more sophisticated mechanisms to leverage probing results across multiple SWIFT nodes.

[3]Note that the objective of wideband carrier sense is not to correctly decode the received signal, but rather to measure received power, which does not require alignment.

(b) Transmitter Identification: The alert reader might have observed that a SWIFT receiver potentially needs to receive and decode packets from multiple transmitters; however, decoding a packet requires knowledge of the exact set of mutually agreed bins over which the data is striped, and this mutual agreement is likely to be different with different transmitters. Hence, the SWIFT receiver needs to identify the transmitter of a packet even before it can decode the packet. This is in contrast to current networks where a node decodes received packet headers to determine if they are intended for itself.

SWIFT adapts the technique of correlation with known pseudonoise sequences, typically used for packet detection, to develop a solution at the link layer. It is well known that pseudonoise sequences exhibit low correlation with each other while showing high correlation with themselves, thereby allowing identification of specific pseudonoise sequences purely by correlation [118]. Transmitter MAC addresses in SWIFT are pseudonoise sequences, and appear in a known and fixed symbol location in the received packet. When a receiver detects a packet, it correlates it against its neighboring nodes' MAC addresses to determine the transmitter, and hence the set of bins. This requires a receiver to maintain a table of neighbor MAC addresses; a receiver learns about a neighbor's MAC address during the initial sync packet where they exchange their mutually usable set of bins. Note that receiving the sync packet itself does not require prior bin agreement, as described in Section 2.5.3.

■ 2.6 Implementing SWIFT

We have implemented SWIFT in a custom wideband radio transceiver platform developed by the WiGLAN research project [37]. The WiGLAN transceiver board, shown in Fig. 2-9, connects to the PC via the PCI bus, and acts like a regular network card. The transceiver [98] consists of three parts: 1) the RF front-end, which captures the analog signal, 2) the data converters, which convert between analog and digital signals, and 3) the digital baseband modem. All digital processing, such as packet acquisition, channel estimation etc., is done in baseband.

Our prototype has two components: the driver and the firmware. The former is implemented in software, and the latter in FPGA.

Driver: The driver presents a standard network interface to the kernel. In addition to

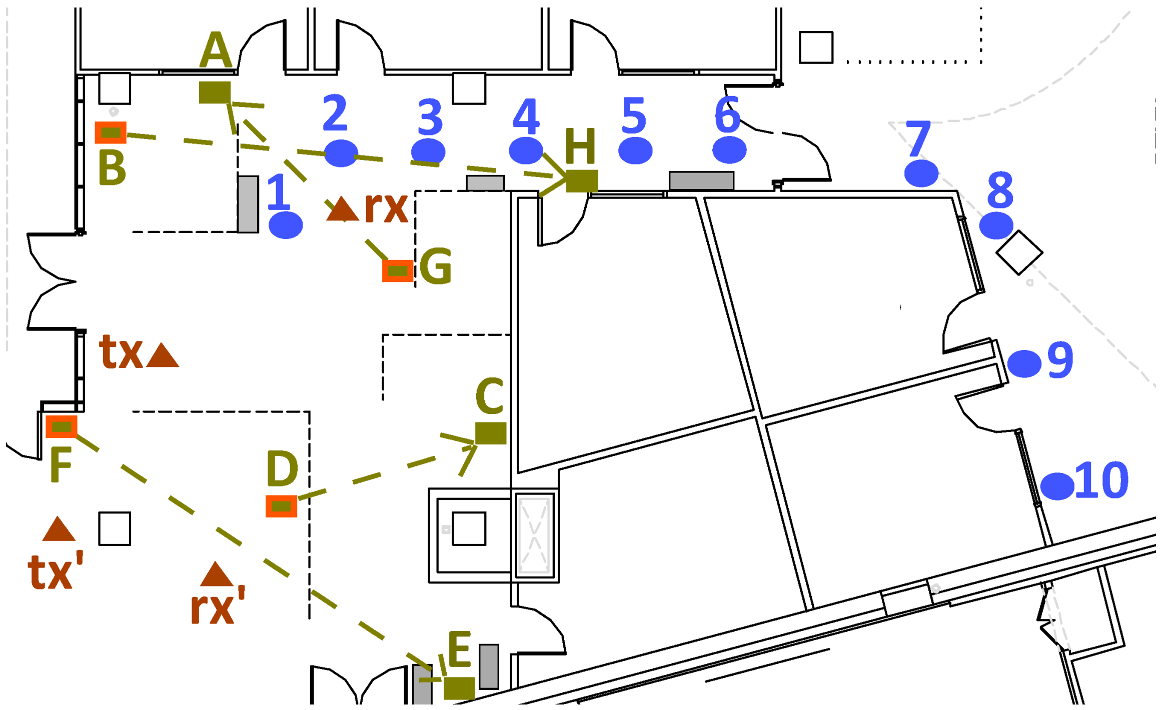
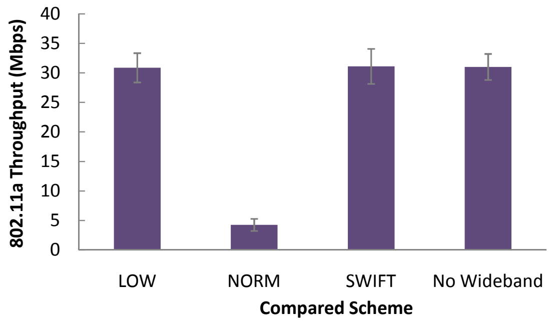


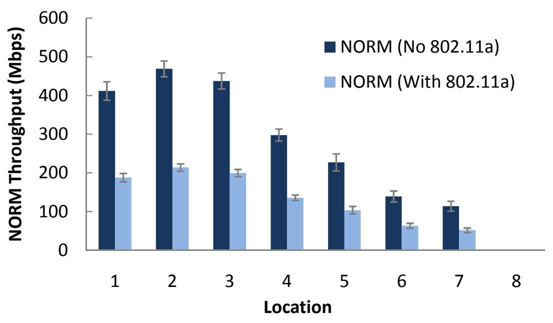
Figure 2-10: Testbed Map: Node Locations are Highlighted.

this typical functionality, the driver offloads from the FPGA any computation that is too complex for hardware and is not on the critical path of an OFDM symbol. For example, the driver implements the metric computation and t-test (Section 2.5.1). Our current prototype implements two metrics: average narrowband power, and probability of transmission immediately after SWIFT.

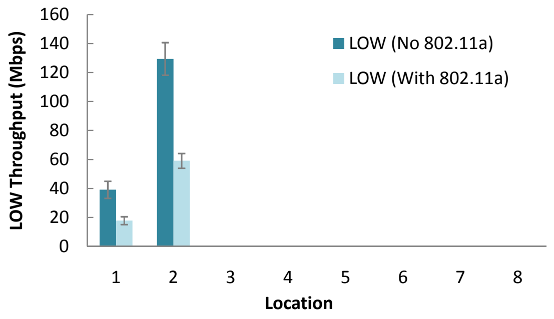
Firmware: Several of SWIFT's major components that need to be on the critical path, such as narrowband power measurement (Section 2.5.1), the cognitive PHY (Section 2.5.2), the band consensus protocol (Section 2.5.3), and the MAC (Section 2.5.4), are implemented on the FPGA. We design SWIFT's algorithms in the Simulink environment, which has a hardware model for the Xilinx Virtex-4 SX35 FPGA that we use. The code is then compiled into an intermediate form using Xilinx tools [4]. We use Verilog to integrate this intermediate form with the PCI subsystem, and create the final hardware representation of our code.



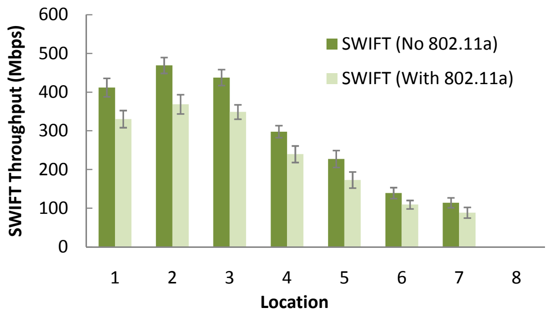
(a) Throughput of 802.11a



(b) Throughput of Non-Adaptive Wideband (NORM)



(c) Throughput of Low-Power Wideband (LOW)



(d) Throughput of SWIFT

Figure 2-11: Approaches to Narrowband-friendliness: Presents the throughput-range tradeoff, and shows that SWIFT, illustrated in (d), is as friendly to 802.11a as LOW, while attaining dramatically higher throughput and operating range.

2.7 Performance Evaluation

We evaluate SWIFT in a 12 node testbed consisting of four wideband nodes, and eight 802.11a nodes. Fig. 2-10 shows the experimental environment, which has high diversity due to the presence of walls, metal cabinets, desks, and various non-line-of-sight node locations. The exact choice of node locations for each experiment will be described along with the results for that experiment.

Wideband Devices. We use the WiGLAN wideband hardware described in Section 5.10, whose specifications are in Fig. 2-9. It has 100 OFDM data bins, numbered from -50 to +50, with bin 0 never being used. For all schemes, the wideband devices are evaluated while continuously sending 10 ms packets with a 1 ms gap between packets.

Narrowband Devices. These nodes run 802.11a in channel 52, corresponding to wideband bins 3 through 23. 802.11a nodes send UDP streams at the highest rate supported by the medium, except for experiment 2.7.5, in which they use TCP. The protocol, signal details, and occupied bands of 802.11a are, of course, unknown to SWIFT.

Compared schemes. We compare the different schemes by configuring our wideband hardware to run one of:

- **SWIFT**: This is the SWIFT protocol implemented as in Section 2.6.
- **Low-power wideband (LOW):** This is a baseline system that operates below the noise level to avoid interfering with narrowband devices. Specifically, it transmits signals with a power spectral density of -41 dBm/MHz, the FCC maximum for UWB devices [1].
- **Non-adaptive wideband (NORM):** This is a system that transmits across a wide band at the normal power of our hardware platform, but does not adapt to narrowband devices.

Note that both LOW and NORM will suffer drastic bit errors in bins used by 802.11a when it is turned on. For conservative comparison in this case, we therefore consider idealized versions of these systems that use the minimal amount of coding required to correct these errors.

■ 2.7.1 Throughput and Range

This experiment explores if it is possible to be as narrowband-friendly as a transmitter operating below the noise level, while preserving the good throughput and range of a normal-powered wideband system.

Method. We place the wideband transmitter in location tx, and test its performance to the wideband receiver which is placed in each of locations 1 through 10. For each location, we measure the throughput of LOW, NORM, and SWIFT with and without interfering 802.11a traffic, and plot the results in Fig. 2-11.

Results. Fig. 2-11 demonstrates that, while both NORM and LOW are flawed, SWIFT delivers on the fundamental goal of simultaneously achieving the high throughput and wide range of NORM, while being as narrowband friendly as LOW. In particular, we see that:

- **Throughput and range of LOW are limited:** Fig. 2-11(c) shows that LOW gets no throughput after location 2, and has $3.6 - 10.5\times$ lower throughput than SWIFT and NORM.

- **NORM is not narrowband friendly:** We can see from Fig. 2-11(a) that NORM significantly reduces 802.11a throughput.

- **SWIFT has high throughput and range:** From Figs. 2-11(b) and 2-11(d), we can see that in all locations, SWIFT achieves the same or greater throughput than NORM, with or without 802.11a.

- **SWIFT is narrowband friendly:** From Fig. 2-11(a), we can see that 802.11a throughput is unaffected by SWIFT.

We see from Figs. 2-11(b) and 2-11(d) that SWIFT surprisingly achieves higher throughput than NORM in the presence of 802.11a. This is because SWIFT intelligently avoids 802.11a occupied bins, while NORM uses these bins, suffers errors due to high narrowband power, and hence incurs additional overhead to correct errors in these bins.

■ 2.7.2 Power Threshold Sensing

In Section 2.5.1, we discussed the intractability of a threshold based algorithm. Here, we present results validating that claim, first showing the difficulty of picking a threshold, and, second, showing that a single threshold cannot simultaneously be safe for narrowband, and efficient for wideband.

Difficulty in Using Thresholds

Method. This experiment uses one pair of SWIFT nodes at location tx and rx in Fig. 2-10, and one pair of 802.11a nodes which is moved among locations 1-10. At each location, we measure two quantities: (a) *Correct Bin Choice*: We disable adaptive sensing on SWIFT and manually try all possible usable bin settings until we find the maximal set of usable bins that does not affect 802.11a throughput.
(b) *Ideal Threshold*: This is defined for each location as the highest threshold that results in a bin choice which does not affect 802.11a in that location. This is the threshold that is most efficient for wideband, while still being safe for narrowband. We record the time average of the power SWIFT sees in each bin when 802.11a transmits, and calculate the ideal threshold as the minimum power across all bins that must be left unused to ensure safe 802.11a operation.

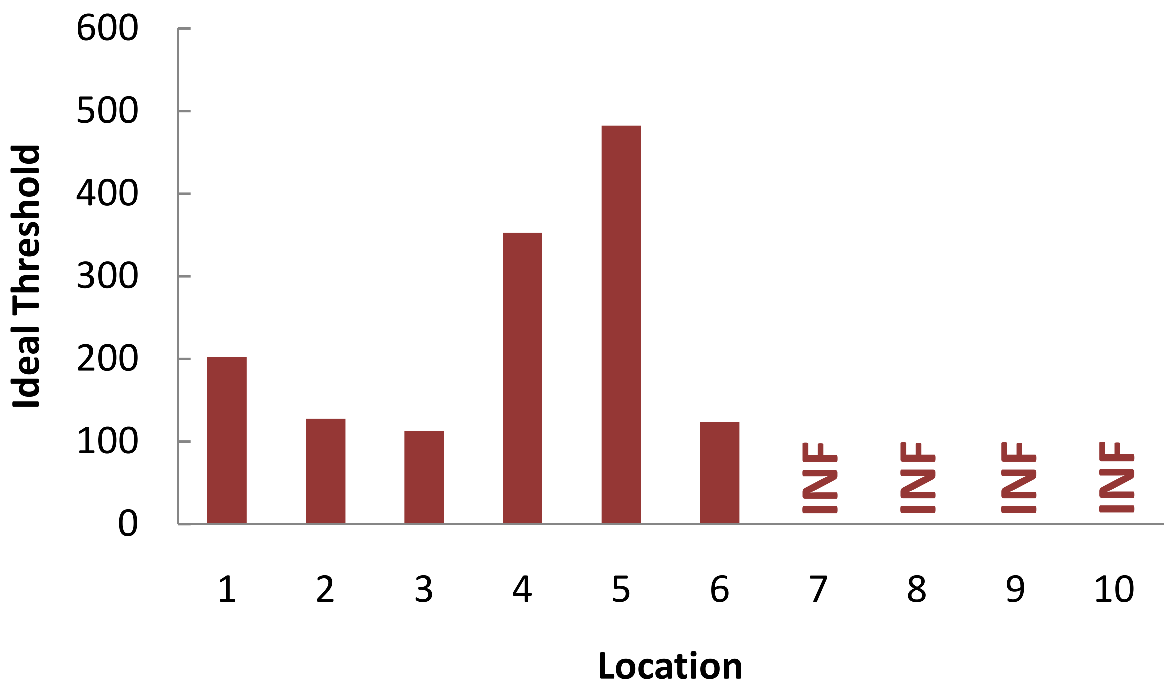
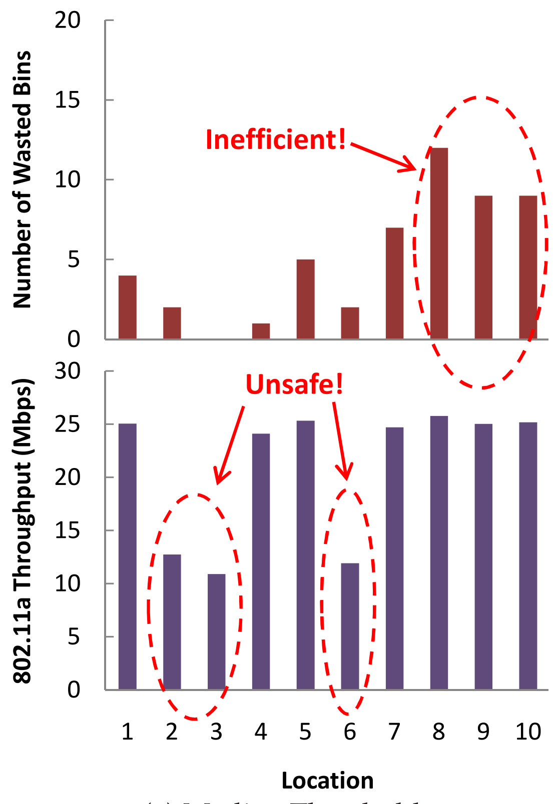


Figure 2-12: No Single Threshold Works Across Locations: This figure plots the ideal threshold that ensures safe narrowband operation while maximizing bins usable by wideband. 802.11a nodes at locations 7-10 are not affected by wideband, and hence the ideal threshold for these locations is infinity.

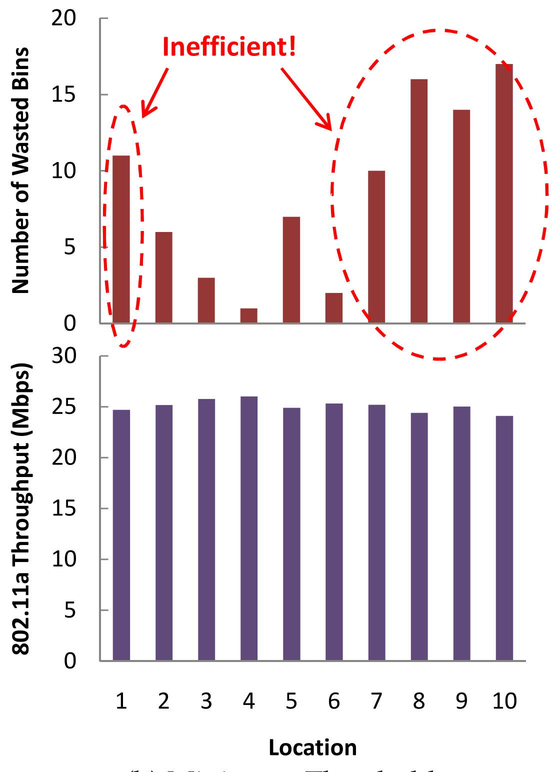
Results. Fig. 2-12 shows the difficulty in choosing a single threshold across locations: the ideal threshold varies by as much as $4.3\times$ in our testbed; furthermore, the thresholds do not correlate with distance, because of the reflection and shadowing typical in an indoor environment.

No Single Threshold is Both Safe and Efficient

In this section, we illustrate how a particular choice of threshold forces a compromise between safe narrowband operation and efficient wideband performance across locations.
Method. We use the same placement of wideband nodes as in Section 2.7.2. We consider two thresholds based on our experiments in Section 2.7.2 above, setting the threshold to either the median, or the minimum of those in Fig. 2-12. We then determine the set of bins that would be marked as usable for each threshold setting and location. We disable adaptive sensing in SWIFT, and at each location, manually set it to use the set of bins resulting from the chosen threshold, and measure the 802.11a throughput.
Results. Fig. 2-13 compares the number of wasted bins, *i.e.*, bins that the threshold unnecessarily marks as unusable by wideband, at each location, against the corresponding



(a) Median Threshold



(b) Minimum Threshold

Figure 2-13: No Threshold is Safe and Efficient in All Locations

802.11a throughput, for both the median and minimum thresholds from Fig. 2-12. The median threshold leads to a dramatic reduction in 802.11a throughput in locations 2, 3, and 6, while simultaneously producing over 10 wasted wideband bins in each of locations 8, 9, and 10. Bins are wasted in these locations because the 802.11a nodes, being too far, are no longer affected by wideband transmissions, but this threshold still causes many bins to be marked as unusable. Note that a threshold-based design can be both unsafe and inefficient in the same location. In particular, with the median threshold it is unsafe in locations 2 and 6, but also wastes a few bins in those same locations. This is because a blip in power in any bin outside of those occupied by the narrowband device causes that bin to be wasted.

A lower choice of threshold would increase the likelihood of safe narrowband operation at the cost of increased inefficiency. For example, using the minimum threshold among all measured locations ensures safe 802.11a operation in all of these locations, but almost doubles the bandwidth wastage. In our example, in addition to wasting bins in locations 7, 8, 9, and 10 where 802.11a is out of range, it also wastes bins in location 1. This wastage is because 802.11a transmissions leak significant power into bins adjacent to those it uses.

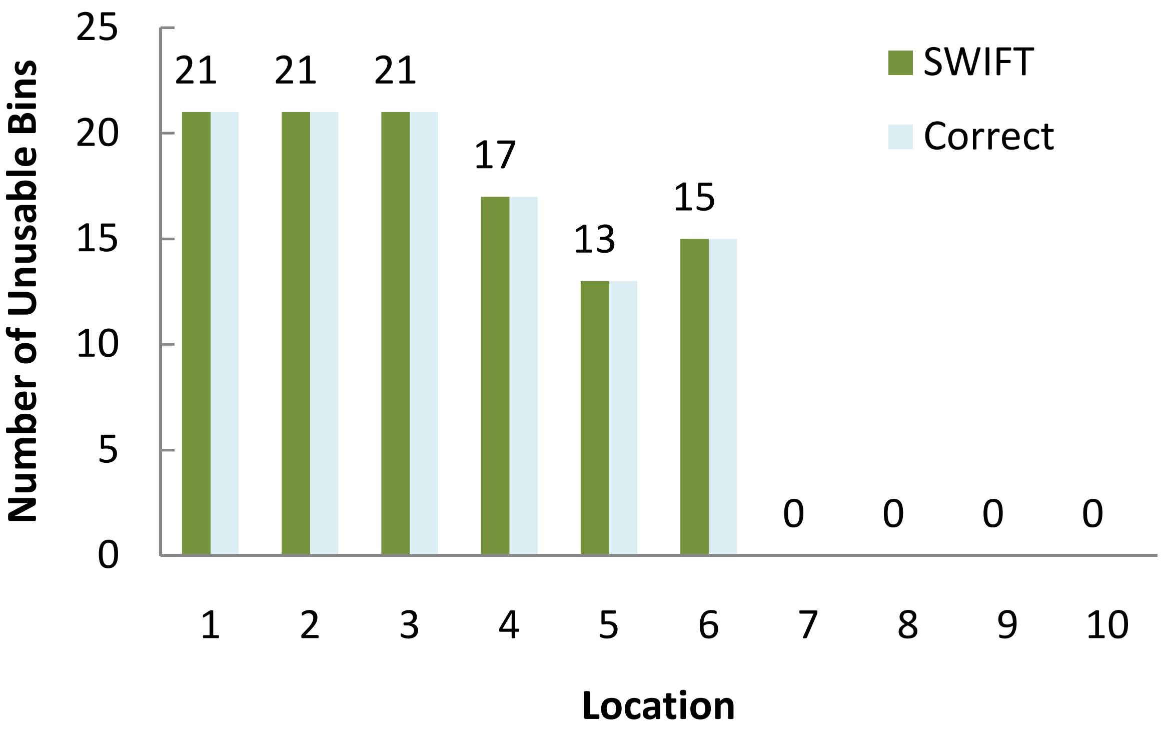


Figure 2-14: Adaptive Sensing is Robust: At each location, SWIFT finds the correct unusable bins, *i.e.* those that interfere with 802.11a.

Additionally, this minimum threshold may be unsafe for locations outside the measured set, or for a different 802.11a transmitter.

■ 2.7.3 Adaptive Sensing

In this section we show how the adaptive sensing algorithm allows SWIFT to use a maximal set of bins with almost no impact on 802.11a, and hence is both safe and efficient.

Method. The setup is similar to the previous experiment, except that the SWIFT nodes now have adaptive sensing turned on. We run one experiment at each location, by first starting the SWIFT node, and then starting the 802.11a transmission 5 seconds later. We record the `UsableBins` setting on which SWIFT settles, and compare it with the *correct* bin setting for each location as determined in Section 2.7.2.

Results. Fig. 2-14 shows that SWIFT finds the exact set of unusable bins, *i.e.*, bins that interfere with 802.11a, at all locations. Note further that SWIFT detects when 802.11a goes out of range, as in locations 7-10, and can reclaim all occupied bins.

Fig. 2-15 shows the typical dynamics of adaptive sensing, using results from an experiment with 802.11a at location 3. SWIFT conservatively backs away from bins used by

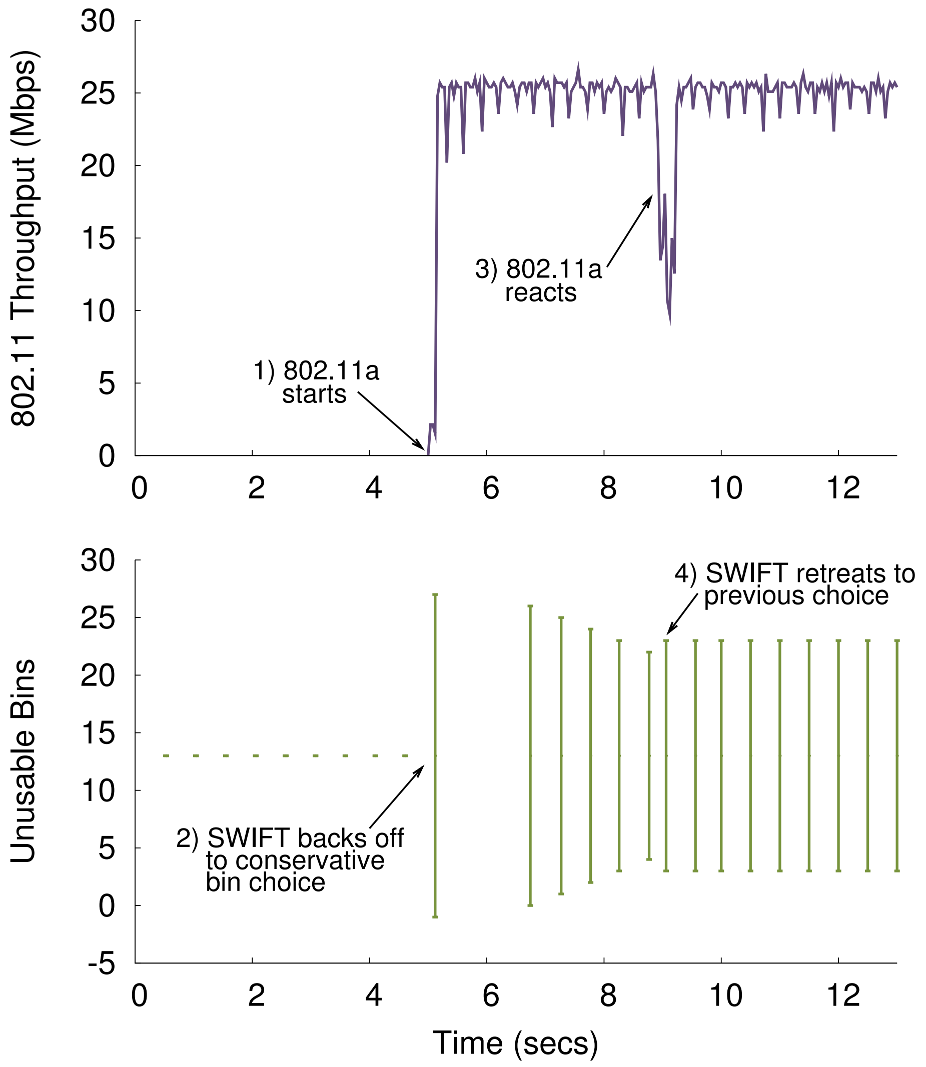


Figure 2-15: Responsiveness of Adaptive Sensing: The top graph shows that 802.11a throughput is not hindered for longer than 0.5 seconds by SWIFT. The bottom graph shows that, when 802.11a first appears, SWIFT backs off to a conservative bin choice within 120 ms, but quickly converges to a maximal set of safe bins.

802.11a within 120 ms of 802.11a commencing transmission. Additionally, within 4 seconds, it finds the ideal bin selection and then sticks with this selection. Over 60% of this time is a result of the communication overhead from our prototype PCI driver, and can be mostly eliminated with an optimized implementation.

Specifically, the bottom graph shows the SWIFT bin selections over time. SWIFT starts out using all bins, (1) until it first detects the 802.11a transmissions. (2) At this point, SWIFT immediately backs off using a conservative threshold, and avoids bins -2 through 28. As it gathers more data, and determines that 802.11a is unaffected, SWIFT decreases its set

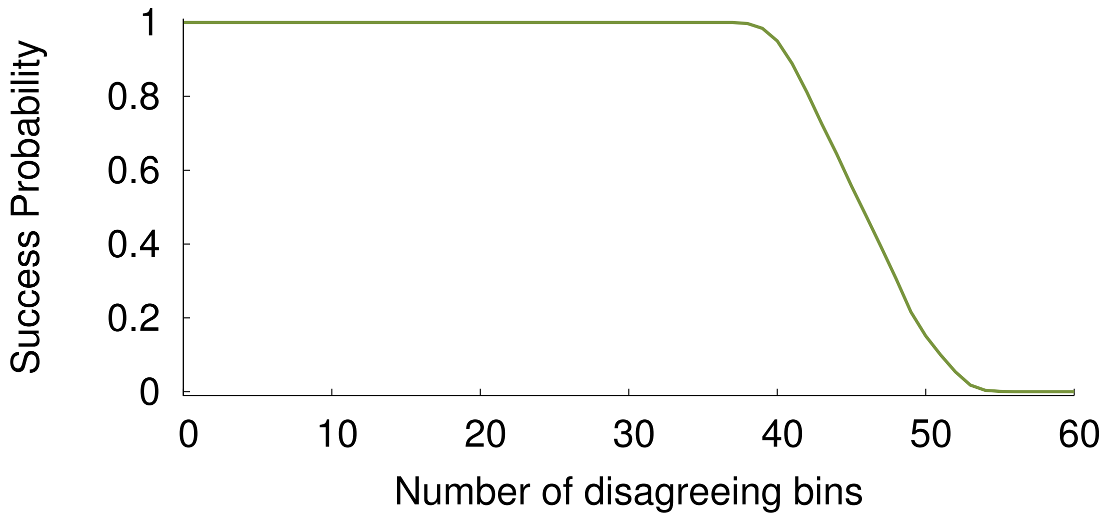


Figure 2-16: Robustness to disagreement: The figure shows the probability of a transmission succeeding as a function of the number of disagreeing bins. It shows that SWIFT is robust to as much as 40% disagreement between the set of transmitter and receiver bins.

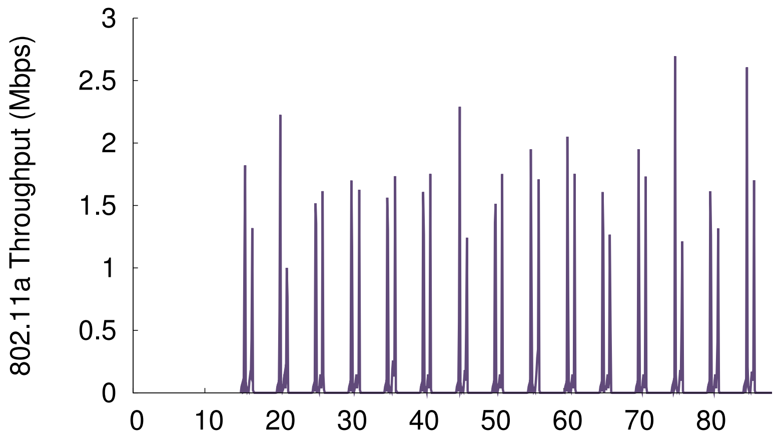
of unused bins gradually, till it begins avoiding only bins 4 through 22. (3) At this point, we see from the top graph that the throughput of 802.11a is affected for the first time. (4) SWIFT immediately relaxes its bin selection to avoid bins 3 through 23, and this returns the throughput of 802.11a to normal. As a result, SWIFT stabilizes at a state that avoids bins 3 through 23, which is the tightest bin selection that does not affect 802.11a.

2.7.4 Dealing with Bin Disagreement

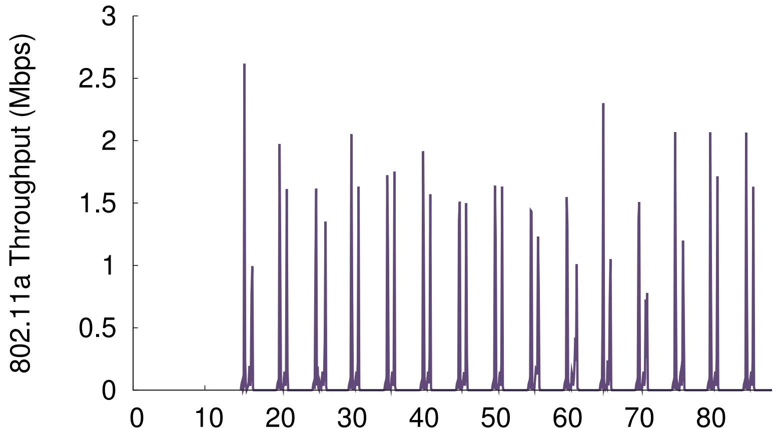
We evaluate the impact of disagreement between communicating pairs on SWIFT's band consensus protocol.

Method. We place the wideband transmitter and receiver within a few feet of each other so that they can communicate with each other with very low probability of channel bit errors. We do this to ensure that almost all bit errors are likely to be introduced purely due to disagreements. We initialize the transmitter and receiver to agree to use the entire wide band, consisting of 100 bins.

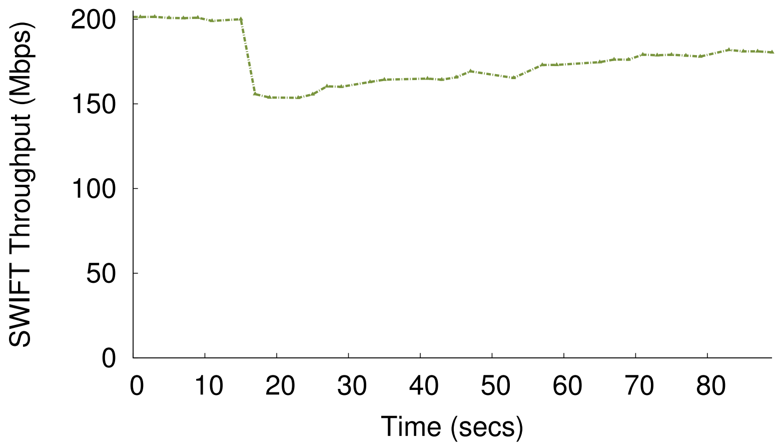
We then configure the adaptive sensing module to update the transmitter with a new set of usable bins with a sequence of K consecutive bins marked as narrowband-occupied, to simulate the appearance of a narrowband transmitter with a band of size K. Since the transmitter cannot use these bins whereas the receiver continues to expect data in them,



(a) 802.11a Throughput without SWIFT



(b) 802.11a Throughput with SWIFT



(c) SWIFT Throughput with TCP

Figure 2-17: SWIFT reaction to TCP web downloads: (a) and (b) show that, even in the face of intermittent 802.11a traffic, SWIFT avoids affecting 802.11a transmissions, while (c) shows that it does this while still achieving 90% of its original throughput.

the size of the disagreement between the nodes is K. We send a random coded sequence from transmitter to receiver using this disagreeing set of bins, check whether it is received correctly, and repeat this operation with a large number of random coded sequences for increasing values of K. We declare a transmission to have succeeded if it is decoded correctly, and compute the probability of a successful transmission for a disagreement of size K.

Results. Fig. 2-16 shows that SWIFT's band consensus works robustly for a large range of disagreements. When K is small, the consensus scheme sees a very small number of errors which can be easily corrected. As K grows, the receiver sees a burst of errors in the disagreeing bins, but the number of errors in any single code word is limited because transmitted data bits are interleaved across the frequency bins. This allows successful transmissions even when the fraction of disagreement is as large as 37% (37 of the 100 total bins). Such a large amount of disagreement is extremely unlikely, and hence SWIFT's low overhead handshake mechanism can almost always achieve band consensus. It is only when the extent of disagreement becomes large (56 bins in our case) that SWIFT nodes will need to reestablish connectivity using a sync packet.

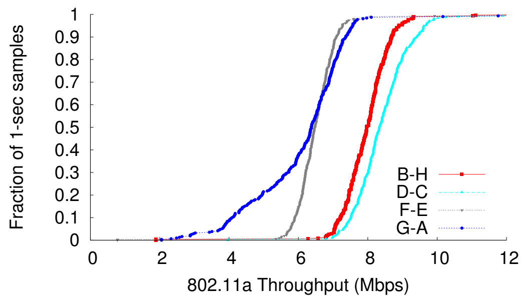
■ 2.7.5 Intermittent Narrowband TCP Web Downloads

This experiment evaluates SWIFT's ability to adapt correctly to intermittent and bursty traffic patterns.

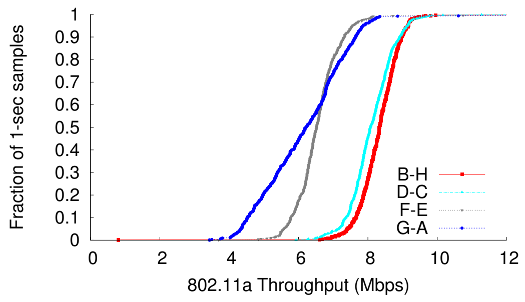
Method. We model a typical home scenario, using an 802.11a node that accesses the Internet by connecting to a Linksys wireless router. We first start the SWIFT node, and at time $t = 15$ seconds, the 802.11a node begins periodic web downloads. We download the home page from `www.apple.com` every 3 seconds. We average the throughputs of the TCP downloads and SWIFT over 100ms intervals, and plot them as a function of time.

Results. Fig. 2-17 shows that SWIFT adapts to intermittent and bursty web traffic, without causing any performance impact on the narrowband user. Notice that the narrowband traffic is indeed intermittent, and that the TCP downloads are too short for narrowband to achieve a peak throughput higher than 2-3 Mbps, despite the fact that the auto-rate algorithm is sustaining 48 or 54 Mbps in this case.

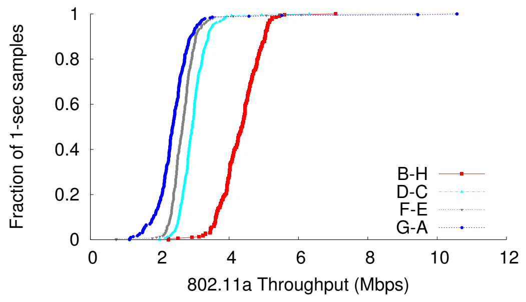
We see that SWIFT throughput drops as soon as the user begins her web download. This is because SWIFT falls back to a conservative set of bins. SWIFT throughput then



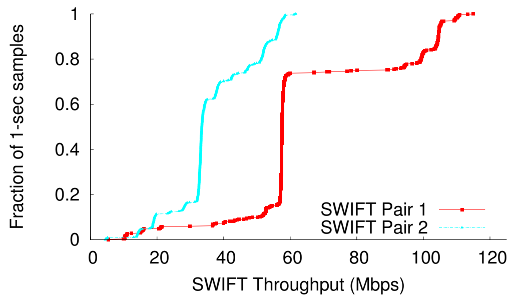
(a) 802.11a Throughput alone



(b) 802.11a Throughput with SWIFT



(c) 802.11a Throughput with NORM



(d) SWIFT concurrent with 802.11a in (b)

Figure 2-18: Throughputs in a Network: (a) and (b) show the throughputs of the four 802.11a pairs, with and without SWIFT. SWIFT has no impact on 802.11a, while, still getting good throughput as seen in (d). In contrast, (c) shows that non-adaptive wideband transmitters reduce 802.11a throughput by around 50%.

gradually increases as it tightens its set of bins. However, this process is slower than the example in Fig. 2-15 because SWIFT only uses measurements in the vicinity of a narrowband transmission, as described in Section 2.5.1. It therefore needs to wait for a longer time to acquire enough data points for each bin choice. SWIFT converges on the right set of bins, and its throughput stabilizes around $t = 75$ seconds. This throughput is lower than the throughput that SWIFT achieved prior to the web downloads because SWIFT is now avoiding bands that could affect 802.11a performance. Throughout this process, SWIFT remains safe to 802.11a and does not cause any noticeable impact on the TCP throughput.[4]

■ 2.7.6 Network Results

Here, we show that SWIFT performs well even in a chaotic environment with multiple 802.11a devices, and multiple SWIFT nodes.

Method. In this experiment we use four wideband nodes and eight 802.11a nodes, creating six pairs of communicating nodes. We place the four 802.11a pairs at locations A-H, and

[4]The differences in TCP throughput with and without SWIFT are caused by varying queue lengths in the wired Internet. In particular, note that the variations in downloads between the two graphs are no greater than the variations within any one graph.

the two wideband pairs at the locations labeled tx/rx and tx′/rx′ in Fig. 2-10. We then measure the throughputs when running the network without any wideband transmitters, with the wideband transmitters running NORM, and with the wideband transmitters running SWIFT.

Results. Fig. 2-18(c) shows that, when NORM transmits simultaneously with 802.11a, it significantly reduces 802.11a throughput. While the throughput reduction of 802.11a pairs at different locations is different, all pairs are impacted, with an overall average loss in throughput of around 50%.

Figs. 2-18(a) and (b) show the throughput of the four 802.11a pairs, with and without SWIFT. In this case, both pairs of SWIFT nodes move away from the bins occupied by the 802.11a nodes, allowing all 802.11a pairs to have essentially the same performance as in the absence of SWIFT. Additionally, Fig. 2-18(c) shows that by utilizing all bins not occupied by 802.11a, the SWIFT nodes are each still able to get reasonable throughputs of 30-100 Mbps in the face of 802.11a.

This result shows that SWIFT can deliver an operational wideband network, while ensuring that it does not affect multiple competing narrowband nodes.

■ 2.8 Discussion

This chapter addresses the problem of coexistence between emerging wideband networks and narrowband devices with which they share the unlicensed bands. We show that overly conservative designs that avoid interference by running below the noise floor needlessly sacrifice the throughput and the range of the wideband radios. In contrast, a design based on cognitive aggregation, which adapts its frequency bands and weaves together multiple non-contiguous bands into one wireless link, can be as narrowband-friendly as the conservative approaches, while achieving a significant increase in operating range and throughput.

Our results can be extended in multiple directions:

(a) Non-reactive narrowband devices: This chapter addresses narrowband technologies that react to interference in their band. Of course, not all devices react to interference. We envision that SWIFT can be extended to deal with such devices in one of two ways: either by being configured to avoid known non-reactive bands if they are present, or by

having adaptive sensing recognize a device as non-reactive if all narrowband bins can be reclaimed without any identifiable reaction. In this case, SWIFT can fall back to a conservative bin setting that avoids all bins with non-reactive narrowband power.

(b) Coexistence of multiple wideband protocols: SWIFT selectively avoids frequency bands used by narrowband devices, and shares the spectrum with other cooperating wideband devices using the SWIFT protocol. However, the future may bring a variety of wideband protocols. These systems need to find a way to share spectrum among different wideband technologies even when they do not use the same protocol.

(c) Dynamic Range: Like other techniques that allow a node to receive multiple concurrent signals [144], SWIFT's nodes deal with a wide range of signal powers and hence their performance improves with a wider dynamic range of the system.

CHAPTER 3

Frequency-Aware Rate Adaptation and MAC Protocols

There has been burgeoning interest in wireless technologies that can use wider frequency spectrum. This chapter presents FARA, a cooperative wideband wireless system that can dynamically share spectrum across wireless nodes in a performance-aware manner.

■ 3.1 Overview

Wireless technologies are pushing toward wider frequency bands than the 20 MHz channels employed by existing 802.11 networks. 802.11n already includes a 40 MHz mode that bonds together two 20 MHz bands [68]. Emerging ultra-wideband (UWB) technologies employ hundreds of MHz to support multimedia homes and offices [69, 152, 31, 121]. The FCC has recently permitted unlicensed use of digital TV whitespaces that occupy 100-250 MHz of spectrum vacated by television bands in the analog-to-digital transition [42]. Furthermore, recent empirical studies show that the 802.11 channelization model which limits each node to a single 20 MHz channel can lead to severe load imbalance [58, 77, 108]. They advocate discarding channelization and allowing all nodes to access the entire 802.11 spectrum based on demand [58, 108]. This push towards wider bands is further enabled by the constantly lowering prices of high-speed ADC and DAC hardware [114, 85].[1] In particular,

[1]The wider the band, the faster the ADC and DAC have to sample the signal.

