

RICE UNIVERSITY

**Design, Implementation and Characterization of a  
Cooperative Communications System**

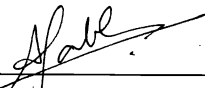
by

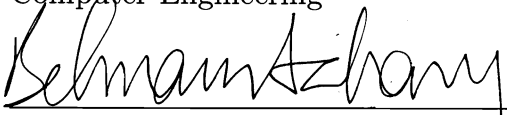
**Patrick O. Murphy**


A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE


**Doctor of Philosophy**

APPROVED, THESIS COMMITTEE:

  
\_\_\_\_\_  
Ashutosh Sabharwal, Chair  
Associate Professor of Electrical and  
Computer Engineering

  
\_\_\_\_\_  
Behnaam Aazhang  
J.S. Abercrombie Professor of Electrical  
and Computer Engineering

  
\_\_\_\_\_  
Edward W. Knightly  
Professor of Electrical and Computer  
Engineering

  
\_\_\_\_\_  
David B. Johnson  
Professor of Computer Science

Houston, Texas

December 2010

## ABSTRACT

### Design, Implementation and Characterization of a Cooperative Communications System

by

Patrick O. Murphy

Cooperative communications is a class of techniques which seek to improve reliability and throughput in wireless systems by pooling the resources of distributed nodes. While cooperation can occur at different network layers and time scales, physical layer cooperation at symbol time scales offers the largest benefit. However, symbol level cooperation poses significant implementation challenges, especially in the context of a network of distributed nodes.

We first present the design and implementation of a complete cooperative physical layer transceiver, built from scratch on the Wireless Open-Access Research Platform (WARP). In our implementation fully distributed nodes employ physical layer cooperation at symbol time scales without requiring a central synchronization source. Our design supports per-packet selection of non-cooperative or cooperative communication, with cooperative links utilizing either amplify and forward or decode and forward relaying. A single design implements transmission, reception and relaying, allowing each node to assume the role of source, destination or relay per packet.

We also present experimental methodologies for evaluating our design and extensive experimental results of our transceiver's performance under a variety of topologies and propagation conditions. Our methods are designed to test both overall perfor-

mance and to isolate and understand the underlying causes of performance bottlenecks. Our results clearly demonstrate significant performance gains (more than 50× improvement in PER in some topologies) provided by physical layer cooperation even when subject to the constraints of a real-time implementation.

As with all our work on WARP, our transceiver design and experimental framework are available via the open-source WARP repository for use by other wireless researchers.

# Contents

Abstract	ii
List of Illustrations	viii
List of Tables	xviii
<b>1 Introduction</b>	<b>1</b>
1.1 Objectives and Challenges . . . . .	2
1.2 Summary of Contributions . . . . .	3
<b>2 Background</b>	<b>5</b>
2.1 Platform Selection . . . . .	5
2.2 Cooperative Schemes . . . . .	6
2.3 OFDM . . . . .	7
2.4 Distributed Space-Time Block Code . . . . .	9
2.5 Networking Support . . . . .	10
<b>3 Carrier Frequency Offsets</b>	<b>12</b>
3.1 Origin of CFO . . . . .	13
3.2 Impact of CFO in OFDM . . . . .	15
3.3 Expectations for CFO on WARP . . . . .	17
3.4 Measured CFO in Hardware . . . . .	18
3.5 Time Domain CFO Correction . . . . .	20
3.5.1 Time Domain CFO Estimation Algorithm . . . . .	21
3.5.2 Performance Expectations . . . . .	23
3.5.3 Radio Transients . . . . .	24

3.5.4	Performance Measurements . . . . .	34
3.6	Frequency Domain Phase Correction . . . . .	37
3.7	CFO in a Cooperative System . . . . .	41
3.7.1	Temporal Properties of CFO . . . . .	42
3.7.2	Mitigating CFO in a Cooperative System . . . . .	46
3.7.3	CFO with Amplify and Forward . . . . .	48
3.7.4	CFO with Decode and Forward . . . . .	49
3.7.5	Frequency Domain Residual CFO Estimation . . . . .	54
3.8	Conclusions . . . . .	61
<b>4</b>	<b>Physical Layer Transceiver Design</b>	<b>62</b>
4.1	Key Subsystems . . . . .	64
4.1.1	Alamouti Encoding . . . . .	65
4.1.2	Packet Buffers . . . . .	67
4.1.3	Frame Format . . . . .	70
4.1.4	Energy Detector . . . . .	73
4.1.5	Receiver State Machine . . . . .	74
4.1.6	Packet Timing Correlator . . . . .	76
4.1.7	Waveform Buffer . . . . .	82
4.2	Auto-Response System . . . . .	83
4.2.1	Rx→Tx Turnaround in a Cooperative System . . . . .	84
4.2.2	Header Match Units . . . . .	85
4.2.3	Actions . . . . .	86
4.2.4	Header Translation . . . . .	88
4.2.5	Timing . . . . .	89
4.3	Designing for Characterization . . . . .	90
4.3.1	Carrier Frequency Offset . . . . .	91
4.3.2	Packet Detection . . . . .	91

4.3.3	Random Payload Generation . . . . .	92
4.3.4	Per-Packet Measurements . . . . .	93
<b>5</b>	<b>Experimental Methodologies and Metrics</b>	<b>96</b>
5.1	Node Design . . . . .	96
5.2	Channel Emulator . . . . .	99
5.2.1	Connections . . . . .	99
5.2.2	Channel Models . . . . .	101
5.3	Topologies . . . . .	103
5.4	Cooperative Schemes . . . . .	108
5.5	Methodology . . . . .	110
5.5.1	WARPnet Framework . . . . .	111
5.5.2	Node Behaviors Across Time Slots . . . . .	114
5.5.3	Interleaving Modes . . . . .	117
5.5.4	Interleaving Trials . . . . .	119
5.6	Metrics . . . . .	120
5.6.1	Packet Error Rate . . . . .	121
5.6.2	Bit Error Rate . . . . .	122
<b>6</b>	<b>Experiments with Full Cooperative Transceiver</b>	<b>124</b>
6.1	Experimental Parameters . . . . .	125
6.2	Co-located Source/Relay Topology . . . . .	127
6.2.1	PER/BER with 1412 Byte Payloads . . . . .	128
6.2.2	PER/BER with 692 Byte Payloads . . . . .	131
6.2.3	Observations . . . . .	133
6.3	Equidistant Nodes Topology . . . . .	135
6.3.1	PER/BER with 1412 Byte Payloads . . . . .	136
6.3.2	PER/BER with 692 Byte Payloads . . . . .	138
6.3.3	Observations . . . . .	141

6.4	Linear Topologies . . . . .	142
6.4.1	PER/BER with 10.4 m SD Separation . . . . .	142
6.4.2	PER/BER with 18 m SD Separation . . . . .	146
6.4.3	Observations . . . . .	149
6.5	Analysis of Performance Bottlenecks . . . . .	151
6.5.1	CFO Pre-Correction Errors . . . . .	151
6.5.2	Dynamic Range of Channel Frequency Response . . . . .	154
6.5.3	Bit Error Densities . . . . .	161
<b>7</b>	<b>Future Work</b>	<b>166</b>
7.1	Physical Layer Cooperation in a Network . . . . .	166
7.1.1	Early MAC Results . . . . .	166
7.1.2	Rate Adaptation . . . . .	168
7.1.3	Open Questions . . . . .	170
7.2	Transceiver Extensions . . . . .	171
7.2.1	Temporal Combining . . . . .	171
7.2.2	Error Correcting Codes . . . . .	172
7.2.3	Full Duplex . . . . .	173
<b>A</b>	<b>Additional Plots for Full Transceiver Characterization</b>	<b>176</b>
A.1	Co-located Source/Relay Topology . . . . .	177
A.1.1	1412 Byte Payloads . . . . .	177
A.1.2	692 Byte Payloads . . . . .	181
A.2	Equidistant Nodes Topology . . . . .	185
A.2.1	1412 Byte Payloads . . . . .	185
A.2.2	692 Byte Payloads . . . . .	189
A.3	Linear Topologies . . . . .	193
A.3.1	10.4 m SD Separation . . . . .	193
A.3.2	18 m SD Separation . . . . .	198

## Illustrations

3.1	Simplified models of a direct conversion RF transmitter and receiver.	13
3.2	Simulation of SISO OFDM performance loss due to CFO-induced inter-carrier interference (ICI).	16
3.3	Hardware connections for measuring CFO between two WARP nodes.	18
3.4	Typical distributions of carrier frequency offsets between two pairs of WARP nodes.	19
3.5	Block diagram of time domain CFO estimator implementation.	22
3.6	Phase differences whose mean is used as the CFO estimate, showing the distortion which skews the CFO estimate.	25
3.7	Time domain CFO estimate as a function of offset from the beginning of a received packet.	26
3.8	Transients in instantaneous frequency offset immediately after enabling the radio transmit path, as calculated by the VSA.	29
3.9	Same data as Figure 3.8, zoomed in on first 50 $\mu$ sec after the transmitter is enabled.	30
3.10	Measuring the instantaneous voltage on the WARP Radio Board.	31
3.11	Oscilloscope measurement of the instantaneous transmitter state (top trace) and supply voltage (bottom trace) on the WARP Radio Board during the start of a transmission.	32
3.12	Hardware configuration for characterizing performance of the time domain CFO estimation implementation.	34



3.13	Distribution of time domain CFO estimates for fixed CFO at high SNR, as measured in hardware. . . . .	36
3.14	Estimation error ( $2\sigma$ ) for the time domain CFO estimator for multiple CFOs and SNRs, as measured in hardware. . . . .	37
3.15	BER vs. CFO, for small CFOs and multiple attenuations. . . . .	38
3.16	Subcarrier mapping of pilot tones, interleaved in time and across antennas. . . . .	41
3.17	Simple two slot cooperative exchange, indicating each inter-node CFO from the perspective of the receiving nodes. . . . .	42
3.18	Measurements of actual CFO between two WARP nodes, captured at 2.1 ms intervals for 40 minutes. Plots (b)-(d) each show a subset of data from plot (a) over smaller time intervals. . . . .	44
3.19	Distribution of CFO drift, from the same experiment as Figure 3.4(b). . . . .	45
3.20	Signals for the path through the relay in an amplify and forward link. . . . .	48
3.21	Hardware configuration for characterizing the destination's tolerance for CFO between source and relay transmissions. . . . .	51
3.22	Experimental PER/BER performance vs. CFO between transmitting nodes. . . . .	52
3.23	Experimental PER/BER performance vs. CFO between transmitting nodes, where one transmitting node applies CFO pre-correction based on its time domain CFO estimate. . . . .	53
3.24	Inferring CFO from phase error estimates calculated per OFDM symbol. . . . .	55
3.25	Performance of various frequency domain CFO estimation schemes vs. SNR for various packet durations (durations are measured in OFDM symbols). . . . .	57
3.26	Experimental distribution of frequency domain residual CFO estimates. . . . .	58
3.27	Experimental performance of the frequency domain residual CFO estimator vs. SNR for multiple packet durations. . . . .	59

3.28	Experimental performance of the frequency domain residual CFO estimator vs. packet duration for multiple SNRs. . . . .	60
4.1	Example System Generator block diagram, showing the CRC-32 calculation subsystem from our OFDM transmitter. . . . .	63
4.2	Block diagram of the OFDM transmit and receive signal processing pipelines. . . . .	64
4.3	Addressing of OFDM transmitter time domain sample buffers implementing frequency domain conjugation for Alamouti encoding. . . . .	67
4.4	Schematic of the OFDM transceiver packet buffer subsystem. . . . .	69
4.5	Field descriptions and durations for OFDM frames. . . . .	70
4.6	Mapping of data symbols and pilot tones across subcarriers. . . . .	72
4.7	OFDM receiver flow chart . . . . .	74
4.8	Auto-correlation of the preamble's long training symbol (LTS). . . . .	76
4.9	Packet timing correlator output for full preamble, comparing full precision calculation (a) to 1-bit $\times$ 1-bit quantized version (b). . . . .	78
4.10	Frame format for cooperative transmissions, where two nodes each transmit one of the streams simultaneously. . . . .	79
4.11	Output of full precision (a) and 1-bit $\times$ 1-bit (b) correlators when processing overlapping preambles for 2 $\times$ 1 or cooperative transmission. . . . .	80
4.12	Output of 3-bit $\times$ 1-bit correlator for a single transmission (a) and two transmissions (b). . . . .	81
4.13	Experimental results for probability of error during packet timing estimation for non-cooperative (NC) and amplify-and-forward (AF) links, using the original 1-bit $\times$ 1-bit (a) and new 1-bit $\times$ 3-bit (b) correlators. . . . .	81
4.14	Block diagram of OFDM receiver's auto-responder subsystem. . . . .	85
4.15	Block diagram of OFDM transmitter's header translation subsystem. . . . .	88

4.16	Relative start times of two automatic transmissions from cooperating nodes, triggered by independent receptions of the same packet. . . . .	90
4.17	Example observation of MAC and PHY behaviors, captured by observing state signals from multiple nodes in real-time on an oscilloscope. . . . .	94
5.1	Block diagram of the overall FPGA design for our implementation on WARP, consisting of C code for the FPGA's PowerPC (1), custom logic designs in the FPGA fabric (2) and key peripherals on the WARP hardware itself (3). . . . .	97
5.2	Connections between WARP nodes and the Azimuth channel emulator for cooperative experiments. . . . .	101
5.3	Hardware setup for our experiments, with three WARP nodes and the Azimuth channel emulator. . . . .	101
5.4	Power delay profiles for TGn channel models B, C and D . . . . .	102
5.5	Co-located source/relay topology, modeling a source and relay at a small, fixed distance cooperating to communicate with a distant destination. . . . .	105
5.6	Equidistant nodes topology, with source, relay and destination each separated by a common distance . . . . .	105
5.7	Linear topology for experiments modeling fixed source and destination, with the relay at various points along the line connecting them. . . . .	106
5.8	Errors between desired and actual distances in the linear topology using only attenuators (1.0 dB step) and attenuators plus model gains (0.1 dB step). . . . .	108
5.9	Timing of Tx/Rx modes for cooperative experiments. . . . .	116

5.10	State transitions in the OFDM receiver for a given transmission, indicating which end states are counted as packet errors. . . . .	122
6.1	Structure of each experiment, with four nested loops sweeping over multiple iterations of every combination of experimental parameters. .	125
6.2	Co-located source/relay topology. . . . .	127
6.3	Packet error rates for co-located source/relay topology with 1416 byte, QPSK modulated payloads. . . . .	128
6.4	Packet error rates for co-located source/relay topology with 1416 byte, 16-QAM modulated payloads. . . . .	129
6.5	Bit error rates for co-located source/relay topology with 1416 byte, QPSK modulated payloads. . . . .	129
6.6	Bit error rates for co-located source/relay topology with 1416 byte, 16-QAM modulated payloads. . . . .	130
6.7	Packet error rates for co-located source/relay topology with 692 byte, QPSK modulated payloads. . . . .	131
6.8	Packet error rates for co-located source/relay topology with 692 byte, 16-QAM modulated payloads. . . . .	132
6.9	Bit error rates for co-located source/relay topology with 692 byte, QPSK modulated payloads. . . . .	132
6.10	Bit error rates for co-located source/relay topology with 692 byte, 16-QAM modulated payloads. . . . .	133
6.11	Equidistant nodes topology. . . . .	135
6.12	Packet error rates for equidistant nodes topology with 1412 byte, QPSK modulated payloads. . . . .	136
6.13	Packet error rates for equidistant nodes topology with 1412 byte, 16-QAM modulated payloads. . . . .	137

6.14	Bit error rates for equidistant nodes topology with 1412 byte, QPSK modulated payloads. . . . .	137
6.15	Bit error rates for equidistant nodes topology with 1412 byte, 16-QAM modulated payloads. . . . .	138
6.16	Packet error rates for equidistant nodes topology with 692 byte, QPSK modulated payloads. . . . .	139
6.17	Packet error rates for equidistant topology nodes with 692 byte, 16-QAM modulated payloads. . . . .	139
6.18	Bit error rates for equidistant topology nodes with 692 byte, QPSK modulated payloads. . . . .	140
6.19	Bit error rates for equidistant topology nodes with 692 byte, 16-QAM modulated payloads. . . . .	140
6.20	Linear topologies with 10.4 m source/destination separation. . . . .	143
6.21	Packet error rates for linear topology QPSK modulated payloads and 10.4 m SD separation. . . . .	144
6.22	Packet error rates for linear topology 16-QAM modulated payloads and 10.4 m SD separation. . . . .	144
6.23	Bit error rates for linear topology QPSK modulated payloads and 10.4 m SD separation. . . . .	145
6.24	Bit error rates for linear topology 16-QAM modulated payloads and 10.4 m SD separation. . . . .	145
6.25	Linear topologies with 18 m source/destination separation. . . . .	146
6.26	Packet error rates for linear topology QPSK modulated payloads and 18 m SD separation. . . . .	147
6.27	Packet error rates for linear topology 16-QAM modulated payloads and 18 m SD separation. . . . .	148
6.28	Bit error rates for linear topology QPSK modulated payloads and 18 m SD separation. . . . .	148

6.29	Bit error rates for linear topology 16-QAM modulated payloads and 18 m SD separation. . . . .	149
6.30	Probability distribution of frequency domain CFO estimation error vs. Rx power, for full-length packets in a flat fading channel. . . . .	153
6.31	Packet success and error rates for NC/DF/AF as a function of channel frequency response dynamic range, for full length 16-QAM payloads in the co-located source/relay topology . . . . .	157
6.32	Block diagram of the signal chain from the antenna to OFDM receiver. The RF receiver in the MAX2829 applies gain in two stages, with gains selected by the OFDM receiver's AGC block. . . . .	158
6.33	Channel estimates for high-dynamic range channel response. . . . .	160
6.34	Packet error rate (a) broken down into contributing errors (b)-(d) for full length 16-QAM payloads in the co-located source/relay topology. . . . .	162
6.35	Distribution (a) and cumulative distributions of number of bit errors per packet for co-located source/destination/relay topology and full length payloads modulated at 16-QAM. . . . .	164
6.36	Pseudo packet error rates for co-located source/relay topology with 1416 byte, QPSK modulated payloads, for four different bit error thresholds. . . . .	165
7.1	Throughput improvement using DOC in a three node network, tested with a fixed source/destination and a relay at various points along the line connecting them. . . . .	167
7.2	Packet error rates for NC and DF schemes in co-located source/relay topology with 692 byte payloads modulated with QPSK and 16-QAM. . . . .	169
A.1	Probability of destination receiving packets with good headers but bad payloads for QPSK modulation. . . . .	177