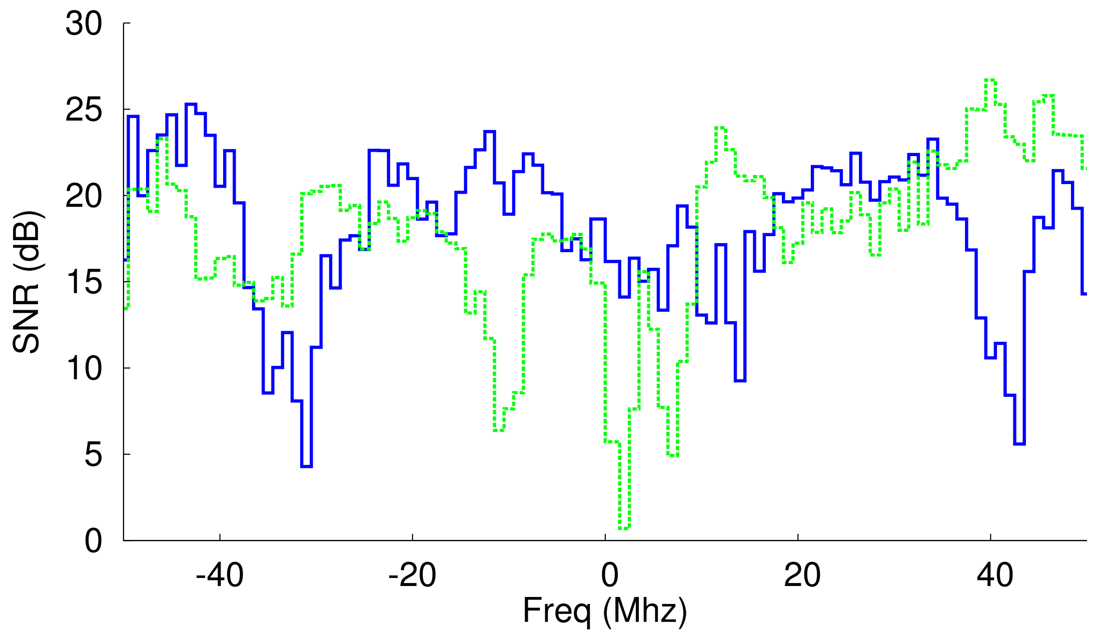


Figure 3-1: Frequency diversity across 100 MHz of 802.11a spectrum as observed by two receivers for transmissions from the same sender. The figure shows that the SNRs of different frequencies can differ by as much as 20 dB on a single link. Further, different receivers prefer different frequencies.

today, wireless cards that span over 100 MHz of spectrum can be built using off-the-shelf hardware components [98].

As wireless networks push towards wider bands, we can no longer afford to ignore frequency diversity. Specifically, multipath effects cause frequencies that are far away from each other in the spectrum to experience independent fading. Thus, different frequencies can exhibit very different SNRs for a single sender-receiver pair. Further, the frequencies that show good performance for one sender-receiver pair may be very different than the frequencies that show good performance for another pair. Fig. 3-1 shows empirical measurements of the SNRs across 100 MHz of the 802.11a spectrum, as observed by 2 clients for transmissions from the same AP (see Section 3.9 for experimental setup). The figure reveals that different frequencies show a difference in SNR of over 20 dB both for a single link and across links. Existing bitrate adaptation and MAC protocols however are frequency-oblivious. They assign the same bitrate to all frequencies and allocate the medium in a time-based manner, ignoring the fact that different frequencies work better for different sender-receiver pairs. Thus, current rate adaptation and MAC protocols can neither deal with the challenge nor exploit the opportunities introduced by the frequency diversity of wide bands or unchannelized 802.11.

This chapter presents the design and implementation of FARA, a frequency-aware wireless architecture. FARA is beneficial for both wireless LANs and mesh networks. Its design focuses on the 802.11a/b/g/n spectrum, but it can also be used in a cognitive mode over white spaces as discussed in Section 3.10.

FARA has four key components that together allow it to improve network throughput and balance load.

- *Per-frequency SNR estimation:* FARA leverages the existing OFDM system, which divides the entire frequency band into many narrow subbands. It devises a new approach to allow a receiver to use normal data packets, whether received correctly or incorrectly, to robustly estimate the SNR in each OFDM subband.

- *Frequency-aware rate adaptation protocol:* FARA uses its per-frequency SNR measurements to enable a transmitter to use different bitrates across different OFDM subbands. Specifically, a FARA receiver measures the SNR in each subband, maps it into an optimal bitrate using characterization tables for the receiver hardware,[2] and periodically reports this optimal bitrate for each subband to the transmitter.

- *Frequency-aware MAC:* A FARA transmitter acquires the medium using carrier-sense. However, once the medium is acquired, a transmitter that has traffic for multiple receivers can simultaneously transmit to all these receivers by preferentially allocating frequencies to receivers to maximize the overall throughput across these receivers.

- *Load-aware contention:* In contrast to existing channelized 802.11 networks, in FARA the entire frequency spectrum is available to all nodes without channelization. As a result, load balancing can be done using a small modification to CSMA where an AP or a router in a mesh network contends for the medium proportionally to its load.

We implemented FARA using the WiGLAN radio platform [98], and compared it to the current frequency-oblivious 802.11. Measurements from our indoor testbed reveal the following findings:

[2]These characterization tables need to be calibrated only once for a particular receiver hardware.

- Frequency diversity exists, and is stable over time. Specifically, our results show that different subbands in the 802.11a spectrum can show a difference in SNR of over 20 dB. Further, the SNR profile for individual subbands is relatively stable for periods up to 5 seconds, and hence can be communicated from receiver to transmitter with low overhead.

- FARA is effective at harnessing frequency diversity, delivering a median throughput gain of $3.1\times$ in our testbed.

- FARA's gains come both from exploiting frequency diversity within a single sender-receiver pair (frequency-aware rate adaptation), as well as across sender-receiver pairs (frequency-aware MAC). Typically, for our experimental scenarios, about 70% of the gains are due to frequency-aware rate adaptation, and 30% are due to the frequency-aware MAC.

- FARA's load-aware contention protocol is fair even when APs have a wide load disparity.

To the best of our knowledge, FARA is the first system to present frequency-aware rate adaptation and MAC protocols, and show through a prototype implementation and experimental evaluation, that frequency-awareness can improve the throughput of an 802.11 network.

■ 3.2 Related Work

Related work falls in the following areas.

(a) Measurement and Analysis of Frequency Diversity. The impact of multipath effects in creating varying signal strength across frequencies is well understood theoretically [144]. Also, multiple measurement studies [99, 134, 30, 80, 17] have demonstrated the existence of this frequency diversity in practice, showing that it occurs both at the low end of the RF spectrum, as in white spaces [15, 117], as well as the high end of the spectrum, as in 60 GHz technologies [99, 134]. Among these, the most relevant to our work are measurement studies in the 2.4 and 5.2 GHz spectrum [30, 80, 17] (corresponding to 802.11b/g and 802.11a, respectively), which report a difference in signal strength of as much as 20 dB between frequency bands both in line-of-sight (LOS) and non line-of-sight (NLOS) scenarios. Our work is motivated by these results, but differs from them significantly because it presents a rate-adaptation and MAC that can leverage frequency diversity to improve network throughput.

(b) Subband Adaptive Modulation and Coding. Wired cable modem standards, such as Asymmetric Digital Subscriber Line (ADSL), High bit-rate DSL (HDSL), and Very high-speed DSL (VDSL) [136] adapt modulation and coding independently across OFDM subbands. Some prior theoretical or simulation studies have proposed a similar approach for wireless channels [76, 44, 29, 65, 83, 161, 86, 88, 95, 39], and a few papers [156, 147] have investigated the complexity of hardware implementations of such designs. Our work builds on this foundation but differs in two ways. First, FARA presents a rate adaptation algorithm that works in real-time and supports its design with empirical evaluation. Second, FARA augments its frequency-aware rate adaptation with a frequency-aware MAC, while all these prior studies focus only on a single link, and none of them exploits frequency diversity across multiple users.

(c) Opportunistic Communication Schemes. Prior work on opportunistic communication considers a scenario where two nodes that can hop between frequencies try and identify the best channel on which to communicate [127, 57]. This work proposes schemes to minimize exploration overhead while maximizing the probability of finding a high-performance frequency band. While FARA is similar in that it exploits frequency diversity, it differs from these schemes in both objective and mechanisms. First, these schemes focus on finding and using a small set of good frequencies assuming that they all have similar SNRs, while FARA allows a sender-receiver pair to operate over a wide set of frequencies that may differ drastically in SNR. To achieve this goal, FARA provides a rate adaptation scheme that uses different bitrates on frequencies with different SNRs. It also extends the 802.11 protocol to allow a transmitter to transmit simultaneously to multiple receivers, taking advantage of frequency diversity across them. Furthermore, FARA is implemented and evaluated in a wireless testbed, while prior work is simulation-based.

(d) Non-channelized 802.11 Protocols. Recent work has advocated using the 802.11 spectrum as a whole, and discarding the traditional fixed-width channel model [58, 108]. Specifically, ODS [58] allows all nodes to simultaneously access the entire 802.11 spectrum, and share it using code division multiple access (CDMA), while Moscibroda *et. al.* [108] dynamically assign non-overlapping frequencies to different APs proportional to their load. Similarly to these schemes, FARA allows all nodes access to the entire 802.11 spectrum based on their demands, and hence can provide load balancing, but, in contrast to the frequency-obliviousness of the prior work, FARA can exploit frequency diversity both for rate adaptation, and medium access, hence providing additional gains even when loads are balanced.

(e) FDMA Cellular Networks. Multiple cellular technologies such as Flash OFDM, GSM and WiMax [144, 11, 124] use frequency division multiplexing for medium access. Flash OFDM and GSM use pseudorandom frequency hopping, rather than assigning to each user those frequencies that work best for the user at that instant. WiMax on the other hand has two modes. For mobile and fast-changing channels, WiMax randomly assigns frequencies to users, rather than assigning the best instantaneous frequencies to each user. For static or moderately dynamic environments, WiMax allocates to each user a chunk of contiguous frequencies that work best for that user. FARA is designed for static or moderately dynamic networks and is similar to this latter mode of WiMax in that it assigns to each sender-receiver pair the frequencies that work best for that pair. FARA however differs from WiMax in two ways: First it does not limit a user to a contiguous chunk of frequencies, and allows a sender-receiver pair to use non-contiguous frequencies. Second, in contrast to WiMax, which uses the same bit rate across the whole chunk allocated to a user, FARA adapts the bit rate independently across different frequencies used by a sender-receiver pair.

■ 3.3 Frequency Diversity

Frequency diversity is an intrinsic characteristic of RF propagation in multipath environments [144]. The wireless channel both attenuates the RF signal and changes its phase. Specifically, the channel shifts the signal's phase by $2\pi f \tau$, where f is the signal's frequency and τ is the path delay. In environments with multipath effects, the receiver ends up with multiple copies of the signal that traversed different paths with different delays, as shown in Fig. 3-2. These copies have different phases and hence may add up constructively or destructively. Since the phase of the received signal is a linear function of its frequency,

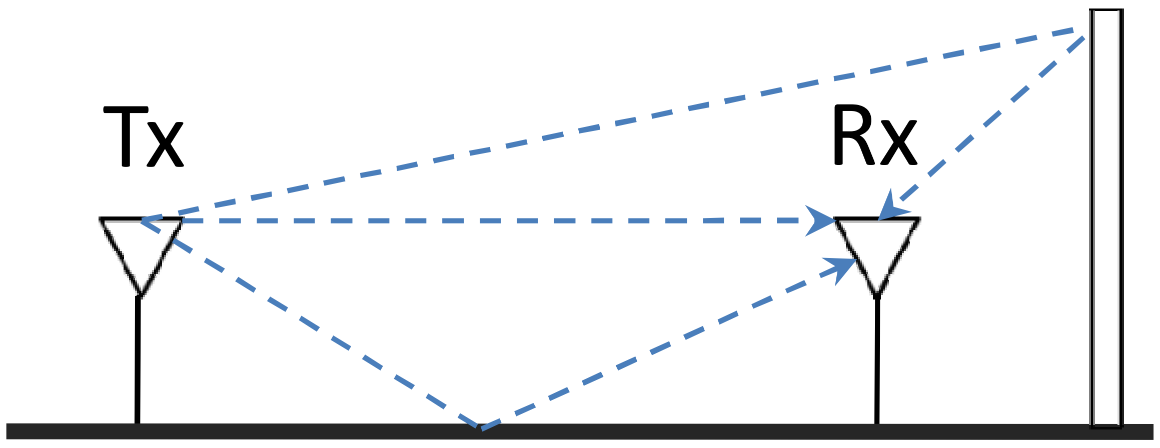


Figure 3-2: Multipath Effect Causes Frequency Diversity. Signals from different paths combine at the receiver constructively or destructively depending on their phases. Since the phase is a linear function of the frequency, the destructive and constructive patterns differ across frequency bands causing different frequencies to have different SNRs.

different frequencies show different degrees of constructive and destructive signal patterns.

The effect of this frequency diversity is significant when examined across a wide spectrum, such as the entire 300 MHz of spectrum usable by 802.11a, or the 80 MHz usable by 802.11g. Past measurements show that different frequency bands within the wide 802.11a/b/g spectrum can differ by as much as 20 dB of SNR [17, 30, 80]. These results align with our own measurements shown in Fig. 3-1. The figure shows the SNRs of two 100-MHz channels in the range of 802.11a, for subbands of one MHz wide. The measurements are taken for two links in our testbed (from transmitter `tx` to receivers `A2` and `B3` in Fig. 3-6). The figure reveals that the SNR difference between frequencies is significant. Furthermore the SNR pattern is highly diverse both for a single link and across the two links.

Frequency diversity motivates bit rate adaptation schemes and MAC protocols that can leverage SNR differences across frequencies to increase network throughput.

■ 3.4 FARA

FARA is a new architecture for static and moderately dynamic wireless networks, i.e., typical 802.11 environments. Similarly to recent proposals for channel bonding [68] and load balancing [58, 108], FARA advocates discarding the current channel notion and allowing all nodes to access a larger chunk of the 802.11 spectrum. FARA however recognizes that a wider spectrum increases frequency diversity. Its design harnesses frequency diversity via four components: per-frequency SNR estimation algorithm, frequency-aware rate adaptation, frequency-aware MAC protocol, and load-aware contention. Together these components significantly increase network throughput and balance the utilization of the 802.11 spectrum. In the following sections, we explain each of these components in detail.

3.5 Per-Subband SNR Estimation

FARA introduces a novel algorithm that allows a sender-receiver pair to estimate the performance of each frequency, *i.e.*, its SNR, using normal data packets, whether received correctly or incorrectly. To do so, FARA leverages Orthogonal Frequency Division Multiplexing (OFDM), which is already implemented as part of the 802.11a/g/n physical layer [13, 56]. OFDM divides the used frequency spectrum into many narrow subbands. A subset of these subbands are called pilots and used to transmit a known bit pattern modulated at BPSK to allow the receiver to track the channel [63]. The other subbands are used for data transmission. A FARA receiver estimates the SNR for each OFDM data subband, for each sender, by estimating the signal power from all of the transmitted data, and leveraging the known bit pattern in the pilot bins to estimate noise.

In particular, the SNR in subband i, SNR_i is the ratio of the signal power in subband i, S_i, to the noise power, N_i. The receiver cannot directly measure the signal power; however, it can measure the received power in subband i, R_i, which is the sum of the signal power, S_i and the noise power, N_i. Thus,

$$SNR_i = \frac{S_i}{N_i} = \frac{R_i - N_i}{N_i}$$

$$SNR_i = \frac{R_i}{N_i} - 1 \tag{3.1}$$

Note that the noise in a communication channel is typically the same for all subbands, *i.e.*, white noise.[3] This is because noise comes from thermal noise in the receiver hardware,

[3]While channel noise is typically white, interference due to other technologies, say Zigbee, can differ across subbands. In this dissertation, we deal with interference the same way 802.11 does, *i.e.*, via carrier sense. FARA however can leverage our previous work on SWIFT [121] to identify subbands occupied by other technologies and avoid them.

quantization, and digital computation errors, which are all independent of frequency. Thus, we can rewrite Eq. 3.1 as

$$SNR_i = \frac{R_i}{N_0} - 1, \text{where } N_0 = N_i, \forall i \tag{3.2}$$

The received power, R_i, in a particular subband can be easily estimated by taking the square of the signal corresponding to that subband, and averaging this value across all data symbols in a packet.

We can get an accurate estimate of the noise power, N_0, by exploiting the fact that OFDM uses some subbands as pilots, which contain known data bits. Specifically, the received signal sample, $y_i[k]$, in subband i can be written as:

$$y_i[k] = H_i x_i[k] + n_i[k] \tag{3.3}$$

where H_i is the channel, $x_i[k]$ is the kth transmitted signal sample in subband i, and $n_i[k]$ is the corresponding noise sample. The receiver knows H_i for all subbands because it is estimated using known OFDM symbols in the preamble [63]. In the case of a pilot subband, $x_i[k]$ is also known at the receiver since pilot subbands contain a known data sequence. As a result, the receiver can estimate the noise samples, $n_i[k]$, and the noise power, N_0, as:

$$n_i[k] = y_i[k] - H_i x_i[k] \tag{3.4}$$

$$N_0 = E_{i,k}(n_i[k]^2) \tag{3.5}$$

where the function $E(.)$ is the mean computed using all pilot bits across all symbols in the data packet.

Thus, every received packet allows the receiver to obtain a new SNR measurement for each OFDM subband. The receiver maintains a time weighted moving average of the SNR in each subband, which it updates on the reception of a data packet.

A few points are worth noting:

(a) What happens when the data packet is corrupted (i.e. does not pass the checksum test)? Even when the packet is corrupted, the receiver can still compute an accurate estimate of the per-subband SNRs. This is because the receiver can compute the average received power, regardless of whether the packet is corrupted or not. Furthermore, the receiver can still

obtain an accurate estimate of the noise power since this only requires the pilots which are known, and sent at BPSK, which is the most robust modulation rate and hence allow synchronization and packet recovery even at low SNRs. Thus, FARA can get accurate estimates of the per-subband SNRs from every captured packet, including corrupted packets.

(b) How accurate are FARA's SNR estimates? We note that since FARA has access to the PHY layer, it can collect accurate SNR estimates. In particular, traditional estimates of the SNR use RSSI readings, which measure the received power of a few samples at the beginning of the packet (i.e., the AGC gain) [24], or infer the SNR using just the correlation of header symbols in the preamble of the packet [150]. In contrast, FARA exploits the known pilot bits to accurately estimate the noise power and utilize it in its SNR computation. Furthermore, FARA computes its signal and noise estimates over the whole packet and not just a few samples at the beginning of the packet, which allows it to obtain more stable estimates.

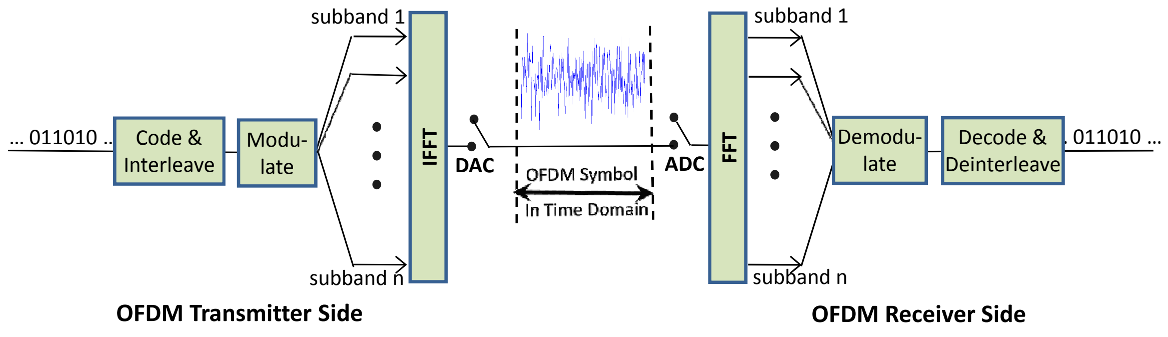
(c) Do different choices of bitrate affect the accuracy of FARA's SNR estimation? OFDM data subbands use a different modulation scheme depending on the choice of bitrate. The modulation scheme in a subband, however, does not affect our per-subband SNR estimate. The estimation of SNR involves only the measured power in each subband and hence can be performed on any packet independent of the modulation and coding schemes used by the transmitter.

■ 3.6 Frequency-Aware Rate Adaptation

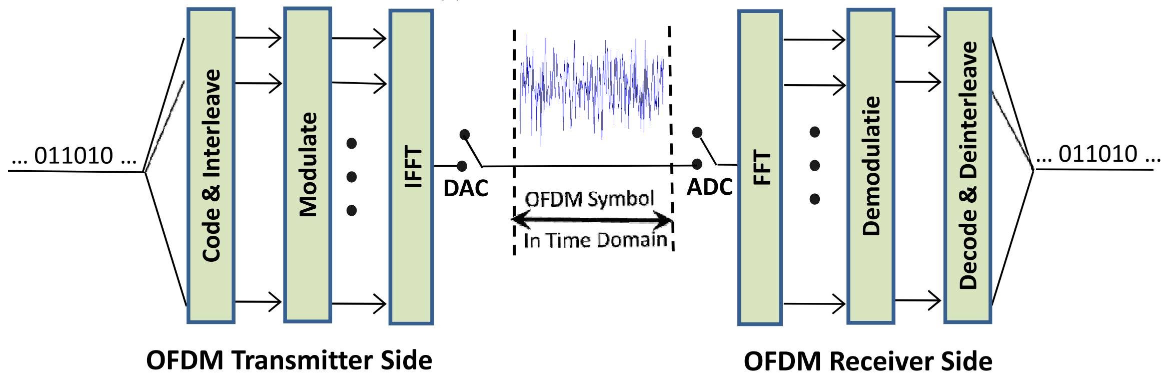
The goal of rate adaptation is to determine the highest bitrate that a channel can sustain at any point in time. Traditional 802.11 rate adaptation schemes are frequency-oblivious, and use the same modulation scheme and coding rate across all frequencies. Thus, they cannot exploit the frequency diversity present across the 802.11 spectrum. In contrast, FARA exploits this frequency diversity via a frequency-aware rate adaptation scheme that picks different bitrates for different frequencies depending on their SNRs.

■ 3.6.1 PHY Architecture

In 802.11, a particular bit rate implies a single modulation scheme and code rate over all OFDM subbands in the entire packet. For example, a bitrate of 24 Mbps corresponds to 16-QAM modulation scheme and a half-rate code. 802.11 has 4 possible modulation schemes



(a) Schematic of 802.11 PHY



(b) Schematic of FARA-enabled 802.11 PHY

Figure 3-3: OFDM PHY semantics with and without FARA. In FARA-enabled devices, the choice of modulation and FEC code rate is done independently for each OFDM subband.

(BPSK, 4-QAM, 16-QAM, and 64-QAM), and 3 possible code rates (1/2, 2/3, and 3/4). In current 802.11, a transmitter implements a particular bitrate by first taking the input bit stream, passing it to the convolutional coder, and puncturing to achieve the desired coding rate. The bits are then interleaved, modulated and striped over the OFDM subbands, as shown in Fig. 3-3(a). The process is reversed on the receiver as shown in the figure.

FARA makes a few modifications to the existing 802.11 PHY layer, as shown in Fig. 3-3(b). Specifically, FARA employs the same set of modulation schemes and code rates supported by the existing 802.11. However, it allows each OFDM subband to pick a modulation scheme and a code rate that match its SNR, independently from the other subbands. Note that this design does not require additional modulation/demodulation or coding/decoding modules in the PHY layer. In particular, since we use standard 802.11 modulation and coding options, we only need to buffer the samples and process them through the same pipeline.

Minimum Required SNR	Modulation	Coding
<3.5 dB	Suppress subband	
3.5 dB	BPSK	1/2
5.0 dB	BPSK	3/4
5.5 dB	4-QAM	1/2
8.5 dB	4-QAM	3/4
12.0 dB	16-QAM	1/2
15.5 dB	16-QAM	3/4
20.0 dB	64-QAM	2/3
21.0 dB	64-QAM	3/4

Table 3-1: Minimum required SNR for a particular modulation and code rate (i.e., bitrate). Table is generated offline using the WiGLAN radio platform by running all possible bit rates for the whole operational SNR range. The SNR field refers to the minimum SNR required to maintain the packet loss rate below 1% (see Section 3.9 for experimental setup).

■ 3.6.2 Mapping Subband SNRs to Optimal Bitrates

The receiver needs to map the average SNR in each subband to the optimal bitrate for that band. To do so, the receiver uses an SNR characterization table like the one in Table 3-1 that lists the minimum SNR required for a particular combination of modulation and coding rate, i.e., a particular bitrate. For each subband, the receiver picks the highest bitrate that can be sustained by the SNR of that subband. Subbands which have SNR too low to support even the lowest bitrate in Table 3-1 are not used *i.e.*, they do not have any power or data assigned to them, as such a decision will improve the overall throughput. Said differently, subband suppression is simply a special case of FARA's ability to use different modulation and coding rates for subbands based on their SNR.

Many hardware manufacturers already perform this calibration and can provide it as part of hardware specification sheets [38]. Even when the manufacturer does not provide the SNR characterization table, it can be computed using brute force by varying the transmission power and bitrates, and measuring the observed throughput and SNR [75]. We show in the results section that the table does not change with location or time, and thus the measurements can be done only once for each receiver.

■ 3.6.3 Rate Adaptation Protocol

FARA's rate adaptation is receiver driven: a FARA receiver computes the optimal choice of bitrate on each subband, and feeds it back to the sender in ack packets. Specifically, FARA extends the 802.11 synchronous ack format with a field for bitrate feedback.

When a sender first initiates communication with a receiver, it makes a conservative choice and uses the lowest bitrate on all subbands. The receiver uses this to obtain its first estimate of the SNRs, and hence, the bitrate, in each subband. In order to allow the sender to quickly jump to the correct bitrate, the receiver then sends the appropriate bitrate for each subband immediately in the ack response.

After this initialization, receiver feedback is sent in 802.11 synchronous acks, which we augment with a feedback field. FARA reduces the feedback overhead by exploiting the fact that bitrates typically do not change from one packet to the next, and even when they do, are likely only to change to neighboring bitrates on either side (*i.e.,* jump up or down by one bitrate) As a result, the subsequent ack packets only need to use a 2-bit field per subband to represent one of three choices: stay at the current bitrate, move up to the next highest bitrate, move down to the next lower bitrate. Further, since most of these field values are likely to represent staying at the current bitrate, the feedback information can be compressed drastically using run-length encoding, which is easy to implement in hardware.

We note that sending variable-length synchronous acks does not affect 802.11 behavior. An 802.11 network only requires acks to start within a fixed (SIFS) interval; variable-length acks will not affect network function as the next packet transmission will not start until the medium becomes idle.

What happens when packets or acks get lost? Since FARA's acks are incremental, loss of these acks could lead a sender and receiver to go out of sync. To address this problem, the receiver includes a sequence number with each ack, and stores the bitrate state that the sender would compute as a result of receiving that ack. The sender, in turn, includes the sequence number of the last ack that it has received in its data packets. Thus, when the receiver gets a data packet, it can look up the included ack sequence number in its stored state, and thereby infer the sender's bitrate state. It can then compute the incremental feedback to be included in the new ack with reference to that state. Note that, since FARA's acks are synchronous, the receiver only needs to store the state corresponding to the most recent ack sequence number received from the sender. It therefore needs only a small amount of state to track the sender's bitrate. The receiver also includes the original ack sequence number with respect to which the increment is computed, so that the sender can update its state correctly upon reception of the new ack.

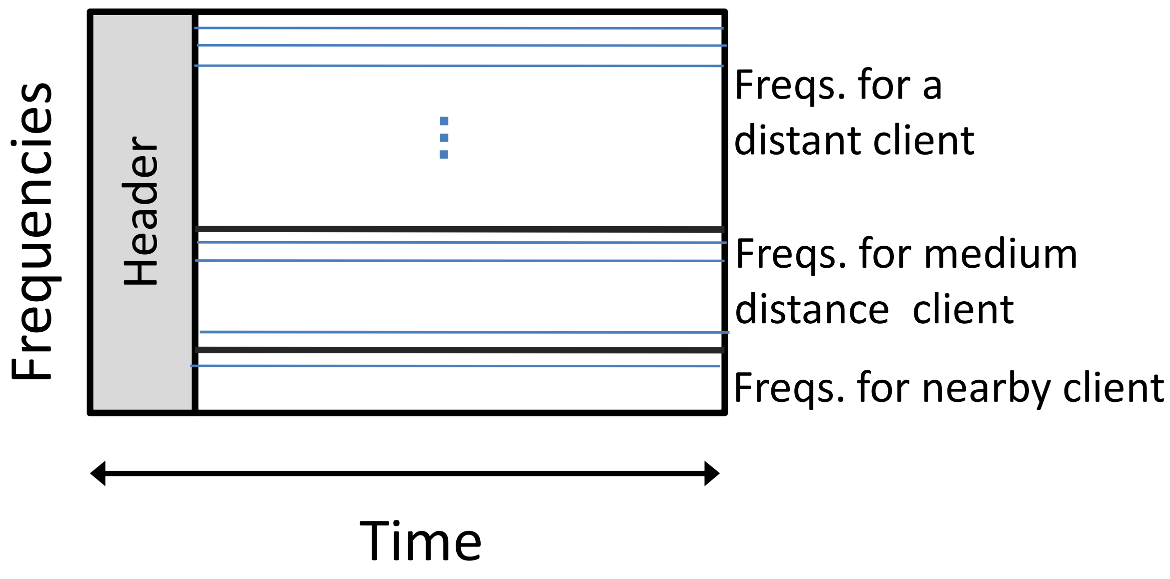

Frequencies
Header
Freqs. for a distant client
Freqs. for medium distance client
Freqs. for nearby client
Time

Figure 3-4: Simplified FARA frame with three concurrent packets. A FARA sender transmits concurrently to multiple receivers, by allocating to each receiver a subset of the OFDM subbands. To ensure fairness, *i.e.*, to ensure that all clients obtain the same average rate, distant receivers that experience lower per-frequency SNRs are allocated more frequencies. Note that the frequencies used for one receiver are not necessarily consecutive. They are made consecutive in the figure to simplify the drawing. The frame header includes metadata about the intended nexthops as well as their allocated subbands.

■ 3.7 Frequency-Aware MAC

Similar to 802.11, a FARA sender uses carrier-sense to access the medium. However, different from 802.11, when it accesses the medium, it transmits simultaneously to multiple nexthops, assigning each of them a non-overlapping set of OFDM subbands. The choice of concurrent nexthops, as well as the set of subbands assigned to them, aims to maximize throughput.

Determining the optimal assignment of subbands to concurrent receivers is a difficult problem. To see why, say that a FARA sender wants to deliver 3 packets to 3 nexthops in a single transmission. Fig. 3-4 shows the transmitted frame across time and frequency. As can be seen, each frame contains multiple concurrent packets intended for different nexthops. The rate for a particular nexthop is the sum of the rates of all the subbands assigned to that nexthop. The transmission time for a packet to that nexthop is therefore the packet size divided by the rate to that nexthop. Since all 3 packets in the figure are being transmitted concurrently, we would like to minimize wastage of medium time by equalizing the transmission time of the 3 packets. For equal packet sizes, this implies that the total rate assigned to the different nexthops are equal. If packet sizes are not equal, the rates need to be proportional to the packet size. This problem is NP-hard as can be demonstrated by a trivial reduction to the bin-packing problem [48]. Hence we seek a heuristic solution.

Assigning subbands to concurrent receivers: FARA's MAC protocol works as follows. A sender is configured with a maximum allowed number, N, of concurrent packets in a transmission. In practice, N is a small number between 2-5. FARA maintains per-nexthop packet queues, as well as a global FIFO transmission queue which contains pointers to packets in the per-nexthop queues. For each transmitted frame, the sender picks upto N concurrent packets. It first picks the packet at the head of the global FIFO transmission queue, and determines the associated nexthop. It then randomly chooses upto $N - 1$ other nexthops with non-empty queues, and picks the packets at the head of these queues. These packets will be transmitted concurrently in one frame. The random choice of nexthops ensures that FARA is fair to all nexthops while providing significant throughput gains, as we show in Section 3.9.4.

The FARA sender now needs to assign subbands to each packet as to equalize the transmission rate to all N receivers. The sender also wants to assign to each receiver its preferred frequencies, *i.e.*, the frequencies that achieve high SNRs for that receiver.

We use a randomized greedy approach for the subband assignment problem. The algorithm maintains two data structures:

- `SubbandAssignment`: Stores the current nexthop assignment for each subband.
- `RateCounter`: Stores the total rate currently assigned to each nexthop.

The algorithm first orders the N nexthops randomly. It initially assigns all subbands to nexthop 1. The `RateCounter` for that nexthop is assigned the sum of the rates that it would have obtained from all these subbands, and all other `RateCounter` values are set to 0.

At each step, we pick the nexthop $nmin$ with the smallest `RateCounter` value, breaking ties randomly. We now need to assign an additional subband to this nexthop so that it can achieve a higher rate. To do this, we pick the nexthop $nmax$ with the largest `RateCounter` value. For each subband i assigned to $nmax$, we compute $\Delta rate[i] = rate_{nmin}[i] - rate_{nmax}[i]$, where $rate_{nmin}[i]$ and $rate_{nmax}[i]$ are the rates that nexthops $nmin$ and $nmax$ would obtain from subband i respectively. We then change the

`SubbandAssignment` for the subband with the largest $\Delta rate$ from $nmax$ to $nmin$, and update the corresponding `RateCounter` values accordingly.

We keep repeating this process and stop when we cannot increase the minimum rate, which means that the receivers have as close a rate to each other as possible.

The algorithm above aims to allocate to each nexthop the frequencies that work better for it than for other nexthops. This is achieved by assigning subbands to a receiver according to the decreasing order of $\Delta rate[i]$. It also aims for equal rates to all concurrent nexthops. This is achieved by moving subbands to the nexthop that has the minimum rate so far, and repeating until we can no longer increase the minimum rate.

The header of each transmitted frame includes the number of nexthops and their addresses, as well as a bitmap with the frequency assignment. This allows each nexthop to learn the frequency subbands used for its packet. In contrast to traditional 802.11 where each data packet is followed by one synchronous ACK, a data frame that encapsulates N packets is followed by N synchronous acks from the corresponding nexthops. The acking order is determined by the order of the nexthops in the header of the data frame, and the acks are separated by a SIFS.

3.7.1 Wireless LANs vs. Mesh Networks

FARA can be used both in wireless LANs and mesh networks. Further, our description of the protocol directly applies to both. We note, however, that the benefits of applying FARA differ between these two scenarios. Specifically, in a mesh network, any node typically has multiple neighbors which constitute its potential nexthops. Hence, a mesh sender can derive gains from both FARA's frequency-aware rate adaptation and MAC protocols.

In contrast, a wireless LAN has two types of nodes, APs and clients. Since an AP is associated with many clients, the downlink, which carries the bulk of the traffic, can benefit from both frequency-aware rate adaptation and MAC protocols. On the uplink, however, the client is associated with a single AP, and hence has only one potential nexthop. While a client does not benefit from a frequency-aware MAC, it can still benefit from a frequency-aware rate adaptation protocol.[4]

[4]One extension to FARA would be to allow concurrent senders, in addition to concurrent receivers. A fine-grained allocation of OFDM subbands to concurrent senders, however, would require the senders to be synchronized to within an OFDM symbol to avoid power leakage between subbands. We therefore leave this for future work.

3.8 Load-Aware Contention

Since all senders in a FARA network have dynamic access to the entire frequency band, FARA naturally eliminates the problem of underutilizing the frequency spectrum due to inefficient 802.11 channel allocation [58, 108].

It is also straightforward to improve load balancing in a FARA network by exploiting prior work on load balancing for CSMA networks [64]. Specifically, FARA's contention-aware load balancing is based on two simple techniques. First, each AP or router contends for the medium by simulating contention from as many clients as have packets in its queue. Tracking the number of active clients is relatively simple. The AP or router keeps a hash table of counters. Whenever it receives a packet, it hashes the IP address of the nexthop, and increments the corresponding entry in the hash table. Whenever it transmits a packet, it hashes its nexthop IP and decrements the corresponding entry. Packets which arrive into a full queue are not counted. The number of active clients is equal to the number of non-zero entries in the hash table. This value needs to be updated only when an entry changes to or from zero. Say the number of active clients is N, the FARA AP picks N random contention slots and transmits in the smallest one as long as no other node transmits first.

The second technique scales the size of the contention window as a function of the number of contenders for the medium. Specifically, since FARA nodes contend for the entire medium without channelization, the average contention is higher. To deal with this issue, we leverage prior research on scaling the contention window with the level of contention. Specifically, IdleSense [64] updates the size of the minimum contention window depending on how long the medium is idle. FARA can use this result directly for its contention window scaling.

Combined, these two techniques allow a node to compete for the medium in proportion to its load, while ensuring that CSMA contention avoidance stays efficient.

3.8.1 Hidden Terminals

One concern with discarding channelization is that it might increase hidden terminal scenarios. FARA uses a simple solution that extends adaptive RTS-CTS activation [154], a commonly used mechanism to detect and address hidden terminals. Specifically, since FARA's SNR based rate adaptation allows the sender to converge to the correct rate within

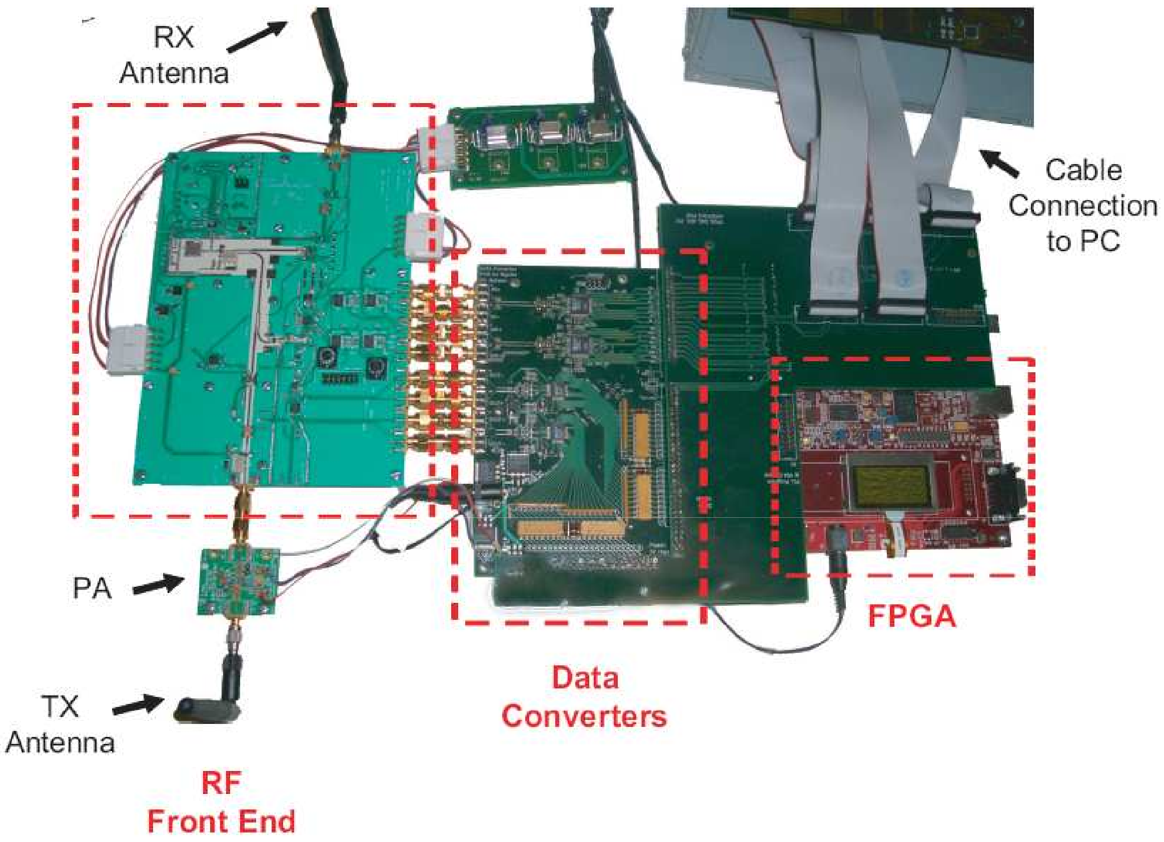


Figure 3-5: The WiGLAN radio platform used in FARA's evaluation.

a few packets, a persistently high loss rate is a good indication of interference due to hidden terminals. Hence, a sender turns on RTS-CTS to a receiver whenever the loss rate to that receiver exceeds a configured threshold (20% in our case). FARA can also additionally leverage recent techniques to solve the hidden terminal problem such as [53].

■ 3.9 Performance

We have implemented a prototype of FARA in FPGA using the WiGLAN radio platform [98], and evaluated it in a wireless testbed.

(a) Hardware: We use the WiGLAN transceiver platform shown in Fig. 3-5. The radio board connects to the PC via the PCI bus, and acts like a regular network card. The radio spans 100 MHz of bandwidth around the 802.11a spectrum and its FPGA code implements standard 802.11 transmit and receive chains, including OFDM over BPSK, 4-QAM, 16-QAM, and 64-QAM modulations. It however differs from traditional 802.11 cards in that it does not use channelization, and hence allows a node to directly access the medium over a 100 MHz of spectrum.

(b) Implemented Infrastructure: Comparing FARA to a frequency-oblivious 802.11 system requires implementing an evaluation infrastructure that is suitable for running both standard 802.11 and FARA.

(a) Supporting 802.11: The WiGLAN board does not implement the 802.11 convolutional codes. Thus, we implement the 802.11 convolutional codes in software and apply them on the signal before passing it to the radio board to be modulated. Matlab has a reference implementation of 802.11 convolutional codes as part of its communication toolbox. It includes the scrambler, the convolutional coder, and the interleaver. We use this reference implementation to ensure that packets receive the same error protection that they would receive with a complete 802.11 implementation.

(b) Supporting FARA: We have implemented both FARA's rate adaptation algorithm and MAC protocol. Specifically, we augmented the FPGA code on the radio board to measure the SNR in each OFDM subband as explained in Section 3.5. The FPGA is also programmed to use SNR measurements to predict the optimal bit rate for each OFDM subband using the table in Fig. 3-1 and communicate it back to the sender. Finally, the frequency-aware MAC is implemented partially in software in the driver and partially in FPGA. The driver divides the subbands between potential nexthops, whereas the PHY code in the FPGA uses this subband assignment to transmit packets concurrently in one frame.

■ 3.9.1 What is the Opportunity from Frequency Diversity?

Frequency diversity is a known property of wireless channels. However, if the performance of a frequency subband changes too quickly (say every millisecond), it will be hard to track it without excessive overhead. Exploiting frequency diversity in rate adaptation and MAC protocols requires the performance of the subbands to change slowly in comparison with the adaptation timescale.

Method. We use the topology in Fig. 3-6, where the node labeled `tx` transmits and the rest of the nodes receive. Since we have a total of 5 radio boards, we fix one of them as the transmitter and move the other boards to cover all the locations indicated in the figure.

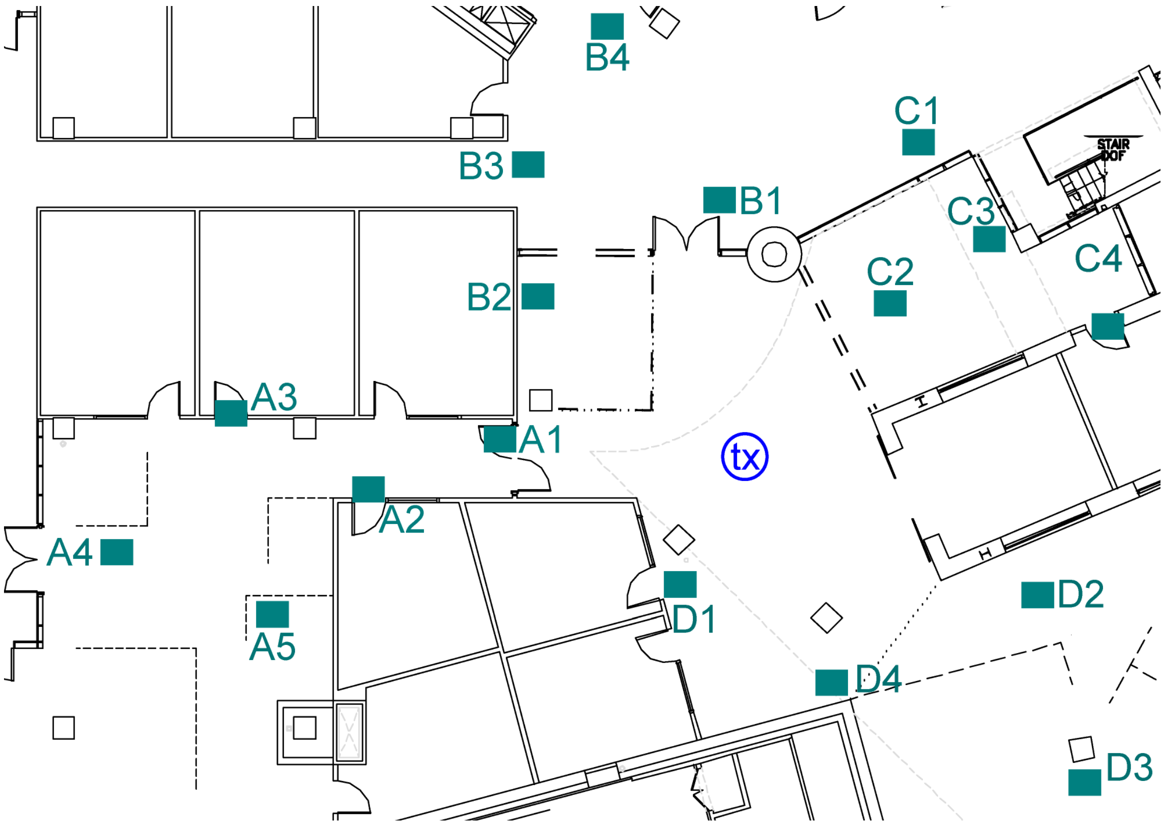
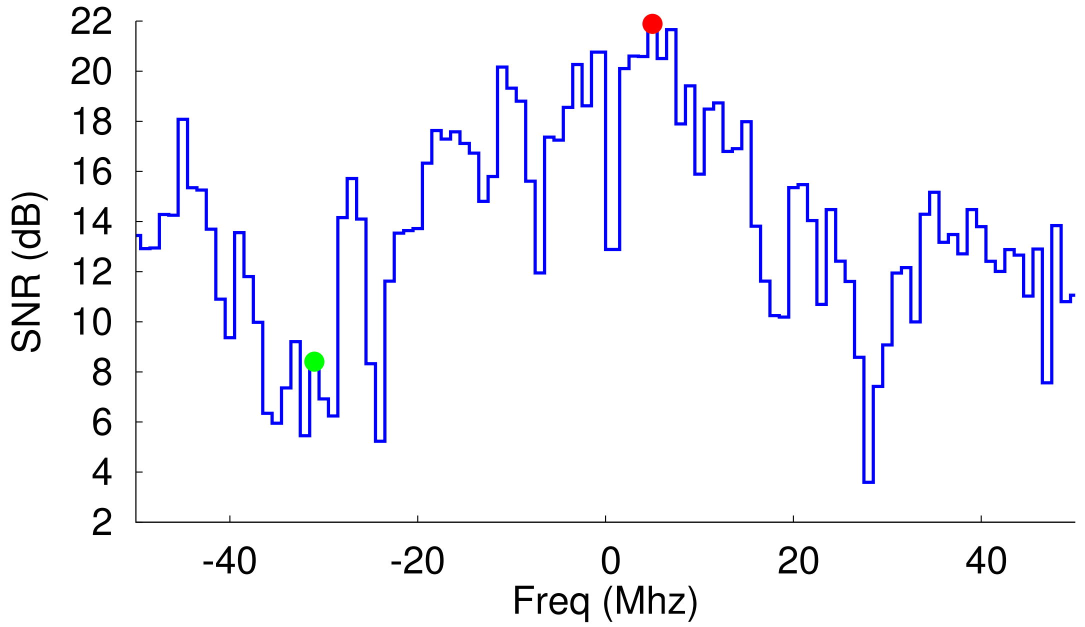


Figure 3-6: Testbed topology showing node locations. The node marked `tx` is used as a transmitter/AP. The other 17 locations are used for receivers.

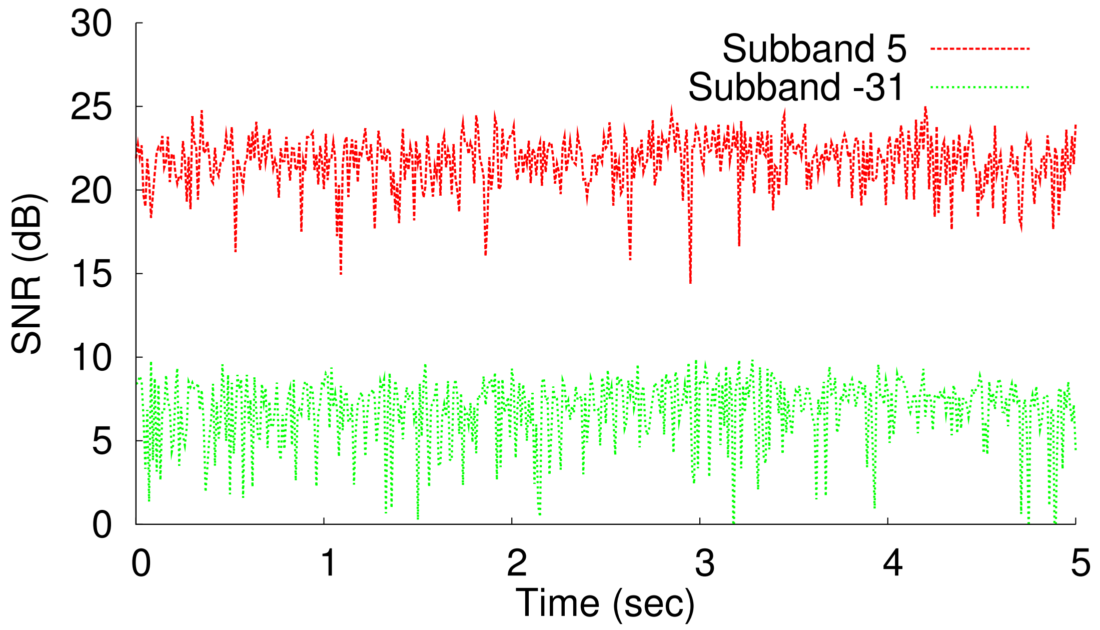
Each run lasts for 10 minutes, and is repeated 5 times. The receivers continuously measure the SNRs in all subbands and report the values as a function of time.

Results. Top graphs in Figs 3-7 (a) and Fig. 3-8 (a) show a plot of subband SNR for both non-line-of-sight (NLOS) and line-of-sight (LOS) channels. The transmission band is depicted as centered on 0 and, the subbands are numbered from -50 to 50, as is conventional for baseband representation. The figures show that SNR differs significantly across subbands for both cases. Differences can be as high as 15–25 dB for the NLOS channel. The LOS channel is less diverse. Nonetheless its subband SNRs can vary by as much as 5–10 dB. Thus, a frequency-aware rate allocation scheme can derive benefits in both these channels.

The bottom graphs in the same figures show how the SNR in a representative subband varies over time. As can be seen, the SNRs largely vary within only a narrow interval even over a period of several seconds, except for the rare deep fade. Hence, a rate adaptation scheme based on SNRs can successfully harness the frequency diversity.

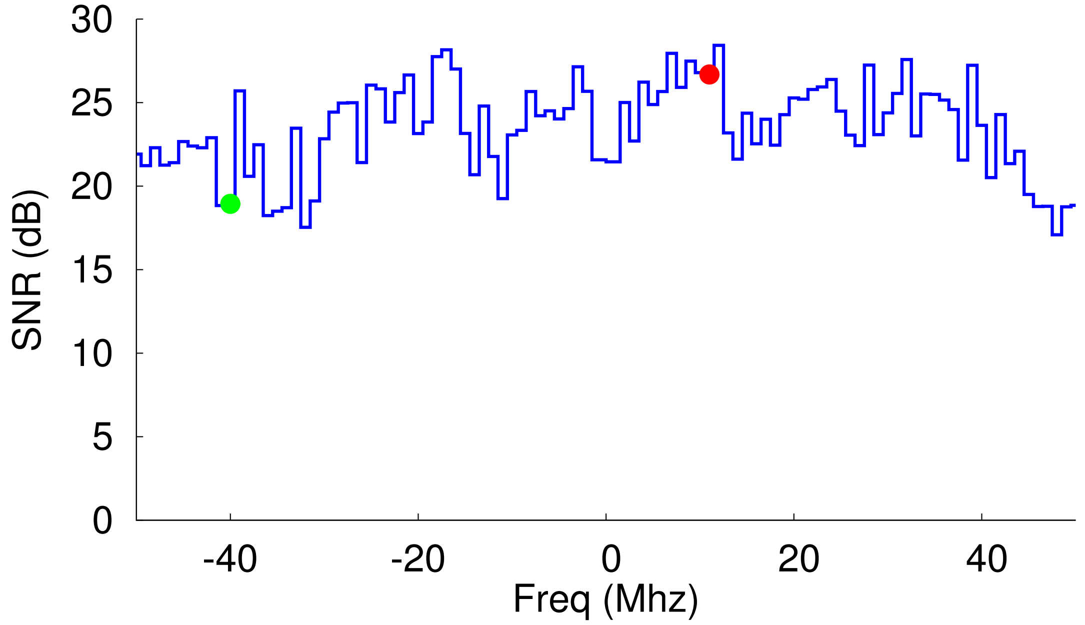


(a) Subband SNR for a Non-Line-of-Sight (NLOS) channel

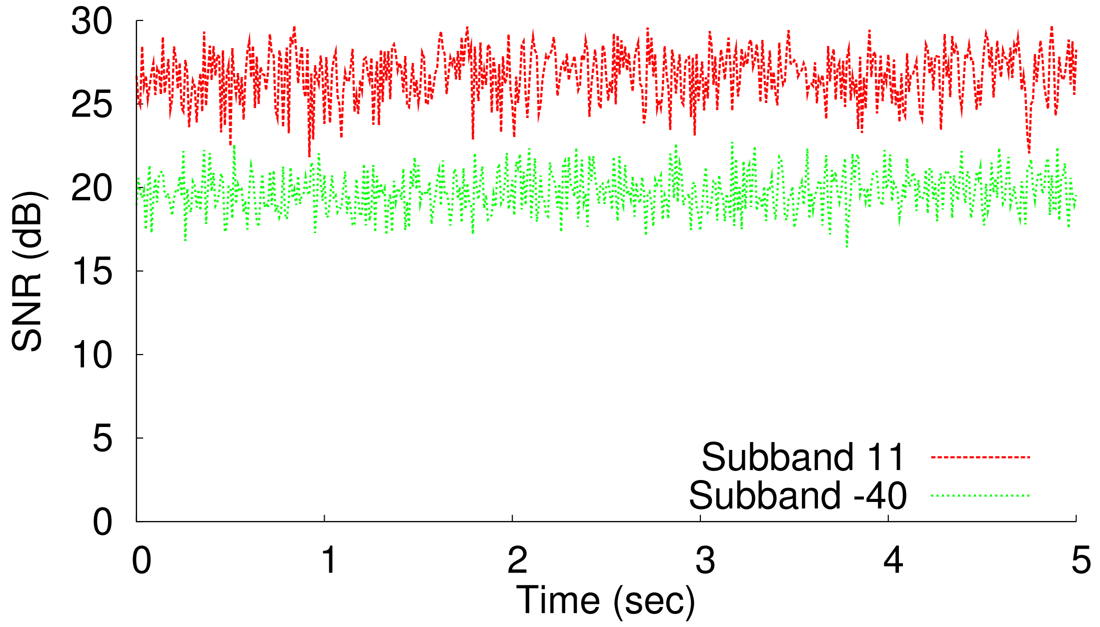


(b) SNR over time for two typical NLOS subbands

Figure 3-7: Frequency Diversity in Non-Line-of-Sight (NLOS): The top graph presents the SNR in each 1-MHz wide subband across the 100 MHz band of our radio for a typical NLOS channel in our testbed. The graph shows that the subband SNRs can differ by more than 20 dB. The bottom graph shows the SNRs of two subbands in the top graph as a function of time. It reveals that the subband SNRs are stable over a multiple-second time period, thereby allowing an adaptive scheme to harness the frequency diversity.



(a) Subband SNR for a Line-of-Sight (LOS) channel



(b) SNR over time for two typical LOS subbands

Figure 3-8: Frequency Diversity in Line-of-Sight (LOS): The top graph presents the subband SNR for a typical LOS channel in our testbed. It shows that the subband SNRs can differ by more than 5-10 dB. While the variation is smaller than in NLOS channels, it is still significant. The bottom graph shows the SNRs of two subbands in the top graph as a function of time. It reveals that the subband SNRs are stable over a multiple-second time period, thereby allowing an adaptive scheme to harness the frequency diversity.

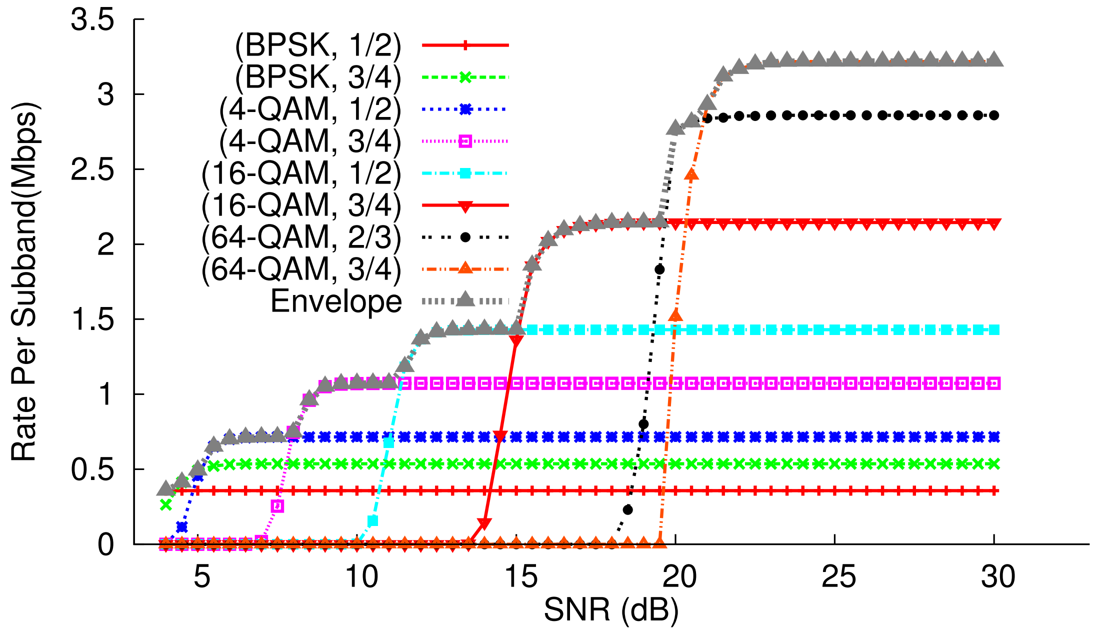


Figure 3-9: Mapping SNRs to a bitrate (i.e., a modulation and code rate): Plots the throughput per 1-MHz subbband as a function of SNR for each choice of modulation and code rate. It shows that for any SNR, the optimal choice of modulation and code rate is fairly clear.

■ 3.9.2 Can We Robustly Map SNR to Best Bitrate?

Method. Harnessing frequency diversity in a rate adaptation scheme requires mapping an SNR value to the maximum sustainable bit rate, *i.e.*, to a combination of modulation and code rate. FARA uses a table look up for this mapping. Underlying our approach is an assumption that given an SNR value, one can determine the optimal combination that maximizes the throughput independent of location and time. Thus, in this experiment, we show that the SNR value robustly determines the best bit rate.

As in the previous experiment, the `tx` node in Fig. 3-6 transmits and the rest of the nodes measure the received SNRs in each subband. For this experiment, we treat each subband completely independently, *i.e.*, we assign it its own modulation, convolutional FEC code, and checksum. The separate checksum allows us to decide whether the bits in a particular subband are decoded correctly, independent from the bits in other subbands. The sender's transmissions use all 802.11 bit rates in a round robin manner, assigning the same rate to all subbands. For each received packet, the receiver reports the SNR in each subband and whether the bits in that subband have passed the checksum test. We aggregate this information across all subbands and all receivers. We plot in Fig. 3-9 the bits per second

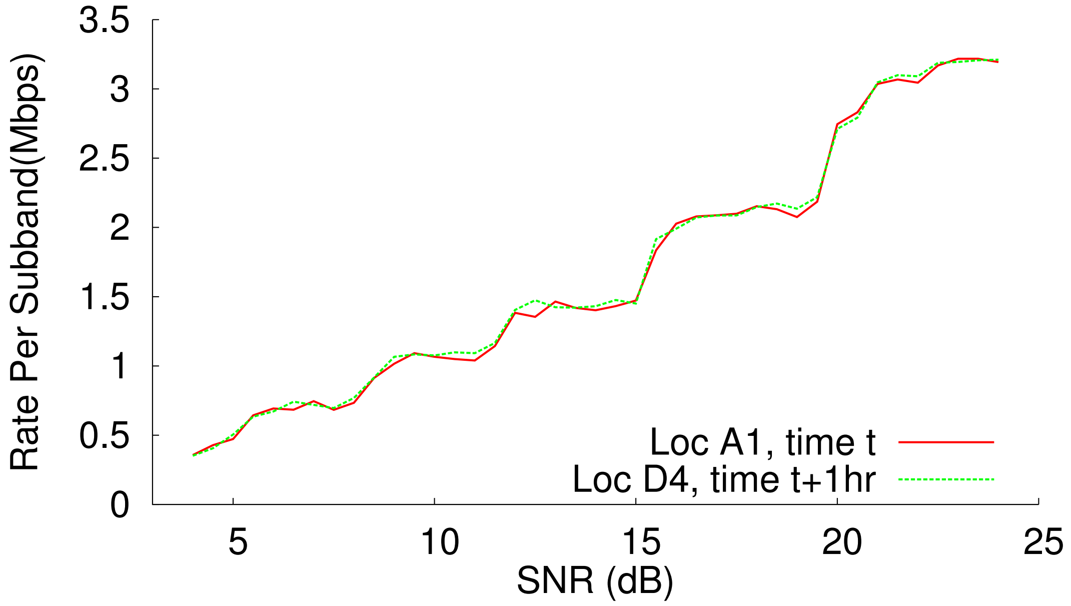
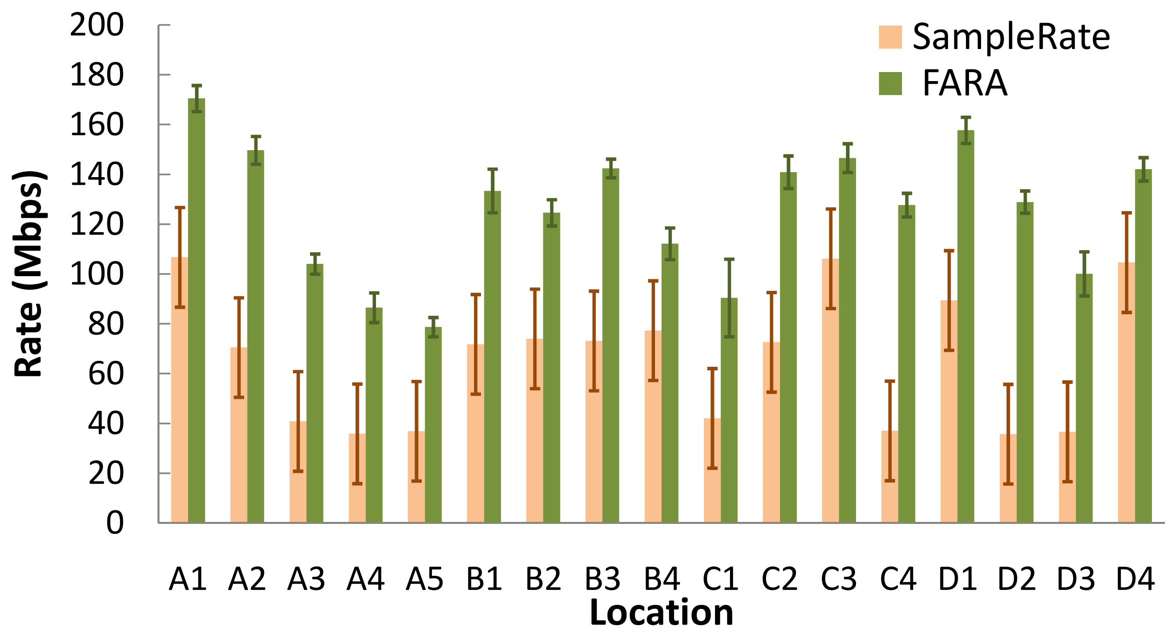


Figure 3-10: Stability of the relation between SNR and the optimal modulation and code rate across time and space. The figure shows the envelope of the functions in Fig. 3-9, for measurement collected at receivers `A1` and `D4` at different times. The two envelopes match closely showing that the SNR dictates the best modulation and code rate.

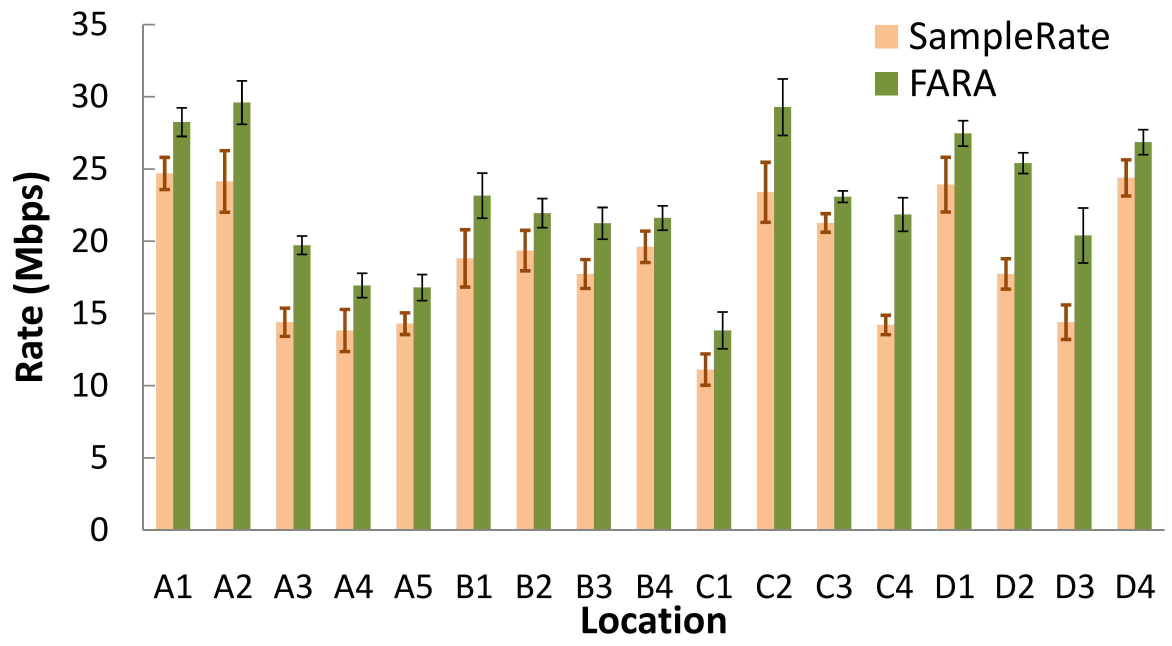
per 1-MHz subband, *i.e.*, the throughput of a single 1-MHz subband as a function of its SNR, for all 802.11 modulation and code combinations.

Results. Fig. 3-9 shows that the per-subband SNR clearly determines the optimal modulation and code rate. For example, when the subband SNR is 17 dB, the optimal choice is the third from the top, i.e., 16-QAM and code rate of 3/4. Using any higher modulation or code rate reduces the probability of decoding the bits in that subband and brings the per-subband throughput close to zero. Using any lower modulation and code rate reduces the subband throughput. On the other hand, when the subband SNR is below 3.5 dB, no combination of modulation or code rate works. In this case, it is better not to transmit in that subband, i.e., to suppress that subband.

Fig. 3-10 plots the envelope of the curve in Fig. 3-9 for two different locations and times. It shows that the mapping of SNR to a modulation and code rate is stable across time and space. Thus, mapping subband SNRs to bit rates requires only a table lookup which reports the SNR values that cause a transition from one set of modulation and code rate to the next. In fact, Table 3-1, which we presented in Section 3.6.2, summarizes the information in the previous figures and is all that a bitrate adaptation protocol needs to map SNRs to bitrates.



(a) Frequency-aware rate adaptation for a 100 MHz channel



(b) Frequency-aware rate adaptation for a 20 MHz channel

Figure 3-11: FARA Rate Adaptation: FARA's frequency-aware rate adaptation achieves higher throughput than SampleRate's frequency-oblivious rate adaptation at all locations, with gains varying from $1.4\times$ to $3.6\times$ for a 100 MHz wide channel, and $1.1\times$ to $1.5\times$ for the 20 MHz channel.

3.9.3 Gains of Frequency-Aware Rate Adaptation

Now that we have established the existence of frequency diversity, its stability which makes it amenable to be harnessed by a rate adaptation protocol, and the robustness of the mapping from SNR to optimal bitrate, we measure the experimental gains from a frequency-aware rate adaptation protocol.

Method. Again we use the topology in Fig. 3-6. We fix the sender in location `tx` and randomly pick a receiver location. We repeat the experiment for all receiver locations shown in Fig. 3-6. For each location, we compare two schemes. The first is FARA's frequency-aware rate adaptation as described in Section 3.6. The second uses SampleRate [18], a well known rate adaptation scheme that assigns the same bitrate to all subbands. Each run lasts for ten minutes, and is repeated five times. We look at the benefit of frequency-aware adaptation for two scenarios: a standard 20 MHz 802.11 channel, and a wide 100 MHz channel.

Results. Fig. 3-11 shows that FARA's frequency-aware rate adaptation achieves significantly higher throughput than a frequency-oblivious algorithm such as SampleRate. Specifically, for a standard 20 MHz channel, a frequency-aware rate adaptation scheme increases the throughput by $1.24\times$. These gains become even higher as we move to wide and bonded channels, where FARA's rate adaptation improves the average throughput by $2.1\times$ over SampleRate.

The throughput gain is larger for receivers with worse channels. For example, some of the worse receivers experience a throughput gain that is as high as $3.5\times$. This is due to FARA's ability to avoid bad frequency bands. Specifically, SampleRate's frequency-oblivious rate adaptation experiences significant errors from subbands that have very low SNRs and hence cannot support even the lowest transmission rate. To compensate for such bad subbands, SampleRate has to drastically lower its average transmission rate and increase coding across all subbands. In contrast, FARA suppresses subbands with less than 3.5 dB SNR and does not need to reduce the rate of every subband to compensate for the extra errors from such bad subbands.

Also, the throughput gain for NLOS channels is typically higher than the gain for LOS channels, because these channels see higher frequency diversity due to the greater prevalence of multiple paths with similar attenuation. Interestingly, location `A2` shows significant throughput gain even though it has a LOS channel to `tx`, because it is within a passage

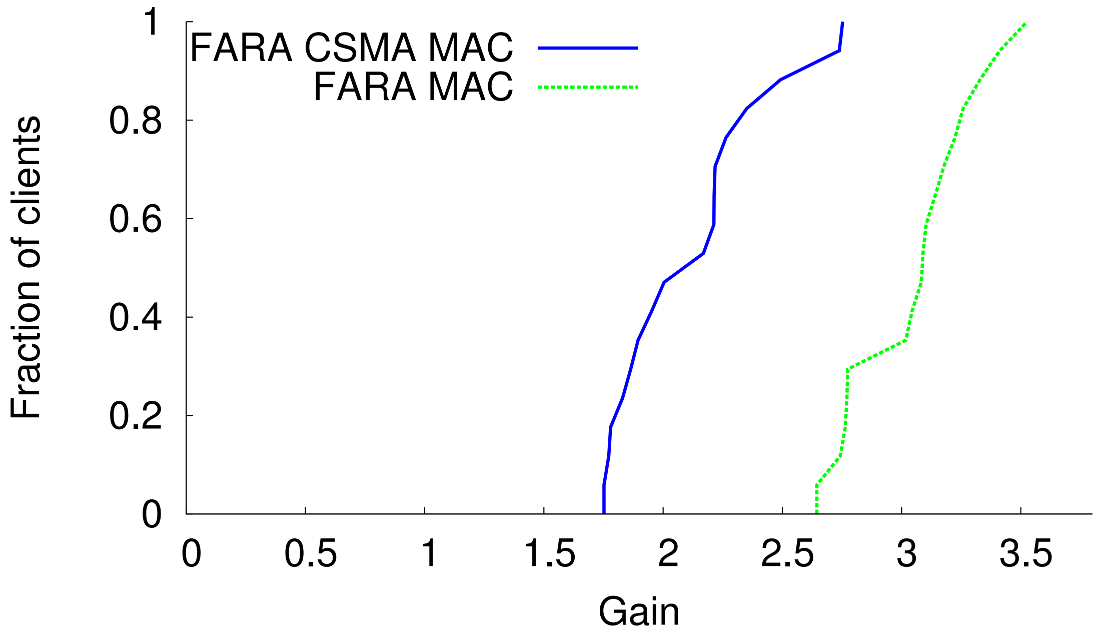


Figure 3-12: Gains from a Frequency-aware Architecture: The figure plots two CDFs. The dashed line is the CDF of the ratio of client throughput under FARA to its throughput in traditional 802.11 networks which use SampleRate and CSMA MAC. The solid line is the CDF of the ratio of client throughput under FARA with a CSMA MAC and traditional 802.11 with SampleRate and CSMA. The CDFs show that FARA provides on average 3× throughput gain. 70% of the gain comes from FARA's frequency-aware rate adaptation, and 30% is due to its frequency-aware MAC protocol.

that provides multiple opportunities for reflected waves that together create significant frequency diversity.

■ 3.9.4 Gains of Frequency-Aware MAC

We now examine the throughput improvement provided by a frequency-aware MAC over a frequency-oblivious MAC.

Method. We again use the topology in Fig. 3-6. We collect measurements by transmitting from node `tx` to four random receiver nodes. We consider only four concurrent receivers because we have a total of five radio boards (including the transmitter). However, we can experiment with various scenarios by choosing different receiver sets. We run the experiment 10 times for each set of receivers, and repeat for a variety of receiver sets. We compare two MAC protocols: first, a frequency-oblivious CSMA MAC, where a sender checks whether the medium is available and transmits the packet at the head of its queue, and second, FARA's frequency-aware MAC as described in Section 3.7. Note that FARA transmits four packets in every frame and hence has less medium sensing overhead. Thus, to ensure that the differences between the two MACs are due only to frequency diversity, and not medium access overhead, we allow the sender to transmit its packets without waiting for an idle medium. This optimization favors the baseline MAC, and is possible because we have only a single sender in each experiment. Note that both FARA and the CSMA MAC use the same spectrum of 100 MHz.

Results. Fig. 3-12 plots the CDFs of the ratio of the throughput in FARA to the throughput in traditional 802.11 which uses SampleRate and a CSMA MAC. The CDF is computed across all receivers in our testbed and all runs. The graph contains two CDFs, one for a full-fledged FARA, and one for FARA after replacing its frequency-aware MAC with a frequency-oblivious CSMA MAC. The figure shows that a full-fledged FARA improves the median throughput by $3.1\times$ over a traditional SampleRate based CSMA MAC. The figure also demonstrates that about 30% of this gain is due to FARA's frequency-aware MAC while 70% is due to its rate adaptation scheme, showing that both mechanisms contribute significantly to the throughput improvements. Finally, it shows that all clients achieve significantly higher throughput with FARA than with traditional 802.11, which shows that FARA is beneficial to all nodes.

■ 3.9.5 Load Balancing

We now demonstrate that, with FARA, multiple APs sharing the same frequency spectrum can achieve load balancing by using the load aware contention scheme described in Section 3.8.

Method. We use a modified version of the topology in Fig. 3-6. Specifically, we put two transmitters (*i.e.*, two APs) around the location `tx`, and place their corresponding receivers at `C3` and `D4` respectively. Since we have a small number of radio boards, we make each board simulate a number of clients. AP1 simulates a varying number of backlogged clients, ranging from 1 to 5, on the link to `C3`, while AP2 always has only one backlogged client at `D4`. This setup allows us to experiment with scenarios with imbalanced loads, where AP1 has up to 5 times the number of clients of AP2. We perform an infinite download to each client and compute the per client throughput averaged over the first 10 minutes.

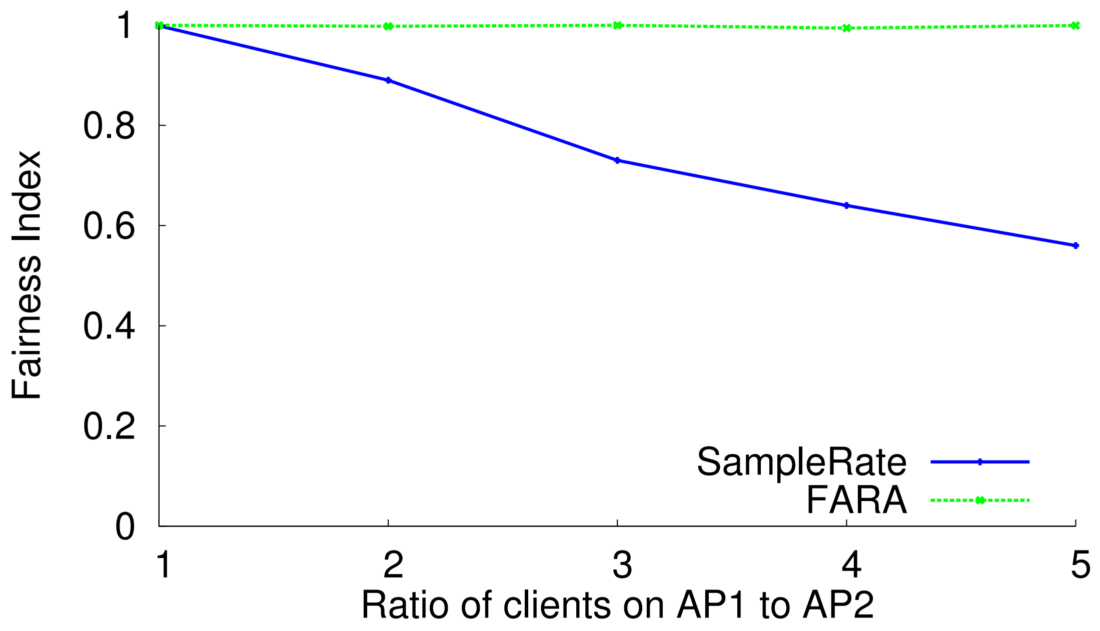


Figure 3-13: Load balancing with FARA: The figure plots the Jain's Fairness Index as a function of the ratio of the number of clients on AP1 to those on AP2. Note that fairness is optimal when the index is 1, and is worst when the index is $1/n$, where n is the total number of clients.

Using the per client throughput, we compute the Jain Fairness Index for the network, as follows [72]:

$$Fairness\ Index = \frac{(\sum_1^n x_i)^2}{n\sum_1^n x_i^2},$$

where x_i is the throughput of client i. The network shows optimal fairness when the index is 1, and is completely unfair when the index is $\frac{1}{n}$, in which case only one client has traffic.

We compare channelized 802.11 to a single channel FARA-equipped 802.11. Since we have only two APs, we limit both FARA and the channelized 802.11 to a total bandwidth of 40 MHz. In the case of channelized 802.11, the 40 MHz band is divided into two channels of 20 MHz and each AP is assigned a different channel. In the case of FARA, only one channel of 40 MHz is used for both APs.

Results. Fig. 3-13 plots the Jain fairness index as a function of the ratio of the number of clients at AP1 to those at AP2. The figure shows that the Jain fairness index remains close to 1 for FARA, whereas, for traditional 802.11, it drops linearly as the difference in load between the APs increases. This is because, in traditional 802.11, different APs operate on different channels and hence the single client on AP2 enjoys a throughput that is about the sum of the throughputs of all clients on AP1. In contrast, FARA discards channelization and further allows each AP to contend for the medium in proportion to the number of active clients. Hence, it allows the nodes to achieve a fairer throughput distribution.

■ 3.10 Discussion

This chapter addresses the challenge and the opportunity of frequency diversity presented by the growing trend of wireless systems to use wider frequency bands. It demonstrates that a frequency-aware design of the physical, link and MAC layers offers significant throughput improvements both for a single client and for a network of clients, as compared to current frequency-oblivious rate adaptation and medium access schemes.

While the results in this chapter have been presented in the context of 802.11, FARA applies to a wider variety of scenarios. Specifically, measurement studies show the existence of frequency diversity in the WiMax, UWB, and the 60 GHz range [11, 45, 99, 134]. All of these technologies use OFDM and have static or moderately dynamic applications, where the per-subband SNRs change relatively slowly [153, 69, 50]. FARA naturally extends to these scenarios.

FARA can also be extended for cognitive operation, and applied to the newly introduced whitespaces. The FCC has recently opened up for unlicensed access 100-250 MHz of digital whitespaces vacated by television bands as part of the analog-to-digital transition [42]. These whitespaces demonstrate significant frequency diversity [15, 117]. Further, they are expected to be used for several static and low mobility scenarios such as fixed wireless broadband access in rural and urban areas, as well as data connectivity inside the home, where FARA could provide significant throughput benefits. To do so, FARA needs to detect which subbands are occupied by the primary owner of the whitespace, and avoid these occupied subbands. FARA can leverage much prior work on detecting and agreeing upon occupied subbands [9, 121], and avoiding them by suppressing these occupied subbands, as in the cognitive PHY of [121], and as discussed in Section 3.6.2.

CHAPTER 4

SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity

Channel diversity is an intrinsic property of wireless networks. Recent years have witnessed the emergence of many distributed protocols like ExOR, MORE, SOAR, SOFT, and MIXIT that exploit channel diversity across multiple receivers in 802.11-like networks. In contrast, the dual of receiver channel diversity, sender channel diversity, has remained largely elusive to such networks. This chapter presents SourceSync, a distributed cooperative architecture for harnessing sender channel diversity. SourceSync enables concurrent senders to align their transmissions to symbol boundaries, and cooperate to forward packets in the same spectrum at higher data rates than they would have achieved by transmitting separately.

■ 4.1 Overview

Diversity across nodes is an intrinsic property of wireless networks. The wireless environment exhibits channel diversity both across receivers and senders. Receiver channel diversity is the property that a single transmitted packet traverses different channels to different receivers, and hence is unlikely to suffer fading at all receivers at the same time. Sender channel diversity, on the other hand, is the property that a packet transmitted *si-*

multaneously from multiple senders traverses different channels to the same receiver, and hence is unlikely to suffer fading from all senders at the same time. In the context of 802.11 networks, the ability to have multiple transmitters simultaneously forward a packet to a receiver can harness both frequency diversity and power gains. Specifically, 802.11 channels span a relatively wide bandwidth (20–40 MHz), where different senders experience deep fading in different frequencies. Enabling multiple transmitters to simultaneously forward a packet to a receiver ensures that no frequency is deeply faded at the receiver, and reduces the overall bit error rate for a particular transmission power. Second, simultaneously forwarding a packet enables senders to combine their transmission power and thereby deliver a higher SNR to the receiver, as compared to a single sender.[1]

Despite the benefits of simultaneous forwarding from multiple transmitters, existing approaches for sender diversity in 802.11 networks restrict themselves to only one sender transmitting at a time, using mechanisms like picking the sender with the best channel [107]. This is in sharp contrast to receiver diversity where many practical systems like ExOR, MORE, SOAR, SOFT, and MIXIT [19, 26, 125, 155, 79] leverage simultaneous reception across multiple receivers.

Simultaneous transmission from multiple senders has challenged 802.11 for three main reasons.

- First, senders need to be synchronized to the symbol level in order that their signals combine on the medium in a manner that reduces the overall packet error rate. Such fine-grained transmitter synchronization is difficult to achieve in a distributed manner, as has been observed by past research [36, 78, 53, 119]. The difficulty arises because the different transmitters need to time their transmissions so that they are synchronized accurately (to within tens of *ns*) [36] at the receiver. In the absence of a shared clock or a central controller, the only mechanism for synchronization is for senders to use packet reception as a reference. However, such a mechanism requires transmitters to compensate for differences in propagation delays, and hardware turnaround times from reception to transmission. These measurements are challenging because a node does not detect packet reception at the exact instant when the signal arrives at its antenna, but rather incurs a random delay depending on the

[1]The FCC limits the maximum transmission power of a single sender, and combining transmissions therefore increases the maximum received power.

noise in the environment and the receiver hardware. This variability is usually on the order of hundreds of *ns* [151], which is too high for accurate symbol-level synchronization.

- Second, the received signal is a combination of signals from multiple senders. Each of these signals has traversed a different path, and has hence experienced a different channel. One might think that the receiver could compensate for the channel distortion of the composite signal in the same manner as it would compensate for the channel distortion of a signal from a single sender. Unfortunately, this approach does not work since the composite channel has fundamentally different characteristics from single sender-receiver channels. Specifically, unlike single sender-receiver channels, which have a constant attenuation throughout a packet, the attenuation of the composite channel varies even within a single packet. This is because the oscillators of different senders naturally have slightly different operating frequencies, and hence the signals from different senders continuously rotate relative to each other.
- Finally, transmitted signals are complex numbers which have phases. Unless these signals are carefully orchestrated at the senders, they can add up constructively, enhancing each other, or destructively, weakening each other.

This chapter introduces SourceSync, a practical architecture for harnessing sender diversity. SourceSync is designed for OFDM, which is the transmission scheme for most modern wireless networks, including 802.11 a/g/n, WiMax, LTE etc. SourceSync has three components that harness sender diversity in a distributed manner:

Symbol Level Synchronizer (SLS). SourceSync has a distributed synchronization algorithm that leverages packet reception as a time reference, computes robust estimates of the propagation delays from all senders to the receiver, as well as hardware turnaround times at each of the senders, and compensates for these delays at the senders prior to transmission, in order to ensure that the packets arrive synchronized at symbol boundaries at the receiver. The key feature that allows SourceSync to achieve tight synchronization is that it can prevent the inherent variability in packet detection from inducing variability in its propagation delay and turnaround time estimates. SourceSync has a mechanism that allows it to accurately measure the delay between the first sample of a packet and when the receiver detects that packet, and account for the delay when computing its estimates.

Further, SourceSync can leverage data packets to track changes in propagation delay over time, and hence keep senders synchronized without the need for active measurements.

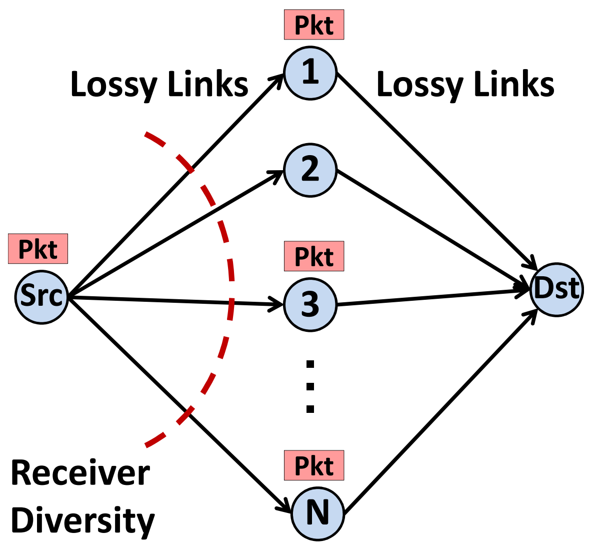
Joint Channel Estimator (JCE). A SourceSync receiver decodes the combined signal from multiple synchronized senders. However, SourceSync differs from prior schemes, where transmitted signals interfere, and hence decoding the signals either requires multiple transmissions from each sender, as in ZigZag [53], or a large difference in power (or code rate) between them, as in Successive Interference Cancellation [59, 149]. In contrast, SourceSync does not need to treat senders as interfering, and can decode a single simultaneous transmission from multiple senders, even when they have comparable powers. It estimates the individual channels from each sender, computes how they interact to create the composite channel, and tracks the variations of the composite channel through the combined packet.

Smart Combiner (SC). Since signals from multiple senders rotate continuously relative to each other, naively transmitting the same packet from all senders will cause the signals to combine destructively at some points within the packet. Therefore, senders need to have a joint strategy for manipulating the phase of the signal prior to transmission to ensure that their transmitted codewords do not combine destructively. SourceSync leverages the rich body of research on space-time block codes [142, 10, 71], which are typically used in MIMO systems to control how signals from different antennas on a single transmit node combine at a receiver. In contrast to MIMO systems, however, SourceSync uses these codes in a distributed manner across multiple transmit nodes.

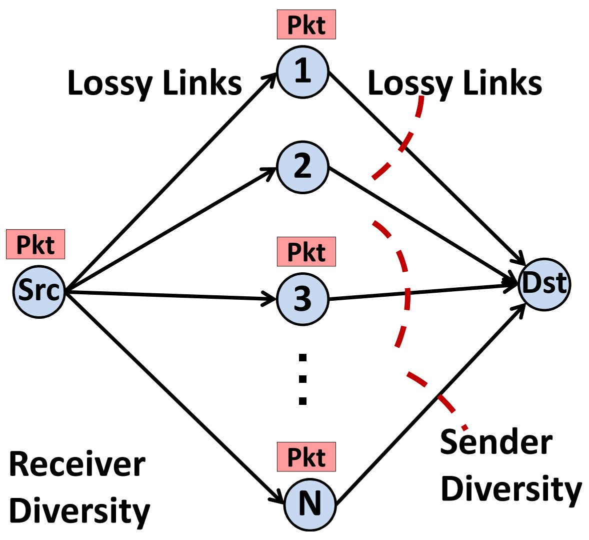
We use SourceSync to develop the following two protocols.