A.2	Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation. . . . .	177
A.3	Probability of destination receiving packets with bad headers for QPSK modulation. . . . .	178
A.4	Probability of destination receiving packets with bad headers for 16-QAM modulation. . . . .	178
A.5	Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads. . . . .	179
A.6	Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads. . . . .	179
A.7	Probability of relay transmitting for QPSK payloads. . . . .	180
A.8	Probability of relay transmitting for 16-QAM payloads. . . . .	180
A.9	Probability of destination receiving packets with good headers but bad payloads for QPSK modulation. . . . .	181
A.10	Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation. . . . .	181
A.11	Probability of destination receiving packets with bad headers for QPSK modulation. . . . .	182
A.12	Probability of destination receiving packets with bad headers for 16-QAM modulation. . . . .	182
A.13	Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads. . . . .	183
A.14	Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads. . . . .	183
A.15	Probability of relay transmitting for QPSK payloads. . . . .	184
A.16	Probability of relay transmitting for 16-QAM payloads. . . . .	184
A.17	Probability of destination receiving packets with good headers but bad payloads for QPSK modulation. . . . .	185

A.18 Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation. . . . .	185
A.19 Probability of destination receiving packets with bad headers for QPSK modulation. . . . .	186
A.20 Probability of destination receiving packets with bad headers for 16-QAM modulation. . . . .	186
A.21 Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads. . . . .	187
A.22 Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads. . . . .	187
A.23 Probability of relay transmitting for QPSK payloads. . . . .	188
A.24 Probability of relay transmitting for 16-QAM payloads. . . . .	188
A.25 Probability of destination receiving packets with good headers but bad payloads for QPSK modulation. . . . .	189
A.26 Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation. . . . .	189
A.27 Probability of destination receiving packets with bad headers for QPSK modulation. . . . .	190
A.28 Probability of destination receiving packets with bad headers for 16-QAM modulation. . . . .	190
A.29 Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads. . . . .	191
A.30 Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads. . . . .	191
A.31 Probability of relay transmitting for QPSK payloads. . . . .	192
A.32 Probability of relay transmitting for 16-QAM payloads. . . . .	192
A.33 Probability of destination receiving packets with good headers but bad payloads for QPSK modulation. . . . .	193



A.34 Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation. . . . .	194
A.35 Probability of destination receiving packets with bad headers for QPSK modulation. . . . .	194
A.36 Probability of destination receiving packets with bad headers for 16-QAM modulation. . . . .	195
A.37 Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads. . . . .	195
A.38 Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads. . . . .	196
A.39 Probability of relay transmitting for QPSK payloads. . . . .	196
A.40 Probability of relay transmitting for 16-QAM payloads. . . . .	197
A.41 Probability of destination receiving packets with good headers but bad payloads for QPSK modulation. . . . .	198
A.42 Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation. . . . .	198
A.43 Probability of destination receiving packets with bad headers for QPSK modulation. . . . .	199
A.44 Probability of destination receiving packets with bad headers for 16-QAM modulation. . . . .	199
A.45 Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads. . . . .	200
A.46 Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads. . . . .	200
A.47 Probability of relay transmitting for QPSK payloads. . . . .	201
A.48 Probability of relay transmitting for 16-QAM payloads. . . . .	201

## Tables

4.1	Alamouti STBC encoding . . . . .	65
5.1	FPGA resource usage . . . . .	98
5.2	TGn channel model parameters . . . . .	102
5.3	Payload and timing parameters for cooperative experiments. . . . .	117
5.4	Relay auto-responder action configurations for cooperative tests . . .	118
6.1	Attenuations configured in the channel emulator for each relay position in the 10.4 m SD separation linear topology. . . . .	143
6.2	Attenuations configured in the channel emulator for each relay position in the 18 m SD separation linear topology. . . . .	147

# Chapter 1

## Introduction

Cooperative communications is the general idea of pooling the resources of distributed nodes to improve the overall performance of a wireless network. Applications of this general idea have been widely studied in the literature, with some of the most prominent [1–4] having already garnered many thousands of citations. The survey in [5] provides an excellent overview of the field from a theory-centric perspective.

While cooperative communications has a rich theoretical history in the literature, efforts to actually implement cooperative systems have been much more limited. A number of papers have been published in recent years describing cooperative implementations [6–9]. Each one, however, falls short of realizing the complete, real-time transceiver we seek to build.

For example, in our own previous work [9], we built a real-time cooperative system utilizing a dedicated relay. While this system achieved significant performance gains, it required a wired connection between the source and relay to synchronize their simultaneous transmissions. It also supported only amplify and forward relaying, as it lacked the receiver features necessary to address the carrier frequency offset challenges imposed by decode and forward relaying.

Two implementations are presented in [10]. In the first, the authors focus on cooperation at the MAC layer. This approach is constrained by using standards compliant wireless interfaces whose physical and link layer behaviors cannot be modified in any substantial way. The second implementation uses a software defined radio platform

which allows custom physical layer designs. Unfortunately the software implementation of the physical layer does not operate in real-time, significantly constraining the achievable data rates and channel conditions which can be evaluated.

In [7] the authors present the performance of a decode and forward system built using GNU Radio. They demonstrate clear a BER improvement using DF, but their transceiver design allows only a single transmission per time slot. The authors state that while simultaneous source and relay transmissions would improve performance, “...distributed synchronization is a large implementation hurdle to overcome.” They continue, saying “this is a particular area of cooperative diversity research in which implementation work can prove particularly fruitful.” We agree, and seek to address precisely these challenges in our design.

## 1.1 Objectives and Challenges

Our work seeks to achieve three primary objectives:

1. Design and implementation of a real-time cooperative physical layer transceiver;
2. Development of experimental methodologies to accurately test the transceiver;
3. Rigorous characterization and analysis of the transceiver’s performance.

Each of these objectives presents significant challenges which must be overcome in our work. A few challenges, discussed below, are apparent even before starting the transceiver design. Others are more subtle and are discussed in context throughout the chapters that follow.

- **Carrier Frequency Offset:** Correcting for differences in carrier frequencies between a transmitter and receiver is a challenge in most wireless systems; it is

especially challenging when two independent nodes (source and relay) transmit simultaneously to a common destination.

- **Synchronization:** The timing of transmissions from cooperating nodes must be aligned; any offset can degrade performance, and large offsets can cripple communication altogether.
- **Full Node Integration:** Every physical layer subsystem, from core signal processing blocks to front-end systems for energy detection and synchronization, must interoperate and be integrated into a single, real-time design.
- **Experiment Design:** The hardware configuration, independent variables, metrics and methodologies for our experiments must be designed to gather results we can be certain reflect the true performance of our implementation.

We have addressed all of these challenges in our work, as detailed in the following chapters.

## 1.2 Summary of Contributions

Our contributions can be broadly divided into three groups.

First is the design and FPGA implementation of a full, real-time cooperative OFDM transceiver. To the best of our knowledge, this transceiver is the only open-source implementation of a complete MIMO OFDM physical layer, and certainly the only one to support physical layer cooperation. Our OFDM design is one of the key deliverables of the overall Rice Wireless Open-Access Research Platform (WARP) effort and has already had significant impact among the hundreds of researchers using WARP worldwide [11].

Our transceiver design addresses all of the challenges associated with building a real-time OFDM system in an FPGA. It also addresses additional challenges unique to physical layer cooperation. These challenges and our solutions to each are detailed in Chapters 3 and 4.

Our second key contribution is the development of methodologies for evaluating the performance of our cooperative system. Our methods are designed to accurately measure performance over the course of long experiments which sweep topological and propagation conditions. We also develop techniques to identify performance limitations and to isolate and understand their underlying causes. While our experimental framework was designed for our own research, the tools are useful for many other types of experiments. In fact, a number of other WARP users have already adopted our methodologies for use in their own experiments. Our experimental methodologies are discussed in Chapter 5.

Our final primary contribution is an extensive set of experimental results detailing the performance of our cooperative design. Our experiments test a variety of node topologies and propagation conditions, each designed to model realistic scenarios for modern wireless networking devices. Our results clearly demonstrate substantial performance gains when using cooperation. These results and related discussions are presented in Chapter 6.

## Chapter 2

### Background

We can state our primary goal quite simply: we aim to build and characterize a complete cooperative communications transceiver. However, the distillation of this general goal into an actual implementation requires we make some early, fundamental design decisions.

#### 2.1 Platform Selection

We use the Rice Wireless Open-Access Research Platform (WARP) [12] for our cooperative transceiver implementation and characterization.

The WARP hardware combines a large FPGA, flexible RF interfaces and various support peripherals. The FPGA is a “blank-slate,” providing substantial computational resources for the implementation of fully custom transceivers. The RF interfaces provide waveform-level access to a radio transceiver; these interfaces impose no wireless standard and require all waveform generation and processing be handled by custom designs in the FPGA. We implement the full physical layer transceiver entirely in the FPGA. The design operates in real-time, transmitting and receiving wideband waveforms without any off-chip processing.

Other programmable wireless platforms have been developed in recent years, but none meets the requirements of our work. For example, the combination of GNU Radio [13] and the Ettus USRP hardware [14] enables flexible wireless designs. Their

approach focuses on using PC-based software for the majority of signal processing, with the hardware performing RF up/downconversion, rate change filtering and exchange of samples with a PC. This approach enables rapid iteration on transceiver algorithms, as they are implemented entirely in software. However the latency in moving samples between the RF front-end and PC is long, on the order of many milliseconds [15].

As discussed in Chapter 1 we seek to build a cooperative system which operates at data rates and time scales comparable to modern wireless networking standards. Operation at these time scales requires nodes switch between receiving and transmitting with turnaround times of tens of microseconds (the SIFS interval between DATA and ACK packets in IEEE 802.11a is  $19 \mu\text{s}$ , for example). Meeting these timing constraints requires moving the physical layer processing closer to the antenna. On WARP, this means implementing the full transceiver in the FPGA with direct connections to the radio's ADCs and DACs.

In short, WARP uniquely enables our implementation of a complete, real-time, cooperative transceiver.

## 2.2 Cooperative Schemes

A huge variety of cooperative schemes have been proposed; [5] provides a good overview. Most schemes can be characterized along two dimensions: whether cooperating nodes transmit simultaneously and by the amount of physical layer processing performed at relay nodes.

We focus on two schemes: **amplify and forward (AF)** and **decode and forward (DF)**. The key difference between AF and DF is the amount of processing at the relay. In AF the relay simply captures the waveform received from the source,



amplifies it, then re-transmits it coincident with a second transmission by the source. In DF the relay implements a full physical layer transceiver. It decodes a transmission by the source, buffers the actual payload bytes, then re-transmits the payload using its own transmitter. Each scheme presents unique implementation challenges, as discussed in Chapter 4.

We use versions of both schemes where cooperating nodes transmit simultaneously. Our approach is analogous to “virtual MIMO,” where distributed single-antenna nodes transmit together, seeking to create a waveform indistinguishable from that of a single multi-antenna transmitter. Our motivation for this approach is to enable a receiver which can seamlessly process SISO, cooperative and multi-antenna transmissions without a-priori knowledge of the transmitter configuration.

Finally, we focus on schemes which employ a single relay (to facilitate use of the Alamouti space-time block code, discussed below).

## 2.3 OFDM

We use orthogonal frequency division multiplexing (OFDM) as the underlying modulation technique for our physical layer design. This choice is motivated by the same reasons that led many of the latest wireless networking and cellular standards to adopt OFDM for their physical layer specifications. Our primary motivations for using OFDM are discussed below.

**Moderate Complexity:** With OFDM, a wideband spectrum is divided into many narrow subcarriers, each of which is processed independently. The frequency division is performed with fast Fourier transforms (IFFT for Tx, FFT for Rx), which are well suited for FPGA implementation. With careful selection of subcarrier spacing and overall bandwidth, the channel frequency response for each subcarrier can be

modeled with a single complex value. This allows a much easier implementation than that required for processing multi-tap time domain channel responses. Our design uses **10 MHz bandwidth** divided into **64 subcarriers**.

**Delay Spread Tolerance:** A transmitted OFDM waveform is of a sequence of “OFDM symbols.” Each symbol is the sequence of samples output from the transmitter’s inverse FFT, consisting of one sample per subcarrier. Guard intervals are inserted between each transmitted OFDM symbol to provide tolerance for multipath fading. Longer guard intervals provide more multipath tolerance, but incur higher overhead. OFDM systems typically fill the guard intervals with a cyclic prefix, a repetition of the final samples of the following OFDM symbol. This technique provides tolerance for synchronization errors at the receiver. Our design uses a **16 sample (1.6  $\mu\text{sec}$ )** cyclic prefix per OFDM symbol.

**Cooperative Timing Tolerance:** The delay spread and synchronization tolerance provided by the cyclic prefix also serves to ease reception of cooperative transmissions [16]. Consider a cooperative transmission by two nodes (source and relay). The arrival of their simultaneous transmissions at a common destination may not occur at the same instant. Fortunately, any offset in the arrival times mimics the effects of multipath. Thus, if the combination of difference in arrival times and channel delay spread is shorter than the cyclic prefix, an OFDM receiver will suffer no extra degradation. It is still critical the two transmissions be as well synchronized as possible to preserve the maximum tolerance for actual multipath fading. Our solution to this synchronization challenge is discussed in Chapter 4

## 2.4 Distributed Space-Time Block Code

Our focus on cooperative schemes which employ simultaneous transmissions by cooperating nodes poses a challenge. By definition, our cooperating nodes will transmit the same payload to a common destination. However if these simultaneous transmissions use identical waveforms they could combine destructively at the destination. This issue is analogous to unintentional beamforming. Two transmissions of the same waveform would form spatial beam patterns. If the destination node were located in a null of these patterns it could receive a weaker signal than if only one transmitter were active. In other words, simply transmitting identical waveforms from cooperating nodes could actually hurt performance.

A frequently proposed solution to this issue is an adaptation of space-time coding [4, 16, 17]. Space-time codes provide techniques for designing waveforms for transmission from multiple antennas with the goal of improving reliability relative to that of a single-antenna transmission. Some space-time codes can be readily adapted for use by cooperating nodes, which each node acting as if it were one antenna in a multi-antenna transmitter.

We selected the Alamouti space-time block code [18] for our design. The Alamouti code specifies an encoding process which translates a sequence of modulated data symbols into two waveforms suitable for simultaneous transmission. The code also specifies a simple decoding process which translates the single received waveform (composed of the weighted sum of the transmitted waveforms, with weights corresponding to channel coefficients) back into the original sequence of data symbols.

The Alamouti encoding and decoding processes are relatively simple and lend themselves to straightforward implementation in an FPGA. Applying these processes in an OFDM transceiver requires some extra care, but it remains feasible to implement

a system which exploits all the benefits of both OFDM and the Alamouti STBC.

One of the key benefits of the Alamouti code is that it provides full diversity. In a two-antenna transmitter this means every payload symbol is transmitted from both antennas. In the context of a cooperative link, this feature implies that either the source or relay's transmission is sufficient for the destination to decode a complete payload.

The Alamouti code was conceived for use with a two-antenna transmitter. However, it is straightforward to use the code with a pair of single-antenna cooperative nodes. The key requirement is that the nodes assume the roles of opposite antennas, so that each node transmits a different (but complementary) waveform. Our implementation uses a single relay per cooperative transmission, so it is straightforward to assign one stream to the source and the other to the relay.

## 2.5 Networking Support

One of our research group's overall projects is investigating the benefits and costs of physical layer cooperation in real wireless networks. Our contribution to this effort is the design and characterization of the cooperative transceiver described in this thesis. While our work here does not include development at higher layers or network-scale experiments, it is critical our transceiver design support these efforts in the future. Thus, a major requirement for our physical layer design is that it enable development and evaluation of novel protocols at the MAC layer.

Medium access protocols can be broadly classified as either scheduled or random access. From a PHY perspective random access is the more demanding, requiring the transceiver be ready to receive a packet from any node at any time. Further, it requires that when a packet is received the PHY must establish all necessary syn-

chronization using just the incoming waveform. Our transceiver is designed to meet both requirements, with key front-end subsystems (energy detection, AGC, symbol synchronization, CFO estimation, etc.) which require no external synchronization and enable reception of packets over a wide range of received powers.

A second requirement for supporting MAC protocol development is that, in addition to the cooperative schemes discussed above, the PHY must also support non-cooperative communication. Further, the PHY should be able to assume the role of source, relay or destination per-packet. Together these features enable exploration of a wide variety of MAC protocols and will facilitate real-time experiments at the data rates and time scales of modern wireless networking devices.

While the primary focus of this thesis is the physical layer, our transceiver has already been used in early investigations of a cooperation-aware MAC protocol. This work is briefly discussed in Chapter 7.

## Chapter 3

### Carrier Frequency Offsets

Carrier frequency offset is a common impairment in real wireless systems. It results from variations in frequency across the local oscillators employed by communicating nodes to generate the carrier signals they use for translating signals between baseband and RF. The issues of CFO are well understood problems in wireless systems. However, the impact of CFO and techniques to mitigate it depend heavily on the specific parameters of a given transceiver and the properties of the hardware on which it is realized.

In the sections below we discuss the origins of CFO and expectations for offsets when using the WARP hardware. We also explore CFO issues specific to OFDM and how these impact our transceiver design. Through both simulations and experiments, we demonstrate the performance of the CFO estimation and correction scheme we designed for our transceiver.

Additional issues related to CFO arise in the context of a cooperative communications system. Specifically, when two nodes transmit simultaneously, the physical layer design must consider both the offset between the transmitters and each transmitter's offset relative to their common destination. In Section 3.7 below we discuss this multi-CFO issue in detail and present our design and experimental evaluation of a scheme to manage it.

### 3.1 Origin of CFO

The basis for carrier frequency offsets can be understood by examining a simple model for the inner workings of a radio transmitter and receiver. These are illustrated in Figure 3.1.

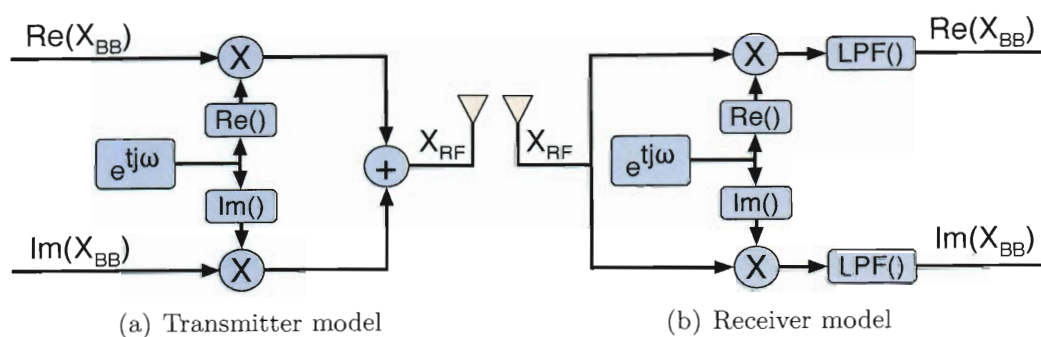


Figure 3.1 : Simplified models of a direct conversion RF transmitter and receiver.

The carrier frequency is denoted as  $\omega$ ,  $X_{BB}$  is a complex baseband signal,  $X_{RF}$  is a real-valued RF signal and  $\text{LPF}(x)$  a low-pass filter with gain 2 and a passband determined by the bandwidth of the baseband signal. These models omit a significant number of other operations performed by real RF transceivers (filters, DC cancellation, transmit and receive gain control, etc.). However, none of these affect the up/downconversion processes as they relate to CFO.