■ 4.1.1 Combining Sender Diversity with Opportunistic Routing

Opportunistic routing protocols leverage receiver diversity; they exploit the fact that since wireless receptions are probabilistic, it is unlikely that all nodes closer to the destination are unable to receive a packet, as shown in Fig. 4-1(a). Protocols like ExOR, MORE, SOAR, and MIXIT therefore allow any downstream node that receives a packet to forward it to the destination. However, none of these schemes take advantage of the analogous opportunity of sender diversity presented by the fact that multiple nodes often receive the same packet. SourceSync complements the opportunistic receptions exploited by current pro-



(a) First-hop receiver diversity



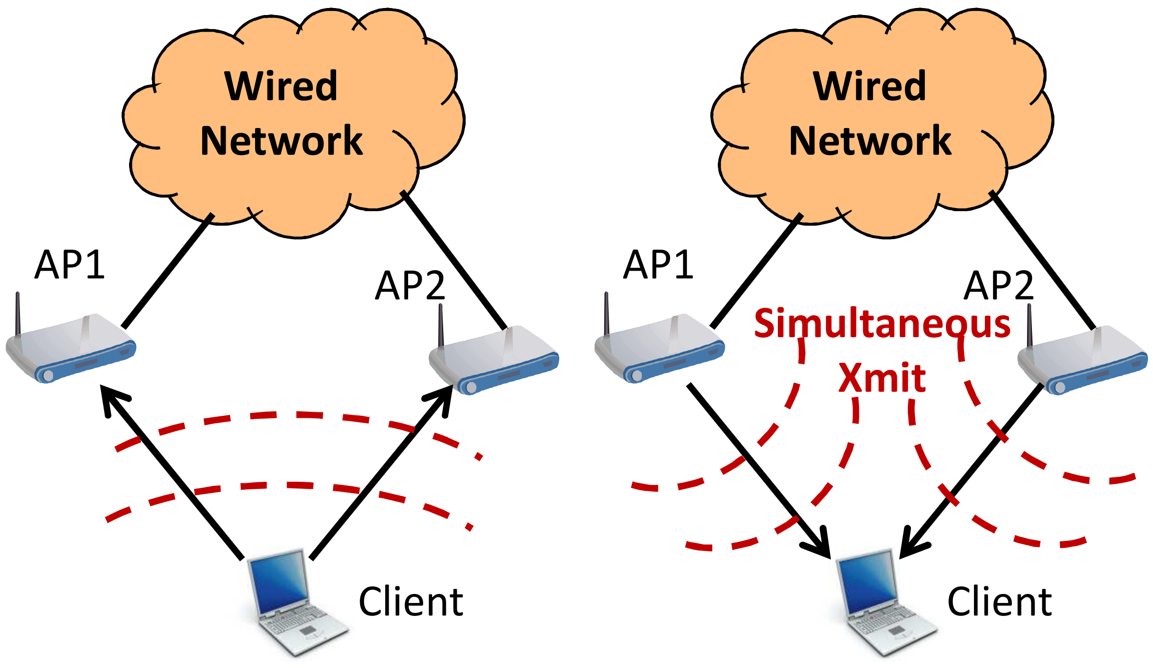
(b) Second hop sender diversity

Figure 4-1: Opportunistic routing with sender diversity. SourceSync enables multiple forwarders to transmit jointly to the destination.

tocols with opportunistic synchronous transmissions by multiple forwarders. Specifically, since multiple forwarders are likely to receive a packet, they can transmit it simultaneously as shown in Fig. 4-1(b). This provides two types of gains. First, since different forwarders experience fades in different frequencies [119], joint transmission reduces the likelihood that a frequency experiences a deep fade at the receiver, and hence decreases the overall bit error rate. Second, since joint transmission allows forwarders to combine their power, it improves the receiver SNR, and thereby its bit rate.

■ 4.1.2 Combining Sender Diversity with Last-hop Receiver Diversity

Protocols like MRD, SOFT and Link-Alike [106, 155, 73] all exploit different aspects of the same concept: last-hop diversity. Consider, for example, a sender with poor connectivity to multiple nearby APs. A transmitted packet is unlikely to reach any specified AP, but is likely to be received by at least one AP. All the above protocols exploit this receiver diversity by allowing APs to combine received bits or packets over the wired network, and hence can increase *uplink* reliability without any retransmissions, as shown in Fig. 4-2(a). However, none of these schemes can similarly address a lossy *downlink* without expending medium time on retransmissions. SourceSync complements all these protocols by harnessing sender diversity to increase downlink reliability without any retransmissions, analo-



(a) Uplink receiver diversity (b) Downlink sender diversity

Figure 4-2: Last-hop with sender diversity. SourceSync enables multiple APs to transmit jointly on the downlink.

gous to receiver diversity mechanisms on the uplink. Specifically, instead of requiring that a client receive packets from only one AP at a time, in SourceSync, multiple neighboring APs can transmit simultaneously to the client as in Fig. 4-2(b), and increase throughput.

■ 4.1.3 Results

We implemented SourceSync on the FPGA of the WiGLAN radio platform [37]. We also implemented proofs of concept of both last-hop diversity, and opportunistic routing with sender diversity. Results from an indoor wireless testbed reveal the following:

- SourceSync's symbol level synchronization is accurate. Testbed evaluations show that two randomly chosen transmitters using SourceSync have a 95^{th} percentile synchronization error of at most 20 *ns* across the range of operational SNRs of 802.11.

- SourceSync increases the gains of opportunistic routing by exploiting sender diversity. Evaluating across multiple deployments with different bitrates and link loss rates, we show that the combination of SourceSync and ExOR achieves a median throughput gain of up to 45% over ExOR alone, and up to 2× over single-path routing.

- SourceSync is effective in harnessing last-hop sender diversity. Specifically, by having two APs transmit simultaneously to a client, SourceSync provides a median throughput gain of 57%. This is because the higher power resulting from simultaneous transmission from APs allows the combined transmission to use a higher 802.11 rate than a transmission by either AP alone.

■ 4.1.4 Contributions

This chapter makes the following contributions:

- It demonstrates via a design, implementation and testbed evaluation the practicality and benefits of simultaneous transmission in 802.11 networks.
- It presents a distributed algorithm for symbol level synchronization and an empirical study of its accuracy.
- It reveals the synergy between opportunistic routing and sender diversity by showing that opportunistic receptions can be further used to enable concurrent forwarding to downstream nodes.

■ 4.2 Related Work

Sender diversity was pioneered by Laneman and Wornell's work on cooperative diversity, which theoretically demonstrated the gains of spatially diverse senders cooperating to relay information [84, 128]. Since then, many papers have analyzed aspects of sender spatial diversity focusing on signal processing and coding algorithms at the relays [132, 82, 129]. These papers focus on theoretical gains, ignore practical issues such as transmitter synchronization and oscillator offsets, and do not present a practical working system. Cellular networks today attempt to exploit sender diversity using Distributed Antenna Systems (DAS) [28]. DAS do not allow separate transmitters to send simultaneously; rather, they consist of a single transmitter with geographically distributed antennas connected using long, low attenuation cables. These systems are expensive and inflexible [34], and hence there is increasing interest in exploiting simultaneous transmissions from multiple senders in future cellular networks. The most recent WiMax multihop relay standard [138] includes simultaneous transmissions from multiple relays as an optional feature, and cooperative relays are also being considered for the future 3GPP LTE-Advanced standards [137]. However, there is no published work currently demonstrating a practical design and implementation of simultaneous transmissions for cellular systems, and further these systems operate under different constraints as they have the benefit of a centralized scheduler and a shared GPS clocking mechanism. 802.11 networks have also shown interest in exploiting sender diversity; however they still restrict themselves to only one sender transmitting at a time, using mechanisms like picking the sender with the best channel [107], which can neither exploit frequency diversity across senders, nor the power gain from combining multiple senders. Concurrent with our work, Zhang *et al.* [160] have demonstrated an implementation of cooperative diversity with nodes connected to a single shared clock. In contrast, our approach requires no shared clocks and applies to practical wireless networks, and also demonstrates the synergy of sender diversity with opportunistic routing.

Additionally, there has been recent work on systems that exploit concurrent transmissions from multiple senders, but cannot provide any sender diversity gains since they do not synchronize transmissions at the symbol level. These include systems like SMACK [36] for group acknowledgments, Message-in-Message [96] for exposed terminals, interference cancellation [59] and ZigZag [53] for hidden terminals, and ANC [78] for network coding.

Finally, SourceSync builds on past work on space-time block codes. These codes are used by different antennas on a single MIMO transmitter and do not extend to different transmitters due to lack of synchronization [142, 8, 10, 71], or because of oscillator frequency offsets [87]. SourceSync addresses synchronization and oscillator offset issues, showing that these codes can be implemented in a distributed manner to collect the gains of sender diversity in practice.

■ 4.3 SourceSync

SourceSync enables multiple senders to concurrently forward a packet to one or more receivers in order to collect diversity and power gains. It does so via a fully distributed joint PHY-MAC architecture.

(a) MAC: Medium access for concurrent transmissions is done by one of the senders, which we call the lead sender. Any node in the network can be a lead sender for a transmission. The lead sender accesses the medium via carrier sense, just as in 802.11. When the lead

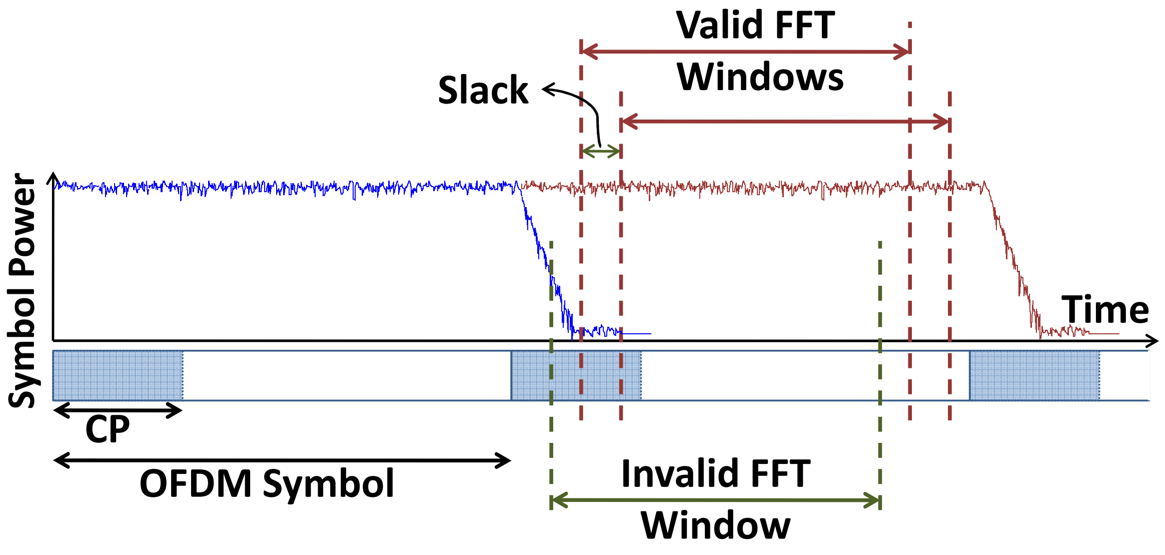


Figure 4-3: FFT windows at a receiver for a single transmitter. Any FFT window within the slack is valid. Any other FFT window would include energy from the previous symbol and hence is invalid.

sender acquires the medium, it transmits a synchronization header. Other nodes that hear the synchronization header, and have the packet being transmitted, can then join the lead sender's transmission.

(b) PHY: The PHY layer ensures that concurrent transmissions are decodable at their intended receiver(s). It does so using three components: (a) a Symbol Level Synchronizer that ensures that transmissions from multiple nodes are synchronized, and can be decoded jointly at the receiver, (b) a Joint Channel Estimator which estimates the composite channel from the concurrent senders, and compensates for the resulting distortions, and (c) a Smart Combiner that encodes the concurrent transmissions to ensure that they combine on the channel in a manner that reduces the error rate at the receiver.

The next few sections describe the PHY in detail. The MAC is a simple extension of 802.11 carrier sense, and is described in the specific context of WLANs (Section 4.7.1) and opportunistic routing (Section 4.7.2).

■ 4.4 Symbol Level Synchronization

■ 4.4.1 Why do we synchronize transmitters?

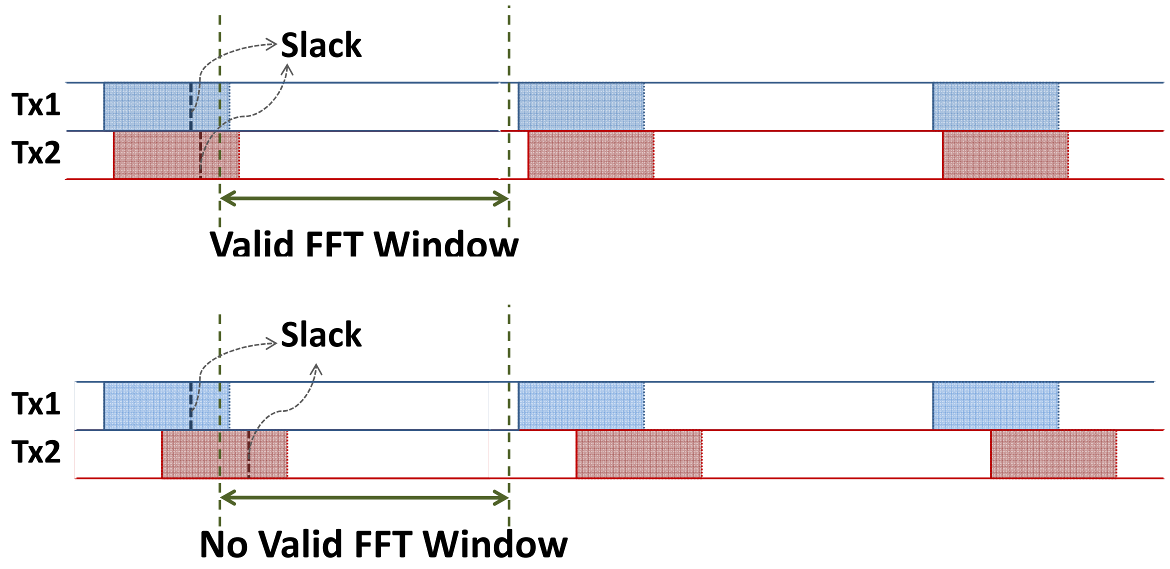
To understand why one needs to synchronize, let us start by explaining what happens with a single sender-receiver pair. When a sender transmits to a receiver, the wireless signal bounces off walls, obstacles etc. and traverses multiple paths to the receiver. This phenomenon, known as the multipath effect, is common in wideband wireless channels such as 802.11. As a result of the multipath effect, different copies of the same signal arrive at the receiver delayed with respect to each other. This means that the energy from one symbol bleeds into the next symbol, and corrupts its signal as shown in Fig. 4-3. Because of this effect, OFDM symbols typically have a guard interval between them, called the cyclic prefix (CP). In a typical network, the value of the CP is chosen to be as small as possible while still accounting for the maximum multipath delay spread of the network, *i.e.*, the maximum delay difference between delayed copies of the signal.

OFDM data is encoded in the frequency domain. An OFDM receiver, in order to decode, converts the received symbol to a frequency representation by taking an FFT of the symbol. In order to do so while ensuring that the symbol is not corrupted by multipath noise from the previous symbol, the receiver should skip the samples in the CP, and take the FFT of the remaining samples.[2] In a typical network, the CP has a small amount of slack to allow for packet detection errors [151]. This means that the receiver has a corresponding amount of slack in the choice of where to align the receiver FFT window in a symbol. Thus, as shown in Fig. 4-3, any FFT window within the slack is valid. Any other FFT window would include energy from the previous symbol and hence lead to erroneous results.

Now, consider two senders transmitting the same symbol to a receiver. If the copies of this symbol from the two transmitters arrive at the receiver aligned within the existing slack of the CP, the receiver can take the FFT as before while still receiving energy only from this symbol, as shown in Fig. 4-4(a). If not, as before, any FFT window that the sender uses would include energy from the previous symbol, as shown in Fig. 4-4(b), and hence would yield incorrect results.

Of course, it is possible to increase tolerance to misalignment and provide more slack by increasing the CP. This approach, however, is problematic for two reasons. First, without sender synchronization, as in existing 802.11 networks, the amount of misalignment between senders can take any value depending on the differences in propagation delays and hardware processing times on different senders. While propagation delays may be bounded in certain environments based on the network diameter, the hardware process-

[2]Since the CP is a cyclic permutation of the symbol, and since FFT is periodic, the FFT yields correct results as long as it is within the symbol.



(b) No valid FFT for these misaligned transmitters.

Figure 4-4: FFT windows at a receiver for two transmitters. In order to decode both transmissions, the symbols from the transmitters must arrive at the receiver aligned within the slack of the CP.

ing times can be significantly different across senders. In fact, 802.11 standards [13, 56, 6] impose only very loose bounds on hardware turnaround times (10 μs in 802.11 a/g/n), and these are far longer than the 802.11 OFDM symbol time (4 μs). The second problem with increasing the CP is that the CP is overhead that is incurred for every OFDM symbol. Hence, the general trend has been to decrease the CP (for example, 802.11n negotiates down the CP if the network topology permits it [6]). Thus, even if one can exactly determine the required increase in the CP, such an approach will increase overhead and may significantly reduce, or even negate, the gains.

■ 4.4.2 Delay Measurements for Accurate Synchronization

At a high level, our synchronization algorithm is simple. One of the senders, called the **lead sender**, acquires the medium and transmits the packet. Upon hearing this signal, other nodes, which we refer to as **co-senders**, join the transmission. The choice of lead sender for a transmission depends on context and is explained in Section 4.7.

The key, however, is that transmissions from the lead sender and co-senders arrive aligned at the receiver. The challenge is that co-senders need to accumulate several samples before detecting the lead sender's transmission, and hence do not detect the transmission at the first sample. Further, different co-senders may take different times to turn

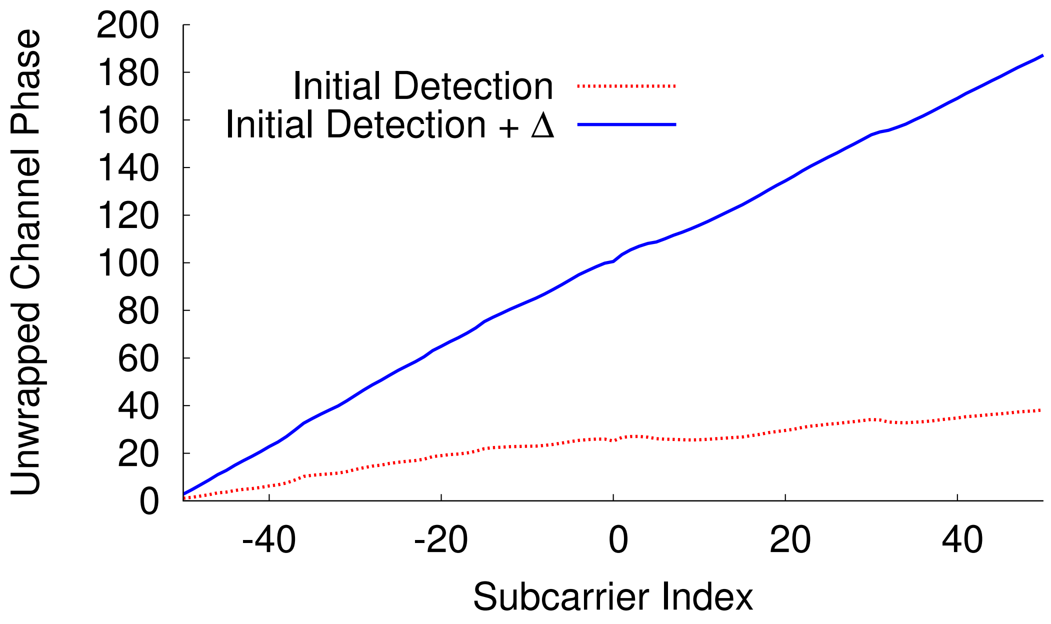


Figure 4-5: Unwrapped channel phase of OFDM subcarriers in a flat fading channel. The slope is a function of the detection delay.

around from receiving the lead sender's transmission to transmitting with the lead sender. Finally, signals from different senders traverse different paths and therefore incur different propagation delays. The co-senders therefore need to measure these different delays, and compensate for them to ensure synchronization at the receiver. In this section, we focus on how to accurately measure the delays, and describe how we compensate for the delays in the next section.

(a) Packet Detection Delay: This is the offset between the arrival of the first sample of the packet at a node, and the instant at which the receiver detects the packet. Estimating packet detection delay is a challenging task as it varies from packet to packet, and depends on the SNR, as well as the multipath characteristics of the channel.

SourceSync exploits a fundamental property of FFTs; a delay in the time domain manifests itself as a phase shift in the frequency domain [111]. To understand how we can leverage this property, let us look at the channel of an OFDM packet whose arrival the receiver detected at a few samples away from the first sample. For clarity, we discuss the case of a flat fading channel. The channel is a complex number, and we will focus on the phase of the channel in each OFDM subcarrier since that is the quantity affected by shifts in time. The dotted curve in Fig. 4-5 shows the receiver channel phases per subcarrier. As can be seen from the figure, the phases increase by a fixed slope. If we artificially induce an additional delay offset and process the packet as if it were detected Δ samples after its actual detection time, the dotted slope of the graph changes to the solid slope as shown in Fig. 4-5. Thus, a delay offset in packet detection has introduced a shift in the phase of each OFDM subcarrier proportional to the index of that subcarrier.

In fact, one can show as a direct consequence of the definition of the FFT [111] that the change in phase of subcarrier i is $\frac{2\pi i\Delta}{N_s}$, where N_s is the number of samples in a symbol. Hence, in the graph in Fig. 4-5, the induced offset Δ introduces an additional slope of [3]

$$\zeta = \frac{2\pi\Delta}{N_s} \tag{4.1}$$

Now, what would the phase slope be if the receiver detects the packet exactly at the first sample? In the case of a flat fading channel (*i.e.* coherence bandwidth larger than channel bandwidth), the different OFDM subcarriers will experience similar channels. Hence, the phase of the subcarriers at different channels will be constant, and the slope will be zero. On the other hand, if the coherence bandwidth is very small, then the different OFDM subcarriers will experience uncorrelated channels. Since the phases of these channels are equally likely to be positive or negative, the slope will be close to zero in this case too. So, how about the intermediate case where the coherence bandwidth is neither too large nor too small? We can treat this case similar to the flat fading case by computing the slope over a small window of consecutive subcarriers that spans a width smaller than the coherence bandwidth, and averaging over several such windows. In fact, we do not need to differentiate between the cases; the solution proposed for intermediate channels works for the other cases too. Hence, in SourceSync, we compute the slope over windows of consecutive OFDM subcarriers that span 3 MHz, which is less than the coherence bandwidth of indoor channels [51], and average multiple such windows to estimate the overall slope. Since the slope should be zero in the absence of detection delay, we can substitute the average slope as ζ in Eq. 4.1, and compute the detection delay offset, Δ.

[3]Note that the contribution of detection offset to channel slope is different from carrier frequency offset (CFO) and sampling offset (SFO) estimation. Specifically, the contribution of detection offset to slope is constant across symbols, unlike CFO which does not change the slope, but only shifts the intercept of the line in Fig. 4-5 from symbol to symbol, and SFO which creates a relative slope between two consecutive symbols [63].

(b) Hardware Turnaround Delay: The turnaround delay is the time required for a co-sender to switch from reception of the lead sender's transmission to transmission of its concurrent signal. This time is dependent both on the speed of the baseband pipeline and the switching time of the radio frontend from reception to transmission. The turnaround time is constant for a particular node and can be measured by locally counting the hardware clock ticks from detection of the lead sender's packet to the beginning of the co-sender's transmission.

(c) Propagation Delay: This is the time of flight of the signal between the nodes. Given a transmitter-receiver pair, one can easily obtain an estimate of the total round trip delay between the nodes by having the sender send a probe and count the number of hardware clock cycles till it gets a response from the receiver. The round trip time elapsed between the transmission of the probe and the processing of the response has multiple components as follows:[4]

$$\begin{aligned}
\text{Delay}_{\text{Probe}\rightarrow\text{Response}} &= \text{Probe Propagation Delay from Tx to Rx} \\
&+ \text{Probe Packet Detection Delay at Rx} \\
&+ \text{Hardware Turnaround Time at Rx} \\
&+ \text{Response Propagation Delay from Rx to Tx} \\
&+ \text{Response Packet Detection Delay at Tx} \qquad (4.2)
\end{aligned}$$

Both sender and receiver can estimate their packet detection delays for the probe and response packets, as well as their hardware turnaround delays as described above. The receiver includes its delay values in the response packet. The transmitter knows the total round trip delay and its own packet detection delay, and can substitute these delays, as well as the delays in the receiver response packet in Eq. 4.2 to obtain the two-way propagation delays. The one-way propagation delay is computed as half the two-way propagation delay.

[4]Eq. 4.2 assumes that the hardware turnaround delay at the transmitter is less than the sum of propagation delays and hardware turnaround delay at the receiver. Note that we can always ensure that this condition holds by adding a constant wait time at the receiver, whose value is known to the transmitter. We drop this detail from the equation for clarity.

4.4.3 Compensating for Different Delays

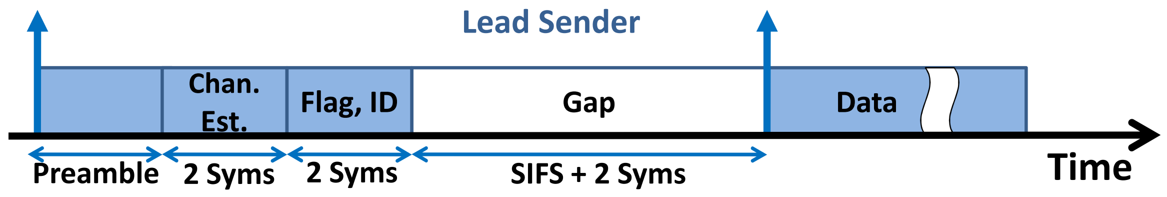
SourceSync uses its measured delays to estimate how long co-senders must wait to ensure that their transmissions arrive synchronized with the lead sender's transmission at the receiver. At a high level, the lead sender initiates transmission by sending a synchronization header. The co-senders hear the synchronization header, switch from reception to transmission, and then begin transmitting their data.

Let d_i be the one-way propagation delay from the lead sender to co-sender i, h_i the hardware turnaround delay of co-sender i, and Δ_i the detection delay for the synchronization header at co-sender i. Co-sender i will not be ready to transmit until after a delay of $d_i + \Delta_i + h_i$. Hence, the lead sender cannot transmit data immediately after the synchronization header, but has to wait for all co-senders to be ready for data transmission. What is the least time necessary to ensure that all co-senders are ready? The 802.11 specification requires that a node should be able to transmit a response within a SIFS after another node transmits a packet to it [13, 56, 6]. Hence, it is sufficient that the lead sender waits for a SIFS (10 μs in 802.11 g/n) after the synchronization header. We will refer to this time, when all co-senders are ready to transmit, as the **global time reference**. Since co-sender i is ready to transmit $d_i + \Delta_i + h_i$ units after the synchronization header, it therefore needs to wait an additional time of $SIFS - (d_i + \Delta_i + h_i)$ to align itself with the global time reference.

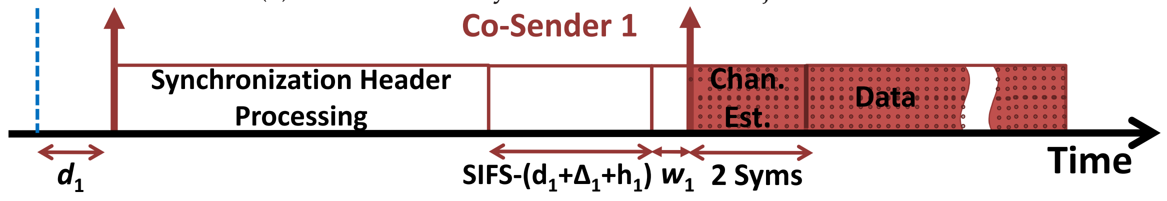
Co-senders however should not begin transmission exactly at the global time reference, since different senders have different propagation delays to the receiver. Specifically, if the co-sender is further away from the receiver than the lead sender, it needs to transmit earlier than the global time reference, and if it is closer to the receiver, it needs to transmit after the global time reference. Exactly how much before or after depends on the one-way propagation delays. Let T_0 be the one-way delay from the lead sender to the receiver, and let t_i be the one-way delay from co-sender i to the receiver. Then, co-sender i simply waits for a time of $w_i = T_0 - t_i$ relative to the global time reference to determine when it should transmit.

The above algorithm requires the co-senders to know the propagation delay from the lead sender to themselves, and the propagation delay from themselves and the lead sender to the receiver. SourceSync computes these delays by having nodes exchange periodic probes. The packet detection delay and hardware turnaround delays are both computed and compensated for locally at co-senders.

1



(a) Transmission by lead sender for the joint frame.


Co-Sender 1
Synchronization Header Processing
Chan. Est.
Data
d_1
SIFS-(d_1+Δ_1+h_1)
w_1
2 Syms
Time

(b) Transmission by co-sender for the joint frame.

Figure 4-6: Joint frame from the perspective of the senders. Symbols in solid blue are transmitted by the lead sender, symbols in dotted red by the co-sender, and symbols in white reflect silence periods. The co-sender hears the lead sender's transmission after a delay of d_1, waits for SIFS$-(d_1 + \Delta_1 + h_1)$ after processing the synchronization header, followed by a wait of w_1, and then begins its transmission.

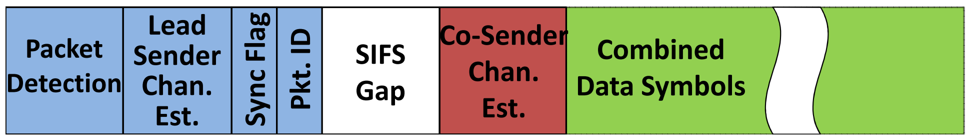


Figure 4-7: Format of joint frame seen by the receiver.

■ 4.4.4 SourceSync's Synchronization Protocol

We now describe SourceSync's synchronization protocol, assuming that all co-senders have computed their wait times. For clarity, we focus on two concurrent senders. The extension to multiple concurrent senders is straightforward.

The lead sender triggers the joint transmission by transmitting a **synchronization header**. The header contains a standard preamble for packet detection and channel estimation, followed by the lead sender identifier, a flag indicating that this is a joint frame, and a packet identifier (16-bit hash of the IP source address, IP destination address, and the IP identifier). After transmitting the synchronization header, the lead sender goes silent for a duration of SIFS to allow the co-sender to switch from reception to transmission. The lead sender stays silent for an additional duration of two symbols to allow the co-sender to transmit its channel estimation symbols, and then begins transmitting data. The co-sender, on its part, starts by listening on the medium. Once it receives the synchronization header, it continues listening till it has received the packet identifier and then switches from reception to transmission mode. The co-sender then waits for its wait time, w_1, computed as above, and transmits its channel estimation symbols, followed immediately by data. Figs. 4-6(a) and (b) show the transmission timeline of the joint frame from the perspective of the lead sender and co-sender respectively. As a result of this procedure, the receiver sees a single joint frame as shown in Fig. 4-7.

Two points are worth noting.

- SourceSync extends directly to more than two senders. In this case, after sending the synchronization header, the lead sender stays silent for the duration of a SIFS to allow all co-senders to switch, followed by two channel estimation symbols for each co-sender.

- The overhead of synchronization is low. In particular, it consists of a SIFS for switching and wait time, and 2 symbols per co-sender channel estimation. For example, in the case of 802.11 using 1460 byte packets, and 12 Mbps transmission rate, the overhead is 1.7% for two concurrent senders, and 2.8% for five concurrent senders.

■ 4.4.5 Delay Tracking and Mobility

The algorithm described so far ensures that senders can transmit synchronized with each other. But what happens when nodes move? It might seem that the changes in propagation delays resulting from node mobility will necessitate constant probe-response exchanges to recompute these delays, and maintain synchronization. However, SourceSync can deal with mobility without additional probes. Instead, it simply uses data transmissions to continuously adjust wait times at co-senders and keep transmitters synchronized.

Specifically, for each received joint frame, a SourceSync receiver detects the start of the synchronization header, and computes the channels of the lead sender and the co-sender. It then measures the slopes of both these channels, and translates the measured slopes to symbol offsets using the technique described in Section 4.4.2. If the lead sender and co-sender are perfectly synchronized, their symbol boundaries will be aligned, and therefore their computed symbol offsets will also be equal. Otherwise, the difference of the offsets corresponds exactly to the misalignment between the senders. The receiver includes the measured misalignment in its ACK, and the co-sender uses this update to appropriately change its wait time for the following transmission.

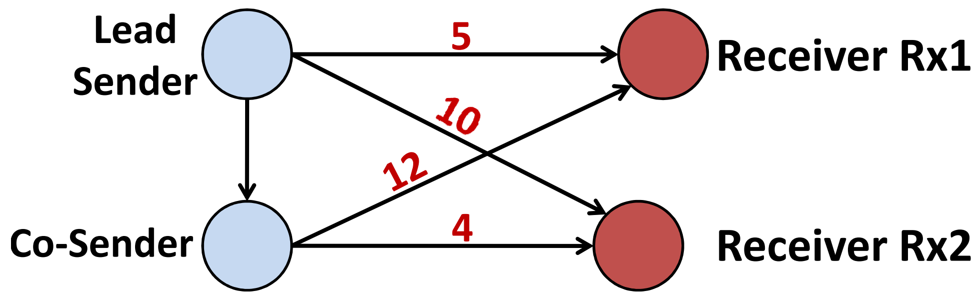


Figure 4-8: Synchronization at two receivers. One-way delays are shown. No choice of wait time allows perfect alignment at both receivers.

■ 4.4.6 Synchronization at Multiple Receivers

So far, we have focused only on synchronization at a single receiver. However, applications such as opportunistic routing would benefit from synchronization at multiple receivers.

In contrast to synchronization at a single receiver, where an appropriate choice of wait times at co-senders can achieve perfect alignment at the receiver, propagation delays may prevent us from achieving perfect synchronization simultaneously at multiple receivers. Consider the senders in Fig. 4-8 with one-way delays as shown. To synchronize at Rx1, the co-sender has to start data transmission before the lead sender. But to synchronize at Rx2, the co-sender has to start data transmission after the lead sender. Thus, it is not always feasible to synchronize senders simultaneously at multiple receivers. However, one can still leverage sender diversity gains from joint transmissions by increasing the CP to account for the residual misalignment. The objective of SourceSync in this case is to pick wait times at co-senders so as to minimize the maximum misalignment at all receivers.

SourceSync formulates this problem as a linear program that estimates the optimal wait time, w_i, for co-sender i. Define t_{ij} as the one-way delay from co-sender i to receiver j, and T_j as the one-way delay from the lead sender to receiver j. These values are estimated as in the single-receiver case described in Section 4.4.2. The pair-wise misalignment at receiver k of co-sender i with the lead sender can be written as $|(w_i + t_{ik}) - T_k|$, and similarly the pair-wise misalignment with another co-sender j can be written as $|(w_i + t_{ik}) - (w_j + t_{jk})|$. The linear program then chooses the w_i's so as to minimize the maximum pair-wise misalignment across the lead sender and all co-senders. This optimization is a linear program, and can be solved efficiently, especially since the number of co-senders and receivers is usually small, say, < 5. Note that to determine the potential receivers to synchronize at, we use the ETX metric as described in Section 4.7.2.

The lead sender performs this optimization and computes the necessary increase in CP as the maximum misalignment across all senders. In order to ensure that all senders in a joint transmission are synchronized throughout the joint frame, it communicates the new CP to co-senders as a field in the synchronization header. Co-senders use this increased CP for the concurrently transmitted data symbols.

4.5 Joint Channel Estimation

Now that senders are synchronized, the next step is to decode the joint frame at each receiver.We focus on a single OFDM subcarrier since OFDM subcarriers can be decoded independently. For simplicity of exposition, we consider two concurrent senders for the rest of this section. Our technique generalizes to multiple concurrent senders.

Say the two senders are already synchronized, and they both transmit the same symbol x_i in subcarrier i. After the FFT, the receiver receives a symbol y_i in subcarrier i, which is related to the transmitted symbol x_i as $y_i = H_i x_i + n$, where H_i is the composite channel experienced by x_i and n is noise. If the receiver knows the composite channel H_i, it can extract x_i from its received signal as $x_i = \frac{y_i}{H_i}$.

The composite channel, however, is affected by two factors. The first is the individual channels traversed by symbol x_i from each of the senders. The second factor is that each sender has a different oscillator crystal. It is unlikely that different crystals have exactly the same carrier frequency [102], and therefore, each sender has a different frequency offset with respect to the receiver. Hence, the composite channel can be written as:

$$H_i(t) = H_{i,1}e^{j2\pi\Delta f_1 t} + H_{i,2}e^{j2\pi\Delta f_2 t}$$

where $H_i(t)$ is the composite channel in subcarrier i at time t, $H_{i,j}$ are the individual channels in subcarrier i from sender j, $(j = 1, 2)$, and Δf_j is the frequency offset of sender j relative to the receiver.[5]

Since different senders have different frequency offsets, the two components of the composite channel will keep rotating relative to each other. SourceSync addresses this issue by

[5]The frequency offset is normalized in units of the subcarrier width.

leveraging the observation that the frequency offset is relatively stable over long periods of time. Therefore it can be computed at the same time as the initial pair-wise propagation delay estimation and communicated to each sender, which can then correct for the offset before transmitting by multiplying its transmitted symbol at time t by $e^{-j2\pi\Delta f_i t}$.

Once the transmitter corrects the offset, the receiver can estimate each sender's channel by using the corresponding channel estimation symbols in the joint frame. It can then add the individual channels to estimate the combined channel.

However, this is not sufficient. One can never correct completely for the frequency offset because, even if the estimate is relatively accurate, a small residual error in frequency accumulates over time leading to large phase errors and unrecoverable decoding errors throughout the packet. This is why, even for a single sender-receiver pair, OFDM decoders have to perform phase tracking to correct for residual errors in frequency offset throughout the packet. SourceSync performs phase tracking for the same reason. The difference, however, is that it has to perform independent phase tracking for each of the senders.

To do so, we augment the traditional OFDM algorithm for phase tracking. Specifically, OFDM allocates some subcarriers known as pilots in every data symbol for phase tracking. The exact algorithm for phase tracking is in [63], but the important point here is that the algorithm is designed to correct the residual frequency offset from a single sender. Hence, this algorithm cannot work as such for concurrent senders, since each sender has a different residual frequency offset. We address this issue by sharing the pilots between the concurrent senders across symbols. This is feasible since senders are synchronized and have a common understanding of symbol boundaries. For example, the lead sender can use pilot subcarriers in odd symbols, and the co-sender can use pilot subcarriers in even symbols. The receiver now maintains two residual frequency offset estimates which it applies to the individual channels of the corresponding senders before summing them to compute the composite channel.

4.6 Smart Combiner

As stated earlier, even when the senders correct for the frequency offset, there is always a residual frequency error that, over time, causes the channel from each sender to rotate relative to the other. Further, the initial phase of the channel for the two senders at the beginning of a joint frame is random. The consequence of these two behaviors is that the signals from the concurrent senders can combine constructively or destructively depending on the random initial phase and the rotation of the two channels, and the senders cannot know how the signals are going to combine *a priori*. Thus, if the two senders naively send the same signal, some unlucky symbols will observe a deeply faded channel due to destructive combining and the receiver will be unable to decode those symbols.

Let us consider a scenario where the channels from the two senders happen to cancel each other, *i.e.*, $H_{i,1} = -H_{i,2}$. In this case, if the transmitters sent the same data symbol, x_i, the receiver receives $H_{i,1}x_i + H_{i,2}x_i$, which equals 0. Of course, one way to address the problem would be for one transmitter to transmit x_i and the other to transmit $-x_i$. This transformation would transform the destructive composite channel to a channel where the two signals reinforce each other at the receiver. But such a strategy does not always work; if the channels were originally aligned with each other, sending x_i and $-x_i$ would result in a 0 signal at the receiver, transforming the constructive channel into a destructive one! Since the transmitters cannot track the individual channels and their phases ahead of transmission, they need a coding strategy that will provide high throughput irrespective of the relative orientations and magnitudes of the channels.

SourceSync addresses this issue by leveraging space time block codes [142] that cleverly code data across symbols to eliminate deep fades due to destructive combination of signals. Specifically, in the case of two senders, SourceSync uses the Alamouti code [10], which is known to provide the optimal throughput in such a scenario, and has low encoding and decoding complexity. In the case of more than two senders, SourceSync uses a quasi-orthogonal space-time block code [71] that is a simple extension of the Alamouti coding scheme, and retains its simplicity of encoding and decoding. Given a sequence of data symbols, a SourceSync lead sender uses codeword 1 from the replicated Alamouti codebook specified by [71], and co-sender i uses the $(i+1)^{th}$ codeword from this codebook. This sequence of codewords also has the property that the receiver can decode the received frame even if only a subset of intended senders participate in the concurrent transmission. Note that a receiver can determine whether an intended co-sender participates in a transmission based on the presence of energy in the time slots corresponding to the channel estimation symbols of that co-sender.

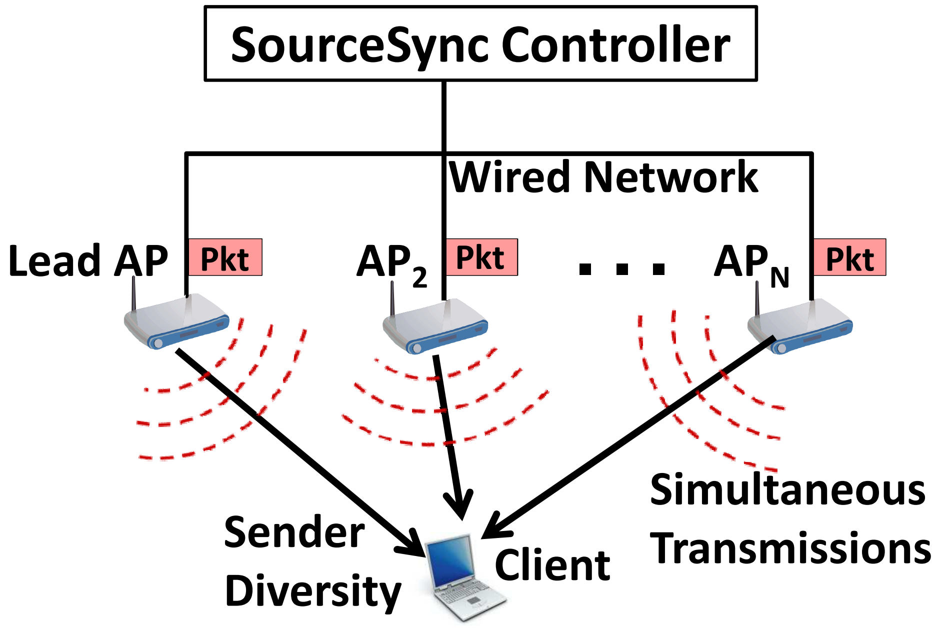


Figure 4-9: SourceSync for the last hop. SourceSync can harness sender diversity using concurrent transmissions from many APs.

■ 4.7 Using SourceSync to harness sender diversity

Now that we have described the components of SourceSync, we explain how SourceSync can be used to harness sender diversity for opportunistic routing and wireless LANs. As we do so, we also explain how SourceSync integrates with the MAC for both scenarios.

■ 4.7.1 Combining SourceSync with Last Hop Diversity

Consider a client that is in the neighborhood of multiple APs, but has poor connectivity to them. Uplink receiver diversity schemes like MRD, SOFT, and Link-Alike [106, 73, 155] exploit the fact that, while a transmitted packet has low probability of being received correctly by a specific AP, it is likely to be received by at least one AP, and all such APs can combine received packets or bits over the wired network. SourceSync complements these schemes by enabling sender diversity on the downlink, *i.e.*, instead of a client receiving packets from only one AP at a time, multiple neighboring APs can transmit simultaneously to the client and increase downlink reliability.

SourceSync exploits last-hop diversity using the architecture shown in Fig. 4-9. We leverage the high bandwidth of the wired network connecting the access points. A SourceSync controller resides on the wired network, and uses it to forward packets arriving from the wired uplink to all the APs in a neighborhood. This enables multiple APs to transmit the same data to a wireless client. Further, the APs have a static ordering that decides which codeword of the space-time block they will utilize for their transmission.

MAC and Association: When a client first joins the wireless network, it associates with multiple, say K, APs in its neighborhood, where K is a tunable parameter. One of these APs, say the one with the best link to the client, is chosen as the lead AP for this client and this information is disseminated to all other APs. All the APs estimate the propagation delays to their associated client. Additionally, the APs can offline estimate their hardware turnaround delays and propagation delays to each other. Each AP then uses this information to calculate its delay compensation, as described in Section 4.4.4.

The APs use a contention-based MAC similar to 802.11. The only difference is that when there is a downlink packet destined to a client, only the lead AP contends for the medium. Once the lead AP acquires the medium, it transmits its synchronization header followed by the data. Upon hearing the synchronization header, all other APs join the transmission as described in Section 4.4.4.

Similarly to 802.11, a client acknowledges successful receptions. Note that since the ACK is on the uplink, APs can use standard receiver diversity techniques like SOFT [155] or MRD [106] to increase the reliability of ACK reception. Received ACKs are communicated to the lead AP over the wired network. The lead AP initiates retransmissions when it does not receive an ACK, and these retransmissions are joined by the other APs, similarly to the original transmission.

Rate Adaptation: The APs coordinate rate adaptation since all simultaneously transmitted packets must have the same set of data symbols. Rate adaptation in SourceSync is controlled by the lead AP. Specifically, the lead AP runs a standard rate adaptation algorithm such as SampleRate, RRAA or SoftRate [18, 154, 150] which makes rate decisions based on the feedback from the receiver (acknowledgment, soft rate hint *etc.*). The lead AP then includes the chosen rate for the packet in the synchronization header when it initiates transmission. Other APs use this information to pick the right transmission rate. Note that, since SourceSync can leverage power and diversity gains across APs, the combined transmission across APs might be able to use a rate that cannot be used by any individual transmissions.

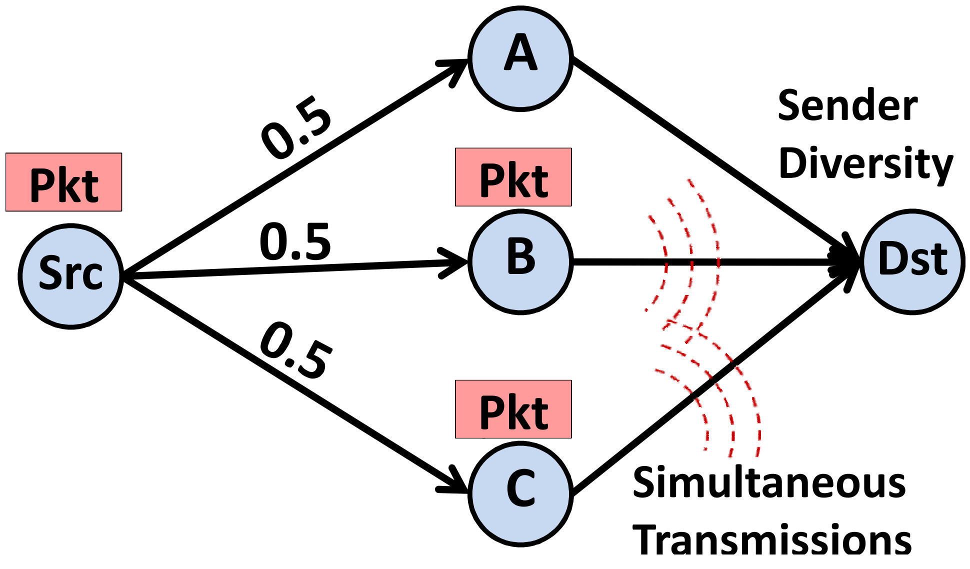


Figure 4-10: SourceSync with Opportunistic Routing. SourceSync exploits the fact that many relays hear a packet to improve throughput.

4.7.2 Combining SourceSync with Opportunistic Routing

In this section, we show how to extend opportunistic routing, particularly ExOR [19], to exploit sender diversity. Opportunistic routing has been proposed to deal with lossy links in wireless mesh networks. Consider the example in Fig. 4-10. Since all links have a loss rate of 0.5, a traditional single-path routing protocol will require an average of two transmissions to deliver a packet from the source to its nexthop router. However, when a source broadcasts its packet, the probability that at least one of these routers will receive it is $1 - (0.5)^3$, and hence the expected number of transmissions to deliver a packet is reduced to 1.14. Opportunistic routing protocols exploit this property to decrease loss rates and increase mesh throughput.

However, the same property means that, half the time, multiple routers will receive the packet from the source. Further, the probability of such an event, *i.e.*, multiple routers hearing the same packet increases with network size and density. Existing protocols cannot exploit this property. In contrast, SourceSync can leverage the fact that multiple routers in a mesh overhear the same packet to have these routers transmit the packet simultaneously towards the destination. This form of cooperative forwarding increases the effective transmission power, enabling the packet to make longer jumps towards its destination. Additionally, since the channels from the concurrent transmitters to a downstream node router are unlikely to experience simultaneous deep fading, overall loss rate is reduced.

In the rest of this section, we will describe how to integrate SourceSync with ExOR to provide an opportunistic routing protocol that exploits both sender diversity and receiver diversity. At a high level, ExOR works as follows. Given the link loss probabilities, ExOR computes the ETX metric [35] of each link, and then arranges the nodes in decreasing order of ETX distance from the destination. ExOR is designed for bulk transport. The source operates in batches, and starts by broadcasting all packets in the batch. Any node that overhears the packet can potentially forward it towards its destination. ExOR has a priority scheduler that ensures that each packet is forwarded by the node closest to the destination that has the packet. We refer the reader to [19] for the details of the scheduling algorithm.

MAC: SourceSync retains ExOR's MAC and extends it to allow simultaneous transmission from multiple forwarders. Similar to ExOR, the potential forwarders for a transmission are determined based on ETX measurements, and included in the packet header of a transmission. However, unlike ExOR, SourceSync ensures that when an ExOR forwarder transmits a packet, other nearby forwarders who happen to have overheard this packet join the transmission. This is similar to how neighboring APs join the transmission of a lead AP to provide lasthop diversity as described in Section 4.7.1. There is one key difference, however. Unlike in the last-hop scenario where AP transmissions need to be aligned at one receiver, in opportunistic routing, transmissions from multiple forwarders need to be aligned at multiple receivers. Hence, SourceSync uses the SLS described in Section 4.4 to determine both the wait compensation at the forwarders, and the minimum necessary increase in the CP to compensate for misalignment between the receivers. This computation requires forwarders to know the delay differences between various nodes in their neighborhood, and the set of concurrent forwarders and potential receivers for each transmission.

SourceSync computes the delay differences between nodes by running periodic measurements, similar to existing loss rate measurements by mesh routing protocols. SourceSync however does not need to perform delay measurements between all node pairs. A node needs to compute delay differences only to nodes that are potential co-forwarders or potential nexthops. The size of this set dictates the measurement overhead. So, in SourceSync, only nodes that are connected by links with loss probability below a threshold perform pairwise delay measurements. Further, SourceSync leverages data packets from concurrent forwarders to keep updating its estimates of delay differences as described in Section 4.4.5.

What happens when all forwarders do not hear a transmission? It is likely that not all forwarders selected during the measurement phase hear all of their intended transmissions. Exchanging information for every packet about exactly which forwarders heard that packet in order to determine the increase in CP, as well as the transmission codeword and wait time to be used by each forwarder will introduce high overhead. SourceSync eliminates the need for such exchanges by leveraging the measurement phase to pick the required wait time and additional CP assuming all forwarders hear a transmission, and also determines the ordering (and therefore codeword) of the forwarders. After this assignment, whenever the lead forwarder transmits, other forwarders hear the synchronization header, which contains the additional CP and identifier of the packet to the transmitted. If a node is in the set of co-forwarders and has the transmitted packet, it joins the transmission using the appropriate wait-time compensation. The node also knows exactly which codeword to use for its transmission based on the precomputed ordering of co-forwarders. For example, say the lead forwarder is node i, and the size of the co-forwarder set is k. The lead forwarder then uses the first codeword, node $i-1$ uses the second codeword, and so on. Of course, not all nodes in the set of potential co-forwarders might hear the packet, or the transmission of the lead forwarder. Note that this does not affect the correctness of SourceSync; a receiver can still decode the concurrent transmission, and garner the benefits of sender diversity from co-forwarders that actually join the transmission.

■ 4.8 Performance

We have implemented a prototype of SourceSync in FPGA using the WiGLAN radio platform [37] and evaluated it in a wireless testbed.

(a) Hardware: The radio board of our transceiver platform connects to the PC via the PCI bus, and acts like a regular network card. The radio operates in the 802.11a spectrum, has a maximum operating bandwidth of 128 MHz and a symbol time of 1 μs. We configure the radio to use 20 MHz of bandwidth, which is the bandwidth of 802.11 channels. The FPGA is clocked at 128 MHz, and the implementation supports standard 802.11 transmit and receive chains.

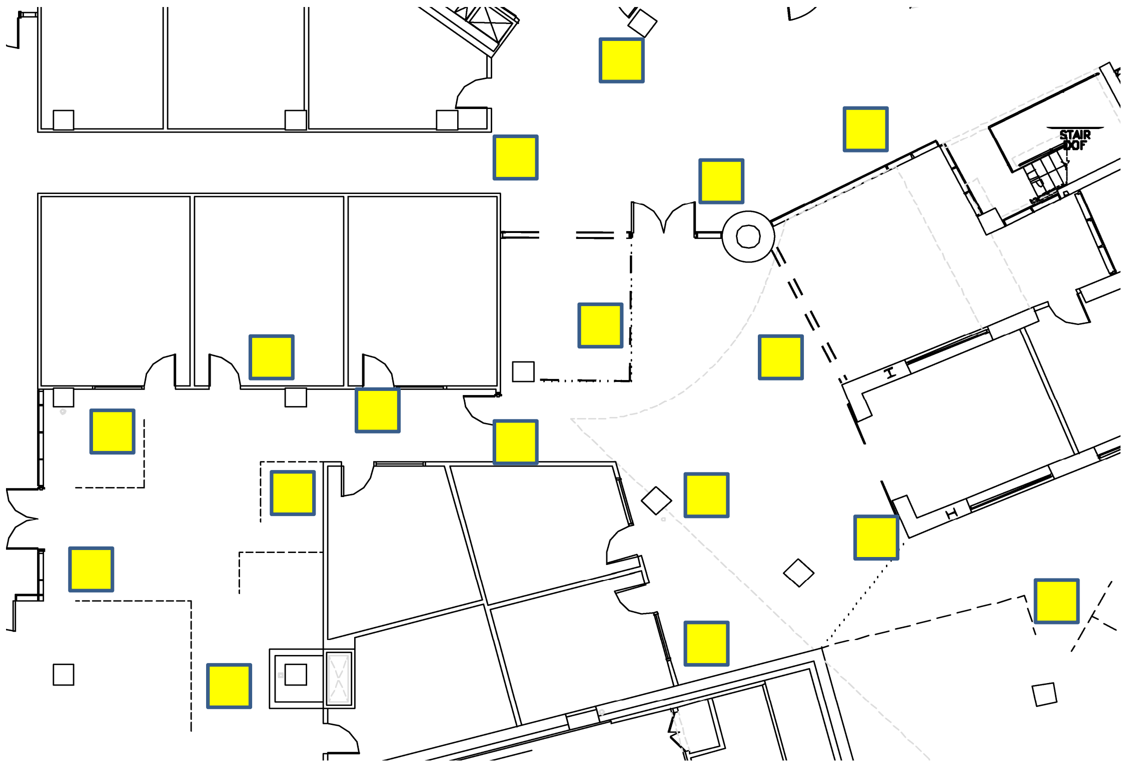


Figure 4-11: Testbed map. Node locations are highlighted.

(b) Implemented Infrastructure: We implement the components of SourceSync and an infrastructure to evaluate it for last-hop diversity and opportunistic routing. Since symbol-level synchronization requires fine-grained sample level timing, we implement SourceSync in the FPGA using Verilog and Simulink. In order to evaluate last-hop and opportunistic diversity, we also implement the following additional components:

(a) SampleRate: We implement SampleRate in our driver, using MadWifi as a reference. We modify SampleRate for SourceSync last-hop diversity to perform rate adaptation only on the lead AP.

(b) ExOR: We use the reference ExOR code and implement a simplified version for our topology, including ETX measurement, forwarder computation, and a priority scheduler.

We evaluate SourceSync in an indoor testbed. Fig. 4-11 shows the node locations in the experimental environment, which exhibits high diversity due to the presence of walls, metal cabinets, desks, and various combinations of line-of-sight and non-line-of-sight configurations. The exact evaluation methodology and topologies used for each experiment are described below.

■ 4.8.1 Symbol Level Synchronization

In this section, we show that SourceSync can provide tight symbol level synchronization across nodes, and that without such tight synchronization the system may suffer significant reduction in SNR.

SourceSync provides tight synchronization

First, we investigate whether SourceSync provides accurate symbol-level synchronization across transmitters.

Method. In this experiment, we place a pair of SourceSync nodes acting as lead sender and co-sender, and one node acting as a SourceSync receiver at three randomly chosen locations in our testbed. We synchronize the two transmitters at the receiver using SourceSync, as described in Section 4.4.4 and Section 4.4.5. Next, we want to measure the resulting synchronization error (*i.e.*, the time difference between transmitters' symbol boundaries). Recall, however, that SourceSync works by measuring synchronization errors and feeding them back to the transmitters in the ACK so they can synchronize their next transmissions, as explained in Section 4.4.5. Thus, to measure SourceSync's synchronization error, we need an algorithm that is more accurate than SourceSync in measuring synchronization errors. How do we find such an algorithm? And if such an accurate algorithm exists, why don't we use it in SourceSync?

We can obtain such a highly accurate algorithm if we incur very large overhead. Specifically, instead of computing synchronization errors using only a few symbols at the beginning of each packet, as in SourceSync, we can replace all the data in the packet with known symbols and use the full packet to compute synchronization errors. A SourceSync packet starts with an initial header consisting of the lead sender's synchronization header followed by the co-sender's channel estimation symbols, after which the two senders jointly transmit their data. The regular SourceSync algorithm obtains an estimate of the synchronization error using only the lead sender's synchronization header and the co-sender's channel estimation symbols, as described in Section 4.4.4 and Section 4.4.5. The error estimation algorithm, on the other hand, replaces the data in each packet with 200 repetitions of the initial header (*i.e.*, the lead sender's synchronization header and the co-sender channel estimation symbols). Since the synchronization error does not change within a packet, the new algorithm can obtain 200 estimates of the synchronization error for each

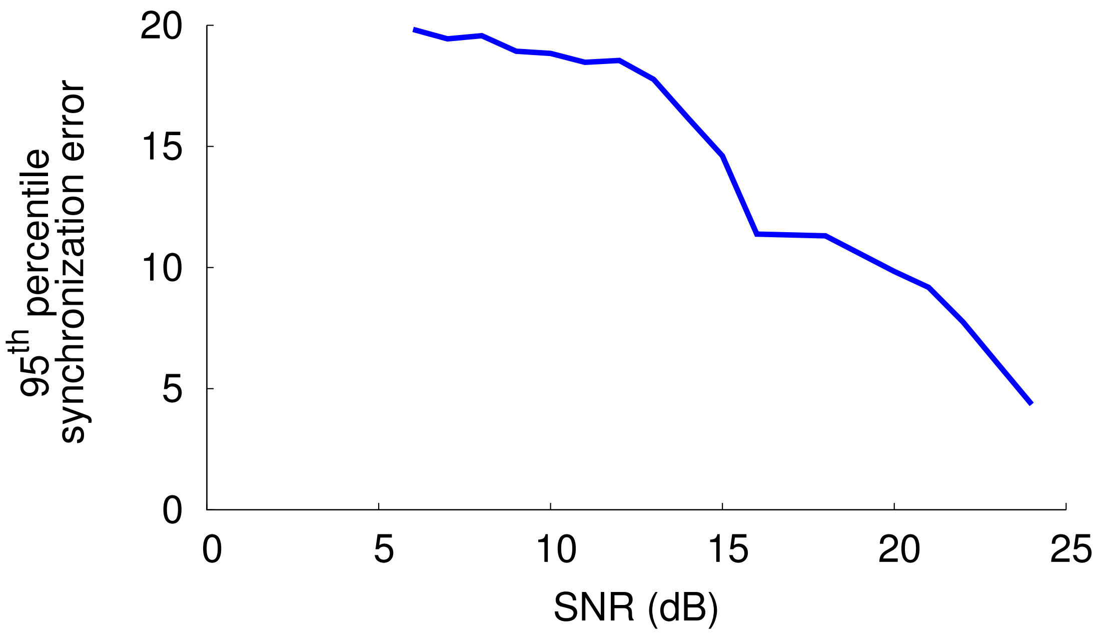


Figure 4-12: 95^{th} **percentile synchronization error.** SourceSync ensures that the synchronization error is less than 20 *ns* across the operational range of 802.11 SNRs.

estimate of SourceSync. By taking the average of these 200 estimates, the new algorithm dramatically reduces the estimation noise, and hence obtains an almost error free estimate of synchronization error for that packet. Such an algorithm is fine to evaluate the extent of synchronization error, but its overhead precludes its use in a practical system. For every set of locations, we transmit 2000 such packets and measure the average SNR from the two transmitters, as well as the transmitters' synchronization errors using both SourceSync and the new algorithm. We consider the new algorithm as the ground truth and compute SourceSync's synchronization errors with respect to the new algorithm. We repeat the experiment with multiple randomly chosen location triplets in our testbed.

Results. Fig. 4-12 shows the synchronization error between the two transmitters when using SourceSync, as a function of the average SNR. The graph shows that SourceSync's synchronization algorithm is robust across a wide range of SNRs. Specifically, the 95^{th} percentile of the synchronization error is less than 20 *ns* for the operational range of 802.11 SNRs. Thus, SourceSync's estimates can be used to perform highly accurate symbol level synchronization.

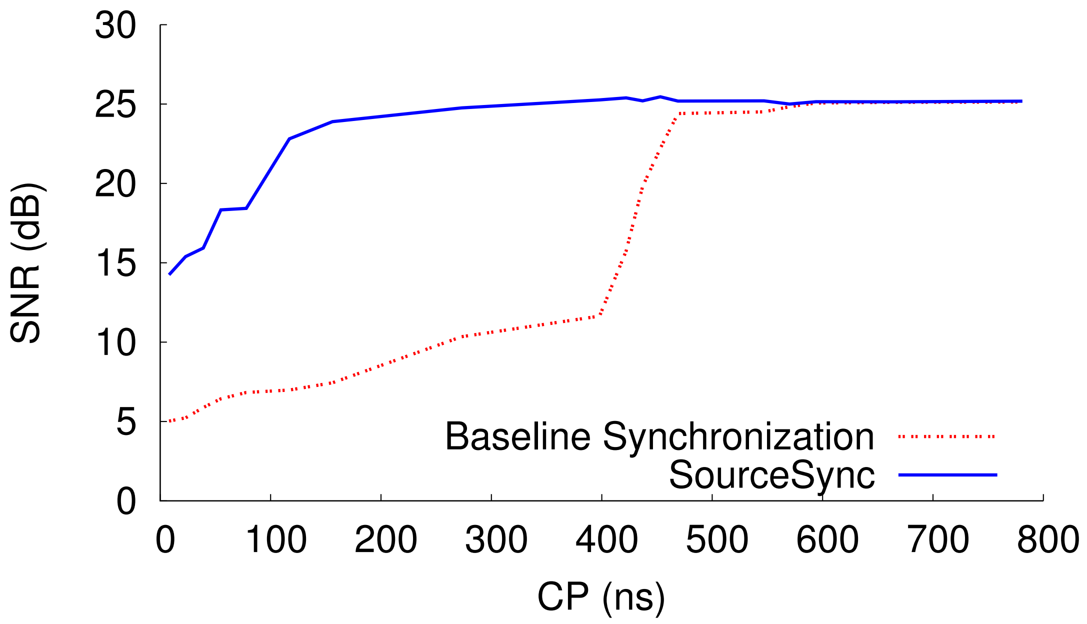


Figure 4-13: CP reduction with SourceSync. SourceSync enables concurrent transmissions to achieve high SNR with a significantly lower CP than an unsynchronized baseline that does not compensate for delay differences.

The need for accurate synchronization

SourceSync compensates for delays at senders to synchronize symbols at the receiver, and so that the joint transmission has multipath tolerance that is as good as with a single transmitter. In this section, we evaluate the consequences of loose vs. tight synchronization.

Method. We place two transmitters and the receiver in a random line-of-sight configuration in our testbed. We label one transmitter a lead sender, and the other a co-sender. Both transmitters have identical hardware, and hence the same hardware turnaround delay. The only difference in delays between the transmitters is due to propagation. We compare two schemes: a baseline scheme where the lead sender transmits a synchronization header, and the co-sender joins the transmission without compensating for delay differences, and SourceSync's symbol level synchronization scheme where the co-sender joins the transmission after an appropriate wait time as described in Section 4.4.4. For both schemes, we calculate the average receiver SNR of a joint transmission, and perform this calculation for various values of the cyclic prefix (CP).

Results. Fig. 4-13 plots the SNR of the joint transmission as a function of CP, for SourceSync, and for the baseline. We see that SourceSync requires a far lower CP to achieve

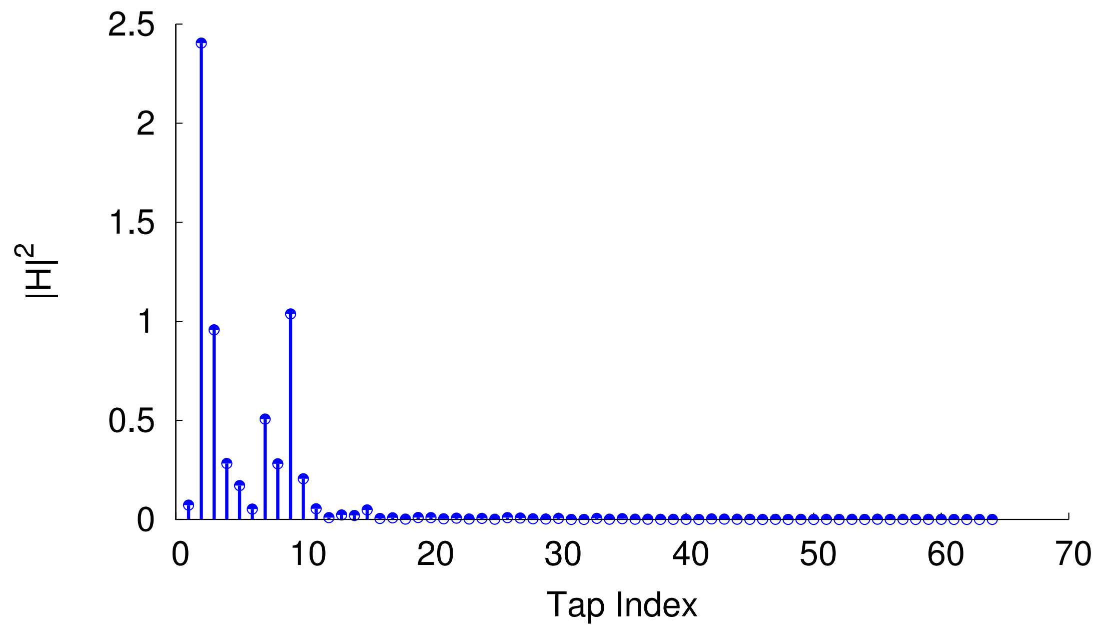


Figure 4-14: Delay spread of a single sender. The OFDM channel in the time domain has 15 significant taps, which corresponds to the CP length required with synchronization.

the peak SNR of the combined transmission, in comparison with the baseline. In particular, SourceSync requires only a CP of 117 ns (15 samples in our system) to achieve an SNR within 95% of the maximum, whereas the baseline requires a CP of 469 *ns* (60 samples in our system). Two points are worth noting. First, even when the transmitters have identical turnaround times, the baseline increases the required CP by 352 *ns* (45 samples) over what is required by SourceSync. By compensating for delay differences, SourceSync can operate with a much smaller CP, thus significantly increasing the benefits of sender diversity. Second, the baseline has no mechanism to identify the required increase in CP. Without this knowledge, one may pick a CP that is too small, in which case the communication system stops working. To prevent this scenario from occurring, one cannot simply set the CP to 469 *ns* since this value may not work for a different set of senders and receivers. One has to pick a conservative CP that works for any network, and hence incur a large overhead.

Finally, it might seem that SourceSync's SNR decreases at a CP lower than 15 samples due to residual synchronization error. However, this is not the case. The SNR reduction is due to the multipath delays in the channel. One can see this by looking at the time domain representation of the channel from one of the transmitters. Fig. 4-14 shows the magnitude of the time domain channel as a function of tap index. We see that the channel has around

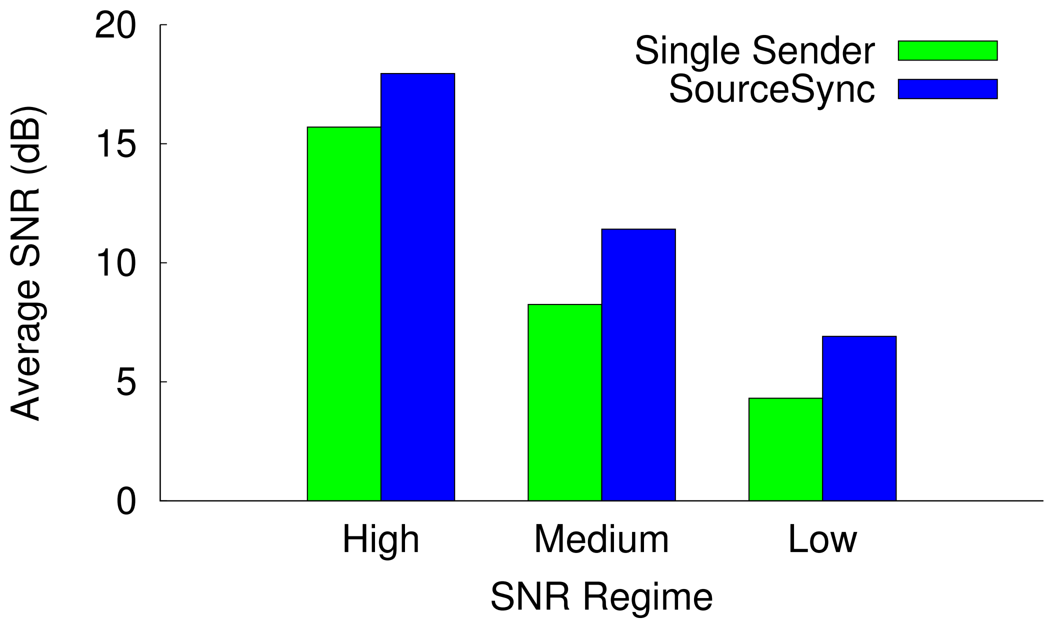


Figure 4-15: Power gains. SourceSync achieves a 2–3 dB gain over a single sender across the range of SNRs.

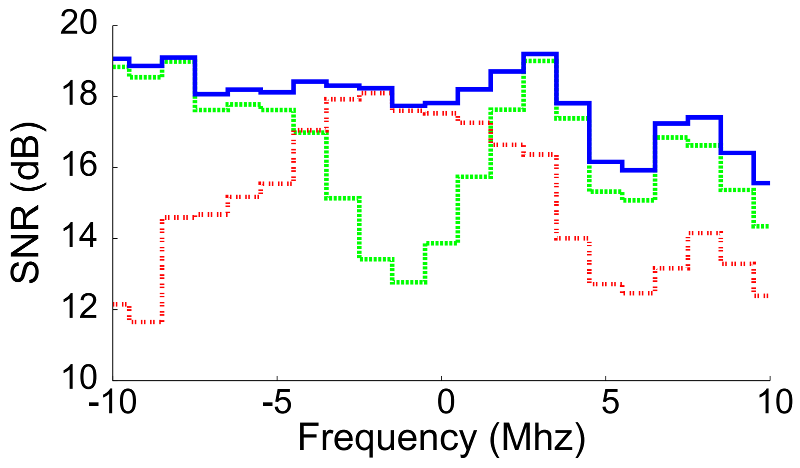
15 significant taps. Reducing the CP below 15 samples causes symbols to leak into each other, and hence reduces the maximum achievable SNR of the system.

4.8.2 Power and Diversity Gains

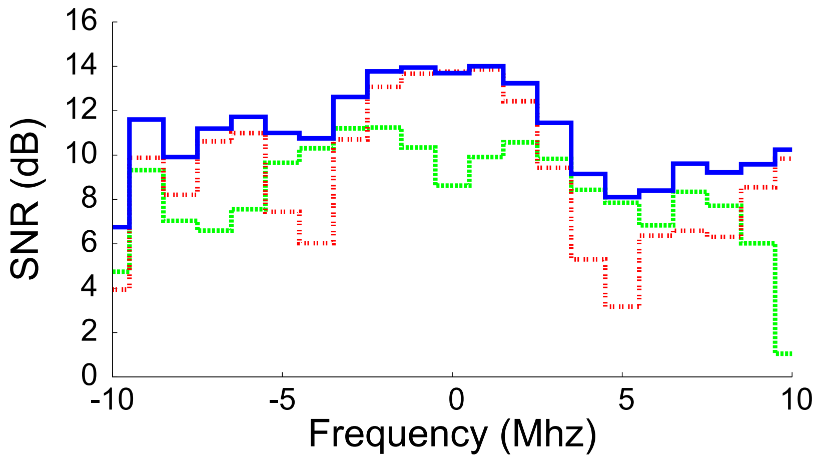
As explained earlier, allowing multiple senders to transmit simultaneously provides both power gains from the addition of the senders' powers, and frequency diversity gains because it is unlikely that the same frequency experiences a fade from all senders to the receiver. In this section, we verify that SourceSync actually provides these gains.

Method. We place the receiver and two transmitters at various random locations in our testbed. For each set of locations, we measure the average SNR across subcarriers, as well as the SNR per subcarrier when each sender transmits separately, and when the two senders transmit in combination using SourceSync. We group the locations into three categories based on the SNRs of the senders transmitting separately: low (<6dB), medium (6–12dB), and high (>12dB).

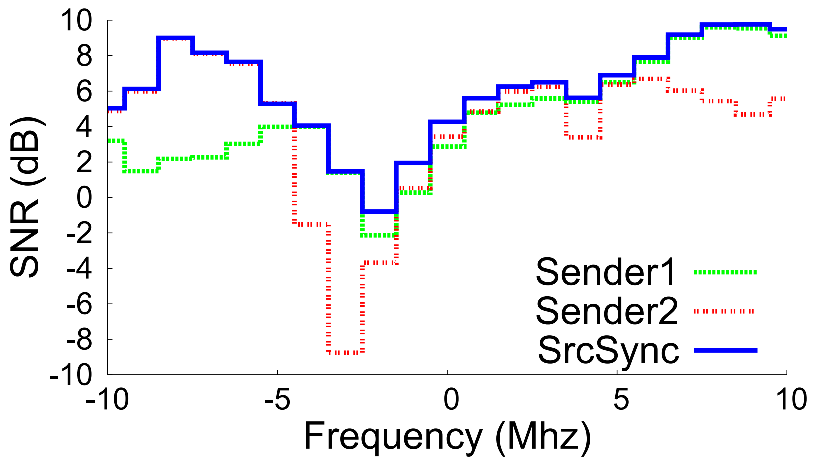
Results. Fig. 4-15 plots the average SNR across subcarriers, both for senders transmitting separately, and for joint transmission using SourceSync. As we can see, SourceSync improves the average SNR by 2–3 dB for all SNR ranges. The increase in SNR is due to the addition of power from both senders to the receiver. In particular, simultaneous transmis-



(a) High SNR



(b) Medium SNR



(c) Low SNR

Figure 4-16: Frequency diversity gains. SourceSync improves the SNR in each sub-carrier and creates a flatter SNR profile.

sion from two senders whose signals arrive at the receiver with equal power results in an SNR increase of 3 dB.

To understand the gains further, we plot the SNR per subcarrier for all three SNR ranges. We see from Figs. 4-16(a)-(c) that SourceSync not only improves the average SNR, but has a flatter SNR profile than that of either sender transmitting separately. This shows that SourceSync is able to exploit sender diversity on a per-subcarrier basis. These gains are due to SourceSync's smart combiner (Section 4.6) that uses space time block codes at a subcarrier granularity to enable signals from multiple transmitters to combine constructively. The flatter SNR profile is important in channels like 802.11, which exhibit frequency selective fading and different SNRs across subcarriers. Since it is unlikely that both senders will simultaneously experience a fade in the same subcarrier, SourceSync has a flatter SNR curve. 802.11 convolutional codes can be affected by even a few bad subcarriers, and hence, a flatter profile allows the system to achieve significantly higher bitrates with SourceSync than without SourceSync.

■ 4.8.3 Last Hop Diversity

We now examine the gains from using SourceSync in a last-hop scenario to harness sender diversity gains.

Method. We place the two transmitters, acting as APs, and the receiver, acting as a client, in random testbed locations. For each set of positions, we compute the throughput with each AP acting alone, as well as the throughput of the combined system with SourceSync, using SampleRate [18] for rate adaptation. We repeat the experiment with different sets of random locations.

Results. Fig. 4-17 shows the CDF of the throughputs obtained for each set of positions using the best AP for the client in that configuration, as well as the throughputs when leveraging diversity across APs using SourceSync. As can be seen, SourceSync provides benefits over selective diversity (*i.e.* using the single best AP) at all client throughputs, with a median throughput gain of $1.57\times$.

■ 4.8.4 Opportunistic Routing with SourceSync

We evaluate the gains of SourceSync with opportunistic routing.

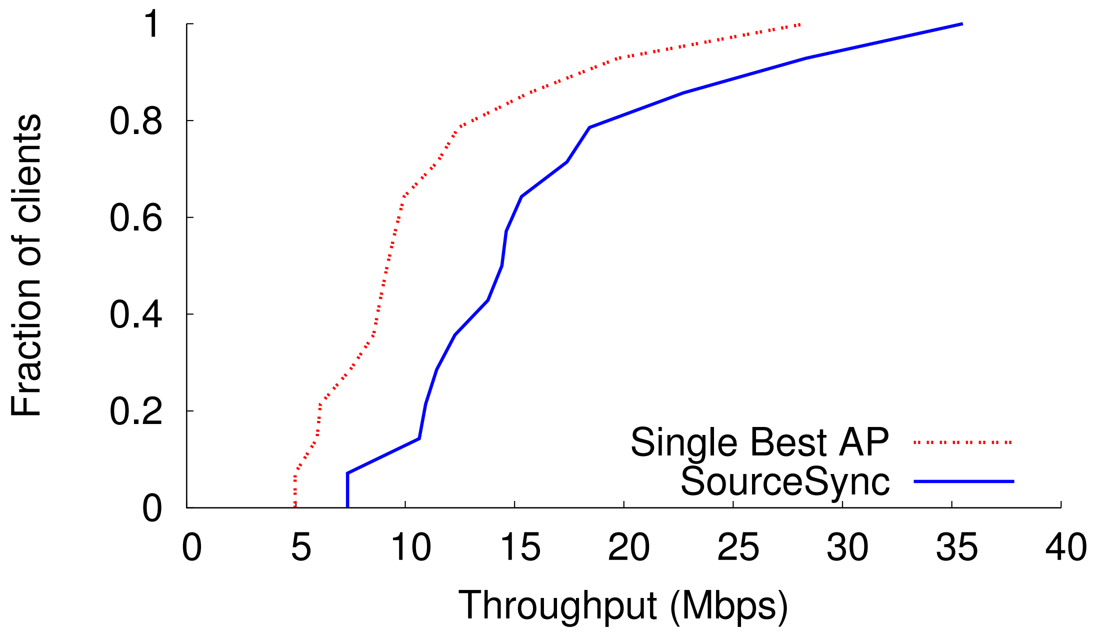
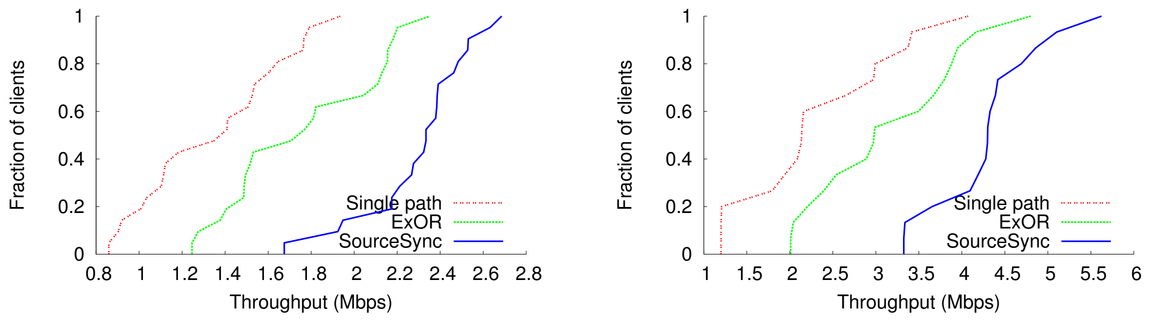


Figure 4-17: SourceSync at the last hop. The red dotted line is the CDF of throughput using selective diversity (*i.e.* single best AP). The blue solid line is the CDF of throughput using sender diversity across both APs with SourceSync. The CDFs show that sender diversity produces a median gain of 1.57× over selective diversity.



(a) Bitrate of 6 Mbps (b) Bitrate of 12 Mbps

Figure 4-18: SourceSync with opportunistic routing. SourceSync together with ExOR provides gains both over ExOR alone, and over traditional single path routing. The median gains are 1.26-1.4× over single path routing, and 1.35-1.45× over ExOR, depending on the bitrate.

Method. We create a five node topology as follows. We place two nodes, acting as source and destination, at random locations in our testbed. For each choice of source and destination, we place nodes acting as relays in three other random locations between the source and destination location. We measure pairwise loss rates between the nodes, compute the ETX metric for each link, and evaluate three schemes: (a) a single path routing scheme that picks the best relay to route the packets from source to destination, (b) ExOR, which opportunistically uses any of the three relays as forwarders, and (c) a combination of ExOR and SourceSync which also exploits sender diversity to forward from relays to the destination. Since rate adaptation for opportunistic routing protocols is still an open area, we configure the entire network to run at 6 Mbps, and at 12 Mbps, and pick the configuration that provides the highest throughput. We repeat the experiment for 20 different topologies at each rate.

Results. Figs. 4-18(a) and (b) show the CDF of the throughputs with single-path routing, ExOR, and the combination of ExOR and SourceSync. As would be expected, ExOR can harness gains from receiver diversity from the source to the relays, and provide a median throughput gain of 1.26–1.4$\times$ over single path routing. SourceSync can provide additional gains of 1.35–1.45$\times$ over the receiver diversity in ExOR by exploiting sender diversity from the relays to the destination. Further, SourceSync and ExOR work in tandem and provide a median throughput gain of 1.7–2$\times$ over single path routing.

■ 4.9 Discussion

This chapter introduces SourceSync, a distributed wireless architecture that exploits sender diversity and demonstrates its practicality via implementation and testbed evaluation. It integrates sender diversity with last-hop diversity and opportunistic routing, showing that this synergy can significantly improve throughput.

We believe that SourceSync has wider implications for wireless design than explored here. Techniques such as distributed beamforming [130] and lattice codes [109] promise significant throughput improvements in theory. However, these techniques have hitherto not been used in practice because they require some form of symbol synchronization. The synchronization mechanisms in this chapter provide a first step toward practical implementations of these techniques.

CHAPTER 5

MegaMIMO: Scaling Wireless Capacity with User Demands

As wireless devices proliferate, the key challenge faced by wireless systems is that they are unable to scale wireless throughput with the growth in user demands. This chapter presents MegaMIMO, a system that enables wireless networks to scale their capacity with the number of transmitters. MegaMIMO enables multiple independent access points (APs) to beamform their signals, and transmit different packets to different end users on the same spectrum without interfering with each other.

■ 5.1 Overview

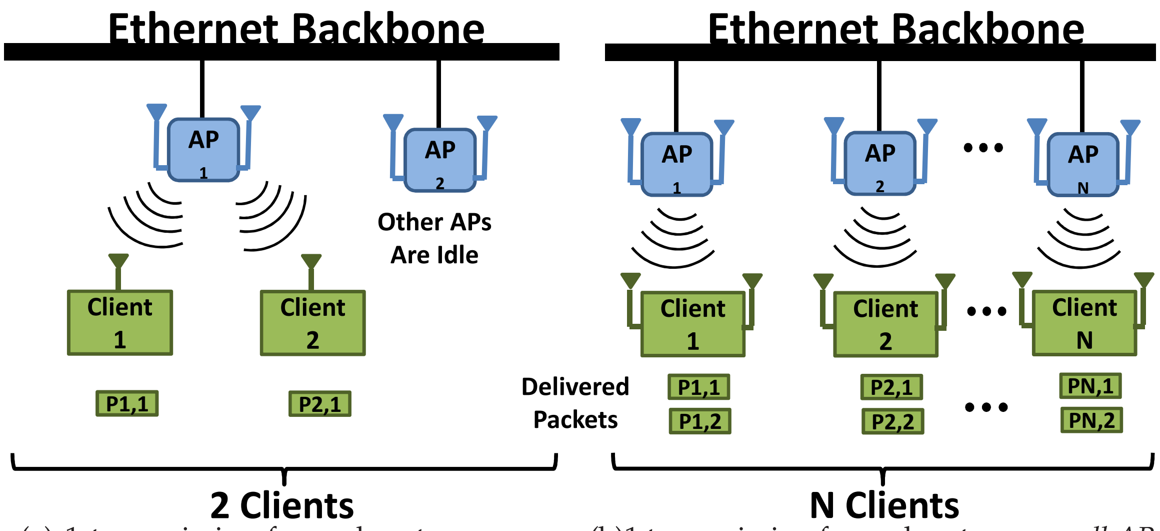
Wireless spectrum is limited; wireless demands can, however, grow unlimited. Busy Wi-Fi networks, for instance, in conference rooms, hotels, and enterprises are unable to keep up with user demands [145, 66], even causing high profile failures like the wireless network collapse during the Steve Jobs iPhone 4 keynote. Cellular networks are in a similar predicament, with their demands forecast to exceed available capacity within the next few years [126]. This is not for lack of improvement in the performance of wireless devices. Indeed, individual wireless devices have improved dramatically in recent years through innovations like the introduction of multi-antenna systems, better hardware, and lower receiver noise. The problem however is that there is a mismatch between the way user demands scale and network throughput scales; user demands scale with the number of devices in the network but network throughput does not. Unless network throughput also scales with the number of devices, wireless networks will always find it hard to keep up with their demands, and projected demands will keep exceeding projected capacity.

In this chapter, we present a system that enables a network to scale its throughput with the number of transmitting devices. We focus on the scenario of typical busy wireless environments such as multiple users in a conference room, enterprise, hotel *etc.* We enable a wireless LAN to keep increasing its total throughput by continuously adding more access points (APs) on the same channel.

The key technical idea behind our system is joint multi-user beamforming. Multi-user beamforming is a known technique that enables a MIMO transmitter to deliver multiple independent streams (*i.e.*, packets) to receivers that have fewer antennas, as shown in Fig. 5-1(a), where a 2-antenna access point delivers two packets concurrently to two single antenna receivers. In contrast, as shown in Fig. 5-1(b), MegaMIMO enables multiple access points on the same channel to deliver their packets concurrently to multiple receivers, without interfering with each other. This system scales network throughput with the number of devices, and delivers as many concurrent streams/packets as the total number of antennas on all APs. Furthermore, it can leverage the continuing performance and reliability improvements of individual devices (*e.g.*, more antennas per device).

The main challenge in implementing MegaMIMO stems from the need to synchronize the phases of distributed transmitters. Specifically, the goal of beamforming is to ensure that each client can decode its intended signal without interference. Thus, at each client, the signals intended for the other clients have to cancel each other out. This requires the transmitters to control the relative phases of their transmitted signals so that the desired cancellation can be achieved. Such a requirement is naturally satisfied in the case of a single device performing multi-user beamforming. However, in the case of MegaMIMO, the transmitters have independent oscillators, which are bound to have differences in their carrier frequencies. If one simply tries to jointly beamform these independent signals from different transmitters, the drift between their oscillators will make the signals rotate at different speeds relative to each other, causing the phases to diverge and hence preventing beamforming.

At first blush, it might seem that it would be sufficient to estimate the frequency offset (i.e., the drift) $\Delta\omega$ between the transmitters, and compensate for the beamforming



(a) 1 transmission for each antenna on a *single* AP

(b)1 transmission for each antenna on *all APs*

Figure 5-1: Traditional vs. Joint Multi-User Beamforming. In a traditional multi-user beamforming system with multiple 2 antenna APs, only 1 AP can transmit on a given channel at any given time. This leads to a maximum of 2 simultaneous packet transmissions regardless of the total number of APs. In contrast, MegaMIMO enables all APs to transmit on the same channel, allowing up to $2N$ simultaneous packet transmissions if there are N 2-antenna APs.

phase errors as $\Delta\phi = \Delta\omega t$, where t is the elapsed time. However, such an approach is not practical. It is well known [6, 139, 63] that frequency offset estimates have errors due to noise, and using such estimates to compute phases causes rapidly accumulating errors over time. Even a small error of, say, 10 Hz (4×10^{-3} ppm, which is several orders of magnitude smaller than the mandated 802.11 tolerance of 20 ppm, or cellular tolerance of 1-2 ppm), can lead to a large error of 20 degrees (0.35 radians) within a short time interval of 5.5 ms. Such a large error in the phase of the beamformed signals will cause significant interference at the receivers, preventing them from decoding.