The transmit and receive processes can be written as follows:

$$\begin{aligned}
X_{RF} &= \text{Tx}(X_{BB}, \omega) \\
&= \text{Re}(X_{BB}) \cos(t\omega) - \text{Im}(X_{BB}) \sin(t\omega) \\
&= \frac{1}{2}(X_{BB}e^{jt\omega} + X_{BB}^*e^{-jt\omega}) \\
X_{BB} &= \text{Rx}(X_{RF}, \omega) \\
&= \text{LPF}(X_{RF}e^{jt\omega}).
\end{aligned} \tag{3.1}$$

In order to understand the basis of CFO, consider a simple transmission of a signal  $S_{BB}$  from a source node with carrier frequency  $\omega_S$  which is received by a destination with carrier frequency  $\omega_D$ :

$$S_{BB} \rightarrow \text{Tx}(S_{BB}, \omega_S) \rightarrow S_{RF} \rightarrow \text{Rx}(S_{RF}, \omega_D) \rightarrow D_{BB}.$$

Ideally, the destination would recover the transmitted baseband signal exactly. Using the expressions for  $\text{Tx}()$  and  $\text{Rx}()$  in Equation 3.1, we can express the received baseband signal  $D_{BB}$  in terms of the transmitted baseband signal  $S_{BB}$  and the carrier frequencies:

$$\begin{aligned}
D_{BB} &= \text{LPF} \left( \frac{(S_{BB}e^{jt\omega_S} + S_{BB}^*e^{-jt\omega_S})(e^{jt\omega_D})}{2} \right) \\
&= S_{BB}(e^{jt(\omega_S - \omega_D)}).
\end{aligned} \tag{3.2}$$

The received baseband signal is equal to the original baseband signal modulated by a complex sinusoid. In the frequency domain, this gives a received spectrum equal



to the transmitted one, only shifted away from DC by the difference in the carrier frequencies of the transmitter and receiver (i.e.  $\omega_S - \omega_D$ ). This shift of the received signal is the baseband manifestation of carrier frequency offset.

### 3.2 Impact of CFO in OFDM

In an OFDM system, the performance degradation due to uncorrected carrier frequency offsets can be substantial [19, 20]. The performance of an OFDM system is impacted by CFO in two primary ways. First, the frequency offset manifests as a phase offset which is constant across subcarriers in an OFDM symbol and increases with each OFDM symbol. These phase offsets must be estimated and corrected in the frequency domain to avoid symbol errors due to rotated constellation symbols. Most OFDM systems allocate a small number of subcarriers for use as pilot tones from which the receiver can extract phase error estimates for each OFDM symbol. Our use of pilot tones is discussed in Section 3.6 below.

The second impact of CFO is the loss of orthogonality between subcarriers in the OFDM receiver's FFT. The result is inter-carrier interference (ICI), which acts as an effective SNR reduction which worsens with increasing CFO [21]. Various schemes have been proposed to mitigate ICI [22]. However, these schemes require complex processing across subcarriers in the frequency domain, deviating from the model of processing subcarriers independently. The inherent simplicity of the independent subcarrier model is one of OFDM's key benefits and is a primary reason we selected it for our implementation.

Figure 3.2 illustrates the performance penalty of this CFO-included ICI. Each trace shows SNR versus BER for a different CFO. These are generated by a Monte Carlo MATLAB simulation of a SISO OFDM link. The simulated OFDM transceiver uses

10 MHz bandwidth and 64 subcarriers, 48 of which are loaded with random 16-QAM data symbols. The only degradations applied between the transmitter and receiver are CFO and AWGN. The receiver model uses perfect knowledge of the CFO to correct the phase offset in each OFDM symbol, but does not implement any correction for ICI. Thus, errors here are due only to noise and CFO-induced loss of orthogonality in the receiver's FFT.

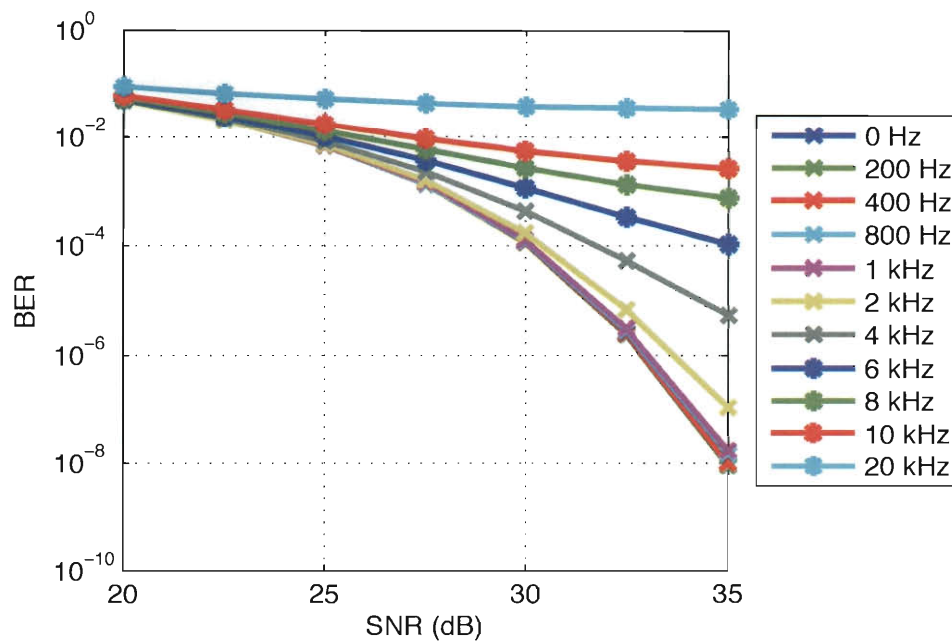


Figure 3.2 : Simulation of SISO OFDM performance loss due to CFO-induced inter-carrier interference (ICI).

The simulation parameters here are intentionally designed to model our actual OFDM implementation. Thus, Figure 3.2 provides useful intuition in understanding the expected impact of various levels of CFO. It is clear from the results that for large CFOs errors caused by ICI dominate performance, even at high SNR. It is also clear that for small CFOs performance is dominated by SNR (i.e. nearly all errors are caused by additive noise). Specifically, for frequency offsets smaller than  $\approx 1$  kHz, the

performance degradation due to ICI is negligible. This observation plays a key role in the discussions below, both in gauging the severity of CFOs observed in hardware (Section 3.3) and in evaluating the performance of the CFO estimation and correction system (Section 3.5).

### 3.3 Expectations for CFO on WARP

In order to implement an OFDM transceiver, we must first understand the range of carrier frequency offsets which can be expected on our hardware platform. As discussed in Chapter 2, we use standard WARP hardware for each of our nodes. Clocking in WARP is managed by a dedicated board (the WARP Clock Board [23]), which uses a Crystek CVT32 temperature-compensated crystal oscillator (TXCO) as the RF frequency reference. This TCXO provides a clock signal at a nominal frequency of 20 MHz. An oscillator's actual frequency, however, varies as a function of multiple factors, and is only specified by the manufacturer with some tolerance. The CVT32 is specified with a frequency tolerance of  $\pm 4$  ppm [24]. This range accounts for variations between devices, device age and operating temperature (specified for  $-20^{\circ}\text{C}$  to  $80^{\circ}\text{C}$ , more than sufficient for our applications). Thus, we must design for a reference frequency of  $20 \pm 0.000080$  MHz.

The MAX2829 transceiver on the WARP Radio Board [25] uses a fractional-N synthesizer to multiply the frequency of the reference clock to generate the RF carrier signal used for up/downconversion between baseband and RF. The multiplication factor is programmable at runtime allowing the radio to tune to arbitrary frequencies in the 2.4 and 5 GHz bands. For a nominal reference frequency of 20 MHz and target carrier frequency of 2452 MHz, for example, the MAX2829 uses a multiplier of 122.6.

The MAX2829 applies the same multiplication factor even when there are small

variations in the frequency of the reference clock. As a result, for a given multiplier (i.e. a given target carrier frequency), the actual carrier frequency will vary across a range defined by the reference clock tolerance and frequency multiplier. For example, for a target carrier frequency of 2452 MHz (multiplier of 122.6), the range of actual carrier frequencies will be  $2452 \pm 0.009808$  MHz (equivalent to  $2452 \text{ MHz} \pm 4 \text{ ppm}$ ). The worst case CFO will occur when the transmit and receive nodes operate at opposite ends of this range. Thus, for operation in the 2.4 GHz band our OFDM transceiver design must be ready to handle any carrier frequency offset up to  $\approx 20$  kHz. Given this observation and the results from Section 3.2 above, we conclude that we will most likely operate in CFO regimes where errors due to ICI cannot be ignored.

### 3.4 Measured CFO in Hardware

The expectations described above are derived from specifications provided by the manufacturers of components in the WARP hardware. We can use the WARP hardware and our OFDM transceiver to directly measure the CFO between WARP nodes to verify these expectations.

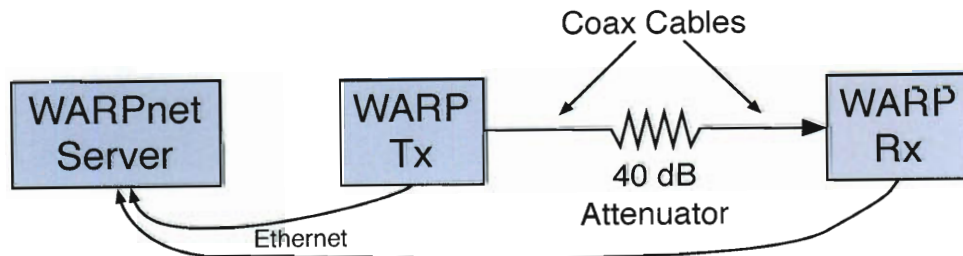


Figure 3.3 : Hardware connections for measuring CFO between two WARP nodes.

The configuration of this experiment, illustrated in Figure 3.3, uses radio boards on two WARP nodes connected via a RF cable and a 40 dB attenuator. This connection

emulates a static, frequency flat high-SNR propagation environment. Each node operates independently, using its local oscillator to generate an RF carrier. One node acts as a transmitter, sending a full-length packet at regular intervals. The other node receives each packet and extracts a single CFO estimate. This estimate is sent via Ethernet to a WARPnet server, where it is recorded to a file for offline processing (the WARPnet framework is discussed in detail in Section 5.5.1).

We ran this experiment using two pairings of kits built with four different WARP Clock Boards over long periods to capture frequency variations due to environmental changes and inherent offsets between different oscillators. The results from two pairs of nodes are shown in Figure 3.4. The plots show the probability distributions of observed carrier frequency estimates for two pairs of WARP nodes, calculated as normalized histograms with 2 Hz bins. Each figure represents  $\approx 1.1$  million CFO measurements gathered during a 40 minute experiment.

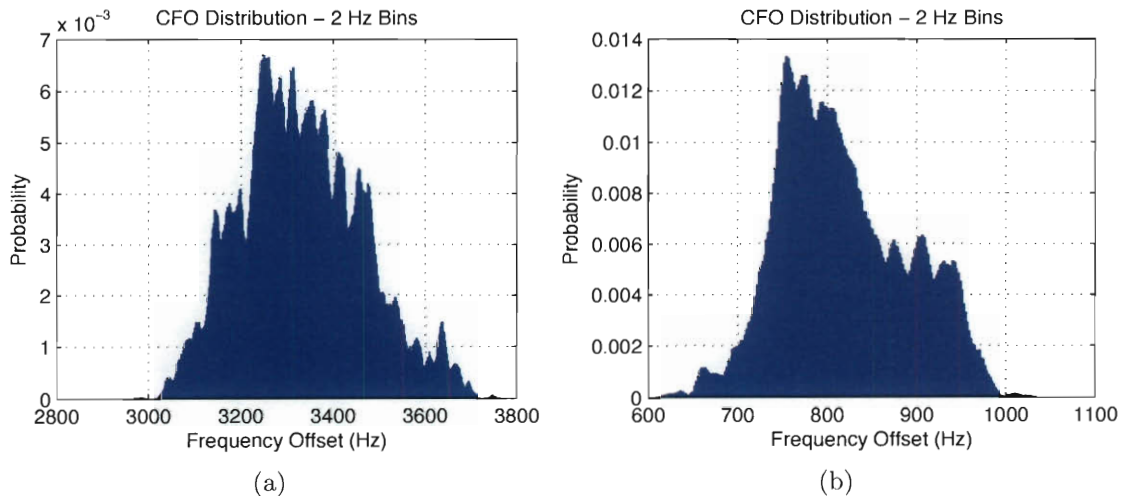


Figure 3.4 : Typical distributions of carrier frequency offsets between two pairs of WARP nodes.

We can make a few important observations from this data. First, every observed

offset falls well within the expected  $\approx 20$  kHz range calculated in Section 3.3 based on the tolerances specified in the reference oscillator datasheet. Second, the offset between nodes varies significantly with time, largely due to environmental changes (primarily temperature). These deviations are within specification for the oscillators. However, understanding the temporal properties of these offsets will be important and is explored in Section 3.7.1 below.

Finally, it is clear from these experiments that the CFOs cannot be characterized by a simple distribution. For example, there is a strong dependence on the individual oscillators, as evidenced by the large difference in mean offset in Figures 3.4(a) and 3.4(b) (note that each histogram represents a different pair of kits). Further, the shapes of the two examples here are significantly different. Given the large number of factors which affect each oscillator's average and instantaneous frequencies, it is not surprising the measured offsets cannot be characterized by a simple distribution.

### 3.5 Time Domain CFO Correction

As explained in Section 3.2, large CFOs significantly degrade performance of an OFDM system due to CFO-induced inter-carrier interference at the receiver. We expect frequency offsets between WARP kits to have offsets large enough for this effect to matter. This is supported by both the specifications for the oscillators used in the WARP hardware and by our own measurements of offsets between WARP kits.

In order to avoid performance degradation due to CFO, it is necessary for an OFDM receiver to estimate and correct CFO in the time domain, before the receiver translates received signals into the frequency domain via its FFT. Time domain CFO estimation is a well studied problem; many estimation algorithms have been proposed, covering a wide variety of OFDM systems [26, 27].

### 3.5.1 Time Domain CFO Estimation Algorithm

We use an adaptation of a standard and frequently cited technique [28] for our OFDM receiver's CFO estimation system. The technique in [28] exploits the periodic nature of the long training symbols included in each packet's preamble. The preamble includes two full copies of the 64-sample training symbol, plus a 32-sample cyclic prefix. The CFO estimate is calculated by comparing the phases of two samples separated by their period of repetition in the preamble. Carrier frequency offset manifests as a phase offset increasing with time. Thus, the phase difference between repeated samples is proportional to the CFO. Thanks to the cyclic prefix, this holds true even in the presence of a dispersive channel response. Our implementation calculates 64 phase differences (one per sample of the long training symbol), averages the differences, then normalizes the sum to calculate the final CFO estimate.

One challenge in implementing this scheme is synchronizing the CFO estimator with the incoming packet. The core problem is that the same long training symbols used for CFO estimation are also used to establish sample-level timing in the receiver. This timing estimation system is discussed in section 4.1.6. When synchronization is established, the last long training symbol will have already been received. At this point, the CFO estimation system needs to use the phases of the *previous* 128 samples to calculate its estimate. Further complicating the implementation is the requirement that the CFO estimate be ready quickly, in time to begin applying a frequency correction to the samples (composing the channel training symbols) which immediately follow the preamble. This correction must be applied before the training symbols are fed into the receiver's FFT. The latency between establishing synchronization and calculating a valid CFO estimate is the critical timing path in our OFDM receiver's time domain processing pipeline.

We address these challenges by realizing the CFO estimation algorithm with an architecture which operates continuously. Our design, illustrated in Figure 3.5, calculates a new CFO estimate with every received sample. It keeps a running sum of the phase differences between the 64 most recent samples and the 64 samples before those. This sum is updated with every received sample. This architecture essentially pretends every received sample is part of the preamble’s long training symbols and uses it to update the CFO estimate. The output of this calculation is actually useless at all times except in the narrow window following the reception of an actual preamble. Identifying this window is identical to identifying the boundary for the first sample of the channel training symbols input to the FFT. This latter task is accomplished by the correlator discussed in Section 4.1.6. We can re-use the correlator’s synchronization decision to capture the CFO estimate and begin applying the frequency correction immediately, before samples are fed into the FFT. Our architecture of continuously calculating a (frequently invalid) estimate, then capturing its output in the small window where its validity is known, minimizes the latency in this critical path. Our implementation produces a valid CFO estimate with a latency of just eight sample periods (seven for a CORDIC arctangent, one for the final multiplication).

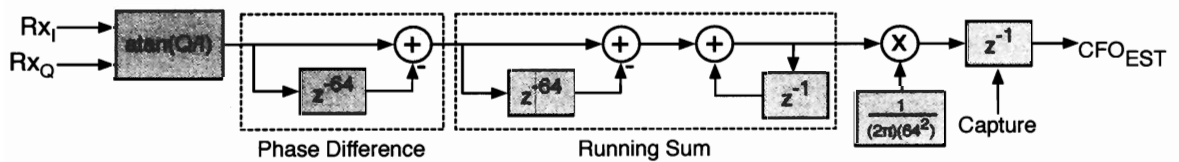


Figure 3.5 : Block diagram of time domain CFO estimator implementation.



### 3.5.2 Performance Expectations

The performance of the CFO estimation algorithm in [28] (and our implementation thereof) is characterized by two primary factors.

First, the range of frequency offsets which can be estimated is limited by the sampling rate and the period of training symbol repetitions in the preamble. Our implementation, with 10 MHz sampling and 64-sample training symbols, can estimate offsets in  $\pm 78.1$  kHz. This significantly exceeds the maximum possible offset ( $\approx 20$  kHz), as determined by the tolerance of the RF reference oscillators.

Second, the performance of the estimator will be subject to real-world degradations, like voltage fluctuations, phase noise and variable SNR. Using WARPnet and our OFDM implementation, we can directly measure the real-time performance of the CFO estimator in hardware.

One challenge in characterizing a CFO estimator is controlling the actual CFO as an experimental parameter. In practice, frequency offsets change constantly and unpredictably, varying with time and temperature on sub-second time scales. However, to measure the error in our CFO estimates, we must know the actual CFO at the instant each estimate is calculated.

We meet this requirement with a combination of hardware and transceiver modifications. On the WARP hardware, we use one Clock Board to drive the sampling and RF reference clocks for two nodes. By sharing these clock signals, we guarantee zero frequency offset between nodes, even as the actual frequencies of the oscillators vary with time. In the OFDM transceiver, we add a subsystem to multiply the transmitter's output waveform by a complex sinusoid with a programmable frequency. This multiplication does not degrade the waveform in any way; it simply shifts the center of the transmitted spectrum away from DC. We can emulate any CFO by changing

the frequency of this sinusoid, and conduct long experiments confident the CFO will remain constant.