MegaMIMO presents a simple and practical approach for synchronizing the phases of multiple distributed transmitters. The key idea underlying MegaMIMO is to elect one of the APs as a lead and use its phase as a reference for the whole system. Other APs (i.e., the slaves) directly measure the phase of the lead AP and change the phase of their signals to maintain a desired alignment with respect to the lead. In particular, MegaMIMO precedes every data packet with a couple of symbols transmitted by the lead AP. The slave APs use these symbols to directly measure the required phase correction for proper beamforming. Since this is a direct phase measurement as opposed to a prediction based on frequency offsets, it has no accumulated errors. After correcting for this phase error, the slave APs use the estimate for their frequency offset to predict any phase changes throughout the packet and correct for it. This bounds the maximum phase error accumulation to the duration of a packet. One can use a simple long term average for the frequency offset to ensure that the phase error accumulated for the duration of a packet is within the desired performance bounds.

In the rest of the chapter, we expand on this basic idea and demonstrate that it can deliver accurate joint beamforming across distributed transmitters. Further, we also extend this idea to work with off-the-shelf 802.11 cards. This would allow organizations to directly leverage MegaMIMO by simply upgrading their AP infrastructure, without requiring any modification to the clients.

We implemented MegaMIMO in two environments:

- The first environment consists of USRP2 APs and receivers, where both APs and clients can be modified. We use this environment to verify the scaling properties of MegaMIMO, and also to perform finer grained analysis of the individual components of MegaMIMO.

- The second environment consists of USRP2 APs and receivers with Intel Wi-Fi Link 5300 adapters. Each AP in this second testbed consists of two USRP2s connected via an external clock and configured to act as a 2-antenna MIMO AP. Correspondingly, each receiver Wi-Fi card has 2 antennas enabled. We use this testbed to verify that MegaMIMO can provide throughput gains with off-the-shelf 802.11n cards, and further, that MegaMIMO can provide these gains with multi-antenna devices.

We evaluated MegaMIMO in an indoor testbed using APs and receivers deployed densely in a room to simulate a conference room scenario. Our results reveal the following findings:

- **USRP testbed:** MegaMIMO's throughput increases linearly with the number of APs. In particular, in our testbed, which has 10 APs, MegaMIMO can achieve a median throughput gain of $8.1 - 9.4\times$ over traditional 802.11 unicast, across the range of 802.11 SNRs.

- **802.11 testbed:** MegaMIMO's ability to linearly scale the network throughput with the number of transmitters applies to off-the-shelf 802.11 clients. Specifically, MegaMIMO can transmit simultaneously from two 2-antenna APs to two 2-antenna 802.11n clients to deliver a median throughput gain of $1.8\times$ compared to traditional 802.11n.

- **Phase Synchronization:** MegaMIMO's distributed phase synchronization algorithm is accurate. The 95^{th} percentile misalignment between APs observed at the receiver is less than 0.05 radians. Further, for the whole range of operational SNRs of 802.11 (5-25 dB), the reduction in SNR at each client due to misalignment, (*i.e.*, the total power of interference from all signals not intended for this client to the noise floor) increases on average by 0.13 dB for every additional AP-client pair.

Contributions: This work presents the first system that scales wireless throughput by enabling joint beamforming from distributed independent transmitters. To achieve this, we design a simple and practical approach for performing phase synchronization across multiple distributed transmitters. Finally, we also show that our system can deliver throughput gains from joint beamforming with off-the-shelf 802.11n cards.

5.2 Related Work

(a) Empirical systems: Recent years have seen a few systems that tried to capture the gains promised by distributed multi-user beamforming [47, 123, 46, 110]. These systems, however, do not address phase synchronization, which is a basic problem in achieving such a system. In particular, they either require the base stations to be tightly synchronized with a Global Positioning System (GPS) clock[1], or assume that all the transmit antennas are driven by a single oscillator [46], or even assume that the receivers can jointly decode the data by exchanging all the received signals [110]. The closest to our work is [74], which addresses phase synchronization, but does not perform distributed joint transmission and achieves large errors (around 20 degrees) that cannot support distributed MIMO. In contrast, MegaMIMO provides the first system that achieves phase synchronization using independent oscillators at the devices in the network. As a result, MegaMIMO can enable devices to operate independently without having to share a common clock or use

[1]While promising, GPS typically does not work indoors, rendering such a GPS-based system hard to use in practice.

external clocks such as GPS. Finally, since MegaMIMO does not require any modifications to existing hardware, it can work with off-the-shelf 802.11n cards.

MegaMIMO is related to work on enabling concurrent transmissions across different nodes in the network like MU-MIMO in LTE and WiMAX [103, 92], SAM [140], IAC [54], multi-user beamforming [12] , and n+ [90]. However, these systems do not scale with the number of devices in the network. In particular, the throughput of these systems is limited either by the number of antennas on a single AP [103, 92, 140, 12], or the maximum number of antennas on any device in the network [90], or twice the number of antennas on any device in the network [54]. In contrast, MegaMIMO is the first system that enables the number of concurrent transmissions to scale with the number of APs, independent of the number of antennas on a single device. This allows MegaMIMO to support multiple independent APs communicating simultaneously with multiple independent clients.

MegaMIMO is also related to work on harnessing channel diversity gains such as distributed antenna systems [94, 34], and SourceSync [120], as well as work on phased arrays [55], which provide directional gain by sending the same signal on different antennas with different, carefully calibrated delays. However, these systems can not provide multiplexing benefits and hence, unlike MegaMIMO, cannot scale network throughput with the number of APs. Finally, recent work has shown how to synchronize concurrent transmissions in time and frequency [139, 120]. MegaMIMO builds on these results to deliver a distributed MIMO system. However, time and frequency synchronization alone are not sufficient, since joint multi-user beamforming intrinsically depends on the ability of the distributed APs to achieve phase synchronization, without which it is impossible to allow independent clients to decode simultaneously.

(b) Theoretical results: There is some theoretical work [143, 16] that addresses distributed phase synchronization, but assumes frequency synchronous oscillators and only provides one-time phase offset calibration. Further, the promise of distributed MIMO to improve the scalability of wireless networks has been explored in the theoretical community [7, 135, 148]. Work by Ozgur *et al.* [113] theoretically proved that such a setup can scale wireless capacity with the number of nodes. While MegaMIMO builds on this foundational work, MegaMIMO is the first empirical system that shows that linear scaling of throughput with the number of transmitters is possible in practical systems with unsynchronized oscillators and resulting time-varying phase differences.

5.3 MegaMIMO

MegaMIMO is designed for the wireless downlink channel. It is applicable to wireless LANs, especially in dense deployments like enterprises, hotels, and conference rooms. MegaMIMO APs can operate with off-the-shelf WiFi client hardware. The techniques in MegaMIMO are also applicable to cellular networks, but the potential of integrating them with off-the-shelf cellular clients and evaluating them in the cellular context are beyond the scope of this chapter.

MegaMIMO APs are connected by a high throughput backend, say, GigE, like APs are today. Packets intended for receivers are distributed to all APs over the shared backend. MegaMIMO enables the APs to transmit concurrently to multiple clients as if they were one large MIMO node, potentially delivering as many streams (*i.e.*, packets) as the total number of antennas on all APs.

In the next few sections, we describe how MegaMIMO works. We start with the basic idea that enables distributed phase synchronization. We then describe our protocol implementing this basic idea for emulating a large MIMO node. We then extend our system to integrate our design with off-the-shelf WiFi cards.

5.4 Distributed Phase Synchronization

The chief goal of distributed phase synchronization is to enable different transmitters powered by different oscillators to emulate a single multi-antenna transmitter where all antennas are driven by the same oscillator. Intuitively our solution is simple: We declare one transmitter the lead, and make all other transmitters synchronize to the oscillator of the lead transmitter, *i.e.*, each transmitter measures the offset between its oscillator and the lead oscillator and compensates for the offset by appropriately correcting the phase of its transmitted signal. This behavior makes all transmitters act as if they were antennas on the same chip controlled by the same oscillator.

We now demonstrate how this intuitive design can deliver the proper MIMO behavior and hence enable each receiver to correctly decode its intended signal without interference. For simplicity, we consider a scenario of 2 single-antenna APs transmitting to 2 single-antenna clients, as shown in Fig. 5-2. Let h_{ij}, where, $i, j \in \{1, 2\}$ be the channel to client i from AP j, $x_j(t)$ the symbol that needs to be delivered to client j at time t, and $y_j(t)$ the

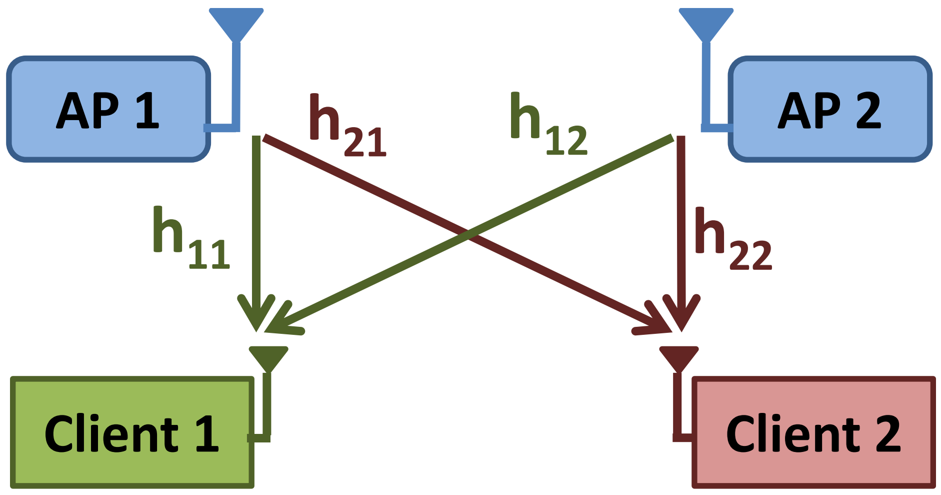


Figure 5-2: Channel matrix with 2 APs transmitting to 2 clients.

symbol that is received by client j at time t. Correspondingly, let $\mathbf{H} = [h_{ij}], i, j \in \{1, 2\}$ be the 2x2 channel matrix, $x(\vec{t}) = [x_1(t)\ x_2(t)]^T$ be the desired symbol vector, and $y(\vec{t}) = [y_1(t)\ y_2(t)]^T$ be the received symbol vector.

No Oscillator Offset: Assume first that there are no oscillator offsets between any APs and clients. If each AP i simply transmits the signal $x_i(t)$, each client will receive a linear combination of the transmitted signals. Since each client has only one antenna, client 1 receives $y_1(t) = h_{11}x_1(t) + h_{12}x_2(t)$ and client 2 receives $y_2(t) = h_{21}x_1(t) + h_{22}x_2(t)$. Each of these equations has two unknowns, and hence, neither client can decode its intended data.

In order to deliver two concurrent packets to the two clients, the APs need to ensure that each client receives only the signal intended for it (*i.e.*, it experiences no interference from the signal intended for the other client). Specifically, we need the effective channel experienced by the transmitted signal to be diagonal, *i.e.*,, it should satisfy:

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} g_{11} & 0 \\ 0 & g_{22} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}, \tag{5.1}$$

where g_{11} and g_{22} are any non-zero complex numbers. In this case, the received signal will simply appear at each receiver as if it has experienced the channel g_{ii}, which each receiver can estimate using standard techniques.

The APs can achieve this result by using *beamforming*. In beamforming, the APs measure all the channel coefficients from the transmitters to the receivers at time 0. Then, instead of transmitting $x_1(t)$ and $x_2(t)$ directly, the APs transmit:[2]

$$\begin{pmatrix} s_1(t) \\ s_2(t) \end{pmatrix} = \mathbf{H^{-1}} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \tag{5.2}$$

In this case, the two clients receive:

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \mathbf{H} \begin{pmatrix} s_1(t) \\ s_2(t) \end{pmatrix} = \mathbf{HH^{-1}} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

Since $\mathbf{HH^{-1}} = \mathbf{I}$, the effective channel experienced by the clients in this case is a diagonal matrix, *i.e.*, Eq. 5.1 is satisfied. Hence, each client can now decode its intended data without interference from the signal intended for the other client.

With Oscillator Offset: What happens when the oscillators of the APs and clients have different frequencies? Let ω_{Ti} be the oscillator frequency of AP i, and ω_{Rj} the oscillator frequency of client j, $i, j \in \{1, 2\}$. In this case, the channel at time t, $\mathbf{H(t)}$, can be written as:

$$\mathbf{H(t)} = \begin{pmatrix} h_{11}e^{j(\omega_{T1}-\omega_{R1})t} & h_{12}e^{j(\omega_{T2}-\omega_{R1})t} \\ h_{21}e^{j(\omega_{T1}-\omega_{R2})t} & h_{22}e^{j(\omega_{T2}-\omega_{R2})t} \end{pmatrix},$$

where $j = \sqrt{-1}$. Because the oscillators rotate with respect to each other, the channel no longer has a fixed phase.

Now, if the APs try to perform beamforming as before, using the channel value they computed at time $t = 0$ and transmitting $\mathbf{H^{-1}}\vec{x}$, the clients receive:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{H(t)H^{-1}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

The product $\mathbf{H(t)H^{-1}}$ is no longer diagonal, and hence the receivers cannot decode their intended signal. Thus, standard MIMO beamforming does not work in this case.

So how can one do beamforming with such a time varying channel? A naive approach would try to make each transmitter compute $\mathbf{H(t)}$ at every t and then multiply its time signal by $\mathbf{H(t)^{-1}}$. Say that the network has N APs and N clients. Then such an approach would require each transmitter to maintain accurate estimates of N^2 frequency offsets of

[2]The APs also need to normalize $\mathbf{H^{-1}}$ to respect power constraints, but we omit that detail for simplicity.

the form $\Delta\omega_{ij} = \omega_{Tj} - \omega_{Ri}$. (Further since nodes can only measure frequency offsets relative to other nodes, but not the absolute frequencies of their oscillators, the number of estimates cannot be reduced to N.) Measurement errors from all of these estimates will accumulate, prevent accuracy of beamforming, and create interference at the receivers. However, according to our intuition at the beginning of this section, we can make multiple transmitters act as if they were one big MIMO node, and hence do accurate beamforming, by having each transmitter estimate only its frequency offset to the lead transmitter. Said differently, our intuition tells us that it should be possible to reduce the number of frequency offset estimates that each transmitter maintains from N^2 to one. Let us see how we can achieve this goal.

Observe that we can decompose the channel matrix at time t as $\mathbf{H(t)} = \mathbf{R(t)HT(t)}$, where $\mathbf{H}$ is time invariant, and $\mathbf{R(t)}$ and $\mathbf{T(t)}$ are diagonal matrices defined as:

$$\mathbf{R(t)} = \begin{pmatrix} e^{-j\omega_{R1}t} & 0 \\ 0 & e^{-j\omega_{R2}t} \end{pmatrix}$$

and

$$\mathbf{T(t)} = \begin{pmatrix} e^{j\omega_{T1}t} & 0 \\ 0 & e^{j\omega_{T2}t} \end{pmatrix}$$

Since $\mathbf{R(t)}$ is diagonal, it can function analogous to the $\mathbf{G}$ matrix in Eq. 5.1. Thus, if the transmitters transmit the modified signal $\mathbf{T(t)^{-1}H^{-1}}\vec{x}$ at time t, then the received signal can be written as:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{R(t)HT(t)T(t)^{-1}H^{-1}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{5.3}$$

which reduces to the desired form of Eq. 5.1:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{R(t)} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Note that $\mathbf{T(t)}$ is also diagonal, and as a result the transmitter phase correction matrix

$$\mathbf{T(t)^{-1}} = \begin{pmatrix} e^{-j\omega_{T1}t} & 0 \\ 0 & e^{-j\omega_{T2}t} \end{pmatrix}$$

is also diagonal. Further, the phase correction entry for each AP depends only on the oscillator phase of that AP. This means that if each AP, i, knows its phase, $e^{j\omega_{Ti}t}$, at time t, it can simply compensate for that phase and the AP will not need any additional frequency or phase measurements. Unfortunately, this is not practical. An AP has no way to measure the exact phase change of its oscillator locally.

We address this difficulty by observing that the channel equation is unchanged when we multiply by $1 = e^{j\omega_{T1}t}e^{-j\omega_{T1}t}$, *i.e*,

$$\begin{aligned}\mathbf{H}(\mathbf{t}) &= e^{j\omega_{T1}t}\mathbf{R}(\mathbf{t})\mathbf{H}\mathbf{T}(\mathbf{t})e^{-j\omega_{T1}t} \\ &= \begin{pmatrix} e^{j(\omega_{T1}-\omega_{R1})t} & 0 \\ 0 & e^{j(\omega_{T1}-\omega_{R2})t} \end{pmatrix} \mathbf{H} \begin{pmatrix} 1 & 0 \\ 0 & e^{(j(\omega_{T2}-\omega_{T1})t} \end{pmatrix}\end{aligned}$$

Since the new observed channel matrix is still diagonal, the clients can still continue to decode the received signal as before.

The resulting system implements our intuition at the beginning of this section.

5.5 MegaMIMO Protocol

We start by describing the protocol at a high level, and follow by the detailed explanation. MegaMIMO's distributed transmission protocol works in two phases:

- MegaMIMO starts with a **channel measurement phase**, in which the APs measure two types of channels:

  1) the channels from themselves to the receivers (i.e., the matrix **H**), which is the beamforming channel whose inverse the APs use to transmit data concurrently to their clients; and

  2) the channels from the lead AP to the slave APs (the h_i^{lead}'s), which enables each slave AP to determine its relative oscillator offset from the lead AP.

- The channel measurement phase is followed by the **data transmission phase**. In this phase, the APs transmit jointly to deliver concurrent packets to multiple receivers. Data transmission uses beamforming after having each slave AP corrects for its frequency offset with respect to the lead AP.

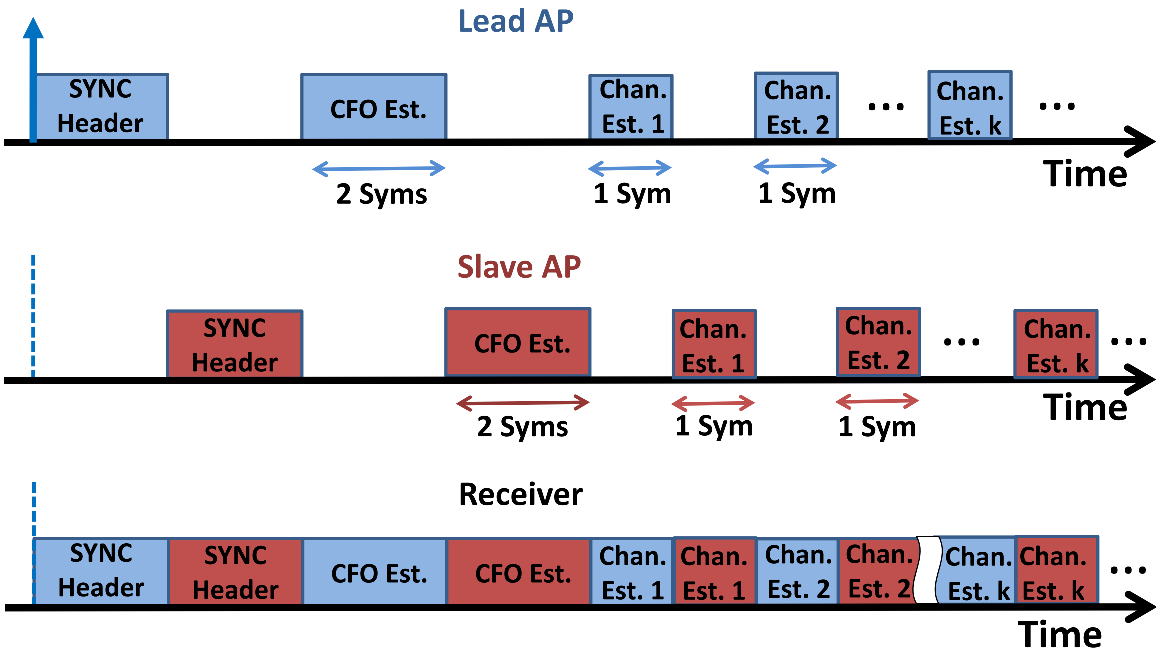


Figure 5-3: Packet Structure from the perspective of APs and the receiver. Symbols in blue are transmitted by the lead AP, symbols in red by the slave AP, and symbols in white reflect silence periods.

Note that a single channel measurement phase can be followed by multiple data transmissions. Channels only need to be recomputed on the order of the coherence time of the channel, which is several hundreds of milliseconds in typical indoor scenarios [52]. Section 5.7 describes how MegaMIMO reduces channel measurement overhead in greater detail.

We now describe the channel measurement and data transmission phases in greater detail. (The description below assumes symbol level time synchronization, for which we use the scheme in [120], which provides tight synchronization up to a few nanoseconds. Our experimental results also incorporate an implementation of that scheme).

5.5.1 Channel Measurement

The goal of channel measurement is to obtain a snapshot of the channels from all APs to all clients, i.e., **H** and the reference channels from the lead AP to the slave APs, i.e., the $h_i^{lead}, \forall i$.

The key point is that *all these channels have to be measured at the same time*, which is the reference time $t = 0$. Otherwise the channels would rotate with respect to each other due to frequency offsets and hence be inconsistent. Below, we divide channel measurement into a few sub-procedures.

(a) Collecting Measurements. The lead AP starts the channel measurement phase with a synchronization header, followed by channel measurement symbols, *i.e.*, known OFDM symbols that the clients can use to estimate the channel. The channel measurement symbols are separated by a constant gap, whose value is chosen to permit the slave APs to send their channel measurement symbols interleaved with the symbols from the lead AP. When the slave APs hear the synchronization header, they know to transmit their channel measurement symbols in the gap, one after another, as shown in Fig. 5-3.

Thus, channel measurement symbols are repeated and interleaved. They are repeated to enable the clients to obtain accurate channel measurements by averaging multiple estimates to reduce the impact of noise. They are interleaved because we want the channels to be measured as if they were measured at the same time. Since exactly simultaneous transmissions will lead the APs to interfere with each other, MegaMIMO performs a close approximation to simultaneous transmission by interleaving symbols from different APs.

(b) Estimating H at the clients. Upon reception of the packet in Fig. 5-3, each client performs three tasks: it computes its carrier frequency offset (CFO) to each AP; it then uses its knowledge of the transmitted symbols and the CFO to compute the channel from each AP to itself; and finally it uses its knowledge of the CFOs to rotate the phase of the channels so that they look as if they were measured exactly at the same time. We detail these tasks below.

Different transmitters (*i.e.*, APs) have different oscillator offsets to receivers, and each receiver needs to measure the frequency offset from each transmitter to correct the corresponding symbols from that transmitter appropriately. To enable this, the channel measurement transmission uses CFO symbols from each AP followed by channel estimation symbols similar to traditional OFDM [63]. The only departure is that the receiver computes and uses different CFO and channel estimates for symbols corresponding to different APs.

Note that these channel estimates are still not completely simultaneous, in particular, the channel estimation symbols of slave AP i is separated from the symbol of the lead AP by $i-1$ symbol widths, as shown in Fig. 5-3. The receiver compensates for this by rotating the estimated channel for AP i by $e^{-j\Delta\omega_i(i-1)kT+D}$ (in each OFDM subcarrier), where T is the duration of one OFDM symbol, k is the index of the interleaved symbol, and D is the duration of the lead AP synchronization header. This ensures that all channels are measured at one reference time, which is the start of the synchronization header. The receiver averages the channel estimates (in each OFDM subcarrier) from each AP to cancel out the noise and obtain an accurate estimate. The receivers then communicate these estimated channels back to the transmitters over the wireless channel.

(c) Estimating the h_i^{lead}'s at the Slave APs. Each slave AP uses the synchronization header to compute the value of the channel from the lead AP to itself at the reference time $h_i^{lead}(0)$.

Note that at the end of the channel measurement phase, each slave AP i has the entire channel matrix to be used for beamforming, as well as a reference channel, $h_i^{lead}(0)$ from the lead AP which it will use during data transmissions, with all channels measured with respect to one reference time.

5.5.2 Data Transmission

Now that the channels are measured, the APs can use beamforming to transmit data concurrently without interference.

(a) AP Coordination: The APs need to agree on which packets are sent concurrently in one beamforming frame. To do this we leverage the bandwidth of the backend Gigabit Ethernet to send all client packets to all APs. The lead AP makes all control decisions and communicate them to the slave APs over the Ethernet. In particular, it determines which packets will be combined in a data transmission and communicates it to the slave APs over the wired backend.

(b) Beamforming: Client packets are transmitted by joint beamforming from the MegaMIMO APs participating in the system. Note that slave APs need to correct the phase of their signal prior to transmission. One way to do this would be for each slave to estimate the frequency offset $\omega_{lead} - \omega_{slave}$ from the lead to itself (using the synchronization header from the previous phase) and then compute the net elapsed phase by calculating $(\omega_{lead} - \omega_{slave})t$, where t is the time elapsed since the channel measurement was taken. However, this would lead to large accumulated errors over time because of any inaccuracies in the measurement of the initial frequency offset. For example, even a small error of 100 Hz in the measurement of the initial frequency offset can lead to a large phase error of π radians in as short a timespan as 20 ms, and hence significantly affect the phase alignment required for correct beamforming. Unless addressed, this error would prevent MegaMIMO from amortizing the cost of a single channel measurement over the coherence time of the channel, e.g., 250 ms, and would force the system to repeat the process of measuring **H** every few milliseconds, which means incurring the overhead of communicating the channels from all clients to the APs almost every packet.

MegaMIMO avoids this issue of accumulating error over large timescales by directly measuring the phase difference between the lead AP and the slave AP. Said differently instead of multiplying the frequency offset $\Delta\omega(= \omega_{lead} - \omega_{slave})$ by the elapsed time (which leads to errors that accumulate over time), MegaMIMO directly measures the phase difference $\Delta\phi(t)(= (\omega_{lead} - \omega_{slave})t)$.

In MegaMIMO the lead AP initiates data transmission using a synchronization header, as in channel estimation. Each slave AP use this synchronization header to measure the current channel, $h_i^{lead}(t)$ from the lead AP to itself. Note that the current channel will be rotated relative to the reference channel because of the oscillator offset between the lead AP and slave AP. In particular, $h_i^{lead}(t) = h_i^{lead}(0)e^{j(\omega_{T1} - \omega_{T2})t}$. Each slave can therefore compute $e^{j(\omega_{T1} - \omega_{T2})t}$ directly, from its two measurements of the lead AP channel. Such an estimate does not have errors that accumulate over time because it is purely a division of two direct measurements. The slave then multiplies its transmitted signal by this quantity, as described in Section 5.4.

Now that all AP oscillators are synchronized at the beginning of the data transmission, the slave AP also needs to keep its oscillator synchronized with the lead transmitter through the actual data packet itself. It does this by multiplying its transmitted signal by $e^{j(\omega_{T1} - \omega_{T2})t}$ where t is the time since the initial phase synchronization at the beginning of the joint transmission. Note that this offset estimate only needs to be accurate within the packet, *i.e.*, for a few hundred microseconds or about 2 ms at most. MegaMIMO APs maintain a continuously averaged estimate of their offset with the lead transmitter across multiple transmissions to obtain a robust estimate that can maintain accurate phase synchronization within a packet.

Two additional points about MegaMIMO's synchronization are worth noting.

First, for ease of exposition, we have discussed the entire system so far in the context of correcting carrier frequency offsets. However, any practical wireless system has to also account for the sampling frequency offsets. Note that any offset in the sampling frequency just adds to the phase error in each OFDM subcarrier. Since our phase offset estimation using the synchronization header, described in Section 5.5, estimates the overall phase, it automatically accounts for the initial phase error accumulated from sampling frequency offset. Within each packet, the MegaMIMO slave APs correct for the effect of sampling frequency offset during the packet by using a long-term averaged estimate, similar to the carrier frequency offset.

Second, as mentioned earlier, in Section 5.5, MegaMIMO APs are synchronized in time using [120]. As described in [120], due to differences in propagation delays between different transmitters and different receivers, one cannot synchronize all transmitted signals to arrive exactly at the same time at all receivers. It is important to note that MegaMIMO works correctly even in the presence of different propagation delays between different transmitters and receivers. This is because the signals from different MegaMIMO APs will arrive within a cyclic prefix of each other at all receivers.[3] The delay differences between the signals from different APs at a receiver translate to a relative phase difference between the channels from these APs to that receiver. MegaMIMO's channel measurement phase captures these relative phase differences in the channel matrix, and MegaMIMO's beamforming then applies the effect of these phase differences while computing the inverse of the channel matrix.

5.5.3 Overarching Principles

In summary, the core challenge met by MegaMIMO's design is to accurately estimate and track the phase differences between each of the N clients and N APs. This challenge is particularly arduous for two reasons: (1) each receiver must simultaneously track the phase of N independent transmitters, and (2) errors in the estimates in the CFO result in phase offsets that accumulate over time, quickly leading to very large errors. Our general approach to tackling these challenges is to have all transmitters and receivers synchronize their phase to that of a single lead transmitter. Our implementation of this approach has been guided by following three overarching principles:

- **Between APs and within a packet we can use estimated frequency and sampling offsets to track phase:** We can measure the frequency and sampling offsets *between APs* accurately enough that the accumulated phase differences within a single packet

[3]In fact, since the common design scenario for MegaMIMO is confined locations like conference rooms and auditoriums, the propagation delay differences between different APs to a receiver are in the tens of nanoseconds, which is smaller than the 802.11 cyclic prefix of 400 or 800 ns, which is designed for worst case multipaths.

(10s to a few 100s of *microseconds*) are not significant enough to harm performance. Specifically, since APs are a part of the infrastructure, and CFOs do not change significantly over time, we can get very accurate estimates of the CFO between APs by averaging over samples taken across many packets.

- **Between APs and *across* packets we *cannot* use estimated frequency and sampling offsets to track phase:** The across packet time scales (10s to 100s of *milliseconds*) are large enough that even with extremely accurate estimates of the frequency and sampling offsets, the accumulated phase differences from residual errors will lead to significant performance degradation. To handle this, MegaMIMO uses a single header symbol to directly estimate the total *phase offset* and re-sync the phases of all nodes at the beginning of each packet.

- **Between a client and an AP, we cannot use estimated frequency and sampling offsets to track phase even through a packet:** Since clients are a transient part of the network, we cannot get accurate enough estimates of frequency and sampling offsets to use for phase tracking even within a single packet. Thus each client uses standard OFDM techniques to track the phase of the lead AP symbol by symbol, Additionally when performing channel estimation, the APs interleave their packets so that the correction of the channels to a common reference time has minimal error.

■ 5.6 Compatibility with 802.11

In order for MegaMIMO to work with clients using off-the-shelf 802.11 cards, MegaMIMO needs to address two challenges:

1. **Sync header:** The sync header transmitted by the lead AP to allow the slave APs to compute their oscillator offset, and trigger their transmission, is not supported by 802.11.

2. **Channel measurement:** Recall that MegaMIMO requires a snapshot of the channel from all transmitters to all receivers measured at the same time. In Section 5.4, we described how to do this with a custom channel measurement packet format with interleaved symbols that allows a receiver to measure channels from all transmitters.

However, such a packet format is not supported by 802.11, and hence 802.11 cards cannot simultaneously measure channels from all APs at the same time.

MegaMIMO solves these issues in the context of 802.11n by leveraging 802.11n channel state information (CSI) feedback for beamforming. We now describe MegaMIMO's solutions to each of the challenges listed above.

■ 5.6.1 Sync Header

The lead AP in MegaMIMO needs to prefix each transmission with a sync header that allows the slave transmitters to measure their relative oscillator offset from the lead, and also triggers their joint transmission. A mixed mode 802.11n packet essentially consists of an 802.11n packet prefixed with 5 legacy symbols. These legacy symbols are only intended to trigger carrier sense in 802.11a/g nodes, and are not used by 802.11n receivers. Thus, the lead MegaMIMO can use these legacy symbols as a sync header. MegaMIMO slave APs use the legacy symbols to measure their oscillator phase offset from the lead, correct their transmission signal, and join the lead AP's transmission after the legacy symbols when the actual 802.11n symbols are transmitted.

■ 5.6.2 Channel Measurement

802.11n does not support the interleaved packet format that allows MegaMIMO to measure a snapshot of the channels from all the transmitters to a receiver simultaneously. Even more fundamentally, an 802.11n receiver with K (at most 4) antennas can measure at most K channels at a time. In a MegaMIMO system, the total number of transmit antennas across all APs is larger than the number of antennas on any single receiver. Thus, a receiver with off-the-shelf 802.11n cards will be unable to simultaneously measure channels from all transmit antennas to itself.

Naively, one could measure the channels from all transmit antennas by transmitting a separate packet from each AP, and then correcting these channel measurements using the estimated frequency offsets to the receiver like in Section 5.5.1. Unlike the scenario in Section 5.5.1 where the transmissions from different APs are separated from each other by only a few symbols (using interleaving), the transmissions from different APs here are separated by at least one packet width. As discussed in Section 5.5.3, it is not practical

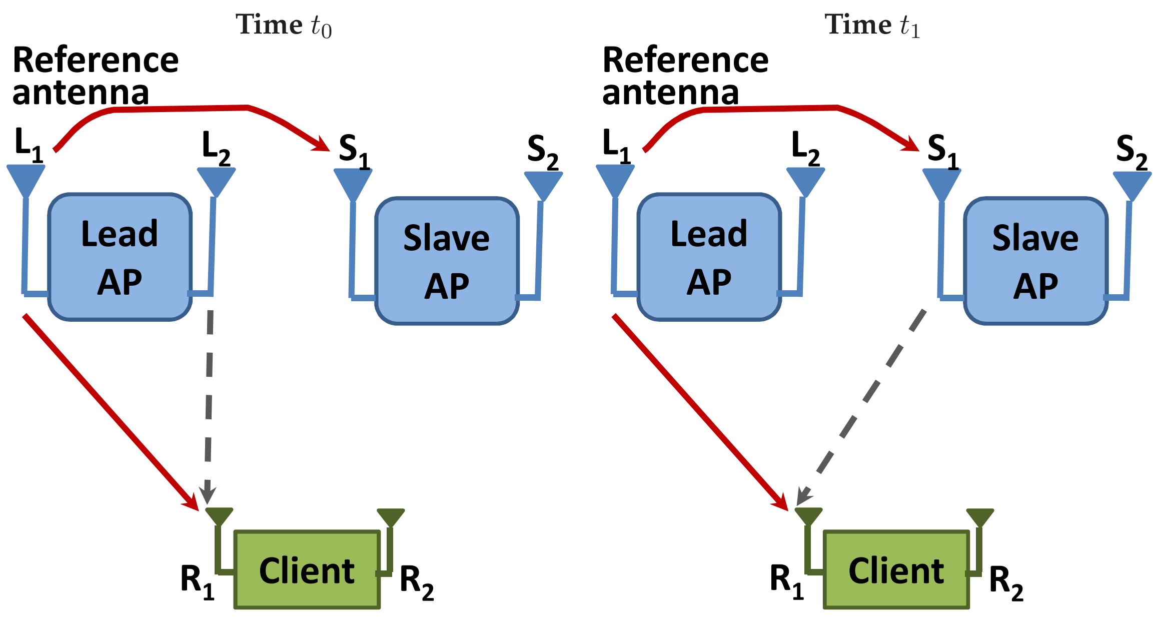


Figure 5-4: 802.11 Channel Measurement. MegaMIMO measures channels with 802.11 clients by sending a series of two-stream transmissions. Every transmission includes the reference antenna, L_1, as well as one other antenna (either L_2 or S_1 in our example). Note that for clarity, the figure does not show the transmissions to/from R_2 and S_2, but MegaMIMO naturally measures the channels to R_2 simultaneously.

to compute receiver frequency offsets accurately enough to ensure that the accumulated phase error across packets will be tolerable.

MegaMIMO instead performs robust channel measurement by "tricking" the receiver into measuring channels from different AP antennas simultaneously. This trick allows MegaMIMO to measure the channel from each AP antenna to the receiver in conjunction with a common reference channel to the receiver. Using such a common reference across all measurements allows MegaMIMO to avoid measuring the *receiver frequency offset*, and instead directly estimate the oscillator *phase offset* between different channel measurements, and therefore compensate for it, as we describe below.

For simplicity, we focus on the scenario in Fig. 5-4 with 2 APs (a lead and a slave) and 1 client, where all nodes have 2 antennas each. In the rest of this discussion, we will focus only on the channel measurements to R_1 since the channels to R_2 are naturally measured simultaneously with R_1 in exactly the same manner.

At time t_0, L_1 and L_2 transmit a 2-stream packet jointly to R_1. This measurement gives us the channels $L_1 \to R_1$ and $L_2 \to R_1$ at time t_0. In addition, S_1 measures the channel $L_1 \to S_1$ using the synchronization header.

At time t_1, L_1 and S_1 trick the receiver by jointly transmitting a 2-stream packet from 2 different APs. This measurement gives us the channels $L_1 \to R_1$ and $S_1 \to R_1$ at time t_1. Again, S_1 measures the channel $L_1 \to S_1$ using the synchronization header.

The challenge is that we would like to obtain the channel $S_1 \to R_1$ at time t_0 but we have only the channel $S_1 \to R_1$ measured at t_1.

We therefore need to correct our measured channel by the accumulated phase offset between S_1 and R_1 in the time interval t_0 to t_1. To do this, we take advantage of the fact that we can compute the accumulated phase offset between both L_1 and R_1, and between L_1 and S_1 in the time interval t_0 to t_1.

- L_1 **and** R_1**:** We can compute this accumulated phase offset using the measurements of the channel $L_1 \to R_1$ at time t_0 and time t_1.
- L_1 **and** S_1**:** We can compute this accumulated phase offset using the measurements of the channel $L_1 \to S_1$ at time t_0 and time t_1.

The difference between these two accumulated phase offsets gives us the desired accumulated phase offset between S_1 and R_1 in the time interval t_0 to t_1.

We can similarly measure the channel $S_2 \to R_1$ in the next time slot, say t_2, and rotate it back to time t_0. We can repeat this process for all AP antennas.

5.7 Decoupling Measurements to Different Receivers

The scheme described in Section 5.4 assumed that the channels from all APs to all receivers are all measured at the same time. In Section 5.6.2, we showed how MegaMIMO could measure channels from different APs to a single receiver at different times and compensate for differences in oscillator offset by using a shared reference measurement across all APs for that receiver. But what about the channels to a different receiver? If this receiver joins the wireless network after the channels to the first receiver are measured, there is no opportunity for a shared reference measurement between the two receivers. It might therefore seem that MegaMIMO's requirement for all channels to be measured at the same time would necessitate measurement of channels to all receivers whenever a receiver joins the network, or when a single receiver's channels change.

In fact, we can show that such full measurement is not necessary, and that MegaMIMO can decouple channel measurements to different receivers. The key idea is that

MegaMIMO can use the channels from the lead AP to slave APs as a shared reference in this case, instead of the channel from the lead AP to a receiver as was the case in Section 5.6.2. We prove below that using such a shared reference allows MegaMIMO to measure channels to different receivers at different times, and still correctly perform multi-user beamforming using distributed phase synchronization.

For simplicity, we focus on the example of two APs and two clients in Fig. 5-2. Let us consider a system where the channels, h_{11} and h_{12}, to receiver 1 are measured at time t_1 and the channels, h_{21} and h_{22}, to receiver 2 are measured at time t_2. For a subsequent transmission at time t, the channels experienced to receiver 1 experience a rotation corresponding to the time $t - t_1$, while the channels experienced to receiver 2 experience a rotation corresponding to time $t - t_2$. In particular, the channel matrix experienced at time t can be written as:

$$\mathbf{H(t)} = \begin{pmatrix} h_{11}e^{j(\omega_{R1}-\omega_{T1})(t-t_1)} & h_{12}e^{j(\omega_{R1}-\omega_{T2})(t-t_1)} \\ h_{21}e^{j(\omega_{R2}-\omega_{T1})(t-t_2)} & h_{22}e^{j(\omega_{R2}-\omega_{T2})(t-t_2)} \end{pmatrix}$$

Recall that for MegaMIMO to perform distributed phase synchronization, we need to decompose $\mathbf{H(t)}$ into the form $\mathbf{R(t)HT(t)}$ where $\mathbf{H}$ is time-invariant, and the time-dependent matrices $\mathbf{R(t)}$ and $\mathbf{T(t)}$ are diagonal, and the i^{th} diagonal entry of $\mathbf{T(t)}$ (similarly) depends only on parameters that the i^{th} AP (similarly i^{th} receiver) can estimate locally. The APs can then all use the time invariant matrix $\mathbf{H}$ to calculate their beamforming signal, and perform correction using the relevant entry of $\mathbf{T(t)}$.

We observe that $\mathbf{H(t)}$ can indeed be written in this desired form. Specifically, we can write $\mathbf{H(t)}$ as $\mathbf{R(t)HT(t)}$, where:

$$\begin{aligned}
\mathbf{R(t)} &= \begin{pmatrix} e^{j(\omega_{R1}-\omega_{T1})(t-t_1)} & 0 \\ 0 & e^{j(\omega_{R2}-\omega_{T1})(t-t_2)} \end{pmatrix} \\
\mathbf{H} &= \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22}e^{-j(\omega_{T1}-\omega_{T2})(t_2-t_1)} \end{pmatrix} \\
\mathbf{T(t)} &= \begin{pmatrix} 1 & 0 \\ 0 & e^{j(\omega_{T1}-\omega_{T2})(t-t_1)} \end{pmatrix}
\end{aligned}$$

Note that $\mathbf{H}$ is now time invariant as desired. The entries only depend on the oscillator offset between times t_1 and t_2. Slave AP i can easily compute this offset by using the reference channel, h_i^{lead} from the lead measured at time t_1 and t_2.

Further, note that the diagonal entries of the matrix $\mathbf{T}(\mathbf{t})$ only depend on the frequency offset of the corresponding slave AP from the lead AP, and hence each slave AP can, as before, observe the channel of the sync header, compute the oscillator offset using the channel measured at time t_1 as reference, and correct its transmission appropriately.

Similarly the diagonal entries of the matrix $\mathbf{R}(\mathbf{t})$ only depend on the frequency offset of the corresponding receiver from the lead AP, and hence each receiver can independently decode its packet as if it were sent from a single transmitter.

Intuitively, this scheme can be understood as the slave AP rotating its measured channel to receiver 2 back to the time t_1 by multiplying h_{22} by $e^{-j(\omega_{T1}-\omega_{T2})(t_2-t_1)}$, and then performing all future channel corrections relative to the time t_1. This is why it corrects by the time dependent quantity $e^{-j(\omega_{T1}-\omega_{T2})(t-t_1)}$ shown in $\mathbf{T}(\mathbf{t})$. The slave AP can, as before, measure this quantity by observing the rotation of the phase of the channel from the lead AP, $T1$, between the reference time t_1 and the current t.

■ 5.8 Diversity

MIMO systems can provide both multiplexing and diversity gains. So far, we have described the use of MegaMIMO for multiplexing. The same discussion applies to diversity except that in this case, we have all the APs transmitting jointly to a single client, say client 1. Each AP then computes its beamformed signal as $\frac{h_{1i}^*}{\|h_{1i}\|}x_1$ and slaves continue to perform distributed phase synchronization as before.

■ 5.9 MegaMIMO's Link Layer

So far, we have described MegaMIMO's physical layer, which enables multiple APs to transmit simultaneously to multiple receivers. We now describe how MegaMIMO's link layer is designed to use this capability.

MAC and Carrier Sense: In MegaMIMO, all downlink packets are sent on the Ethernet to all MegaMIMO APs. Thus, all APs in the network have the same downlink queue. Each packet in the queue has a *designated AP*, which is the AP with the strongest SNR to the client to which that packet is destined. MegaMIMO always uses the packet at the head of the queue for transmission, and nominates the designated AP of this packet as the lead AP for this transmission. The lead AP then chooses additional packets (and corresponding slave APs) for joint transmission with this packet in order to maximize the network throughput. There are a variety of heuristics [159, 131, 158] that can be adopted for selecting the packets for joint transmission, and we leave the exact algorithm for making this choice for future work.

The lead AP contends on behalf of all slave APs, with its contention window weighted by the number of packets in the joint transmission as described in [119]. Clients contend for the medium as they do today using 802.11 CSMA. When the lead AP wins a contention window, it starts transmitting its synchronization header, which causes the slave APs to join the transmission. We note that contention for joint transmission by the lead AP is robust to hidden terminals for two reasons. First, MegaMIMO is intended for dense deployments like conference rooms where access points can hear each other, and the overall wireless capacity is limited by interference between access points. Further, even in the unlikely event of hidden terminals, situations causing persistent packet loss due to repeated collisions can be detected using mechanisms like in [133], and the lead AP can ensure that JMB access points that trigger hidden terminal packet loss above a threshold are not part of the joint transmission. This ensures that both MegaMIMO joint transmissions, as well as other transmissions, do not encounter persistent hidden terminals.