### 3.5.3 Radio Transients

In the course of characterizing the performance of our CFO estimator we encountered an unexpected complication. In early experiments the CFO estimator consistently produced wildly incorrect results. Specifically, the mean of the CFO estimate differed significantly from actual CFO. In order to understand the underlying issues, we added logic to the design which records the 64 phase differences which constitute the average CFO estimate.

Figure 3.6 illustrates these phase differences. Each trace corresponds to an experiment using a different CFO, ranging from 0 to 4.8 kHz. The Y-axis is phase difference, expressed as a CFO estimate. The X-axis is the time offset from the beginning of the received packet. The first 160 samples (16  $\mu$ sec) are short training symbols, which are used for AGC convergence and DC offset correction (and not for CFO estimation). The next 32 samples are the cyclic prefix for the long training symbols. This plot shows the 64 phase differences taken during the second long training symbol (i.e. in the final 64 samples of the preamble), hence the range of time values on the X-axis offset from 0.

Ideally each trace would be constant, taking the value of the actual CFO. Instead, the traces have a clear trend which spans a larger range of CFO values than the actual offset. The shape is consistent across actual CFOs and across experimental trials, suggesting there is another source of degradation which was not accounted for.

Seeking to better understand the shape of the CFO estimate curves, we designed an experiment using WARPLab [29]. WARPLab is a design flow which uses WARP

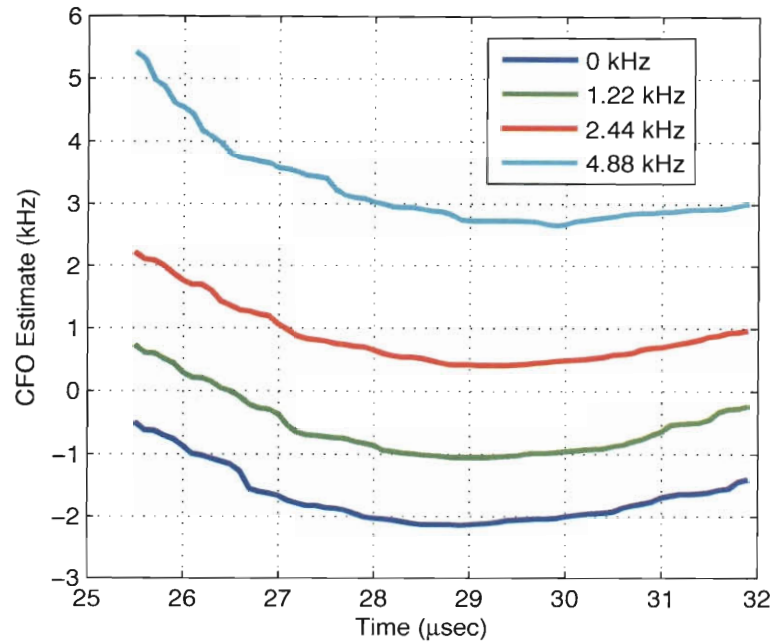


Figure 3.6 : Phase differences whose mean is used as the CFO estimate, showing the distortion which skews the CFO estimate.

nodes for RF transmission and reception, but implements all signal processing in MATLAB. In this test, a transmit node sends a long burst built from many repeated copies of the long training symbol. This transmission acts like a very long preamble from which many CFO estimates can be extracted. As in standard WARPLab, the receive node captures this transmission and offloads the samples to MATLAB for processing. In m-code we implement the same CFO estimation algorithm as in hardware. Because the transmission is composed entirely of periodic training symbols, the CFO estimation algorithm produces a valid CFO estimate for every received sample.

Figure 3.7 illustrates the results of this test. Each trace again corresponds to an experiment using a different CFO. But the time-axis here spans a full 100  $\mu\text{sec}$ , starting with the beginning of the transmitted burst. The large swings in the CFO estimates are apparent and are consistent with the shorter duration traces in Fig-

ure 3.6. Note the range of CFO values produced by the estimator relative to the maximum  $\approx 20$  kHz offset we expect using the WARP hardware. The large deviation in estimated CFO which decays rapidly with time strongly suggests some underlying transient in hardware which affects the frequency of either the transmitted or received waveform.

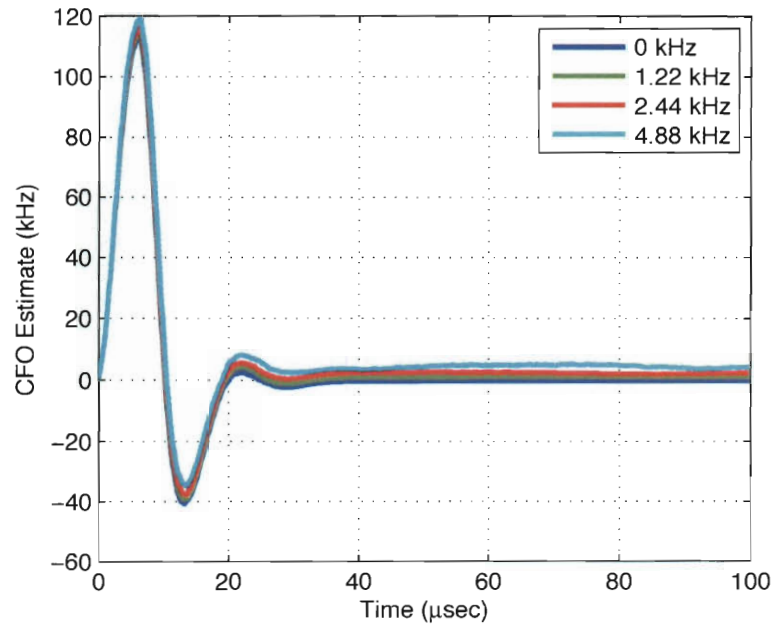


Figure 3.7 : Time domain CFO estimate as a function of offset from the beginning of a received packet.

In order to isolate this issue to either the transmitter or receiver, we conducted a similar WARPLab experiment using just one WARP node. In this experiment, the WARP node acts as a transmitter, and we use an Agilent 89600S vector signal analyzer (VSA) as the receiver. The WARP node is configured to repeat a simple cycle:

1. Begin feeding a constant value to the WARP Radio Board DACs;

2. Enable the WARP Radio Board transmit path (assert the MAX2829 Tx enable and power on the PA);
3. Wait  $\approx 300 \mu\text{sec}$ , leaving the transmit path enabled;
4. Disable the transmitter and PA;
5. Wait  $\approx 1 \text{ msec}$ , leaving the transmit path disabled.

By feeding a constant value into the baseband inputs of the MAX2829 transceiver (via the transmit DACs), the radio board will generate an RF sinusoid centered exactly at the radio's carrier frequency. In normal operation this is undesirable (often called "carrier leakage" or "local oscillator leakage") as it is usually necessary for transmitted waveforms to have zero DC component. This is true for our OFDM implementation, as well as in standards like 802.11a/g/n, where the DC subcarrier is always filled with zero. Further, the MAX2829 actually implements DC-blocking high-pass filters in its receive path to allow better settling of transients in its amplifiers. But for this experiment, transmitting only DC generates the ideal output- a simple, unmodulated sinusoid at exactly the carrier frequency. In order to capture transients at the initiation of a transmit cycle, it is also important the constant input to the DACs be initiated before the radio is enabled; this required a slight modification to the standard WARPLab design.

The VSA can be considered as the "perfect" receiver. The RF performance of the VSA hardware (additive noise, phase noise, quantization artifacts, I/Q gain/phase imbalance, spectral flatness, etc.) significantly exceeds the performance of a system built with WARP Radio Boards. This is no surprise, given their relative costs, sizes, capabilities and intended applications. The VSA software is comparably sophisticated, capable of processing a wide variety of waveform designs and modulation schemes.

The VSA operates by downconverting an RF waveform to baseband, digitizing and buffering it, then offloading it to a PC for offline processing. This is conceptually similar to WARPLab; the VSA hardware captures blocks of waveforms in real-time, but processes them elsewhere much slower than real-time.

For this experiment we configure the VSA to begin its capture upon the reception of an energy spike corresponding to the start of the WARP node's transmit cycle. We use the relatively simple VSA software mode designed for processing analog phase modulation (PM) signals. The VSA's PM demodulator implements a CFO compensation stage which estimates CFO by analyzing the frequency content of the full received waveform (access to the full waveform being a key benefit of offline processing). The VSA then constructs a new signal which is proportional to the instantaneous phase of the captured, downconverted, CFO-corrected waveform. Given that our transmitted baseband signal is a constant, this CFO-corrected, PM-demodulated output should be zero.

The results of this test are shown in Figure 3.8. The top trace shows the received baseband signal before the VSA software applies CFO correction. The low-frequency sinusoid is a direct manifestation of CFO, oscillating at a frequency equal to the difference in carrier frequencies of the transmitting WARP node and VSA downconverter. The second trace shows the received signal power on a log scale, wherein the beginning of the WARP node's transmission is clear. The third shows the output of the VSA's PM demodulator. In all three traces the signals look as expected for the majority of the test. However, there are some irregularities early in the transmission that merit a closer look.

Figure 3.9 presents the same traces, zoomed in to the first 50  $\mu\text{sec}$  of the received waveform. At this time scale the distortions in the received waveform are clear. The

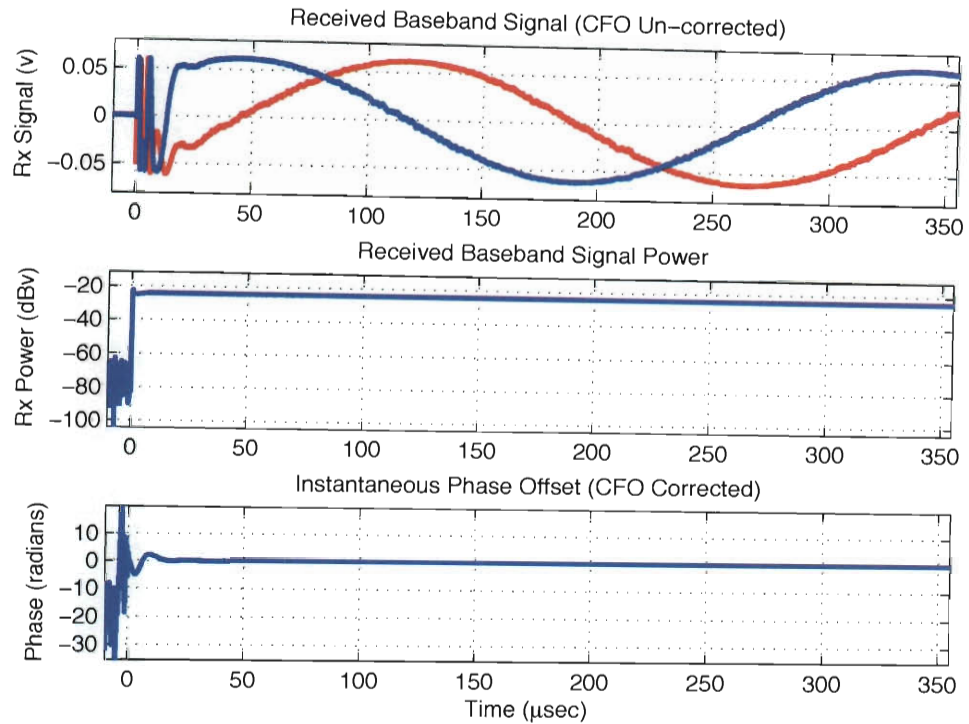


Figure 3.8 : Transients in instantaneous frequency offset immediately after enabling the radio transmit path, as calculated by the VSA.

raw baseband signal (top trace) only settles to the expected low-frequency sinusoid in the latter half of the plot. The amplitude and phase distortions (middle and bottom traces) are likewise apparent. Take note of the timing of the peaks in the phase distortion as measured by the VSA, and compare them to the peaks in the CFO estimate in Figure 3.7. These curves are closely related- rapid phase changes (observed by the VSA) correspond to large deviations in CFO estimates (calculated on WARP), and the timing of the curves align perfectly. These traces conclusively demonstrate the underlying cause of the erroneous CFO estimations are distortions incurred at the transmitter in the first  $\approx 35 \mu\text{sec}$  after the radio transmitter and power amplifier are enabled.

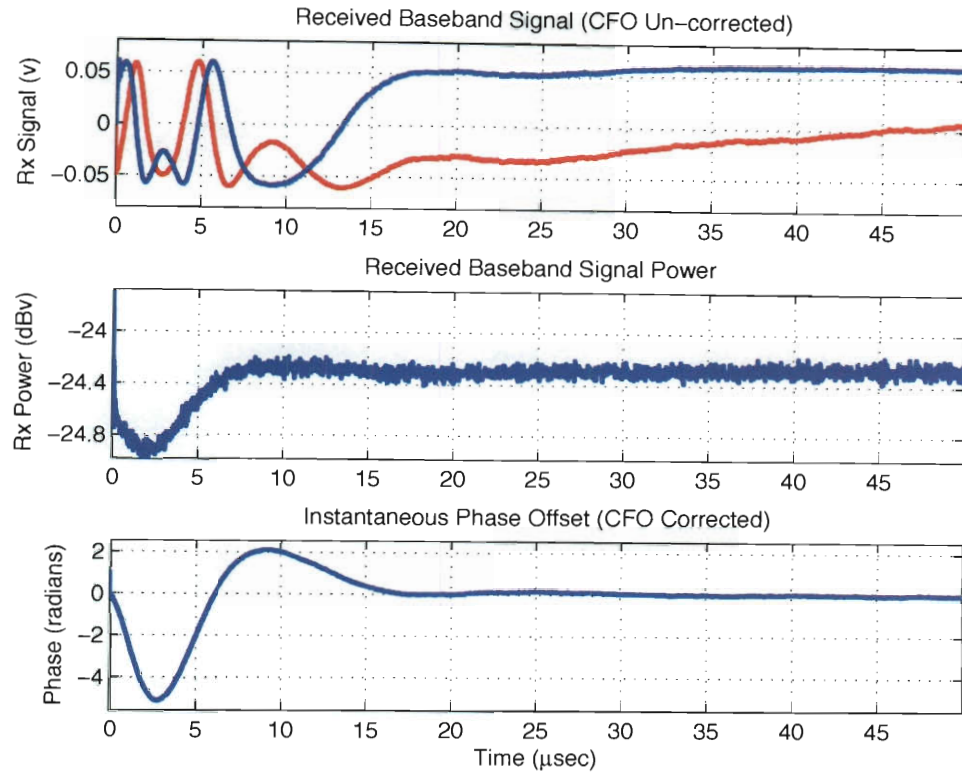


Figure 3.9 : Same data as Figure 3.8, zoomed in on first 50  $\mu\text{sec}$  after the transmitter is enabled.

While the relationship between errors in CFO estimates and phase distortion at the transmitter is clear, the underlying cause of the distortions merits investigation. The rapid phase changes are clearly related to enabling the radio transmitter and power amplifier on the WARP Radio Board, which suggests a source for the distortions in the hardware itself. Thankfully we have access to the right equipment to measure signals in hardware at runtime.

In the WARP Radio Board design, the MAX2829 transceiver and PA share a power supply [25]. This supply is a linear voltage regulator on the radio board which regulates down the board-level 3.3 V supply to a 2.9 V supply used by the MAX2829 and PA. We can measure the instantaneous voltage of this supply in real-



time using a high speed oscilloscope. In fact, we can measure the voltage very near to the MAX2829 by probing across the pins of a decoupling capacitor adjacent to the transceiver. A photo of this probing setup is show in Figure 3.10. The blue tips of the oscilloscope probes are visible on either side of the (very small) capacitor. A third blue probe tip is visible in the upper-left, probing a digital signal which asserts when the MAX2829 and PA are enabled.

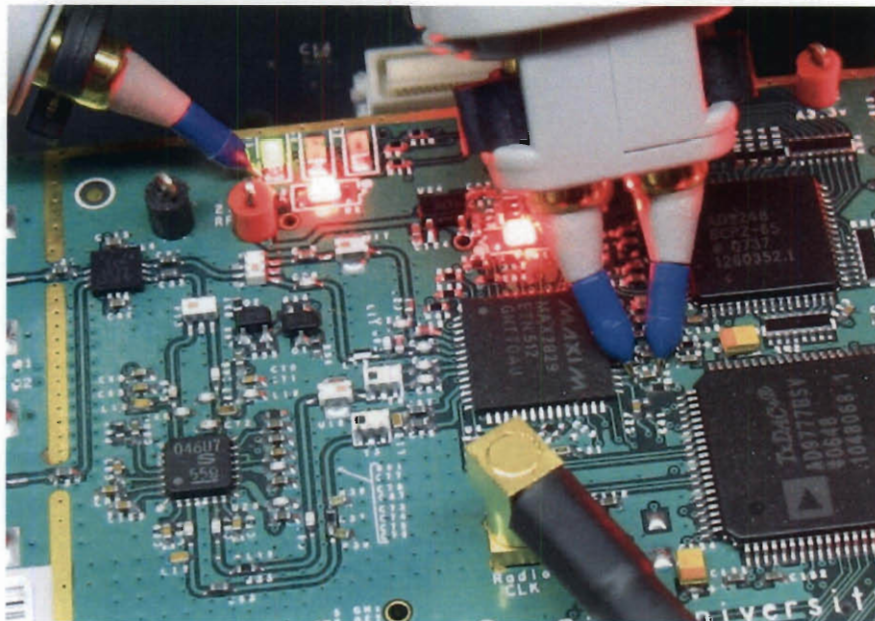


Figure 3.10 : Measuring the instantaneous voltage on the WARP Radio Board.

Figure 3.11 shows the oscilloscope traces for this test. The top trace is the digital signal indicating when the transmit cycle begins. The bottom trace is the voltage of the supply feeding the MAX2829 and PA. The vertical scale is 50 mV per major grid tick, giving a peak-to-peak deviation in the supply voltage of  $\approx 180$  mV. This is a significant fraction of the nominal voltage of 2.9 V. The horizontal scale is 5  $\mu$ sec per major grid tick. Note the time between peaks of the voltage trace, approximately 8  $\mu$ sec. This corresponds exactly to the shapes of the distortions in CFO estimates

(Figure 3.7) and phase distortions calculated by the VSA (Figure 3.9).

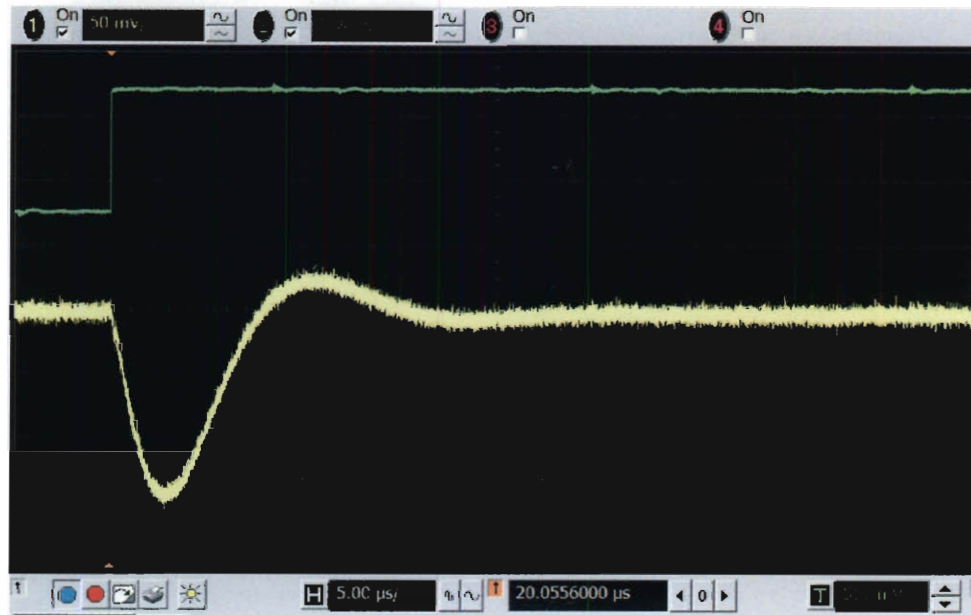


Figure 3.11 : Oscilloscope measurement of the instantaneous transmitter state (top trace) and supply voltage (bottom trace) on the WARP Radio Board during the start of a transmission.

We conclude from this test that the primary cause of the phase distortions we observe early in a transmitted waveform are due to transients in the supply voltage in the radio hardware itself. In retrospect, this result is not surprising. Both the MAX2829 transmit circuitry and the power amplifier draw significant amounts of current from the board's power supply. When these circuits are enabled the current load increases very quickly. The linear regulator requires some time to compensate for the sudden change in load, during which its output voltage droops. As the supply voltage changes, the frequency of the MAX2829 RF carrier drifts as the transceiver's PLL attempts to track the frequency of its VCO relative to the frequency reference. This drift manifests as rapid frequency changes taking on the same shape as the underlying voltage transient. The whole system settles after a few 10's of microseconds,

after which the supply voltage and carrier frequency are stable.

Understanding the underlying cause of the CFO estimation errors allows us to consider potential solutions. One option is to design an algorithm to digitally pre-compensate for the phase distortions in the transmitted waveform. This approach would be very difficult, primarily because the magnitude and timing of the voltage transients are not fixed. The shape of the transient is not constant across WARP radio boards, as it varies with individual component values as well as environmental factors (i.e. temperature). Thus, in order to apply its inverse, any compensation scheme would require frequent re-calibration to measure the current transient profile. And because the WARP Radio Board is a half-duplex transceiver, an additional radio board (or VSA) would be required to act as the receiver in the calibration process.

Instead, we chose the simple solution of introducing a delay between enabling the radio's transmit circuitry and driving the initial samples of our transmitted waveform into the radio board's DAC. Choosing the delay requires placing the preamble's long training symbols late enough in the transmitted waveform that the phase distortions have decayed sufficiently far to not degrade the CFO estimate. The traces in Figure 3.7 provide a good guide on choosing a delay (these curves are representative of all the WARP Radio Boards we tested). In our design, we delay the initial samples by 5  $\mu\text{sec}$ , placing the end of the second long training symbol, and thus valid CFO estimate, at 37  $\mu\text{sec}$  after the transmitter is enabled. At this offset, neither the voltage or phase measurements show any deviation, and the CFO estimate has settled to its expected value.

### 3.5.4 Performance Measurements

Having mitigated the issue of phase distortions early in the transmit waveform, we can finally conduct a meaningful characterization of our CFO estimator's performance.

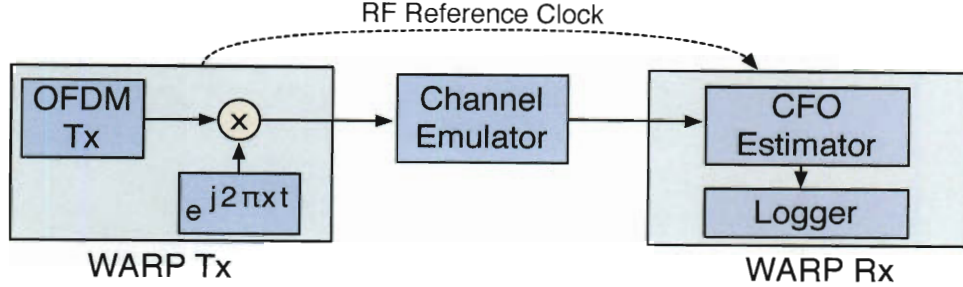


Figure 3.12 : Hardware configuration for characterizing performance of the time domain CFO estimation implementation.

For these experiments, we use a hardware setup illustrated in Figure 3.12. The WARP nodes share an RF reference clock, giving precise control of the CFO via intentional offsets applied at the transmitter. The RF interfaces of the WARP nodes are connected via the Azimuth channel emulator; the emulator is discussed in Section 5.2. The channel emulator enables control of the propagation channel characteristics without having to modify any settings in the WARP nodes themselves. For characterizing our CFO estimator, we use a static channel profile whose only parameter is path loss, realized by a programmable attenuator at the output of each of the emulator's RF outputs.

Using WARPnet we are able to coordinate the behavior of the emulator with our design running on WARP. As explained above, our CFO estimator calculates one estimate per received packet. Thus, to gather sufficient samples of the CFO estimate, we need to transmit and receive many packets per combination of values of our independent variables. For these tests, we sweep both SNR (realized as attenuation in the

inherent phase noise characteristics in the WARP hardware (much less the emulator).

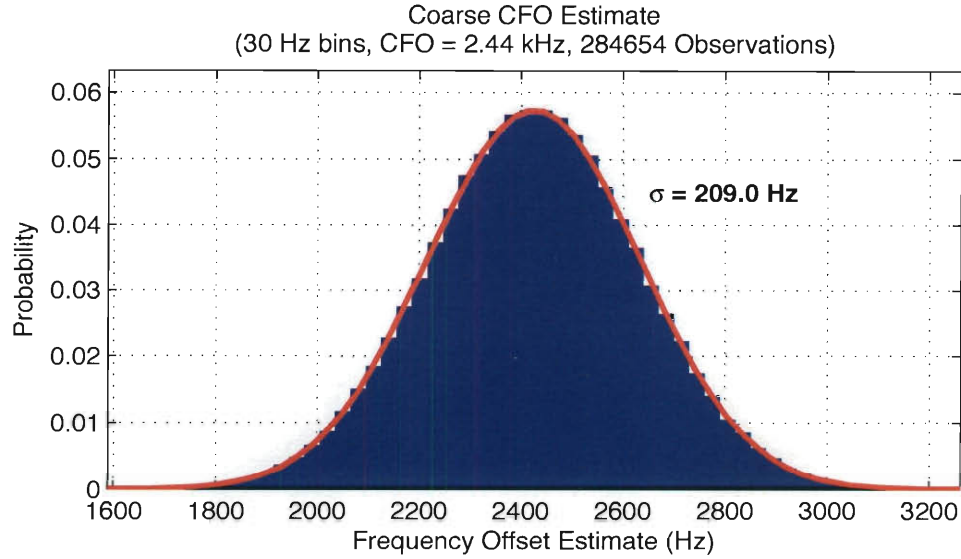


Figure 3.13 : Distribution of time domain CFO estimates for fixed CFO at high SNR, as measured in hardware.

Fortunately, the floor in CFO estimation performance is entirely acceptable for our application. Any estimation error in the time domain CFO correction system will manifest as residual CFO in the frequency domain. An OFDM receiver can tolerate small residual CFOs, where “small” is defined by a negligible degradation in performance due to CFO-induced ICI. For example, consider the case of  $\sigma = 209$  Hz in Figure 3.13, and compare this to the CFO vs. BER curves in Figure 3.2. Assuming a normal distribution for CFO estimates (well supported by the fit in Figure 3.13), our implementation can provide an estimate within 627 Hz ( $3\sigma$ ) for more than  $> 99\%$  of packets. A residual CFO of 627 Hz induces negligible ICI in an OFDM system with subcarrier spacing of 156.25 kHz (64 subcarriers in 10 MHz bandwidth).

For a clearer view of this, consider Figure 3.15. These BER values are drawn from the same data as Figure 3.2, isolating the points for CFOs below 1 kHz. At every

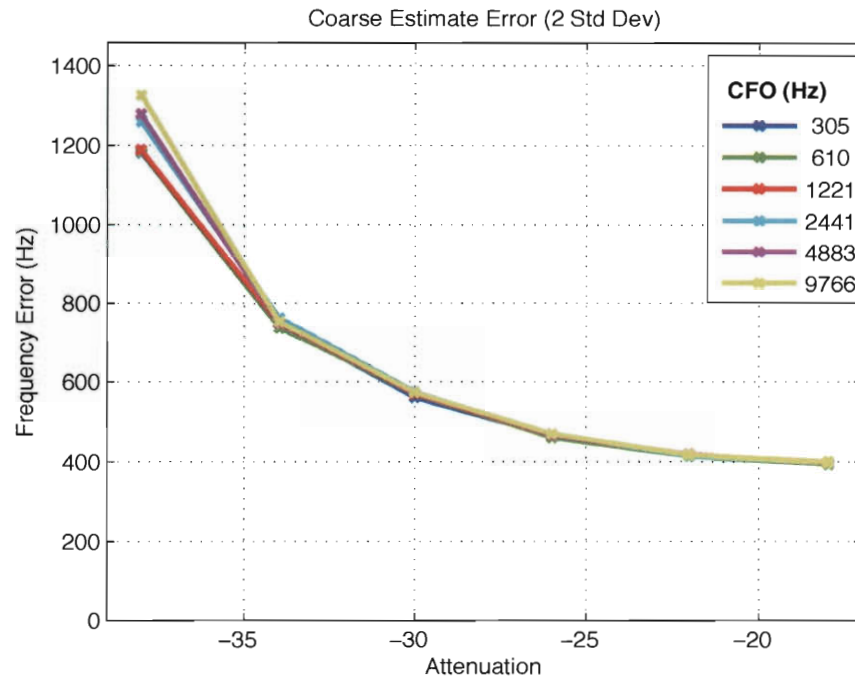


Figure 3.14 : Estimation error ( $2\sigma$ ) for the time domain CFO estimator for multiple CFOs and SNRs, as measured in hardware.

CFO, the BER performance is dominated by SNR, with only the highest SNR curve showing even a small performance degradation at the highest CFO.

### 3.6 Frequency Domain Phase Correction

As discussed above, the time domain CFO estimation system seeks to minimize the impact of ICI by reducing the CFO before samples are fed into the receiver's FFT. However, it is clear this estimator will rarely remove all CFO, leaving a residual offset which will propagate into the frequency domain. The impact of this residual offset is a time varying phase offset. The offset is the same across subcarriers in a single OFDM symbol but increases linearly with each OFDM symbol at a rate proportional to the residual CFO. While the time domain estimator bounds the residual offsets to a



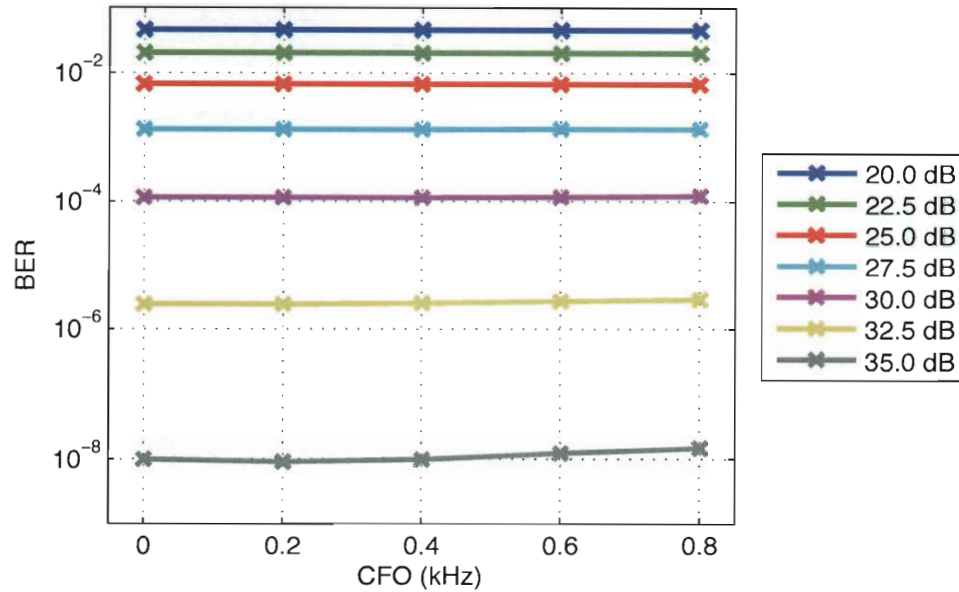


Figure 3.15 : BER vs. CFO, for small CFOs and multiple attenuations.

range for which ICI is negligible, these increasing phase offsets must still be corrected as part of the channel equalization processing.

For example, consider a residual CFO (i.e. time domain CFO estimation error) of 200 Hz ( $\approx 1\sigma$  for the estimate distribution in Figure 3.13) and a full-length packet modulated at QPSK. This packet occupies  $\approx 10,000$  samples which, at 10 MHz bandwidth, span a 1 msec duration. Over the course of 1 msec, a 200 Hz frequency offset will cause phase offsets which increase from zero (at the start of the packet) to  $72^\circ$  (at the packet's end). For QPSK, any uncorrected offset larger than  $45^\circ$  guarantees a symbol error. In a real system, any uncorrected phase offset will degrade performance via reduced noise margins by rotating received symbols away from the centers of the constellation decision regions. Our receiver clearly needs to correct for these phase errors, even when caused by very small residual frequency offsets.

There are two primary components to this process: phase error estimation and

emulator) and CFO (controlled by the artificial CFO applied at the transmit WARP node).

In this experiment, one WARP node periodically transmits a full-length OFDM packet. The other node receives the transmission, attenuated via the channel emulator, and extracts a CFO estimate. This estimate is transmitted via Ethernet using WARPnet for logging and offline processing. As the design runs in real-time, we can gather a significant number of samples from our estimator in fairly short experiments.

Figure 3.13 shows the distribution of CFO estimates for one combination of attenuation and CFO. The bars represent the normalized histogram of the CFO estimates, calculated with 30 Hz bins. The overlaid line shows the Gaussian curve calculated using the empirical mean and standard deviation. The curve clearly fits the data very well. The mean estimate is equal to the actual CFO, as expected.

The results of the full sweep of SNRs and CFOs are shown in figure 3.14. Here, the performance is characterized by the standard deviation of the estimates at each point, drawn as  $2\sigma$ . There are a few things worth noting here. First, the performance of the estimator is largely independent of the actual CFO. This is as expected; the estimation algorithm is designed to function equally well over a wide range of offsets, breaking down only for very large offsets, much larger than our hardware will experience.

Second, the performance clearly improves with increasing SNR, but does show a floor developing at the highest SNRs. This is consistent with our expectations. Increasing the SNR (realized by applying less attenuation between the Tx and Rx nodes) reduces the impact of additive noise at the receiver, but does not reduce the impact of phase noise incurred at both transmitter and receiver. Given that our CFO estimator derives its estimate from phase values, the overall performance should be dominated by phase noise above some SNR. We cannot control or eliminate the



phase error correction. The phase error estimates serve as inputs to the correction calculation, which provides an updated phase correction value for each OFDM symbol to be applied during equalization.

Our transceiver, like most OFDM systems, dedicates subcarriers for use as pilot tones. We use the same subcarrier mapping as IEEE 802.11a, with four pilots per OFDM symbol, each populated by pseudo-random BPSK symbols. The pseudo-random sequence is the same for every packet; the receiver can estimate instantaneous phase errors from the difference in phases between the transmitted and equalized pilots. Our design computes the phase error of each pilot, then takes the average to produce a single error estimate per OFDM symbol.

One complication arises in adapting this scheme for a  $2 \times 1$  Alamouti system. It is generally undesirable to transmit the same waveform from multiple antennas simultaneously, in order to avoid the unintentional formation of beam patterns that may degrade signal quality at receivers via destructive combining. In the context of OFDM, this means we should avoid sending similar signals (like pilot tones) on the same subcarrier from multiple antennas (or multiple nodes) simultaneously.

One option would be to Alamouti-encode the pilot tones, giving them the same space-time orthogonality as the payload symbols. This approach would require the receiver to perform Alamouti combining before useable pilot tones were available for phase error estimation. However, Alamouti coded systems are especially sensitive to phase errors which vary between received symbols, as they do when caused by frequency offsets. This property of the Alamouti STBC has been analyzed at length in the literature [30, 31].

An intuitive explanation for this sensitivity can be found in the quasi-static fading assumption underlying the Alamouti code. The Alamouti code requires channel co-

efficients remain constant (in both magnitude and phase) across two symbol periods. However, a carrier frequency offset induces a phase offset which changes with every received symbol. If the per-symbol phase offsets are left uncorrected before Alamouti combining, the transmitted symbols are not fully re-orthogonalized, resulting in an effective SNR loss due to ISI.

A straightforward way to combat this sensitivity is to apply per-symbol phase corrections before the combining stage. This approach means the pilots cannot be Alamouti encoded, necessitating we use some other orthogonalization method. We use a simple scheme of orthogonalizing the pilots explicitly, by interleaving them between antennas in time and frequency, but never transmitting pilots on the same subcarrier from both antennas (or from both the source and relay nodes) at the same time. Our interleaving pattern is illustrated in Figure 3.16. There are four pilot tones per OFDM symbol period, two on each stream (a stream corresponds to one antenna in a two-antenna system, or to one node in a two-node cooperative transmission). Swapping the subcarrier assignments between streams in alternate symbol periods greatly simplifies the implementation. Note this interleaving pattern assures at least two pilot tones are transmitted in every symbol period, even if one stream is missing. This is an important property as it helps preserve diversity, allowing the receiver to estimate phase errors with every symbol even in cases where one stream is missing (e.g. due to a deep channel fade).

As with time domain CFO correction, the literature offers a wide range of phase error correction schemes. We utilize one developed by researchers at Toshiba [32]. This algorithm is designed explicitly for a real-time system, generating phase correction values per OFDM symbol with very low latency. The implementation of this technique is straightforward, requiring only a few registers and arithmetic blocks, and

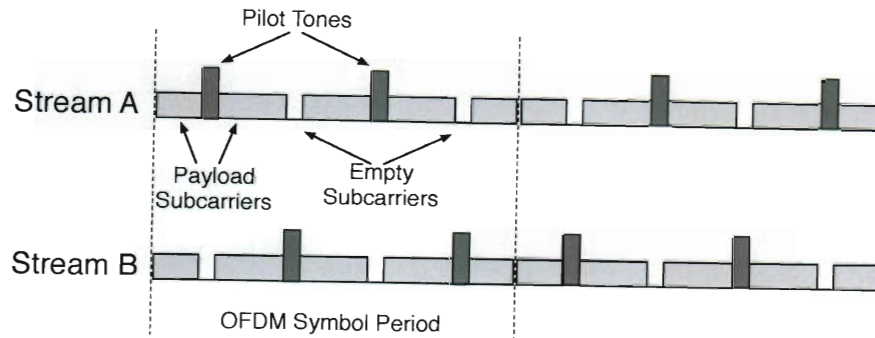


Figure 3.16 : Subcarrier mapping of pilot tones, interleaved in time and across antennas.

functions as expected, providing reliable phase correction values which are applied to received symbols immediately before Alamouti combining.