Rate Selection using Effective SNRs: In systems like MegaMIMO where different sets of APs transmit concurrently for different packets, the rate to a client can change from packet to packet as the effective channel at each client changes as a result of beamforming. Such systems therefore need to use a rate-selection algorithm [90]. MegaMIMO uses the **effective SNR** algorithm, which is designed for rate selection for 802.11-like frequency selective wideband channels [61]. Since APs in MegaMIMO know the full channel matrix, **H**, prior to transmission. APs multiply the signals by $k\mathbf{H}^{-1}$ (k accounts for the maximum power constraint at APs). Thus, the effective channel is $k\mathbf{H}^{-1}\mathbf{H} = k\mathbf{I}$, giving a signal strength of k^2 at each client. Client cards report noise N as in [11]. Clients send the noise N to APs along with the measured channels. Since APs know the signal strength, k^2, in each subcarrier, and the associated noise N, they can compute the SNR in each subcarrier as $\frac{k^2}{N}$. They can then map this set of SNRs to rate by performing a table lookup [61]. Thus, each client in a MegaMIMO joint transmission gets the same rate, which is similar to traditional 802.11 fairness.

Acknowledgments: MegaMIMO disables synchronous ACKs at clients and uses higher layer asynchronous acknowledgments like in prior work such as MRD and ZipTx [106, 91]. Further, similar to systems like Maranello [62], MegaMIMO can modify the firmware on clients to implement an optimized joint synchronous acknowledgment protocol consisting of a single SIFS, followed by back-to-back acknowledgments from all the clients.

Packet losses and retransmissions: It is important to note that, even though APs transmit packets jointly to different receivers, packet losses at different clients are decoupled. Specifically, if APs have stale channel information to a client, only the packet to that client is affected, and packets at other clients will still be received correctly. As in regular 802.11, APs in MegaMIMO keep packets in the queue until they are ACKed. If a packet is not ACKed, they can be combined with other packets in the queue for future concurrent transmissions.

■ 5.10 Testbed and Implementation

We implement MegaMIMO's AP design in software radios and evaluate it with both off-the-shelf 802.11 clients and software-radio clients.

(a) Implementation for the software radio testbed: In this testbed, each node is equipped with a USRP2 board [41], and an RFX2400 daughterboard, and communicates on a 10 MHz channel in the 2.4 GHz range. We implement OFDM in GNURadio, using various 802.11 modulations (BPSK, 4QAM, 16QAM, and 64QAM), coding rates, and choose between them using the effective-SNR bitrate selection algorithm [61].

Our MegaMIMO implementation includes the following modules: distributed phase alignment, beamforming for multiplexing and diversity, and bitrate selection. We do not implement ACKs, CSMA, or retransmissions. To perform correct phase alignment, concurrent transmitters must be synchronized tightly at the sample level. We do this by using USRP2 timestamps to synchronize transmitters despite delays introduced by software. Before every data packet, the lead AP sends a trigger signal on the medium at $t_{trigger}$. All other APs log the timestamp of this signal, add a fixed delay t_Δ to it, and then transmit concurrently at this new time. We select t_Δ as 150μs based on the maximum delay of our

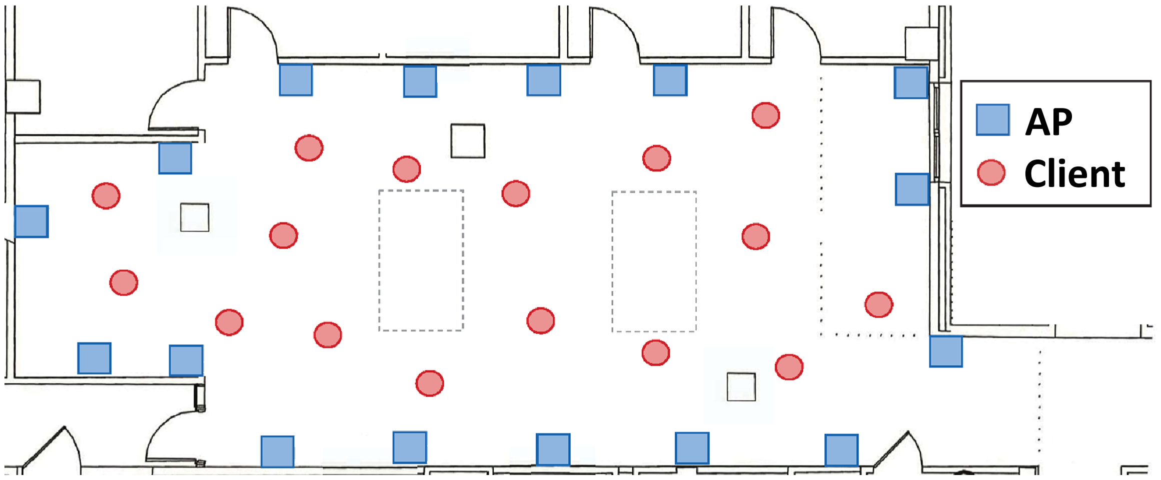


Figure 5-5: Testbed Topology. Client locations are marked by red circles, and AP locations by blue squares. *Note that the figure shows the set of possible locations for clients and APs, and different subsets of locations are picked for different experiments.*

software implementation. Finally, to optimize the software turnaround, we did not use GNURadio, but wrote our own C code that directly interacts with the USRP hardware.

(b) Implementation for the 802.11 testbed: There are two main differences between this testbed and the one above. First, each client in this testbed uses an off-the-shelf 802.11 card. Second, each node in this testbed has two antennas and can act as a MIMO node. Our objective is to show that MegaMIMO extends beyond single antenna systems; For example, it can combine two 2x2 MIMO systems to create a 4x4 MIMO system.

Each AP is built by connecting two USRP2 nodes via an external clock and making them act as a 2-antenna node. Each client is a PC equipped with a Intel Wi-Fi Link 5300 a/b/g/n wireless network adapter on which 2 antennas are enabled. The Intel Wi-Fi Link 5300 adapters are updated with a custom firmware and associated `iwlwifi` driver in order to obtain the channel state information in user space [60].

The AP software implementation is similar to the other testbed except that we make the channel width 20 MHz to communicate with actual 802.11 cards. The packet format is also changed to match 802.11 packets. The client software collects the channel measurements from the firmware and logs correctly decoded packets.

(c) Testbed Topology: We evaluate MegaMIMO in an indoor wireless testbed that simulates a conference room or classroom, with APs deployed on ledges near the ceiling, and clients scattered through the room. Fig. 5-5 shows node locations in the experimental environ-

ment. In every run, the APs and clients are assigned randomly to these locations. Note that the testbed exhibits significantly diverse SNRs as well as both line-of-sight and non line-of-sight paths due to obstacles such as pillars, furniture, ledges etc. The APs transmit 1500 byte packets to the clients in all experiments.

■ 5.11 Results

We evaluate MegaMIMO both through microbenchmarks of its individual components, as well as an integrated system on both USRP and 802.11n testbed.

■ 5.11.1 Microbenchmarks

(a) Necessity of Phase Alignment: A key challenge for a distributed MIMO system is that it must compensate for oscillator offsets between the transmitters. In this section, we demonstrate the impact of misalignment between transmitters on the received SNR.

Method. We simulate a simple 2-transmitter, 2-receiver system where different data is intended for each receiver. The transmitters measure the initial channel matrix to the receivers, and use this matrix to compute their beamforming vectors. We then introduce a phase misalignment at the slave transmitter, and compute the reduction in SNR at each receiver as a result of this misalignment. We repeat this process for 100 different random channel matrices, phase misalignments from 0 to 0.5 radians, and for two systems - one in which the average SNR is 10 dB, and other in which the average SNR is 20 dB.

Results. Fig. 5-6 shows the average reduction in SNR in dB, as a function of phase misalignment. As one would expect, an increase in phase misalignment increases the interference at each receiver. As the graph shows, even a phase misalignment as small as 0.35 radians[4] , can cause an SNR reduction of almost 8 dB at an SNR of 20 dB. This SNR reduction will be greater as we add more and more transmitters to the system. Further, phase misalignment causes a greater reduction in SNR when the system is at higher SNR. This is because the impact of additional noise added by interference is higher when the original noise itself is low, *i.e.*, at high SNR.

[4]0.35 radians is much smaller than the misalignment expected with the mandated 802.11 tolerance of 20 ppm.

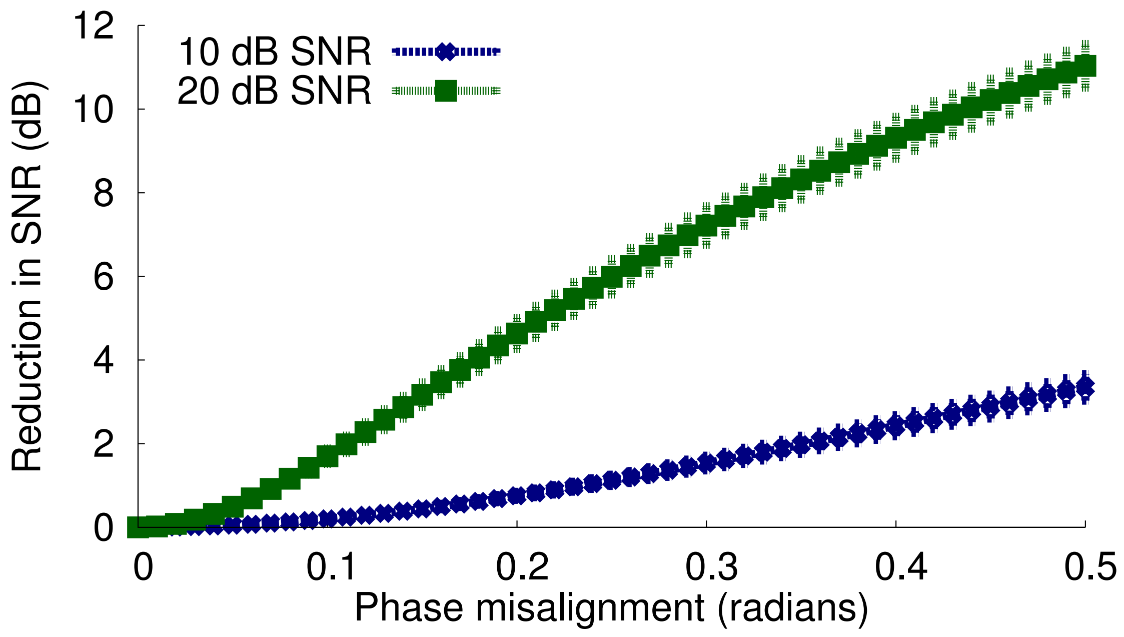


Figure 5-6: Degradation of SNR due to phase misalignment. Even with only 2 transmitters, a misalignment of just 0.35 radians can reduce the SNR by almost 8 dB at the receivers due to interference.

(b) Accuracy of MegaMIMO's Phase Alignment: We now examine the accuracy of MegaMIMO's lightweight distributed phase alignment algorithm.

Method. In this experiment, we place two MegaMIMO nodes at random AP locations and a third MegaMIMO node at a receiver location. We randomly pick one of the two APs to be the lead and the other to be the slave. The slave transmitter implements MegaMIMO's distributed phase synchronization algorithm, and performs phase correction on its transmission before joining the lead transmitter's data transmission. In order to measure the accuracy of MegaMIMO's phase synchronization algorithm, we make the lead and the slave APs alternate between transmitting OFDM symbols. In particular, each transmitter's transmission consists of pairs of an OFDM symbol followed by an OFDM symbol length of silence. The transmissions of the lead and slave transmitter are offset by 1 symbol so that the receiver sees alternating symbols from lead and the slave transmitter. The receiver estimates the lead and slave transmitter's channels separately, and computes the relative phase between them. We then perform several rounds of this measurement. The receiver uses the first measurement of relative phase as a reference, and computes the deviation of relative phase from this reference in subsequent transmissions.

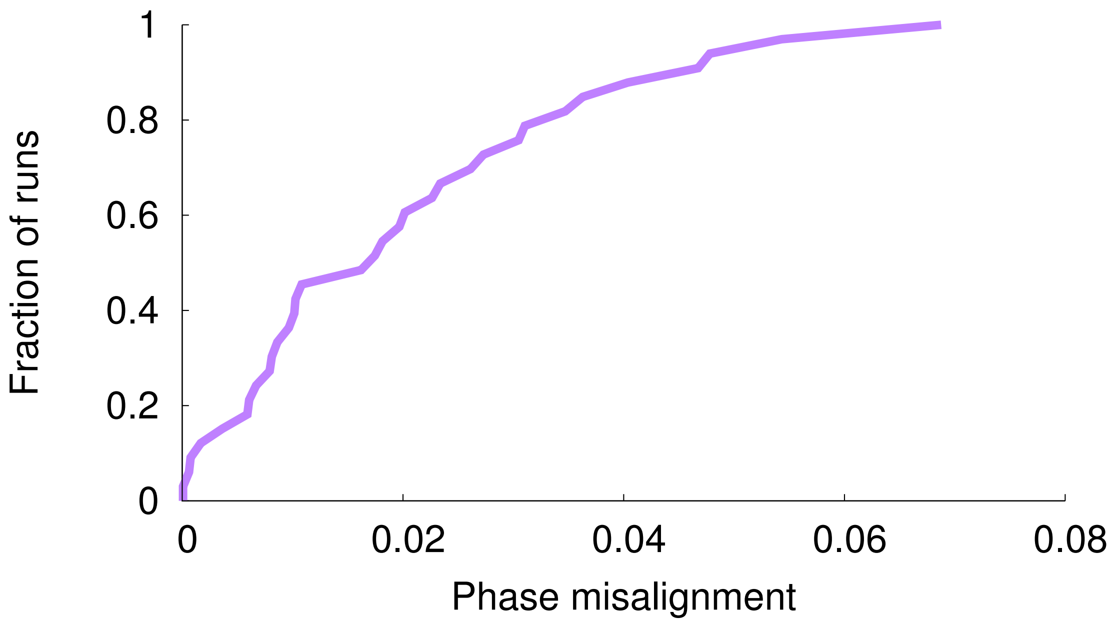


Figure 5-7: CDF of observed phase misalignment. The median misalignment is 0.017 radians, and the 95^{th} percentile misalignment is 0.05 radians.

Results. Fig. 5-7 plots the CDF of the absolute value of the deviation in relative phase across all the experiments. If the lead and slave transmitter are always perfectly aligned, the deviation should be zero. However, estimation errors due to noise and oscillator drift due to the delay from when the slave measures the lead's channel and turns around to jointly transmit data will induce misalignment. As can be seen, however, the median misalignment is less than 0.017 radians, and the 95^{th} percentile misalignment is less than 0.05 radians. Based on Fig. 5-6, with two transmitters, MegaMIMO's phase alignment algorithm can ensure that the SNR of joint transmission is not reduced by 0.4 dB at the 95^{th} percentile.

(c) How does SNR reduction scale? The previous experiments examined in depth the impact of misalignment and MegaMIMO's precise alignment performance in the case of a 2x2 distributed MIMO system. In this experiment, we observe how the SNR reduction grows as we increase the number of transmitters in the system.

Method. We evaluate the SNR reduction in MegaMIMO in three effective SNR [61] ranges: low (6-12 dB), medium (12-18 dB) and high ($>$ 18 dB). For each range, we place several MegaMIMO nodes in random AP locations in the testbed. We then place the same number of MegaMIMO nodes in random client locations, such that all clients obtain an

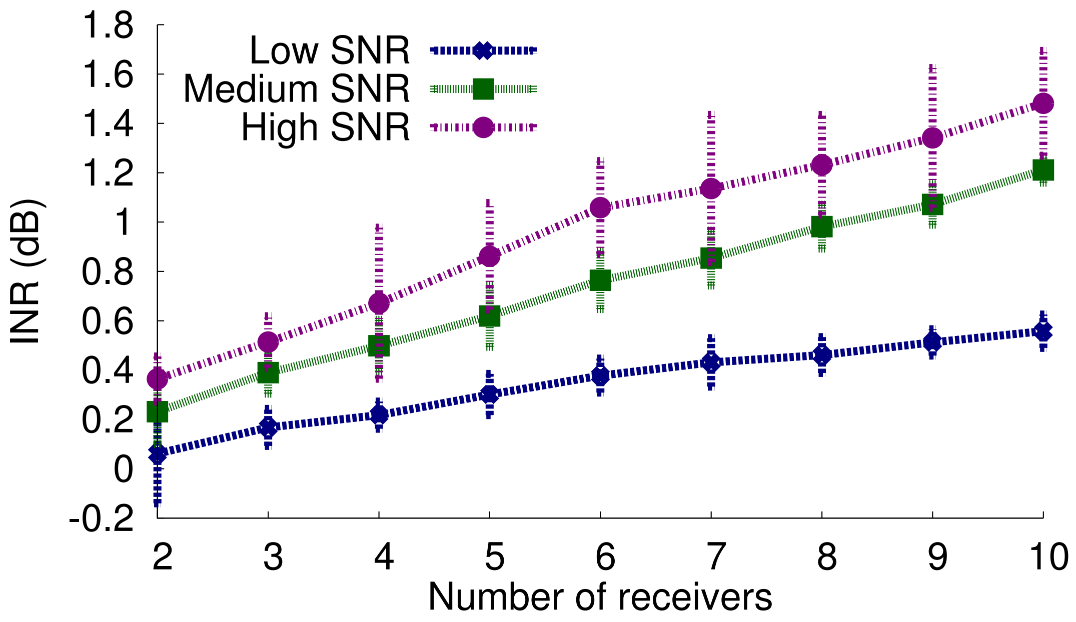


Figure 5-8: Accuracy of Phase Alignment. Average INR at receivers for various numbers of receivers across different SNR ranges. INR stays below 1.5dB across SNRs even with 10 receivers.

effective SNR in the desired SNR range. For each placement, we then choose a client at which all APs null their interference, *i.e.* the expected signal at that client is zero and measure the received signal power at that client. If phase alignment is perfect, the received signal power should be comparable to noise, *i.e.* the ratio of the received signal power to noise should be 0 dB. Any inaccuracy in phase misalignment will lead to interference, manifest as a higher ratio of received power to noise, and produce a corresponding reduction in SNR if data were actually transmitted to that client. For each topology, we null at each client, and compute the average interference to noise ratio (INR) across clients. We repeat this experiment for different topologies, and different numbers of MegaMIMO APs at low, medium and high SNRs.

Results. Fig. 5-8 shows the INR as a function of the number of MegaMIMO APs at low, medium and high SNRs. Note that, as before, the reduction in SNR increases with SNR, but is below 1.5 dB even at high SNR. The INR also increases with the number of APs, as the number of interferers increases, but increases gradually: only ~ 0.13 dB for every additional AP-client pair even at high SNR.

5.11.2 Theoretical Scaling Behavior of MegaMIMO

The key promise of MegaMIMO is that it provides linear scaling of throughput with the number of APs at all SNRs. However, it is known that traditional MIMO does not provide unbounded scaling as the number of transmit and receive antennas is increased [144]. Instead, it is known to provide only at most 4 to 8 degrees of freedom in typical indoor scenarios. A natural question is whether MegaMIMO suffers from the same scaling limitation.

In order to answer this question, we first need to understand how traditional MIMO provides degrees-of-freedom gain over single antenna systems. In traditional MIMO, the transmit (similarly, receive) antennas are placed very close to each other with adjacent antennas separated by roughly half a wavelength. In particular, the distance between adjacent antennas is negligible compared to the distance between the transmitter and the receiver. Thus, in the presence of pure line-of-sight (LOS) channels, traditional MIMO does not provide any additional degrees of freedom in comparison to a single antenna system [144]. In order to obtain additional degrees of freedom, the transmitter therefore needs to have multiple paths with significant angular separation to the receiver. In traditional MIMO, this requires the presence of reflected paths from reflectors in the environment that are far apart from each other. Thus, the degree of freedom gain in traditional MIMO is limited by the number of multipaths in the environment.

However, in MegaMIMO, transmit antennas are located on separate nodes and hence are naturally geographically separated from each other. Thus, we would expect that MegaMIMO has the ability to obtain additional degrees of freedom even in a pure line-of-sight environment since the paths from different transmit antennas have high angular separation to the receiver, and also suffer significantly different attenuation relative to each other.

We formally evaluate this intuition in a linear array scenario, with MegaMIMO APs placed along one edge of a rectangular room, and clients placed along the opposite edge of the room, as shown in Fig. 5-9. This is the MegaMIMO equivalent of a traditional multi-antenna transmitter with a linear antenna array on one side of the room, and a multi-antenna receiver on the opposite side of the room. However, the key difference from traditional MIMO is the large inter-antenna separation in MegaMIMO as compared to traditional MIMO.

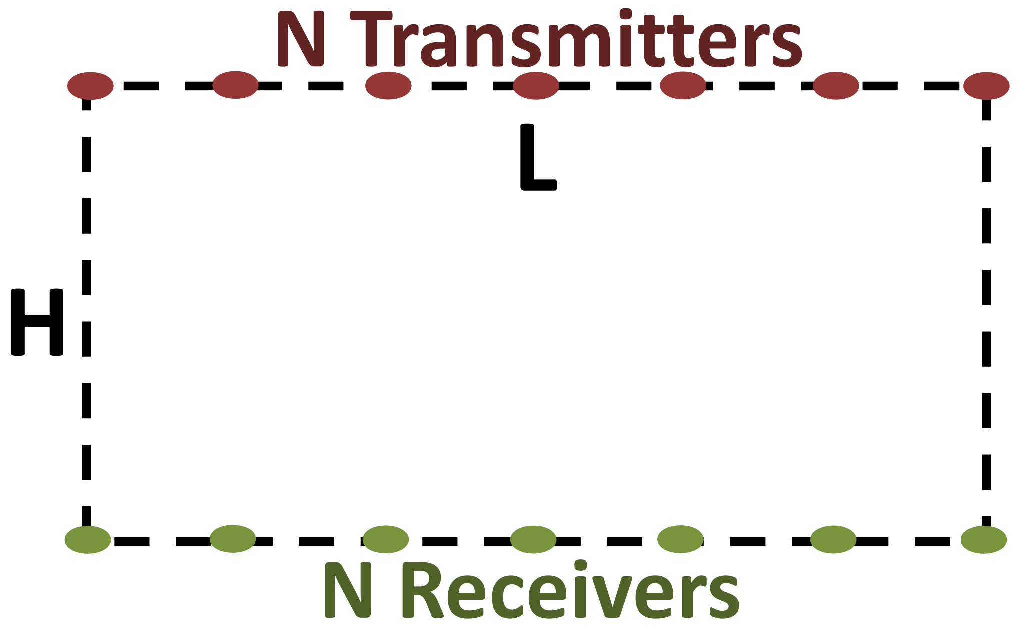
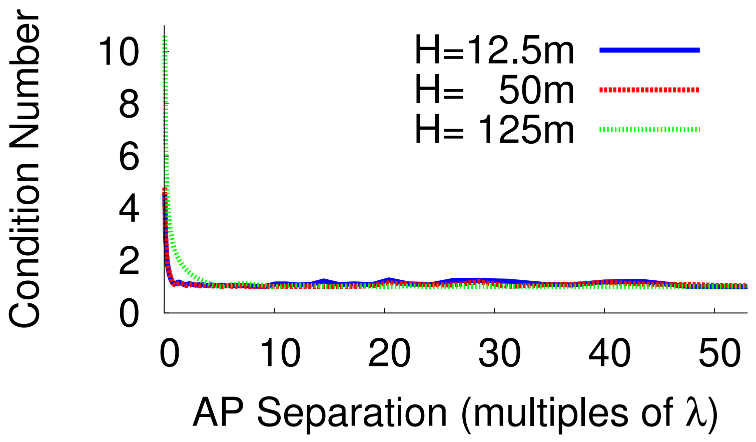


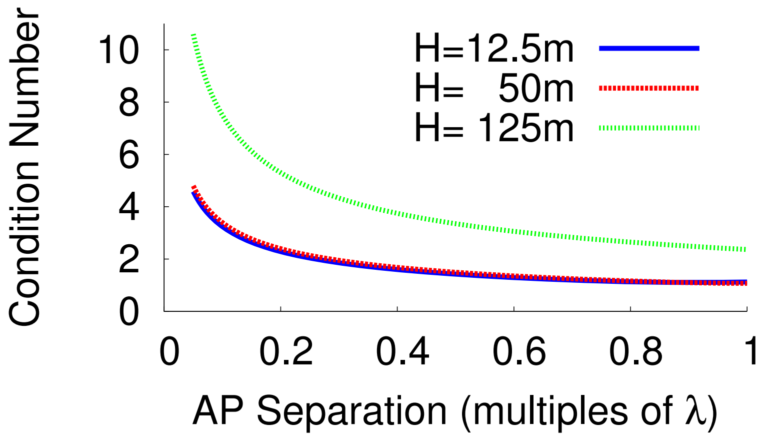
Figure 5-9: Linear Array Topology. This is the MegaMIMO equivalent of a traditional MIMO transmit and receive array. This topology has N APs spaced uniformly along one edge of a rectangular room, and N clients similarly spaced along the opposite edge. The length of the AP array is L meters, and the AP-client separation is H meters.

Our methodology is as follows. We pick a certain number of APs, N. For each N, we lay out the APs uniformly separated on one edge of the room and the clients uniformly separated along the other edge of the room. We model a pure line-of-sight scenario, *i.e.*, there are no reflectors present in the environment. We compute the distance, d_{ij} between each client k and AP l. Following [144], we then compute the corresponding channel between these antennas as $h_{kl} = d_{kl}^{-\rho} \exp(-\frac{j2\pi d_{kl}}{\lambda})$ where λ is the carrier wavelength (0.125 m corresponding to the Wi-Fi carrier frequency of 2.4 GHz in our case) and ρ is the path loss exponent. Since this is a pure line-of-sight environment, the path loss exponent is 1 corresponding to quadratic power loss in free space.

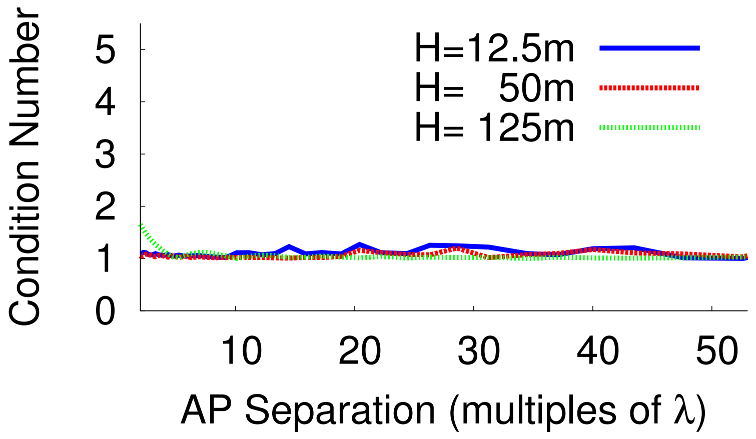
We first estimate how the conditioning of the channel matrix changes as we add more APs and pack them more densely. We compute this in a manner similar to [144]. In particular, for a particular number of APs, N, we compute the channel vectors from two adjacent APs to all N clients. We then determine the two singular values of the $2 \times N$ matrix corresponding to these two channel vectors, and evaluate the condition number as the ratio of the maximum singular value to the minimum singular value. We vary the number of APs, N for a given AP-client separation, H. We repeat this experiment for various values of H.



(a) All AP separations



(b) Small AP separations (zoomed from (a))



(c) Large AP separations (zoomed from (a))

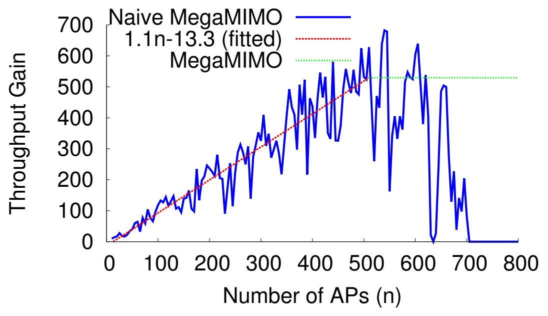
Figure 5-10: Condition number of the channel matrix as a function of inter-AP separation in line-of-sight. The channel matrix becomes better conditioned as the inter-AP separation becomes larger. When the inter-AP separations are very small, the system is poorly conditioned similar to traditional MIMO. At large inter-AP separations, the system is well conditioned even in a pure line-of-sight scenario.

Fig. 5-10 plots the condition number as a function of the inter-AP separation (represented as multiples of the carrier wavelength) for client-AP separations, H, of 12.5, 50, and 125 meters (100, 400, and 1000 wavelengths respectively, at 2.4 GHz) and an antenna array length, L, of 125 meters (1000 wavelengths) . Figs. 5-10(b) and (c) zoom into Fig. 5-10(a) when the inter-AP separations are small (less than a wavelength) and large (greater than 2 wavelengths).

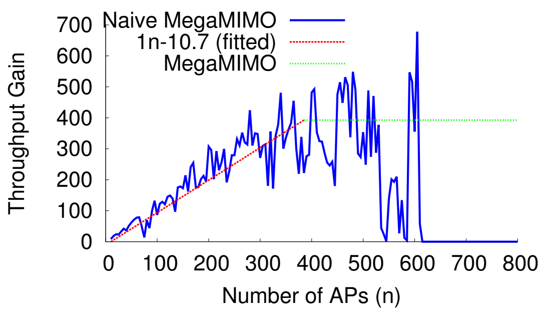
A large condition number indicates that the channel matrix is poorly conditioned, *i.e.*, there is not a significant degrees-of-freedom gain, and a condition number close to 1 indicates that the matrix is well condition and provides degrees-of-freedom gain. There are two points to note from the graph. First, for a given client-AP separation, the conditioning of the matrix improves as the inter-AP separation increases. This is because the channel vectors from the two different APs get increasingly different from each other as the inter-AP separation increases. Fig. 5-10(b) shows the condition number when the AP-client separation is less than a wavelength. When the AP-client separation is very small, the behavior of the system is similar to that of traditional MIMO. The matrix is poorly conditioned, implying that there are no degrees-of-freedom gains from multiple antennas. As the AP-client separation increases, the system moves into a different regime, shown in Fig. 5-10(c). In this regime, the channel vectors from the two APs get increasingly different, and the matrix is much better conditioned. When the APs are separated from each other by multiple wavelengths, the system therefore provides degrees-of-freedom gains even in a pure line of sight scenario. In MegaMIMO, since APs are separated from each other by large distances (tens of centimeters to multiple meters), it operates in the latter regime and can therefore provide degree of freedom gains even in line-of-sight.

Second, for a given inter-AP separation, the matrix is better conditioned when the AP-client separation is smaller. This is because the angular resolution of the client array is greater when the APs are closer to the clients. However, even when the AP-client separation is as large as 1000 wavelengths (125 m, in the case of Wi-Fi at 2.4 GHz), the matrix is well-conditioned for an inter-AP separation of about 10 wavelengths (1.25 m). It is not feasible for a single MIMO AP to have antennas separated by such a large distance, but such a topology is both feasible and natural in MegaMIMO, where degrees-of-freedom gain are obtained from the antennas on separate APs.

Now that we have obtained the intuition for degrees-of-freedom gain in MegaMIMO,



(a) AP-client separation (H) of 50m (400 wavelengths)



(b) AP-client separation (H) of 125m (1000 wavelengths)

Figure 5-11: Throughput gain as a function of number of APs for different AP-client separations. MegaMIMO provides degrees-of-freedom gain and linear throughput scaling even in a line-of-sight scenario because it can take advantage of large inter-AP spacing. Further, as the AP-client distance decreases, the minimum required inter-AP spacing to obtain linear scaling decreases allowing MegaMIMO to pack more APs in the same space. In both cases, as the number of APs becomes large (and the inter-AP spacing small), the gains of a naive MegaMIMO system that uses all available APs reduce and eventually become zero. However, as described in Section 5.9, an actual MegaMIMO system would not use all APs, but adaptively use only as many APs as necessary to achieve the maximum possible gain, as indicated by the green horizontal lines.

we compute the actual throughput scaling provided by MegaMIMO as we increase the number of APs. For this experiment, we fix the length of the AP array, L, to be 125 m (1000 wavelengths), and pick two client-AP separations (H), say, 50m (400 wavelengths) and 125m (1000 wavelengths). In order to ensure that both systems are operating in the same SNR regime, we scale the transmit power so that the client SNR from the nearest AP is the same independent of client-AP separation. This allows us to determine the gains arising purely from channel variation in the two different client-AP separation scenarios.

Figs. 5-11(a) and (b) show the throughput gain of MegaMIMO relative to the throughput of a single AP-client link for an SNR of 20 dB and for the two different client-AP separations. As the figures show, the throughput gain increases roughly linearly initially as the number of APs increases. This is because the matrix stays well conditioned (corresponding to large antenna separation) as we continue to add more APs. Eventually, the matrix starts gradually becoming worse and worse conditioned. While there are still degrees-of-freedom gain as compared to a single AP-client pair, the gain of a naive MegaMIMO system that greedily uses all available APs starts reducing as we increase the number of APs. Finally, beyond a threshold, the APs are packed too closely together and the channel matrix is singular, at which point a naive MegaMIMO system can no longer obtain any throughput gains compared to a single AP-client pair. However, as described in Section 5.9, an actual MegaMIMO system adds additional APs to a joint transmission only as long as they increase the total throughput. Thus, it would not greedily use all available APs, but instead adaptively use the largest number of APs where the system still achieves linear gains, and pick different subsets of APs for transmitting to different clients. Its gain would therefore not drop to zero, but stay at the point indicated by the green horizontal lines in the graphs.

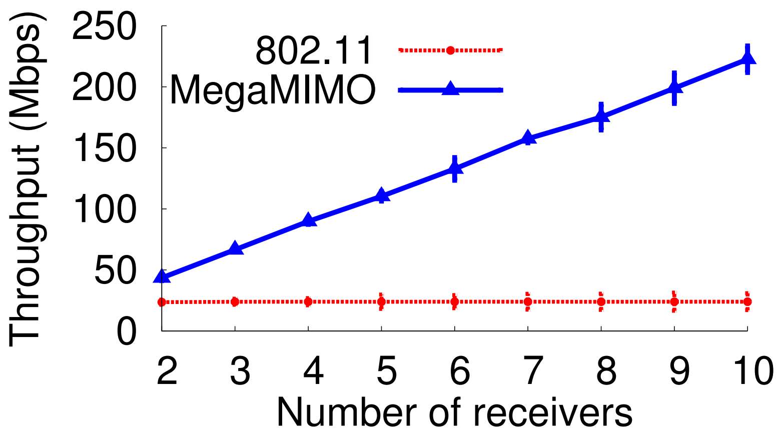
This graph shows that MegaMIMO can obtain linear throughput scaling (indicated by the red line with slope approximately 1 in both graphs) from degrees-of-freedom gains up to several hundreds of APs even in a pure line-of-sight scenario. Further, while the maximum achievable gain is smaller as the AP-client separation increases, MegaMIMO can still achieve large scaling even when the AP-client separation is large, unlike traditional MIMO.

Note that all the results in this section demonstrates the gains of MegaMIMO purely from the channel variations in a line-of-sight scenario with no reflections in the environment. The presence of reflectors and the addition of non line-of-sight paths will provide additional channel diversity, and therefore yield even larger degrees-of-freedom gains than a pure line-of-sight scenario.

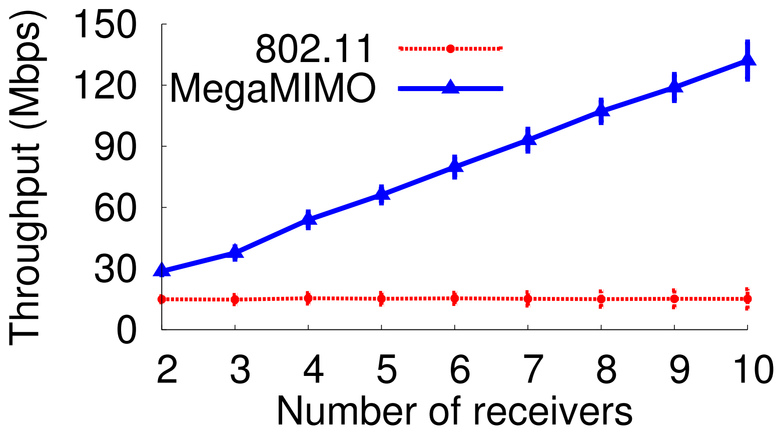
■ 5.11.3 Increase of Network Throughput with the Number of APs

Now that we have seen that MegaMIMO can indeed provide linear scaling of throughput with the number of APs in theory, we verify if MegaMIMO delivers on that promise in practice.

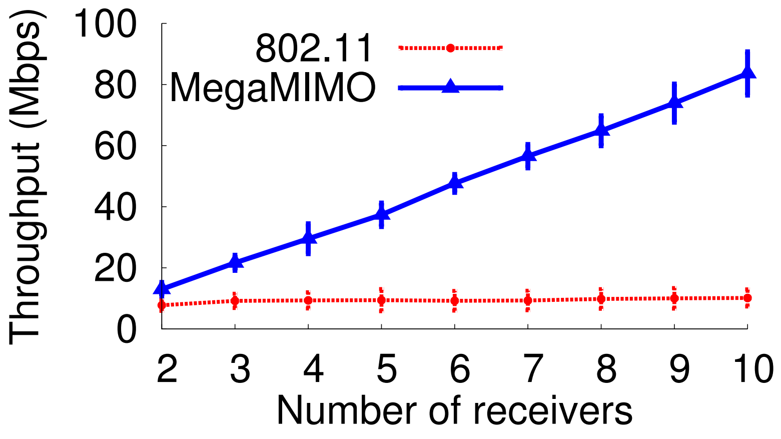
Method. We evaluate MegaMIMO's performance in three effective SNR ranges: low (6-12 dB), medium (12-18 dB) and high ($>$ 18 dB). For each range, we place a certain number of MegaMIMO nodes in random AP locations in the testbed. We then place the same number of nodes in random client locations such that all clients obtain an effective SNR in the desired range. For each such topology, we measure the throughput obtained both with regular 802.11 and MegaMIMO. Since USRP2 cannot perform carrier sense due to software latency, we measure the throughput of 802.11 by scheduling each client so that it gets an equal share of the medium. We repeat the experiment for 20 different topologies and also vary the number of MegaMIMO APs for each SNR range.



(a) High SNR (> 18 dB)



(b) Medium SNR (12-18 dB)



(c) Low SNR (6-12 dB)

Figure 5-12: Scaling of throughput with the number of APs. In this experiment, the number of APs equals the number of receivers. At all SNRs, MegaMIMO's network throughput increases linearly with the number of APs while total 802.11 throughput remains constant.

Results. Figs. 5-12(a), (b), and (c) show the total throughput obtained by 802.11 and by MegaMIMO for different numbers of APs, and different SNR ranges. Note that, as one would expect, the obtained throughput increases with SNR (802.11 throughput at low SNR is 7.75 Mbps, at medium SNR is around 14.9 Mbps, and at high SNR is 23.6 Mbps). There are two main points worth noting:

- 802.11 cannot benefit from additional APs operating in the same channel, and allows only one AP to be active at any given time. As a result, its throughput stays constant even as the number of APs increases. This throughput might vary with the number of APs in a real 802.11 network due to increased contention; however, since USRPs don't have carrier sense, we compute 802.11 throughput by providing each client with an equal share of the medium. In contrast, with MegaMIMO, as we add more APs, MegaMIMO can use these APs to transmit concurrent packets to more receivers. As a result, we see that the throughput of MegaMIMO increases linearly with the number of APs.

- The absolute gains provided by MegaMIMO are higher at high (~$9.4\times$ for 10 APs) and medium (~$9.1\times$) SNRs, than at low SNRs (~$8.1\times$). This is a consequence of the theoretically predicted throughput of beamforming. In particular, the beamforming throughput with N APs scales as $N\log(\frac{\text{SNR}}{K}) = N\log(\text{SNR}) - N\log(K)$, where K depends on the channel matrix $\mathbf{H}$ and is related to how well conditioned it is [144]. Natural channel matrices can be considered random and well conditioned, and hence K can essentially be treated as constant for our purposes. The 802.11 throughput scales roughly as $\log(\text{SNR})$ [144]. The expected gain of MegaMIMO over 802.11 can therefore be written as $N(1 - \frac{\log(K)}{\log(\text{SNR})})$ and hence becomes closer to N as SNR increases. This is why, MegaMIMO's gains at lower SNR grow at a lower rate than the gains at high SNR.

5.11.4 Fairness

In this experiment, we verify if MegaMIMO is fair, *i.e.*, it delivers the above throughput gains to all nodes.

Method. We perform the same experiment as in Section 5.11.3. We then compute the throughput gain of each node as the ratio of its throughput with MegaMIMO to its throughput with 802.11. As before, we vary the number of APs from 2 to 10, and repeat across the full range of SNRs.

Results. Figs. 5-13(a), (b), and (c) plot the CDF of throughput gain for 2, 6 and 10 APs at high, medium, and low SNRs. The results show that MegaMIMO is fair *i.e.* all nodes see roughly the same throughput gains, and these match the gains in total throughput shown in Section 5.11.3. Note that the CDF is wider at lower SNR. This is a consequence of greater measurement noise at low SNR causing larger throughput differences between clients.

■ 5.11.5 Diversity

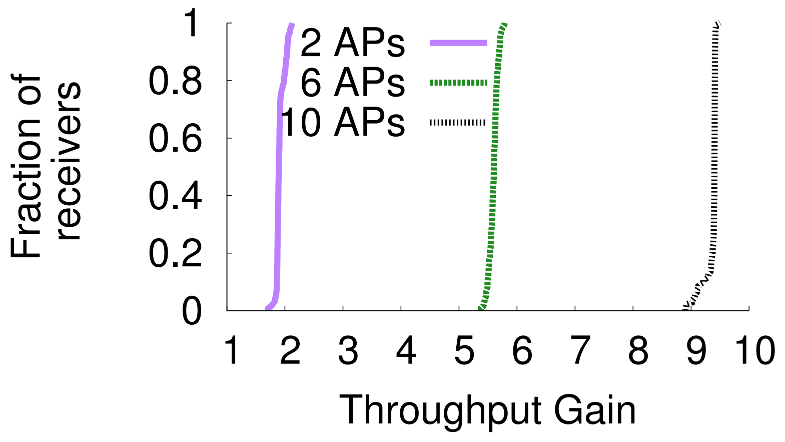
As described in Section 5.8, in addition to providing multiplexing gains, MegaMIMO can also provide throughput gains through diversity. In this section, we investigate MegaMIMO's diversity gains.

Method. We place several APs in random AP locations in the testbed, and one node at a client location, ensuring it has roughly similar SNRs to all APs. We then compute the throughput with regular 802.11 and MegaMIMO. We repeat the experiment with the number of APs varying from 2 to 10, and plot the results for the range of operational SNRs of 802.11.

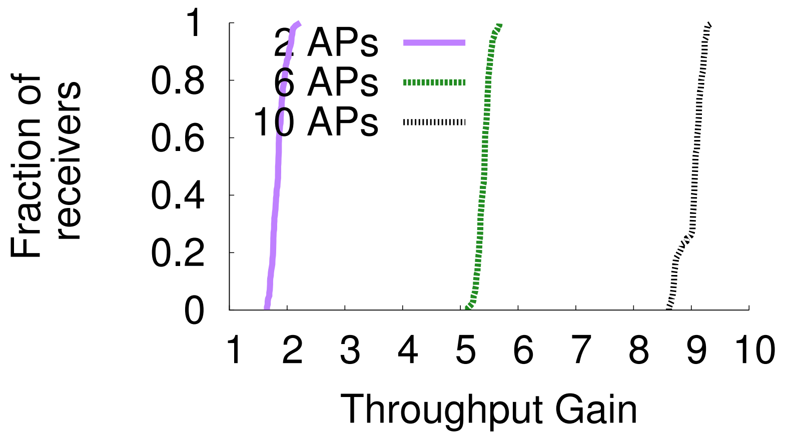
Results. Fig. 5-14 shows throughput of 802.11 and MegaMIMO as a function of SNR for 2, 4, 6, 8 and 10 APs. Note that MegaMIMO provides significant gains over 802.11, especially at low SNRs. For instance, a client that has 0 dB channels to all APs (*i.e.* its received power from each AP is about the same as the noise) cannot get any throughput with 802.11. However the figure shows that, with 10 APs, such a client can achieve a throughput of 21 Mbps with MegaMIMO. Thus, using MegaMIMO for diversity can significantly expand the coverage range of an 802.11 deployment, and alleviate dead spots. This is expected because with MegaMIMO's coherent diversity, using APs to coherent combine the signal can provide a multiplicative increase in the SNR of N^2 [144]. This results in significant throughput improvements in the low SNR regime.