### 3.7 CFO in a Cooperative System

The discussion above presents the issue of CFO for a standard OFDM system, our expectations for CFO between WARP nodes and the successful design and evaluation of a time domain CFO correction system. However, additional CFO issues arise in extending our OFDM design to support cooperative communications.

As discussed in Chapter 2, our work focuses on cooperative communications among fully distributed nodes. These nodes must coordinate their behavior using the same wireless interfaces over which they communicate payloads. Most importantly, they share no back-channel synchronization mechanisms. From the perspective of CFO, this means each node operates with an independent local reference oscillator whose frequency varies relative to every other node. In the case of non-cooperative transmissions, the inter-node CFOs are handled by the estimation and correction systems discussed above. However, when multiple cooperating nodes transmit simultaneously,

a new complication must be addressed.

Consider the simple two slot cooperative exchange illustrated in Figure 3.17. In the first time slot, the source node transmits a packet which is received by the relay and (potentially) by the destination. Both the relay and destination apply the CFO estimation/correction scheme discussed above in attempting to receive the transmission. Each node observes a different CFO, defined by the offset of its local oscillator to that of the source. In the second slot, both the source and relay transmit to the destination. In this case, the destination receives a waveform which is the sum of two transmissions, each with its own CFO. This is a different problem, one which the standard OFDM CFO estimation/correction schemes discussed above cannot address.

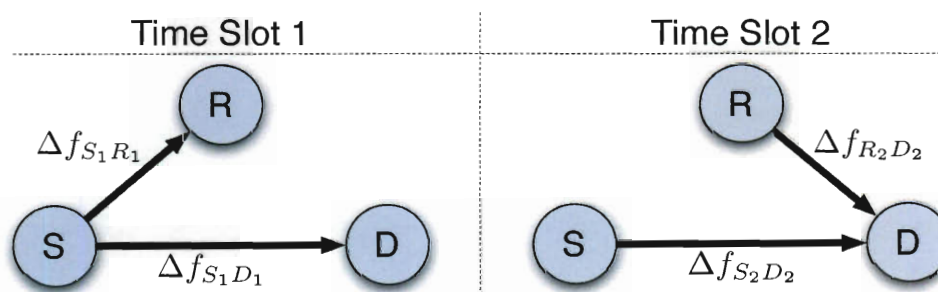


Figure 3.17 : Simple two slot cooperative exchange, indicating each inter-node CFO from the perspective of the receiving nodes.

### 3.7.1 Temporal Properties of CFO

The first step in understanding the issue of dual-CFOs is to investigate whether, in a two time-slot cooperative transmission, knowledge of CFOs from the first slot can be useful in mitigating CFO in the second. Such knowledge will only be useful if the carrier frequencies remain nearly constant across the two time slots. This will be determined by the “coherence time” of the carrier reference oscillators. Unfortu-

nately, short-term temporal stability is not specified for the TCXO's used on WARP (or for any TCXOs we have seen). Instead, we need to characterize the stability experimentally.

Fortunately, we can use the same WARP setup discussed above. In fact, we can use the same data underlying the distributions shown in Figure 3.4. In that experiment, the transmitting WARP node sends a new packet every 2.1 msec for 40 minutes. The receiver computes a new CFO estimate for every packet, which is logged via WARPnet.

Figure 3.18 shows the results from the same experiment, plotted here as individual CFO measurements verses time. These plots present the time-series data used to calculate the histogram in Figure 3.4(b). The four subplots show the data at various time scales, zooming in from the full view of 1.1 million samples in 40 minutes down to 47 samples in 100 msec. The X-axes are all actual time, measured in seconds. The Y-axes are the measured frequency offset in Hz.

A few characteristics stand out. First, at every time scale, the offsets appear random. This is consistent with our expectations, given the random nature of the underlying causes of oscillator frequency variation. Second, on short time scales, the measured offset either changes by very small or very large amounts. For example, note the sudden changes in Figure 3.18(b). On a few occasions, the CFO jumped by more than 100 Hz in a single packet duration. But these jumps are rare; the vast majority of inter-packet CFO changes are small, which is clear in Figure 3.18(d).

One useful way to better quantify the temporal behavior of CFO is to calculate the distribution of inter-packet changes in observed offsets. This is straightforward, given the periodic transmissions used to gather the CFO vs. time data above. The results are shown in Figure 3.19. This distribution is computed as a histogram of

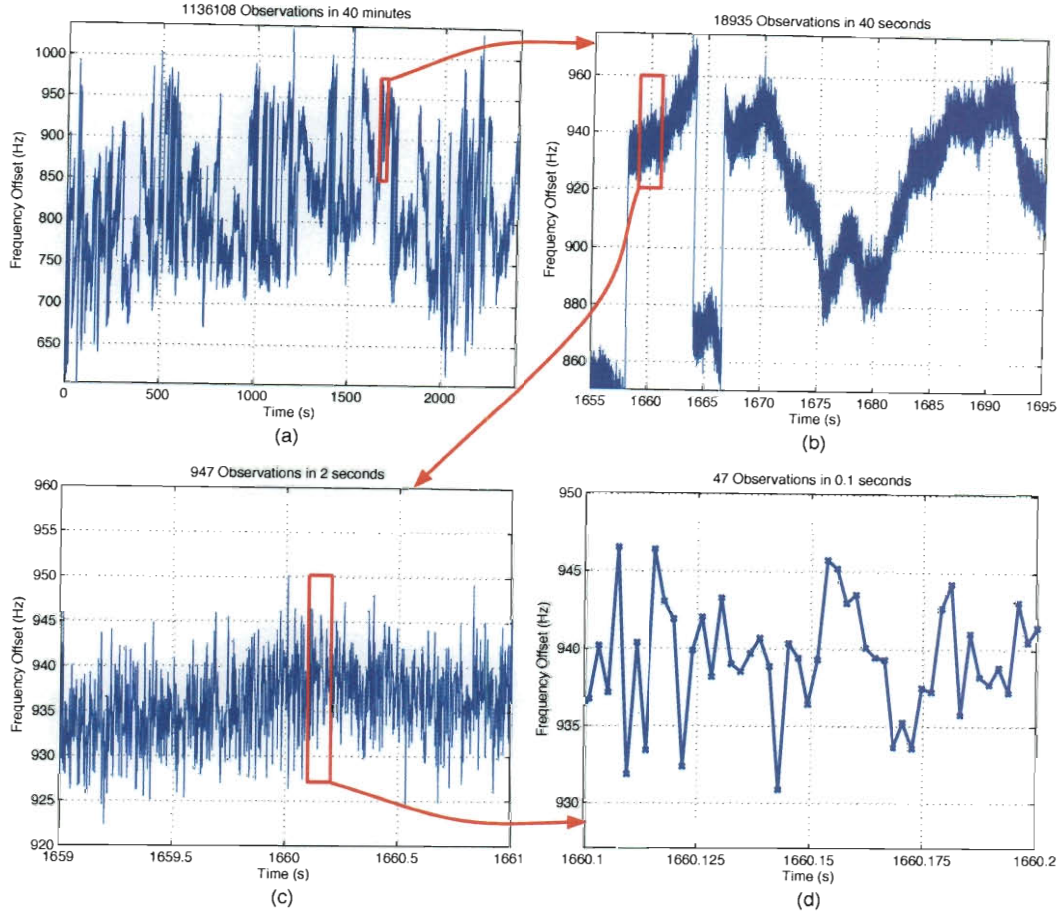


Figure 3.18 : Measurements of actual CFO between two WARP nodes, captured at 2.1 ms intervals for 40 minutes. Plots (b)-(d) each show a subset of data from plot (a) over smaller time intervals.

the absolute inter-packet frequency offset change, with packets spaced at 2.1 msec intervals. The histogram bins are 1 Hz wide.

Of particular importance is how the inter-packet frequency drift is heavily concentrated towards low values (note the Y-axis is probability on a log scale). In this experiment, the probability of a CFO change greater than 100 Hz in 2.1 msec is just  $1.7 \cdot 10^{-4}$ . This is promising, indicating that we can expect very similar CFOs across the two time slots of a cooperative transmission. In fact, this experiment is

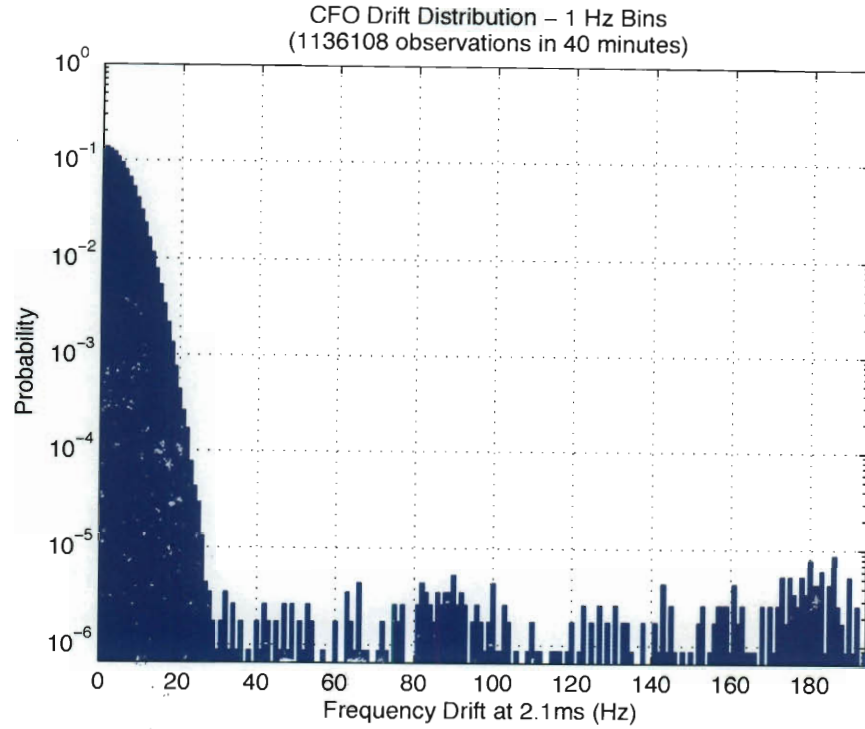


Figure 3.19 : Distribution of CFO drift, from the same experiment as Figure 3.4(b).

rather pessimistic, transmitting once every 2.1 msec. This is the maximum duration of a packet transmission in our design, corresponding to a full length (1500 byte) payload modulated with BPSK. When using faster modulation rates, this duration drops significantly ( $\approx 1$  msec for QPSK,  $\approx 500$   $\mu$ sec for 16-QAM), further reducing the probability of large CFO changes between time slots. These results suggest that it is fair to assume carrier frequencies will remain nearly constant across the two back-to-back slots of a cooperative transmission as illustrated in Figure 3.17 (i.e.  $\Delta f_{X_1 Y_1} \approx \Delta f_{X_2 Y_2}$ ).

### 3.7.2 Mitigating CFO in a Cooperative System

The problem of different carrier frequency offsets among cooperating transmitters and techniques to mitigate their impact have been discussed in the literature [33–35].

In [33], the authors develop an algorithm for mitigating the impact of dual frequency offsets in a cooperative system employing a distributed version of the Alamouti code. Their technique adds an iterative equalizer in the frequency domain which attempts to remove two sources of error: ICI, due to lost orthogonality between subcarriers in the FFT, and ISI, due to lost orthogonality between Alamouti streams. Unfortunately, the estimation process in [33] requires knowledge of the CFO and propagation channel for each transmitter and the number of nodes participating in each transmission. It also adds significant complexity to the OFDM receiver. And while their scheme outperforms previously proposed alternatives the authors acknowledge it still falls short of the performance possible when no CFO exists between transmitting nodes.

In [34] the authors propose a scheme for fully mitigating the impact of dual frequency offsets, but their approach is unsuitable for use in our system for a number of reasons. First, it requires extending the cyclic prefix of every OFDM symbol by a huge amount. For two transmitting nodes, the new prefix would nearly double the duration of a packet. This extra transmission time is all overhead. Second, this technique requires a substantial amount of processing in the time domain, including operations which require knowledge of the time domain response of the channel to each transmitter. This degrades one of the key benefits of using OFDM, wherein a single frequency domain channel coefficient is required for equalizing each subcarrier. While this technique can (in theory) match the performance of a system with no CFO between transmitting nodes, its complexity and estimation requirements render



it unusable for our implementation.

Finally in [35], the authors consider the design of a distributed Alamouti system very similar to ours. They develop an extension of moderate complexity to the standard Alamouti receiver which can compensate for different frequency offsets at the transmitting nodes. The authors also analyze the sensitivity of their scheme to errors in the estimates of these CFOs. In their analysis, they find the scheme to be very sensitive to even small estimation errors. Their results show that for reasonable CFOs and SNRs, CFO estimation errors of just a few percent reduce BER performance below that of a non-cooperative SISO link. Given our experiments with CFO estimation accuracy and temporal oscillator frequency variations, we decided against basing our implementation on this algorithm.

Instead, our design uses a different approach. Rather than require the destination to estimate both the  $\Delta f_{SD}$  and  $\Delta f_{RD}$  carrier frequency offsets and correct their combined impact, our design attempts to mitigate the dual-CFO issue at the relay. This technique requires the relay attempt to remove the effect of its own carrier frequency whenever it cooperates in a transmission with the source. This is rather intuitive, seeking to have cooperating transmitters act as a virtual antenna array operating with a common carrier frequency.

Pre-correcting the CFO at the relay provides two key benefits. First, no extra processing is required at the destination, preserving the very useful feature of a receiver design which does not require knowledge of how many nodes are participating in a given transmission. Second, no extra overhead is required; the relay extracts everything it requires from the source's normal transmission.

The methods required to realize CFO pre-cancellation at the relay depend on the underlying cooperative scheme.

### 3.7.3 CFO with Amplify and Forward

We start our investigation of CFO in an amplify and forward system by constructing a simple model of the relevant baseband and RF signals. Consider a network of three nodes S, R and D. Assuming each node employs an independent oscillator, their local carrier frequencies are  $\omega_S$ ,  $\omega_R$  and  $\omega_D$ , respectively.

It is clear that any signal the destination receives directly from the source will have a frequency offset of  $(\omega_S - \omega_D)$ . The impact of CFO on the signal which propagates through the relay requires more careful consideration. Figure 3.20 illustrates the multiple baseband and RF signals which constitute the path through the relay.

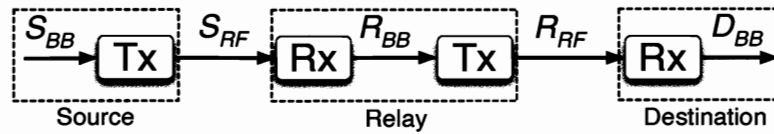


Figure 3.20 : Signals for the path through the relay in an amplify and forward link.

Using the RF transceiver models and expressions from Section 3.1, we can trace the impact of CFO through this path. Expressing  $R_{BB}$  in terms of  $S_{BB}$  and the source/relay carrier frequencies gives

$$R_{BB} = S_{BB}(e^{jt(\omega_S - \omega_R)}). \quad (3.3)$$

Repeating this process to find  $D_{BB}$  in terms of  $S_{BB}$  gives

$$\begin{aligned} D_{BB} &= R_{BB}(e^{jt(\omega_R - \omega_D)}) \\ &= S_{BB}(e^{jt(\omega_S - \omega_R)})(e^{jt(\omega_R - \omega_D)}) \\ &= S_{BB}(e^{jt(\omega_S - \omega_D)}). \end{aligned} \quad (3.4)$$

Notice that  $D_{BB}$  is not a function of  $\omega_R$  and that the frequency offset here ( $\omega_S - \omega_D$ ) is the same as for the direct source-destination path. The process of AF relaying, realized as downconverting then upconverting the same baseband waveform, effectively removes any trace of the source-relay frequency offset, transmitting a signal at exactly the source's carrier frequency.

This is a very useful property. Amplify and forward relaying inherently provides CFO pre-cancellation at the relay without any extra processing. Of course, perfect pre-cancellation requires no change in carrier frequencies between the two time slots. However, any pre-cancellation scheme (including the one we use for DF, discussed below) has the same dependence on the temporal stability of carrier frequencies. With real-world oscillators, no pre-cancellation will be perfect, but AF inherently achieves the base-case.

#### 3.7.4 CFO with Decode and Forward

Our DF implementation essentially attempts to actively mimic the self-cancellation of CFO at the relay which occurs passively with AF. This approach adds some complexity to the relay's receiver, but is far simpler than implementing the alternative destination-based dual-CFO estimation/mitigation schemes discussed above.

The basic process for pre-cancellation of CFO is for the relay to estimate  $\Delta f_{SR}$  (the source-relay CFO) in the first time slot, then apply the opposite frequency shift to its own transmission. Ideally this will result in a relay transmission whose carrier frequency exactly matches that of the source. From the destination's perspective, it will receive the sum of two waveforms, each of which is shifted by the same offset ( $\Delta f_{SD}$ ), just as if a single two-antenna node had transmitted both waveforms.

There are two limiting factors in this approach. First,  $\Delta f_{SR}$  must not change

significantly between time slots. Based on the results presented in Section 3.7.1, we know the CFO is fairly stable on packet time scales. In the vast majority of cases, the CFO drift over 2.1 msec was less than 20 Hz; we need to establish whether even this small drift is tolerable. Second, the accuracy with which the relay can estimate  $\Delta f_{SR}$  is critical. To understand this issue, we must quantify the tolerance for CFO estimation errors at the relay.

Recall that CFO degrades performance in an OFDM two ways: via ICI, due to lost orthogonality in the FFT, and via phase offsets, which increase over the course of a packet reception. We will focus on the impact of the phase offsets here (as will be clear below, the magnitude of CFOs which matter here are much smaller than those for which the effects of ICI dominate).

A thorough analysis of the impact of different CFOs for cooperating nodes is provided in [35]. For our purposes, we only need a rough idea of the CFO tolerance, primarily to decide whether our existing time domain CFO estimator is sufficiently accurate for use in pre-correcting CFO at the relay.

We first test this tolerance experimentally. Recall the experimental setup described above for characterizing the performance of our time domain CFO estimator. We use a similar setup here, illustrated in Figure 3.21. Two nodes (source and relay) share reference clocks, which allows a precise frequency offset to be applied to the source's transmission. A third node (the destination) acts as the receiver. The two transmitters implement a distributed Alamouti encoder, with each node transmitting complementary halves of the usual two-antenna Alamouti transmission. The receiver operates independently, implementing a standard Alamouti receiver. We sweep the frequency offset between the two transmit nodes starting at 0 Hz and record the BER and PER for each offset. As a reference, we also test the BER/PER of a single-

node transmission at each point. These tests are conducted via the channel emulator at high SNR with frequency flat fading, using 1500 byte payloads modulated with QPSK.