■ 5.11.6 Compatibility with 802.11

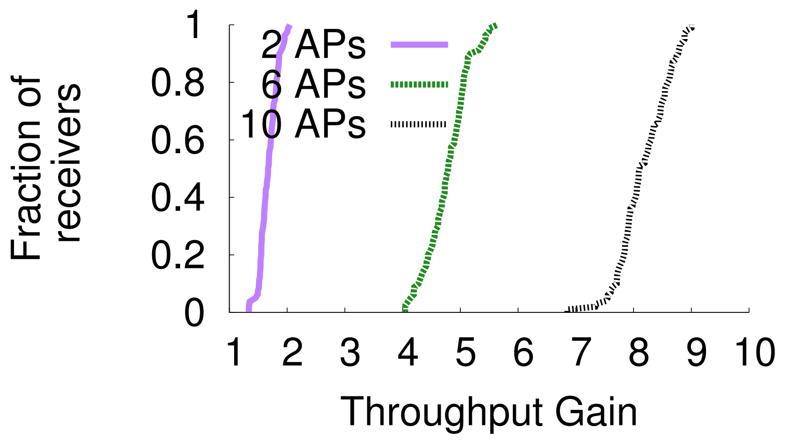
Finally, as described in Section 5.6, MegaMIMO is compatible with existing 802.11n cards. In this section, we investigate whether MegaMIMO can deliver significant throughput gains when used with commodity 802.11n cards. Further, since each AP and each 802.11n



(a) High SNR (> 18 dB)



(b) Medium SNR (12-18 dB)



(c) Low SNR (6-12 dB)

Figure 5-13: Fairness. CDFs of per-client throughput gain. Across all SNRs, MegaMIMO provides all clients with very similar gains.

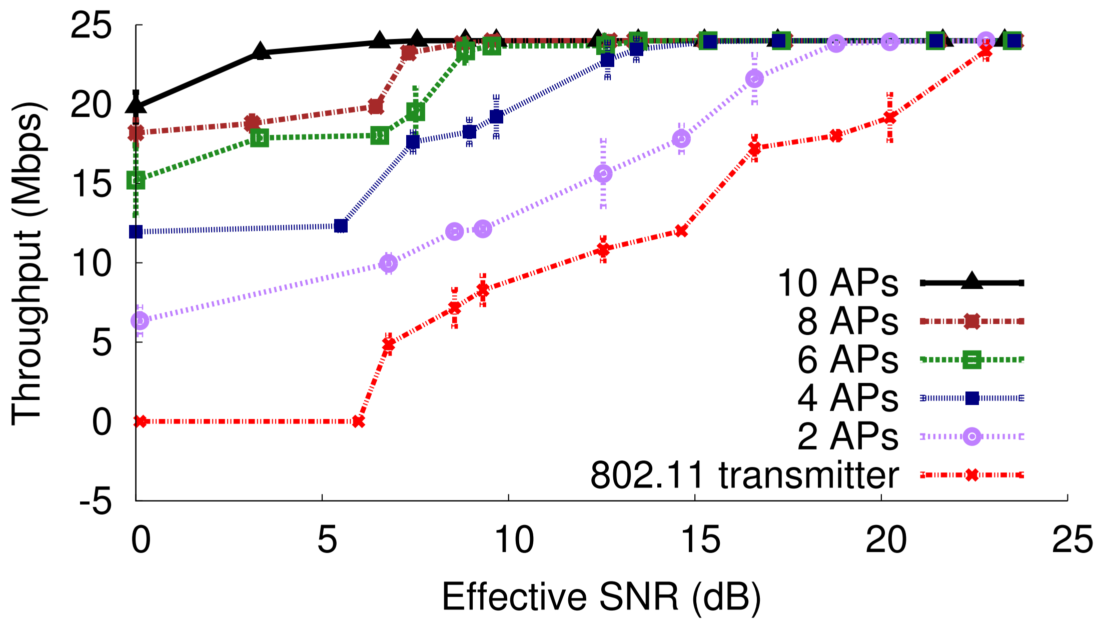


Figure 5-14: Diversity Throughput. Throughput of a MegaMIMO client when using diversity with 2, 4, 6, 8 and 10 APs. MegaMIMO can achieve close to maximum rate even to a client unable to receive any packets with 802.11.

card in this system has 2 antennas, this experiment also verifies that MegaMIMO can provide its expected gains with multi-antenna transmitters and receivers.

Method. We place 2 MegaMIMO nodes at random AP locations in the testbed, and 2 802.11n receivers at random client locations in the testbed. For each topology, we compute the total throughput with 802.11n and with MegaMIMO. As before, we compute the 802.11n throughput by giving each transmitter an equal share of the medium. We repeat the experiment across multiple topologies and the entire range of SNRs.

Results. Fig. 5-15 shows the total throughput with and without MegaMIMOat high, medium and low SNRs. Since we have two receivers in this experiment, the theoretically throughput gain compared to 802.11n is 2×. The chart shows that MegaMIMO delivers an average gain of 1.67-1.83× across all SNR ranges. Similar to the case with USRP receivers, the gains in the high SNR regime are larger than gains in the low SNR regime.

We now investigate MegaMIMO's fairness, *i.e.*, whether MegaMIMO can deliver its throughput gains for every receiver in the network across all locations and SNRs. Fig. 5-16 shows the CDF of the throughput gain achieved by MegaMIMO as compared to 802.11n

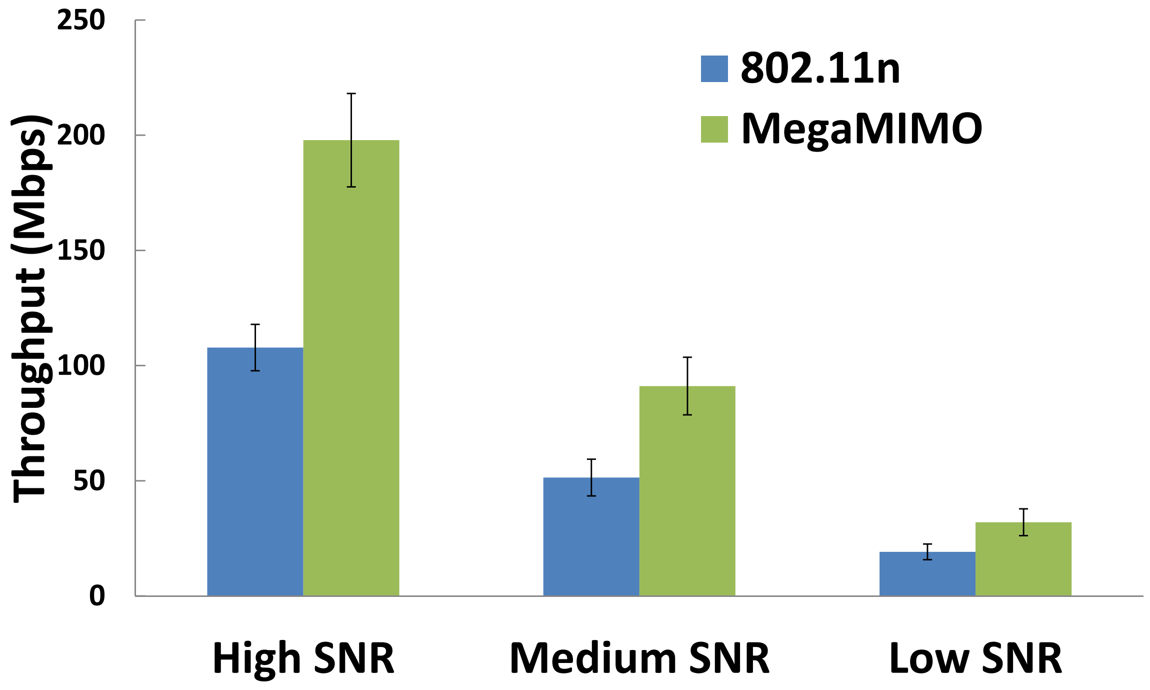


Figure 5-15: Throughput achieved using MegaMIMO on off-the-shelf 802.11n cards. MegaMIMO significantly improves the performance of off-the-shelf 802.11n cards at high (>18 dB), medium (12-18 dB) and low (6-12 dB) SNRs.

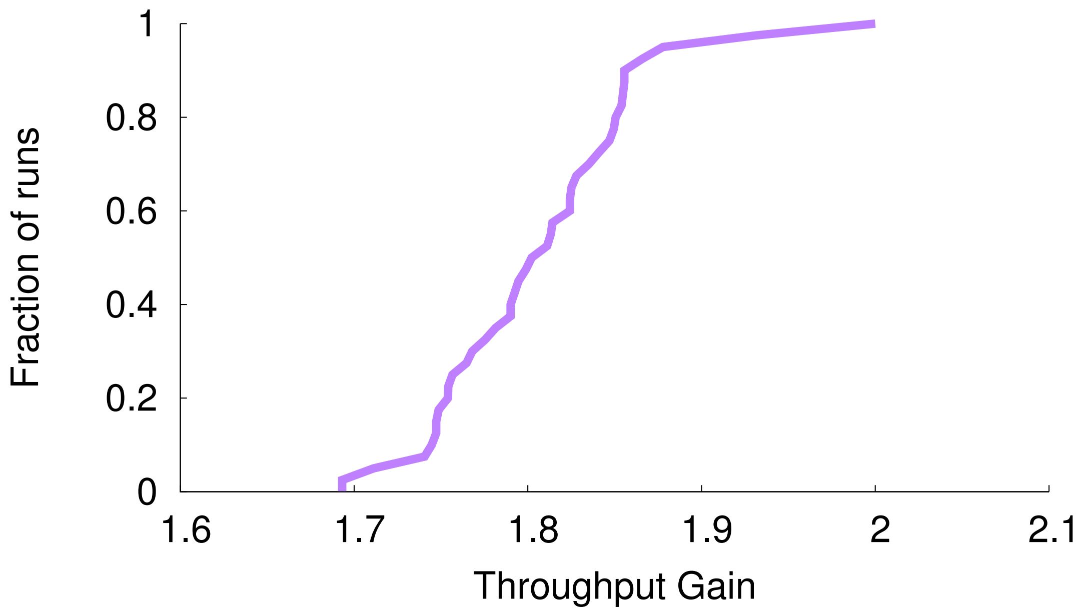


Figure 5-16: Fairness Results. For all nodes in our testbed, MegaMIMO delivers a throughput gain between 1.65-2×, with a median gain of 1.8× across SNRs. This shows that MegaMIMO provides similar throughput gains for every node in the network.

across all the runs. The results show that MegaMIMO delivers throughput gains between 1.65-2$\times$ for all the receivers and hence is fair to the receivers in the network.

■ 5.12 Discussion

This chapter enables joint beamforming from distributed independent transmitters. The key challenge in delivering this system is to perform accurate phase synchronization across multiple distributed transmitters. The lessons learnt from building the system and testing it with real hardware are : 1) Estimates of frequency offset can be made accurate enough to predict (and hence correct) phase misalignment within an 802.11 packet; however, these estimates cannot be used across multiple packets due to large build-ups in phase errors over time; and 2) Joint multi-user beamforming can be achieved by synchronizing the phases of all senders to one lead sender, and does not impose any phase synchronization constraints on the receivers.

We believe that the design of MegaMIMO has wider implications than explored in this chapter. In particular, several areas of information theory like lattice coding, noisy network coding, and transmitter cooperation for cognitive networks [109, 89, 97] assume tight phase synchronization across transmitters. The algorithms presented in this chapter can bring these ideas closer to practice.

CHAPTER 6

Discussion and Concluding Remarks

We have seen a proliferation of wireless technologies and mobile devices in recent years. This has resulted in rapid growth of wireless traffic, putting immense pressure on the limited wireless spectrum available. The looming spectrum crunch necessitates the development of new practical techniques to enhance spectrum utilization, and enable wireless networks to keep up with the burgeoning demand. .

■ 6.1 A Wireless Architecture for Cooperation and Cognition

This dissertation proposes an architecture for cooperation and cognition in wireless networks and shows that it can provide large gains in spectrum utilization in practice. Nodes in today's wireless networks typically only cooperate in a limited manner, orchestrating their transmissions so that they do not transmit at the same time in the same frequency. In contrast, this dissertation demonstrates systems that perform agile and fine-grained cognition and cooperation within and across technologies. Specifically, it introduces the following systems:

- SWIFT, a cognitive cross-technology solution that enables wideband devices to dynamically detect frequency bands occupied by narrowband devices, and weave the remaining, possibly non-contiguous, unoccupied frequency bands into a single high throughput link.

- FARA, a cooperative system for multi-channel wireless systems like 802.11 to dynamically utilize all available spectrum for all devices, allocating to each link the frequency bands that show the highest performance for that link.
- SourceSync, a system that synchronizes wireless nodes and enables them to transmit the same data simultaneously, in order to increase channel reliability.
- MegaMIMO, a system that linearly scales wireless throughput with the number of transmitters by enabling multiple transmitters to transmit simultaneously to multiple receivers in the same frequency bands without interfering with each other.

In addition to increasing spectrum utilization, these systems present novel coordination and synchronization primitives that can serve as a building block for other cooperative wireless systems. These include an adaptive spectrum sensing primitive that exploits higher layer network semantics to dynamically detect frequency bands occupied by narrowband devices using unknown modulation schemes (SWIFT), a distributed consensus scheme to agree on the frequency bands to use for communication in the absence of a control channel (SWIFT), a mechanism for fine-grained time synchronization (within 20 *ns*) of transmissions from distributed senders (SourceSync), and a protocol for phase-coherent transmission from distributed wireless transmitters driven by different oscillators (MegaMIMO).

The dissertation demonstrates how these primitives can be practically implemented using cross-layer techniques, provides prototype implementations of our systems, and shows that these implementations provide large gains in spectrum utilization compared to existing wireless designs. It thus lays a robust foundation for cooperative wireless systems that can dynamically share access to spectrum both within and across technologies.

■ 6.2 Future Work

The work on cooperation and cognition presented in this dissertation can be extended to act as underpinnings of future agile networks.

There has been significant interest in recent years in dynamic access to white space frequencies, which are the bands vacated by television channels after the move from analog to digital. Similarly, a recent PCAST study [115] has advocated dynamic and opportunistic access by secondary users to frequency bands that are currently reserved for use by the government. A key question is how access to these frequency bands will be mediated both between primary and secondary users, as well as between a diversity of secondary users. The FCC currently mandates the use of a spectrum database that maintains a list of unoccupied frequencies in each geographic location to ensure that secondary users do not interfere with primary users [146]. However, the exact mechanism by which different secondary users will share spectrum amongst themselves remains an open question. This is especially challenging since secondary users can use a variety of modulation techniques, frequency bands, transmission protocols *etc.*. A possible solution will be to develop a spectrum sharing etiquette that allows secondary users to adaptively probe the spectrum they are interested in *a la* SWIFT, and determine which parts of the spectrum they can use. Such a spectrum sharing etiquette would be useful not just for whitespaces, but for the large chunks of spectrum that are soon expected to become available for dynamic access by unlicensed devices.

This dissertation has presented systems such as FARA, SourceSync and MegaMIMO that synchronize wireless systems in frequency, time, and phase. The techniques presented in these systems can be used to build a unified wireless architecture that allows multiple wireless nodes to cooperate at the physical layer. Such a system could be useful, for instance, in wireless LANs or cellular networks, where the diversity of network conditions across space and time will require dynamically adopting different techniques and protocols for different transmitters and receivers. For instance, such a cooperative wireless system could use a mechanism like SourceSync for users in dead spots, while leveraging MegaMIMO to scale network throughput to users with good connectivity. Further, an architecture that coordinates multiple transmitters as a first order primitive simplifies handling of network functions like mobility. Building such a wireless network requires addressing various additional questions over and above the work in this dissertation: What are the different network components in such an architecture? What is the API presented by transmitting and receiving nodes to the network, both to expose physical layer information to the network, and for the network to control transmission and reception algorithms at the nodes? What are the algorithms and heuristics that the network should use to switch between different transmission and reception schemes?

With the increasing diversity of wireless devices and traffic patterns, it has become crucial that agility be a primary consideration in the design of wireless networks. In particular, wireless networks need to exploit opportunities for higher spectrum utilization arising from variations in spectral occupancy and channel conditions across time and space. This dissertation demonstrates practical wireless systems that do so using cross-layer algorithms for cooperation and cognition. The ideas underlying these systems will form the foundation of future high-throughput wireless systems.

References

[1] FCC Slides for UWB Spectral Limits. `http://sss-mag.com/uwbslides.html`. (Cited on page 74).

[2] IEEE 802.22 WG. `www.ieee802.org/22/`. (Cited on pages 32 and 53).

[3] Impact of devices using ultra-wideband technology on systems operating within radiocommunication services. (Cited on pages 50 and 53).

[4] Xilinx design tools center. (Cited on page 72).

[5] Uwb - intel standards, 2005. (Cited on page 54).

[6] IEEE standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages c1 –502, 29 2009. (Cited on pages 37, 127, 131 and 155).

[7] S. Aeron and V. Saligrama. Wireless ad hoc networks: Strategies and scaling laws for the fixed SNR regime. *IEEE Transactions on Inf. Theor.*, 53(6), 2007. (Cited on page 158).

[8] D. Agrawal, V. Tarokh, A. Naguib, and N. Seshadri. Space-time coded OFDM for high data-rate wireless communication over wideband channels. In *Proc. IEEE VTC*, volume 3, pages 2232–2236. Citeseer, 1998. (Cited on page 124).

[9] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. NeXt Generation/Dynamic Spectrum Access/Cognitive Radio Wireless Networks: A survey. In *Computer Networks Journal*. Elsevier, September 2006. (Cited on pages 32, 50, 51, 53, 57 and 116).

[10] S. Alamouti. A simple transmit diversity technique for wireless communications. *IEEE Journal on selected areas in communications*, 16(8):1451–1458, 1998. (Cited on pages 120, 124 and 137).

[11] J. G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. PrenticeHall, 2007. (Cited on pages 54, 92 and 116).

[12] E. Aryafar, N. Anand, T. Salonidis, and E. Knightly. Design and experimental evaluation of multi-user beamforming in wireless LANs. In *Mobicom 2010*. (Cited on page 158).

[13] 802.11a. `http://standards.ieee.org/getieee802/download/802.11a-1999.pdf`, 1999. IEEE. (Cited on pages 36, 51, 54, 94, 127 and 131).

[14] P. Bahl, R. Chandra, P. A. Chou, J. I. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu. KNOWS: Kognitiv Networking Over White Spaces. In *IEEE DySPAN, 2007*. (Cited on pages 32, 50 and 53).

[15] S. R. Banerjee, R. Jesme, and R. A. Sainati. Investigation of spatial and frequency diversity for long range UHF RFID. In *IEEE Antennas and Propagation Society International Symposium*, San Diego, CA, 2008. (Cited on pages 90 and 116).

[16] S. Berger and A. Wittneben. Carrier phase synchronization of multiple distributed nodes in a wireless network. In *8th IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC), Helsinki, Finland*, June 2007. (Cited on page 158).

[17] H. L. Bertoni. Propagation effects observed indoors. `http://eeweb.poly.edu/faculty/bertoni/docs/06IndoorObserve.pdf`, 2005. (Cited on pages 90 and 93).

[18] J. Bicket. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Institute of Technology, 2005. (Cited on pages 112, 139 and 150).

[19] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. *ACM SIGCOMM, 2005*. (Cited on pages 39, 47, 118, 140 and 141).

[20] H. Bölcskei. *Principles of MIMO-OFDM wireless systems*. 2004. (Cited on page 54).

[21] V. Brik, E. Rozner, S. Banarjee, and P. Bahl. DSAP: A Protocol for Coordinated Spectrum Access. 2005. (Cited on pages 32 and 53).

[22] M. M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans. DIMSUMNet: New Directions in Wireless Networking Using Coordinated Dynamic Spectrum Access. 2005. (Cited on pages 32 and 53).

[23] D. Cabric, S. M. Mishra, D. Willkomm, R. Brodersen, and A. Wolisz. A cognitive radio approach for usage of virtual unlicensed spectrum, 2005. (Cited on pages 50 and 53).

[24] J. Camp and E. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. In *ACM Mobicom 2008*, San Francisco, CA, September 2008. (Cited on page 96).

[25] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury, 2nd edition, 2002. (Cited on page 62).

[26] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proc. of ACM SIGCOMM 2007, Kyoto, Japan*. (Cited on pages 39 and 118).

[27] M. S.-W. Chen and R. W. Brodersen. A subsampling uwb radio architecture by analytic signaling. In *ICASSP*, 2004. (Cited on page 52).

[28] W. Choi and J. G. Andrews. Downlink performance and capacity of distributed antenna systems in a multicell environment. *IEEE Trans. on Wireless Comms.*, 6(1):69–73, January 2007. (Cited on page 123).

[29] P. Chow, J. Cioffi, and J. Bingham. A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels. *IEEE Transactions on Communications*, 48, 1995. (Cited on page 91).

[30] H. K. Chung and H. L. Bertoni. Indoor propagation characteristics at 5.2 GHz in home and office environments. *Journal of Communication and Networks*, 4:176–188, 2002. (Cited on pages 90 and 93).

[31] CNN. Cutting the cord to flat-screen TVs. `http://www.cnn.com/2008/TECH/01/03/wirelesshigh.def.ap/index.html`, January 2008. (Cited on pages 49, 52 and 87).

[32] S. S. Company. Spectrum occupancy measurement, 2007. (Cited on pages 35, 36, 50 and 53).

[33] T. Cover and A. E. Gamal. Capacity theorems for the relay channel. *IEEE Trans. Inf. Theory*, 25(5):572–584, Sept. 1979. (Cited on page 32).

[34] Distributed Antenna Systems. http://medicalconnectivity.com/2008/02/05/distributed-antenna-systems-no-replacement-for-wireless-strategy. (Cited on pages 123 and 158).

[35] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03*, San Diego, California, September 2003. (Cited on page 141).

[36] A. Dutta, D. Saha, D. Grunwald, and D. Sicker. SMACK: a SMart ACKnowledgment scheme for broadcast messages in wireless networks. In *ACM SIGCOMM*, Barcelona, Spain, 2009. (Cited on pages 39, 118 and 124).

[37] F. Edalat. *Real-time Sub-carrier Adaptive Modulation and Coding in Wideband OFDM Wireless Systems*. PhD thesis, Massachusetts Institute of Technology, 2008. (Cited on pages 47, 49, 52, 54, 71, 122 and 142).

[38] F. Edalat, J. K. Tan, K. M. Nguyen, N. Matalon, and C. G. Sodini. Measured Data Rate from Adaptive Modulation in Wideband OFDM Systems. In *IEEE International Conference on UWB*, Waltham, MA, Sept. 2006. (Cited on page 98).

[39] E. Eleftheriou and S. Olcer. Low-density parity-check codes for digital subscriber lines. In *Proceedings of IEEE International Conference on Communications (ICC)*, volume 3, pages 1752–1757, 2002. (Cited on page 91).

[40] S. C. Ergen. Zigbee/ieee 802.15.4 summary, 2004. (Cited on pages 36 and 51).

[41] USRP. `http://www.ettus.com`. Ettus Inc. (Cited on pages 47 and 176).

[42] Second Rep. and Order and Memorandum Opinion and Order, November 2008. FCC 08-260. (Cited on pages 87 and 116).

[43] Mobile broamobile broadband: The benefits of additional spectrum. `http://download.broadband.gov/plan/fcc-staff-technical-paper-mobile-broadband-benefits-of-additional-spectr` `pdf`. Federal Communications Commission. (Cited on page 31).

[44] R. F. Fischer and J. B. Huber. A new loading algorithm for discrete multitone transmission. In *Proceedings of IEEE Global Telecommunications Conference (GlobeCom)*, volume 1, pages 724–728, 18-22 November 1996. (Cited on page 91).

[45] J. Foerster. Channel modeling sub-committee report, February 2003. IEEE P802.15 Wireless Personal Area Networks. (Cited on page 116).

[46] A. Forenza, R. W. H. Jr., and S. G. Perlman. *System and Method For Distributed Input-Distributed Output Wireless Communications*. U.S. Patent Application number 20090067402. (Cited on page 157).

[47] System for increasing capacity in future mobile communications networks. `http://www.hhi.fraunhofer.de/fileadmin/hhi/downloads/BM/PR_Demonstration_Network_MIMO.pdf`. Fraunhofer Heinrich Hertz Institute. (Cited on page 157).

[48] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. (Cited on page 101).

[49] M. S. Gast. *802.11 Wireless Networks*. O'Reilly, 2nd edition, 2005. (Cited on page 69).

[50] New Wireless Standard Promises Ultra-Fast Media Applications. `http://www.sciencedaily.com/releases/2009/01/090122161953.htm`, January 2009. Science Daily. (Cited on page 116).

[51] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005. (Cited on page 129).

[52] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005. (Cited on page 164).

[53] S. Gollakota and D. Katabi. ZigZag Decoding: Combating Hidden Terminals in Wireless Networks. In *Sigcomm*, 2008. (Cited on pages 39, 104, 118, 120 and 124).

[54] S. Gollakota, S. Perli, and D. Katabi. Interference alignment and cancellation. *ACM SIGCOMM*, 2009. (Cited on page 158).

[55] Greentouch consortium. `https://www.youtube.com/watch?v=U3euDDr0uvo`. GreenTouch Demonstrates Large-Scale Antenna. (Cited on page 158).

[56] 802.11g. `http://standards.ieee.org/getieee802/download/802.11g-2003.pdf`, 2003. IEEE. (Cited on pages 54, 94, 127 and 131).

[57] S. Guha, K. Munagala, and S. Sarkar. Jointly optimal transmission and probing strategies for multichannel wireless systems. *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 955–960, March 2006. (Cited on page 91).

[58] R. Gummadi and H. Balakrishnan. Wireless networks should spread spectrum based on demands. In *Proc. ACM Hotnets, Calgary*, Oct 2008. (Cited on pages 37, 87, 91, 93 and 103).

[59] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: Interference Cancellation for wireless LANs. In *ACM Mobicom*, 2008. (Cited on pages 120 and 124).

[60] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: gathering 802.11n traces with channel state information. *SIGCOMM Comput. Commun. Rev.*, 41:53–53. (Cited on page 177).

[61] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM SIGCOMM*, 2010. (Cited on pages 175, 176 and 180).

[62] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller. Maranello: practical partial packet recovery for 802.11. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 14–14, Berkeley, CA, USA, 2010. USENIX Association. (Cited on page 176).

[63] J. Heiskala and J. Terry. *OFDM Wireless LANs: A Theoretical & Practical Guide*. Sams Publishing, 2001. (Cited on pages 64, 65, 94, 95, 129, 136, 155 and 165).

[64] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless LANs. *SIGCOMM Comput. Commun. Rev.*, 35(4):121–132, 2005. (Cited on page 103).

[65] D. Hughes-Hartogs. *Ensemble Modem Structure for Imperfect Transmission Media*. U.S. Patents no. 4,679,227 (July 1987), 4,731,816 (March 1988), and 4,833,706 (May 1989). (Cited on page 91).

[66] The iPad and its impact on hotel owners and operators. `http://www.ibahn.com/en-us/public/docs/The_Impact_of_iPad.pdf`. iBAHN. (Cited on pages 41 and 153).

[67] J. M. III and G. Q. M. JR. Cognitive radio: Making software radios more personal. *IEEE Personal Communications*, 1999. (Cited on page 32).

[68] Wireless channel bonding. `http://www.intel.com/support/wireless/sb/CS-025343.htm`. Intel Technologies. (Cited on pages 87 and 93).

[69] Enabling high-speed wireless personal area networks. `http://www.usb.org/wusb/docs/Ultra-Wideband.pdf`, 2005. Intel White Paper. (Cited on pages 49, 52, 87 and 116).

[70] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, Aug. 1988. (Cited on pages 36 and 51).

[71] H. Jafarkhani. A quasi-orthogonal space-time block code. *IEEE Transactions on Communications*, 49(1):1–4, 2001. (Cited on pages 120, 124 and 137).

[72] R. Jain, W. Hawe, and D. Chiu. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report DEC-TR-301, DEC, September 1984. (Cited on page 115).

[73] S. Jakubczak, D. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan. Link-alike: Using Wireless to Share Network Resources in a Neighborhood. In *MC2R*, 2008. (Cited on pages 121 and 138).

[74] D. C. Jenn, J. H. Ryu, T. Yen-Chang, and R. Broadston. Adaptive phase synchronization in distributed digital arrays. In *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, pages 199 –204, june 2010. (Cited on page 157).

[75] D. Jiang, Q. Chen, and L. Delgrossi. Optimal data rate selection for vehicle safety communications. In *Vehicular Ad Hoc Networks*, 2008. (Cited on page 98).

[76] I. Kalet. The multitone channel. *IEEE Transactions on Communications*, 37:119–124, February 1989. (Cited on page 91).

[77] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, April 2008. (Cited on page 87).

[78] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *SIGCOMM*, 2007. (Cited on pages 39, 118 and 124).

[79] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level Network Coding for Wireless Mesh Networks. In *ACM Sigcomm*, 2008. (Cited on pages 39 and 118).

[80] S.-C. Kim, H. Bertoni, and M. Stern. Pulse propagation characteristics at 2.4 GHz inside buildings. *Vehicular Technology, IEEE Transactions on*, 45(3):579–592, Aug 1996. (Cited on pages 90 and 93).

[81] G. Kramer, M. Gastpar, and P. Gupta. Cooperative strategies and capacity theorems for relay networks. *IEEE Trans. Inf. Theory*, 51(9):3037–3063, Sept. 2005. (Cited on page 33).

[82] G. Kramer, I. Marić, and R. Yates. Cooperative communications. *Found. in Networking*, 1(3):425, 2006. (Cited on page 123).

[83] S. K. Lai, R. S. Chen, K. B. Lataief, and R. D. Murch. Adaptive trellis coded MQAM and power optimization for OFDM transmission. In *Proceedings of IEEE Vehicular Technology Conference*, volume 1, pages 290–294, 16-20 May 1999. (Cited on page 91).

[84] J. N. Laneman, D. N. C. Tse, and G. W. Wornell. Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *IEEE Trans. on Inform. Theory, Volume 50, Issue 12, Dec 2004.* (Cited on pages 33, 39 and 123).

[85] Lattice announces low-cost FPGA-based ADC interface reference design solution. `http://www.latticesemi.com/corporate/newscenter/productnews/2008/r080114announceslowcostfp.cfm`, 2008. Lattice Semiconductors. (Cited on page 87).

[86] V. K. N. Lau and M. D. Macleod. Variable-rate adaptive trellis coded QAM for flat-fading channels. *IEEE Transactions on Communications*, 49:1550–1560, September 2001. (Cited on page 91).

[87] X. Li. Space-time coded multi-transmission among distributed transmitters without perfect synchronization. *Signal Processing Letters, IEEE*, 11(12):948 – 951, Dec. 2004. (Cited on page 124).

[88] Y. Li and J. Moon. Increasing data rates through iterative coding and antenna diversity in OFDM-based wireless communication. In *Proceedings of IEEE Conference on Global Telecommunications (GlobeCom)*, volume 5, pages 3130–3134, November 2001. (Cited on page 91).

[89] S. Lim, Y. Kim, A. El Gamal, and S. Chung. Noisy network coding. In *IEEE Info. Theor. Workshop*, 2010. (Cited on pages 32 and 194).

[90] K. C.-J. Lin, S. Gollakota, and D. Katabi. Random access heterogeneous mimo networks. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pages 146–157, New York, NY, USA, 2011. ACM. (Cited on pages 158 and 175).

[91] K. C.-J. Lin, N. Kushman, and D. Katabi. ZipTx: Harnessing Partial Packets in 802.11 Networks. (Cited on page 176).

[92] LTE: MIMO techniques in 3GPP-LTE. `http://lteportal.com/Files/MarketSpace/Download/130_LTEMIMOTechniquesFreescaleNov52008.pdf`. (Cited on page 158).

[93] L. Ma, X. Han, and C.-C. Shen. Dynamic open spectrum sharing mac protocol for wireless ad hoc networks. (Cited on pages 32 and 53).

[94] Z. Ma, M. Zierdt, J. Pastalan, A. Siegel, T. Sizer, A. J. de Lind van Wijngaarden, P. R. Kasireddy, and D. M. Samardzija. Radiostar: Providing wireless coverage over gigabit ethernet. *Bell Lab. Tech. J.*, 14:7–24, May 2009. (Cited on page 158).

[95] V. Mannoni, D. Declercq, and G. Gelle. Optimized irregular low-density parity-check codes for multicarrier modulations over frequency-selective channels. *EURASIP Journal on Applied Signal Processing*, 204(10):1546–1556, 2004. (Cited on page 91).

[96] J. Manweiler et al. Order matters: transmission reordering in wireless networks. In *ACM Mobicom'09*. (Cited on page 124).

[97] I. Maric, N. Liu, and A. Goldsmith. Encoding against an interferer's codebook. In *Allerton*, 2008. (Cited on page 194).

[98] N. Matalon. An Implementation of a 5.25 GHz Transceiver for High Data Rate Wireless Applications. Master's thesis, MIT, EECS, July 2005. (Cited on pages 71, 88, 89 and 104).

[99] D. M. Matic, H. Harada, and R. Prasad. Indoor and outdoor frequency measurements for mm-waves in the range of 60 ghz. In *Proceedings IEEE 48th Vehicular Technology Conference*, pages 567–571, 1998. (Cited on pages 90 and 116).

[100] M. McHenry. Frequency agile spectrum access technologies, 2003. (Cited on pages 35, 50 and 53).

[101] E. C. V. D. Meulen. Three-terminal communication channels. *Adv. Appl. Probab.*, 3:120–154, June 1971. (Cited on page 32).

[102] H. Meyr, M. Moeneclaey, and S. A. Fechtel. *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. John Wiley, 1998. (Cited on page 135).

[103] MIMO schemes in 16m (WiMAX 2.0). `http://www.wimax360.com/profiles/blogs/mimo-schemes-in-16m-wimax-20`. WiMax360. (Cited on page 158).

[104] S. Mishra, R. Brodersen, S. Brink, and R. Mahadevappa. Detect and avoid: an ultra-wideband/wimax coexistence mechanism [topics in radio communications]. *Communications Magazine, IEEE*, 45(6):68–75, June 2007. (Cited on page 53).

[105] M. Mittelbacht, C. Mullert, D. Fergert, and A. Fingert. Study of coexistence between uwb and narrowband cellular systems, 2004. (Cited on pages 50 and 53).

[106] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005. (Cited on pages 121, 138, 139 and 176).

[107] A. K. L. Miu, G. Tan, H. Balakrishnan, and J. G. Apostolopoulos. Divert: Fine-grained path selection for wireless lans. In *MobiSys*, 2004. (Cited on pages 39, 118 and 124).

[108] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-aware spectrum distribution in wireless LANs. In *International Conference on Network Protocols*, Oct 2008. (Cited on pages 37, 87, 91, 92, 93 and 103).

[109] B. Nazer and M. Gastpar. The case for structured random codes in network capacity theorems. *European Transactions on Telecommunications*, 19(4), 2008. (Cited on pages 152 and 194).

[110] Network MIMO. http://www.alcatel-lucent.com/wps/DocumentStreamerServlet?LMSG_CABINET=Docs_and_Resource_Ctr&LMSG_CONTENT_FILE=Data_Sheets/Network_MIMO.pdf. Alcatel-Lucent. (Cited on page 157).

[111] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals & systems*. Prentice-Hall, Inc., 1996. (Cited on pages 128 and 129).

[112] A. Ozgur, R. Johari, D. Tse, and O. Leveque. Information-theoretic operating regimes of large wireless networks. *IEEE Trans. on Info. Theor.*, 2010. (Cited on page 33).

[113] A. Ozgur, O. Leveque, and D. Tse. Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks. *IEEE Trans. on Info. Theor.*, 2007. (Cited on pages 33 and 158).

[114] Y. N. Papantonopoulos. High-speed ADC technology paves the way for software defined radio. http://www.rfdesignline.com/showArticle.jhtml?articleID=201202962, 2007. (Cited on page 87).

[115] Realizing the full potential of government-held spectrum to spur economic growth. http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast_spectrum_report_final_july_20_2012.pdf. President's Council of Advisors on Science and Technology. (Cited on page 196).

[116] E. Perahia and M. X. Gong. Gigabit wireless lans: an overview of ieee 802.11ac and 802.11ad. *Mobile Computing and Communications Review*, 15(3):23–33, 2011. (Cited on page 37).

[117] S. Plass, A. Dammann, S. Kaiser, and K. Fazel. Space-time-frequency diversity in the next generation of terrestrial digital video broadcasting. In *Multi-Carrier Systems and Solutions*, pages 101–110. Springer Verlag, Netherlands, 2009. (Cited on pages 90 and 116).

[118] J. Proakis and M. Salehi. *Digital Communications*. McGraw-Hill, 5th edition, 2007. (Cited on page 71).

[119] H. Rahul, F. Edalat, D. Katabi, and C. Sodini. Frequency-Aware Rate Adaptation and MAC Protocols. In *ACM MOBICOM 2009*, Beijing, China, September 2009. (Cited on pages 10, 13, 39, 118, 121 and 175).

[120] H. Rahul, H. Hassanieh, and D. Katabi. Sourcesync: a distributed wireless architecture for exploiting sender diversity. *SIGCOMM*, 2010. (Cited on pages 10, 13, 158, 164 and 168).

[121] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to Share: Narrowband-Friendly Wideband Networks. In *ACM SIGCOMM 2008*, Seattle, WA, August 2008. (Cited on pages 10, 13, 87, 94 and 116).

[122] H. S. Rahul, S. Kumar, and D. Katabi. Megamimo: scaling wireless capacity with user demands. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 235–246, New York, NY, USA, 2012. ACM. (Cited on pages 10 and 13).

[123] Distributed-input distributed-output wireless technology. `http://www.rearden.com/DIDO/DIDO_White_Paper_110727.pdf`. Rearden Companies. (Cited on page 157).

[124] S. Redl, M. Weber, and M. W. Oliphant. *GSM And Personal Communications Handbook*. Artech House, 1998. (Cited on page 92).

[125] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu. SOAR: Simple Opportunistic Adaptive Routing Protocol for Wireless Mesh Networks. In *IEEE TMC* , 2009. (Cited on pages 39 and 118).

[126] Mobile broadband capacity constraints and the need for optimization. `http://rysavy.com/Articles/2010_02_Rysavy_Mobile_Broadband_Capacity_Constraints.pdf`. Rysavy Research. (Cited on pages 31, 41 and 153).

[127] A. Sabharwal, A. Khoshnevis, and E. Knightly. Opportunistic spectral usage: bounds and a multi-band CSMA/CA protocol. *IEEE/ACM Trans. Netw.*, 15(3):533–545, 2007. (Cited on page 91).

[128] A. Sendonaris, E. Erkip, and B. Aazhang. User cooperation diversity. Part I. System description. *IEEE Transactions on Communications*, 51(11):1927–1938, 2003. (Cited on page 123).

[129] A. Sendonaris, E. Erkip, and B. Aazhang. User cooperation diversity. part ii. implementation aspects and performance analysis. *Communications, IEEE Transactions on*, 51(11):1939–1948, Nov. 2003. (Cited on page 123).

[130] S. Shamai, O. Somekh, and B. M. Zaidel. Multi-cell communications: An information theoretic perspective. In *Workshop on Communications and coding*, 2004. (Cited on page 152).

[131] L. Shen-fa and W. Wei-ling. Fast antenna selection algorithms for distributed mimo systems. In *Journal of Beijing University of Posts and Telecommunication*, volume 30, pages 50–53, Jun. 2007. (Cited on page 175).

[132] O. Shin, A. Chan, H. Kung, V. Tarokh, et al. Design of an OFDM cooperative space-time diversity system. *IEEE Transactions on Vehicular Technology*, 56(4):2203, 2007. (Cited on page 123).

[133] V. Shrivastava, N. Ahmed, S. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra. Centaur: realizing the full potential of centralized wlans through a hybrid data path. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, MobiCom '09, pages 297–308, New York, NY, USA, 2009. ACM. (Cited on page 175).

[134] A. G. Siamarou and M. O. Al-Nuaimi. Wideband propagation measurements for indoor Rician fading radio channels at 62.4 GHz. In *Proceedings IEEE 53rd Vehicular Technology Conference*, pages 449–453, 2001. (Cited on pages 90 and 116).

[135] O. Simeone, O. Somekh, H. Poor, and S. Shamai. Distributed MIMO in multi-cell wireless systems via finite-capacity links. In *ISCCSP*, 2008. (Cited on page 158).

[136] T. Starr, J. Cioffi, and P. Silverman. *Understanding Digital Subscriber Line Technology*. Prentice Hall PTR, 1999. (Cited on page 91).

[137] Further advancements for E-UTRA: Physical layer aspects, rel. 9, June 2009. Tech Specification Group Radio Access Network. (Cited on page 124).

[138] Local and metropolitan area networks, part 16: Air interface for fixed broadband wireless access systems: Amendment 1: Multihop relay specification, May 2009. IEEE. (Cited on page 123).

[139] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine-grained channel access in wireless lan. In *ACM SIGCOMM 2010*. (Cited on pages 155 and 158).

[140] K. Tan, H. Liu, J. Fang, W. Wang, J. Zhang, M. Chen, and G. M. Voelker. SAM: enabling practical spatial multiple access in wireless LAN. In *MobiCom 2009*. (Cited on page 158).

[141] R. Tandra and A. Sahai. Snr walls for signal detection. In *IEEE Journal on Special Topics in Signal Processing*, Feb. 2008. (Cited on pages 51, 54 and 57).

[142] V. Tarokh, N. Seshadri, and A. Calderbank. Space-time codes for high data rate wireless communication: Performance criterion and code construction. *IEEE transactions on information theory*, 44(2):744–765, 1998. (Cited on pages 120, 124 and 137).

[143] I. Thibault, G. Corazza, and L. Deambrogio. Phase synchronization algorithms for distributed beamforming with time varying channels in wireless sensor networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 77 –82, july 2011. (Cited on page 158).

[144] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005. (Cited on pages 37, 85, 90, 92, 182, 183, 189 and 190).

[145] Turn off your wi-fi network! `http://www.youtube.com/watch?v=fFiJ5rnIPVw`. Steve Jobs iPhone4 Keynote. (Cited on pages 41 and 153).

[146] White space database administration. `http://www.fcc.gov/encyclopedia/white-space-database-administration`. Federal Communications Commission. (Cited on page 197).

[147] J. Veillcux, P. Fortier, and S. Roy. An FPGA implementation of an OFDM adaptive modulation system. *3rd Intl IEEE-NEWCAS Conference*, 2005. (Cited on page 91).

[148] S. Venkatesan et al. A WiMAX-based implementation of network MIMO for indoor wireless. *EURASIP*, '09. (Cited on page 158).

[149] S. Verdu. *Multiuser Detection*. Cambridge University, 1998. (Cited on page 120).

[150] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-Layer Wireless Bit Rate Adaptation. In *ACM SIGCOMM 2009*, Barcelona, Spain, August 2009. (Cited on pages 96 and 139).

[151] C. Williams, S. McLaughlin, and M. Beach. Robust OFDM timing synchronisation in multipath channels. *EURASIP Jrnl on Wireless Comm. and Networking*, 2008:7, 2008. (Cited on pages 39, 119 and 126).

[152] C. Wilmot. Intel demonstrates fast new UWB WPAN at IDF Taiwan. `http://www.tweaktown.com/articles/968/intel_demonstrates_fast_new_ultrawideband_wpan_at_idf_taiwan/index.html`, 2006. (Cited on pages 49, 50, 52, 53 and 87).

[153] Applications for WiMAX. `http://www.wimax.com/education/wimax/ims`, 2009. WiMax.com. (Cited on page 116).

[154] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006. (Cited on pages 103 and 139).

[155] G. Woo, P. Kheradpour, and D. Katabi. Beyond the bits: Cooperative packet recovery using phy information. (Cited on pages 39, 118, 121, 138 and 139).

[156] M. Wouters, G. Vanwijnsberghe, P. V. Wesemael, T. Huybrechts, and S. Thoen. Real time implementation on FPGA of an OFDM based wireless LAN modem extended with adaptive loading. *Proceedings of the 28th European Solid-State Circuits Conference*, pages 531 – 534, September 2002. (Cited on page 91).

[157] Y. Wu, P. A. Chou, and S.-Y. Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. In *Proc. of CISS 2005, Baltimore, MD.* (Cited on page 33).

[158] Z. Xin-sheng, Y. Xiao-hu, and Z. Ding-qian. An investigation on dynamic rau selection method for distributed radio mobile communications system. In *Journal of Electronics Information Technology*, volume 28, pages 2334–2338, Dec. 2006. (Cited on page 175).

[159] L. Ying-Dong and Z. Guang-Xi. Transmit-receive antenna selection for distributed multi-user mimo downlink channels. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 2767 –2770, dec. 2009. (Cited on page 175).

[160] J. Zhang, J. Jia, Q. Zhang, and E. M. K. Lo. Implementation and Evaluation of Cooperative Communications in Software-Defined Radio Testbed. In *INFOCOM*, San Diego, CA, 2010. (Cited on page 124).

[161] Y. J. Zhang and K. B. Lataeif. Single- and multi-user adaptive pragmatic trellis coded modulation for OFDM system. In *Proceedings of IEEE Wireless Communications and Networking Conference*, volume 1, pages 9–14, 16-20 March 2003. (Cited on page 91).

[162] J. Zhao, H. Zheng, and G. Yang. Distributed coordination in dynamic spectrum allocation networks. (Cited on pages 32 and 53).

[163] Q. Zhao, L. Tong, and A. Swami. Decentralized cognitive mac for dynamic spectrum access. (Cited on pages 32 and 53).

[164] ZigBee Alliance. *ZigBee and Wireless Radio Frequency Coexistence*, June 2007. White Paper. (Cited on page 58).