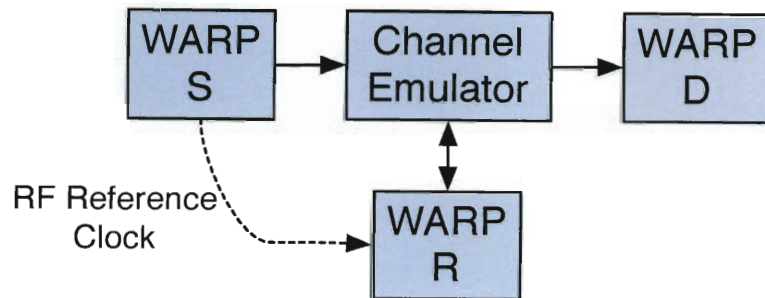


Figure 3.21 : Hardware configuration for characterizing the destination's tolerance for CFO between source and relay transmissions.

The results of this experiment are shown in Figure 3.22. The X-axis is the CFO between transmitters in Hz. The Y-axes are the packet and bit error rates, respectively. It is clear that even a small frequency offset between transmitting nodes has a significant impact on performance. In fact, any offset greater than  $\approx 120$  Hz reduces performance below that of a single antenna transmission.

This observation indicates our existing time domain CFO estimator is insufficient for CFO pre-correction at the relay. Based on the distribution of CFO estimates in Figure 3.13, 54% of time domain CFO estimates deviate from the actual offset by more than 120 Hz under ideal propagation conditions (high SNR, static channel coefficients). We clearly require a more accurate CFO estimator for use in pre-correcting offsets at the relay.

We can verify this requirement with a small modification to our experiment. The two transmit nodes still share reference clocks, and one node (the source) continues to apply a small frequency offset to its transmission. The other (the relay) estimates

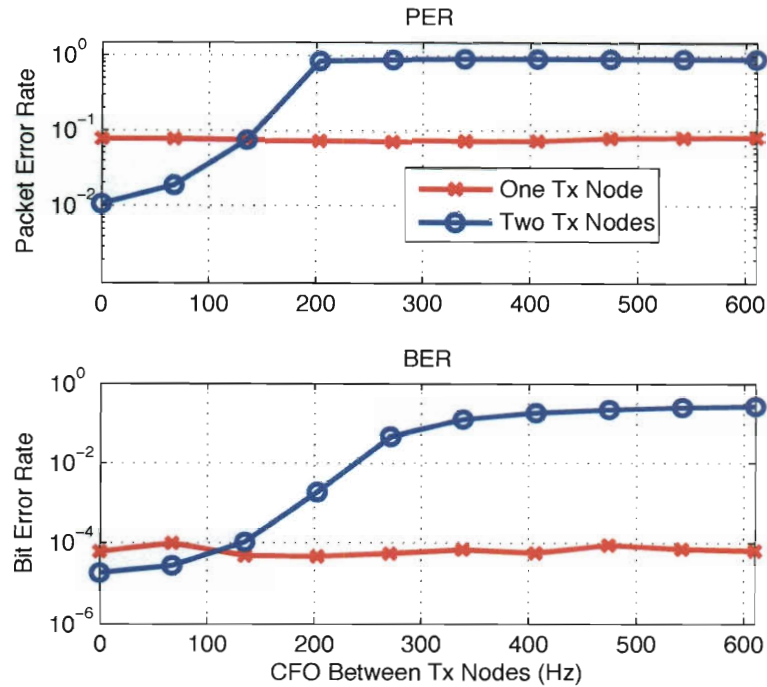


Figure 3.22 : Experimental PER/BER performance vs. CFO between transmitting nodes.

this offset using its time domain CFO estimator while receiving the source's transmission, then applies this offset to its own transmission to the destination. The results of this modified experiment are shown in Figure 3.23. The axes are the same as above, and the PER/BER for a single antenna (source-only) transmission is included for reference. As expected, the performance of the two-node transmission is independent of the source/relay CFO, since the time domain CFO estimator functions equally well over a wide range of offsets. However, the two-node performance is now consistently much worse than that of the single-antenna link. This can be understood by recalling the distribution of CFO estimates produced by the time domain estimator (Figures 3.13 and 3.14), and comparing the distribution to the BER/PER curves in Figure 3.22. With high probability, the error in the time domain CFO estimate will

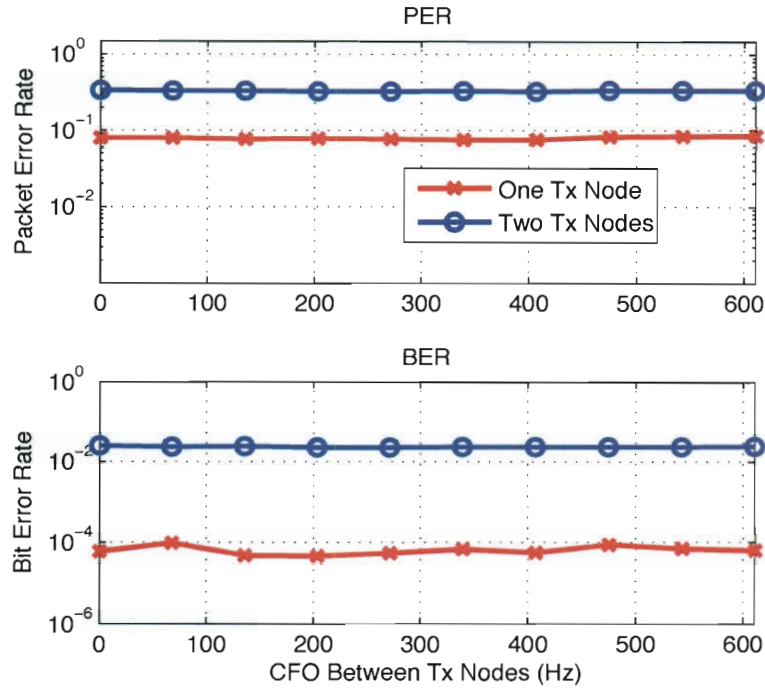


Figure 3.23 : Experimental PER/BER performance vs. CFO between transmitting nodes, where one transmitting node applies CFO pre-correction based on its time domain CFO estimate.

exceed the very small tolerance for inter-transmitter CFO. When the error is sufficiently large, symbol errors due to diverging CFO-induced phase errors will dominate PER/BER performance, as clearly illustrated in Figure 3.23.

One other observation we can make from these results relates to the inter-packet drift in carrier frequencies. Based on Figure 3.22, the performance degradation due to inter-transmitter CFO is very small for offsets smaller than  $\approx 30$  Hz. Compare this to the frequency drifts plotted in Figure 3.19. This is very promising, indicating that the vast majority inter-packet frequency changes will be small enough to have negligible impact on the performance of a two time-slot cooperative transmission.

One final aspect of CFO estimation for pre-correction at the relay is that the

tolerances for estimation errors varies with the modulation order and duration of cooperative transmissions. The effect of CFO at the destination is a phase error which increases linearly with time, where the rate of increase is proportional to the frequency offset. These phase errors manifest as rotations of the constellations in the receiver. Thus, longer packets and those using higher order modulations are more sensitive to errors in the pre-correction of CFO at the relay. This observation plays an important role in evaluating designs for the relay's CFO estimator.

### 3.7.5 Frequency Domain Residual CFO Estimation

The time domain CFO estimator discussed above works for reducing CFO to a tolerable level for successful reception of single-node transmissions, but provides estimates with too high a variance for use in pre-correcting CFO at the relay. We need to develop a secondary CFO estimator for this purpose.

One reason the time domain CFO estimator operates with high variance is that it must calculate the CFO using only a small number of samples from the packet preamble in order to begin correcting the CFO before any data-bearing samples are input to the FFT. In the case of CFO pre-correction at the relay the timing is relaxed significantly, requiring an estimate only before the relay begins its cooperative transmission in the second slot. This allows the relay to use much more information from the packet received in the first slot to derive its CFO estimate.

The obvious place to start with designing this CFO estimator is with the phase error estimator already used in the OFDM receiver. Recall that our OFDM design dedicates four subcarriers for use as pilot tones. These tones allow the receiver to estimate and correct the phase error in each OFDM symbol. This system seeks to correct the phase errors caused by residual CFO, but does not actually estimate the



residual CFO itself.

However, we can use the same per-symbol phase error estimates to calculate the residual CFO. The basic process for this is depicted in Figure 3.24. We can calculate the frequency offset by taking the difference between the phase error estimates for two OFDM symbols, then dividing by the length of time separating them. In the presence of noisy phase estimates, some amount of averaging will be required to realize a sufficiently accurate CFO estimate.

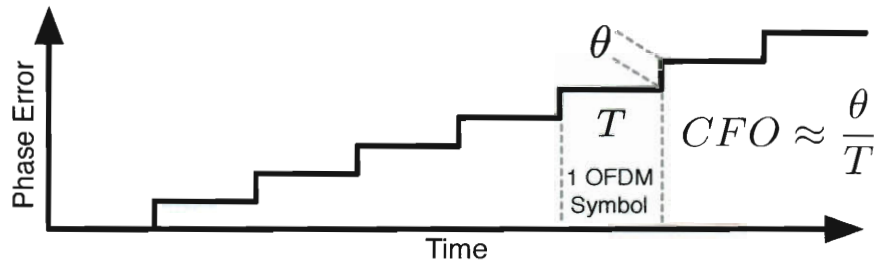


Figure 3.24 : Inferring CFO from phase error estimates calculated per OFDM symbol.

We consider three schemes for calculating the CFO based on the frequency domain phase error estimates:

- **Block Averages:** calculate the slope of the phase errors over the span of the final 8 or 16 OFDM symbols, and take the average of 8 or 16 of these estimates;
- **Last Symbol:** divide the phase error of just the final OFDM symbol by that symbol's time offset from the packet start;
- **N Symbol MRC:** divide the phase errors of the latter half of OFDM symbols by their time offsets from the packet start, and combine the estimates via MRC, weighting later estimates higher.

All three of these methods attempt to calculate the slope of the phase offset vs. time curve, using different combinations of phase error estimates and time spans. We evaluate these schemes in MATLAB via Monte Carlo simulations of a single-antenna OFDM link. The simulation parameters (assignment of data and pilot subcarriers, sampling rate, packet lengths, etc.) are matched to our hardware implementation. Both AWGN and a frequency offset are applied between the simulated transmitter and receiver. We test each scheme at a realistic residual CFO (305 Hz) at various SNRs and for various packet durations. Packet durations are enumerated by the number of OFDM symbols (even in hardware, every transmission consists of an integral number of OFDM symbols). The actual payload length and modulation rates are irrelevant, as the CFO estimates are based only on phase estimates derived from the pilot tones present in every OFDM symbol.

The results of the simulations are shown in Figure 3.25. The X-axes are SNR for the AWGN; the Y-axes are carrier frequency estimation error in Hz. The three plots each show results for a different packet duration. The dotted and dashed lines show the maximum tolerance for CFO estimation error if the payloads were modulated with QPSK or 16-QAM. Notice that the error tolerance is lower for 16-QAM, and both tolerances drop with increasing packet length. The packet lengths here are measured in OFDM symbol periods.<sup>1</sup>

The four traces show results for each estimation scheme, measured as the standard deviation of each estimator's output (in every case, the mean estimates matched the actual CFO). It is immediately clear that every scheme improves with increasing

---

<sup>1</sup>The packet duration for a given payload depends on the modulation rate. For example, a full length payload (1500 bytes) modulated at QPSK occupies 125 OFDM symbols (12 bytes/symbol); for 16-QAM, the same payload fills 63 symbols (24 bytes/symbol). Pilot tones are inserted in every symbol, regardless of the payload modulation rate.

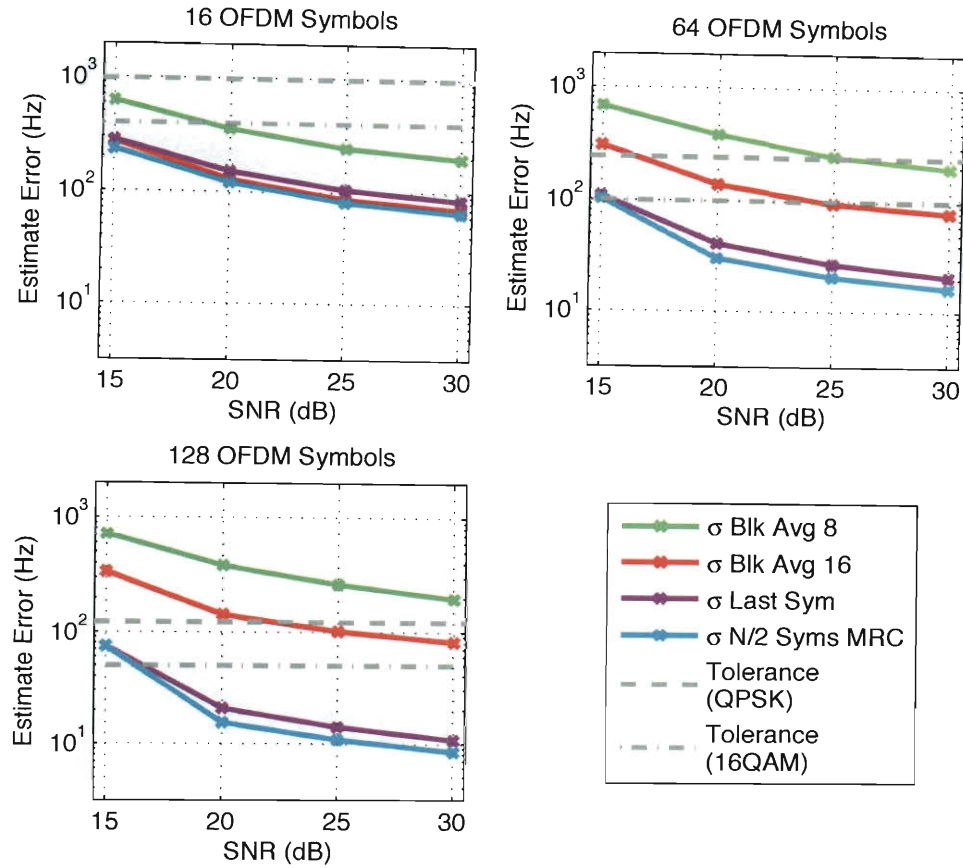


Figure 3.25 : Performance of various frequency domain CFO estimation schemes vs. SNR for various packet durations (durations are measured in OFDM symbols).

SNR. This makes sense, as the underlying pilot-derived phase error estimates will degrade with higher noise levels. It is also clear the performance of the two block averaging schemes are independent of packet duration. Again, this is expected, given that these schemes use a fixed number of phase error values at the end of each packet, for any packet duration. Finally, the final two schemes both show good performance which improves with longer packets. These schemes also consistently perform better than the required tolerances, showing improved performance with increasing packet lengths in line with the tightening tolerances.

Perhaps most promising is the comparable performance of the two best schemes, given their relative complexity. The  $(N/2)$ -MRC scheme performs best, as expected, but is the most computationally complex option. The last-symbol-only scheme performs almost as well, but is, by far, the least complex of all the schemes. From an implementation perspective, the last-symbol-only scheme is very attractive, requiring only the existing phase error estimator, plus a single multiplier and lookup table to store normalization factors for each possible duration. As such, this is the scheme we use in our hardware implementation.

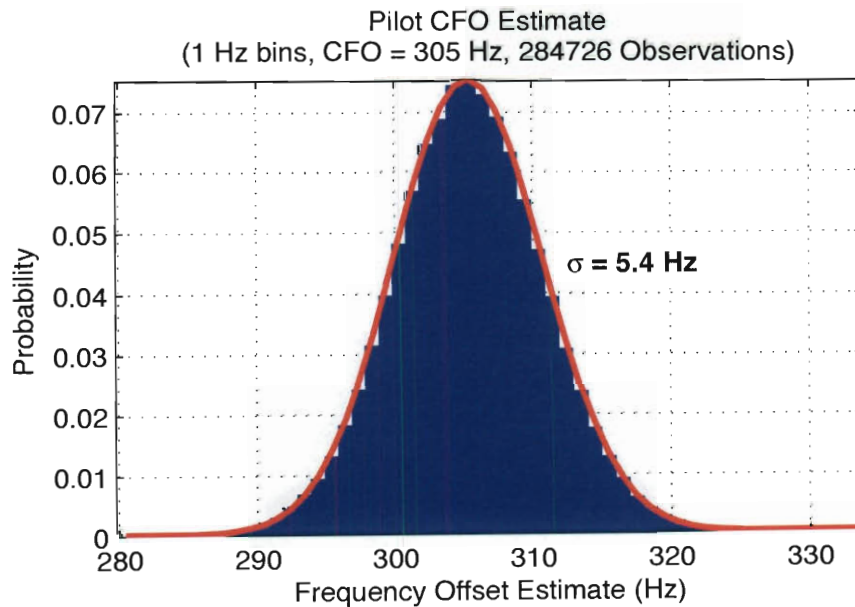


Figure 3.26 : Experimental distribution of frequency domain residual CFO estimates.

We evaluate our implementation of this estimator using the same hardware configuration described in Section 3.5.2. In this test, the artificially induced CFO is smaller (305 Hz) than when evaluating the time domain estimator, modeling a typical residual CFO that would remain after time domain CFO correction. The time domain correction system is disabled, guaranteeing only the small offset propagates

into the receiver's frequency domain pipeline. The resulting distribution of estimates is shown in Figure 3.26. Just as with the time domain estimator, the distribution of estimates here fits a normal curve centered at the actual offset. But notice the standard deviation of just 5.4 Hz, far lower than with the time domain estimator's standard deviation of 209 Hz under comparable test conditions.

The estimator performance over a range of SNRs and packet lengths are shown in Figures 3.27 and 3.28. In these plots the estimator performance is plotted as  $2\sigma$  for the distribution of estimates at each point; in every experiment, the mean estimate matched the actual residual CFO. In both plots the Y-axes are frequency estimation error in Hz. In Figure 3.27, the X-axis is analogous to SNR, parameterized here as the attenuation (in dB) applied at the channel emulator output. Figure 3.28 plots the same data but with an X-axis of packet duration, measured in OFDM symbols. This plot also includes traces indicating the maximum tolerance for CFO estimation error at each duration for packets modulated with QPSK and 16-QAM.

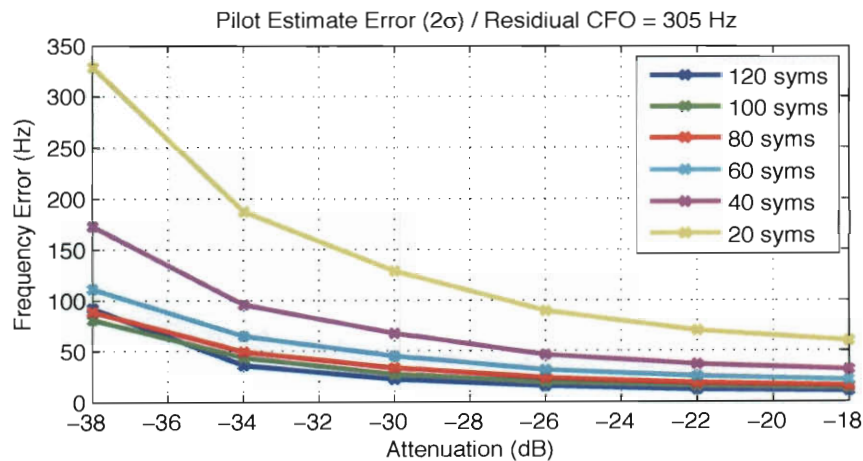


Figure 3.27 : Experimental performance of the frequency domain residual CFO estimator vs. SNR for multiple packet durations.

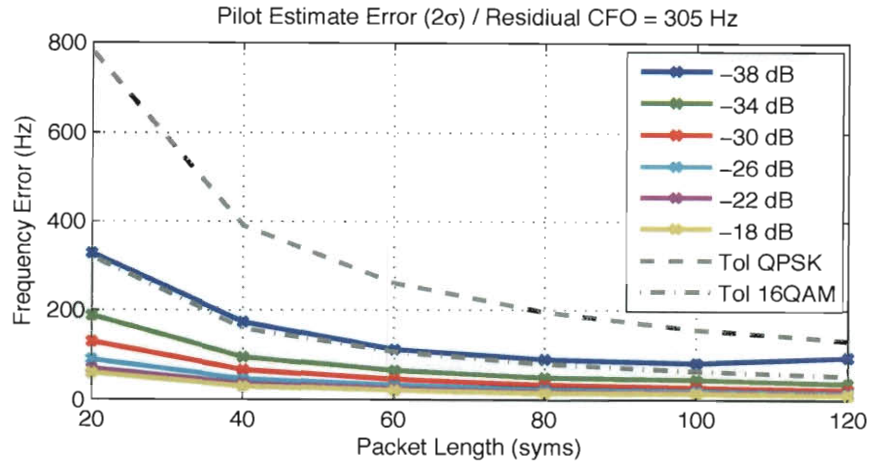


Figure 3.28 : Experimental performance of the frequency domain residual CFO estimator vs. packet duration for multiple SNRs.

This estimator performance meets our requirements, as judged in two ways. First, for all cases except very low SNR, the estimation error falls well within the range established earlier where source/relay CFO have minimal impact on performance (see Figure 3.22). Second, remember that in a cooperative transmission, this estimator will operate in the first slot while its estimate will be used in the second. The range of estimation errors we observe is comparable to the expected frequency drift between time slots (see Figure 3.19). Even if this estimator were perfect, inter-slot frequency changes will have the same impact as estimation error.

Our design reduces the effects of inter-transmitter CFO in a cooperative link to the same order as those caused by unavoidable inter-packet oscillator drift. This is a pleasing balance. Our design requires a small increase in complexity of the relay implementation but requires no extra overhead. This compares favorably with the alternatives discussed in Section 3.7.2, which required substantial complexity increases or huge increases in signaling overhead to mitigate dual CFOs in the receiver.

A thorough performance evaluation of a decode and forward relay employing our

CFO pre-correcting scheme is presented in Section 6.

### **3.8 Conclusions**

As discussed in Section 1, mitigating CFO is one of the major challenges in building a complete cooperative transceiver. We have presented the design and evaluation of two CFO correction systems. The first corrects frequency offsets in the time domain at the receiver and is used for both non-cooperative and cooperative links. The second operates at the relay, allowing simultaneous transmissions from cooperating nodes to operate with a nearly identical carrier frequency. This technique allows the destination to use its existing CFO compensation systems with no modification for cooperative links. We have evaluated both systems in hardware, demonstrating performance which will facilitate cooperative communications in a wide range of scenarios.

## Chapter 4

# Physical Layer Transceiver Design

A significant fraction of our overall effort focused on the FPGA implementation of our OFDM transceiver. Our goal of a fully self-contained cooperative node requires the transceiver implement complete real-time signal processing, synchronization and control systems in the fabric of the WARP hardware's FPGA.

We selected Xilinx System Generator [36] as the primary FPGA design tool for our OFDM transceiver. System Generator enables graphical FPGA design entry using a special blockset for Mathworks Simulink. Each block in the System Generator library integrates a simulation model and an FPGA implementation. The simulation models allow Simulink to execute a block diagram built with System Generator blocks exactly as if it had been built using standard Simulink blocks. Once the designer is satisfied with the simulation results, System Generator can produce an FPGA implementation of the full model. This flow guarantees that the simulated and HDL designs will operate identically, down to individual bits and clock cycles.

Although it enables graphical design entry, System Generator is still an FPGA design tool which allows hardware architectures to be designed at the same level of detail as with hand-coded HDL. Many blocks in the System Generator blockset, such as flip-flops, multiplexers and bitwise logical operations, map directly to FPGA primitives. Other blocks abstract common combinations of FPGA elements, such as adders and ROMs, which are built using FPGA resources like lookup tables and block RAMs. Only a few blocks provide higher levels of abstraction. We use two such



blocks: an FFT (implementing forward and reverse transforms) and a DDS (direct digital synthesizer, for generating sinusoids). Aside from these two blocks, the entire transceiver is implemented at the level of individual logical, arithmetic and memory operations.

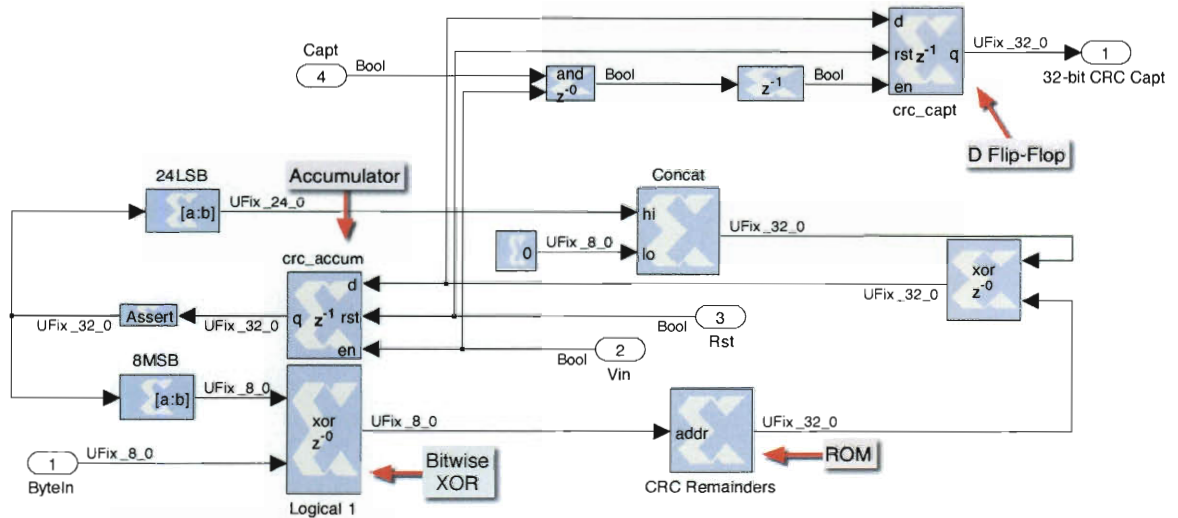


Figure 4.1 : Example System Generator block diagram, showing the CRC-32 calculation subsystem from our OFDM transmitter.

Figure 4.1 shows an example of a simple System Generator design. This block diagram is the CRC-32 checksum calculation subsystem in our OFDM transmitter. Each rectangular block is a System Generator primitive. A few blocks are labeled (an accumulator, flip-flop, ROM, etc.) for clarity. The thin lines are wires which connect ports of individual blocks. The bit widths of each wire are specified in the configuration of each block's outputs and are denoted by the small labels attached to each wire. The ellipse-shaped terminals indicate ports which connect to other subsystems higher in the design hierarchy. This is a snapshot of just one of the hundreds of subsystems in the overall transceiver design, and demonstrates the level at which every subsystem is designed.

## 4.1 Key Subsystems

A block diagram of the overall transceiver architecture is shown in Figure 4.2. The basic flow is straightforward. The transmitter starts with data from a packet buffer and ends with samples output to the Radio Board DACs. The receiver does the reverse, starting with samples from the radio ADCs and finishing with received data written to a packet buffer.

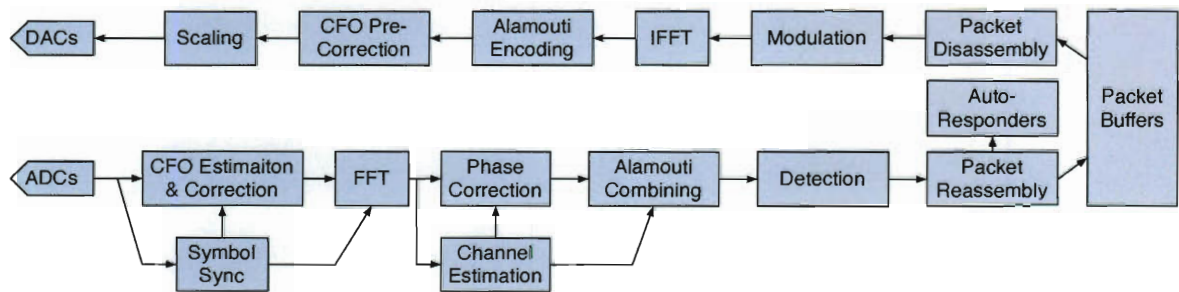


Figure 4.2 : Block diagram of the OFDM transmit and receive signal processing pipelines.

Many of the blocks in between the radio and packet buffers are common to any OFDM implementation. These blocks include digital modulation, the IFFT and output filtering in the transmitter and input filtering, the FFT and channel estimation in the receiver. These operations are standard parts of any OFDM design and lend themselves to straightforward implementation in an FPGA.

Some other blocks, however, pose unique implementation challenges. For example, the carrier frequency offset estimation and correction systems described in Chapter 3 required careful design to accommodate both actual CFO (frequency differences between nodes) and transients in the instantaneous carrier frequency of a node itself.

The sections below discuss additional blocks with particular implementation challenges. Some of these challenges surface only in a cooperative system (the design of

the symbol timing correlator in Section 4.1.6, for example). Others are more general, requiring careful implementation for both cooperative and non-cooperative links (the efficient architecture for Alamouti encoding in Section 4.1.1, for example).

#### 4.1.1 Alamouti Encoding

The Alamouti space-time block code (STBC) specifies an encoding process which translates a sequence of modulated symbols into two spatial streams of encoded symbols [18]. Every data symbol is included in both spatial streams. The encoding process is shown in Table 4.1 for data symbols  $x_n$  and symbol periods  $t_n$  ( $x^*$  is complex conjugation).

	$t_0$	$t_1$	$t_2$	$t_3$	...
Stream A	$x_0$	$-x_1^*$	$x_2$	$-x_3^*$	...
Stream B	$x_1$	$x_0^*$	$x_3$	$x_2^*$	...

Table 4.1 : Alamouti STBC encoding

The code operates across pairs of symbols and symbol periods. In the first period, unmodified data symbols are fed to each stream. The same symbols are used in the second period but are swapped across streams and either conjugated or negated and conjugated. This process repeats for every pair of data symbols to be transmitted.

In an Alamouti OFDM transmitter the STBC encoding process described above must be applied in the frequency domain (before the IFFT) across full OFDM symbols (i.e. the  $x_n$  and  $t_n$  above refer to OFDM symbols and symbol periods, respectively). This requirement implies that every data symbol must be fed through an IFFT twice: once unmodified (first symbol period), once conjugated/negated-conjugated (second symbol period), as shown in Table 4.1.

This presents an implementation challenge. Our OFDM transmitter is designed as a single long pipeline, operating continuously from when a transmission is initiated until the last sample is output to the DACs. Specifically, payload bytes are unloaded from the packet buffer individually, decomposed into groups of bits and translated into modulated symbols. This process operates just-in-time, generating symbols and feeding them immediately into the IFFT. This architecture minimizes latency by avoiding buffering wherever possible.

Extending this architecture to support Alamouti encoding in the frequency domain would require either buffering modulated symbols or adding support to “rewind” the control logic to re-modulate the same payload bytes. Neither approach is appealing; the former would consume substantial memory resources, the latter would significantly complicate the design of the modulator.

We address this complication by moving the Alamouti encoding to the time domain. This is possible thanks to a useful Fourier property. If  $\mathcal{F}^{-1}(X[f]) = x[n]$  is the inverse DFT of the frequency domain signal  $X[f]$ , then  $\mathcal{F}^{-1}(X^*[f]) = x^*[-n]$  (where  $*$  is complex conjugation). In other words, the IFFT of the conjugate is equal to the conjugate of the IFFT indexed in reverse.

The output of the IFFT is buffered in order to append a cyclic prefix to every OFDM symbol before transmission. Our design buffers pairs of OFDM symbols  $x_n$  and  $x_{n+1}$ . In the first symbol period, the symbols are read from the buffers in natural order and fed to the rest of the transmit pipeline. In the second period, the symbols are read in reverse order (realizing  $x[-n]$ ). As the samples are unloaded either the real or imaginary parts are alternately negated, realizing  $-x^*$  and  $x^*$ , respectively. The combination of reverse indexing and output negation completes the Alamouti encoding process, generating exactly the same samples as if the process had been

implemented in the frequency domain.

Figure 4.3 depicts the addressing scheme described above as implemented in our transmitter. The IFFT outputs for two OFDM symbols are written to the two buffers (A and B). Each symbol is unloaded twice, once per stream, with cyclic prefixes prepended.

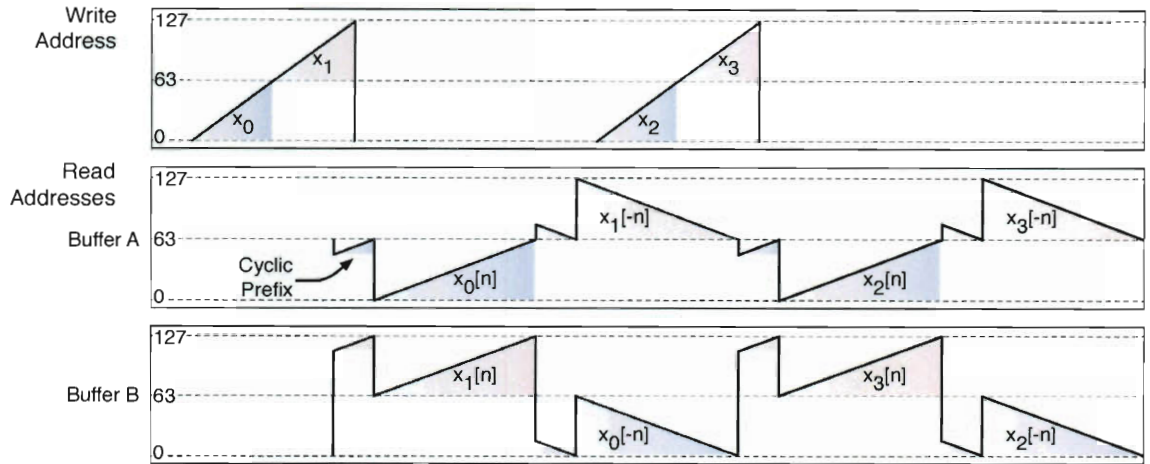


Figure 4.3 : Addressing of OFDM transmitter time domain sample buffers implementing frequency domain conjugation for Alamouti encoding.

#### 4.1.2 Packet Buffers

At the core of our OFDM transceiver are separate signal processing pipelines for the transmitter and receiver. These pipelines are designed to handle single packets. The transmitter accepts a steady stream of bytes as an input, translating each into a series of modulated symbols. At the end of its pipeline the receiver does the reverse, assembling groups of demodulated symbols into a sequence of bytes, which are then output from the core.

This view of the PHY design (stream of bytes in, stream of bytes out) is a good way to design the signal processing pipelines, but it's clear some kind of buffer is

necessary to have these byte streams be useful elsewhere in a system (e.g. to a higher network layer). The obvious solution is a pair of packet buffers, one each for the transmitter and receiver, which are mapped into the address space of the processor executing the medium access control (MAC) protocol.

We used this dual-buffer architecture in early generations of the OFDM transceiver design. It worked fine for debugging the PHY, but an issue arose when we began to implement MAC designs. In many MAC protocols, a node may have two transmit packet in-flight. In CSMA, for example, a node in timeout (having transmitted a DATA packet but received no ACK) could receive a DATA packet for which it must transmit an ACK. If there is a single transmit packet buffer, the original DATA packet must be copied elsewhere (so the ACK can be created and sent), then copied back for eventual re-transmission. All this packet shuffling takes time, of which there is little to spare in meeting MAC timing constraints.

We considered the alternative of ping-pong buffering, which would use four buffers (two each for transmit and receive). This would eliminate the contention between two transmit packets. However, only a subset of MAC protocols would benefit. Specifically, any protocol with more than two packets in flight, or one which re-transmits received packets, would still need to shuffle packets among limited buffers.

We designed a packet buffering system which alleviates these issues. Our architecture, illustrated in Figure 4.4, is based on the observation that the PHY's transmit and receive pipelines only care about byte addresses relative to the start of a packet. This design provides an array of 32 buffers, each 2 kB in size, all of which can be accessed by both the transmit and receive pipelines. Each pipeline still provides just byte addresses; these addresses are concatenated to a buffer index to construct an actual memory address for the 64 kB RAM. There are separate buffer indices for

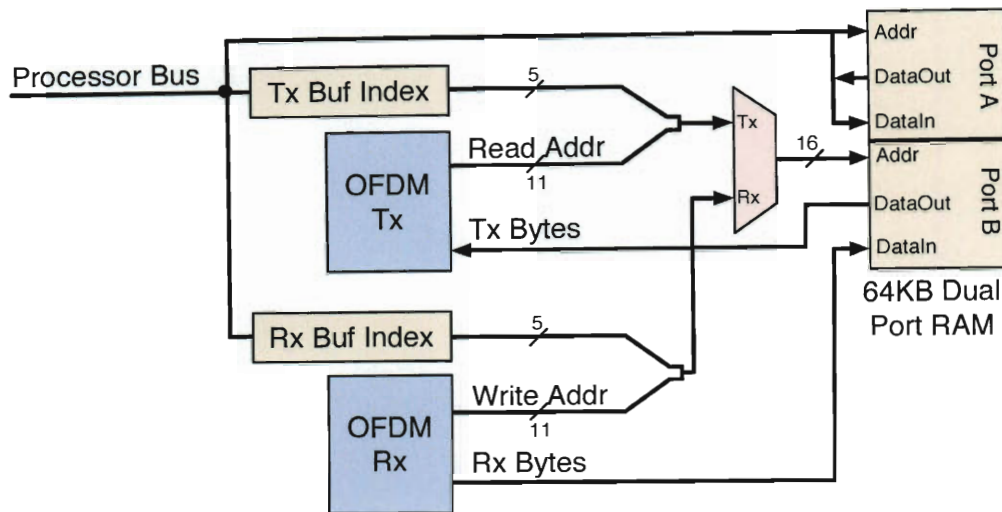


Figure 4.4 : Schematic of the OFDM transceiver packet buffer subsystem.

transmit and receive, both controlled via memory-mapped registers. The MAC software can update these registers per packet, selecting the active transmit and receive buffers as mandated by the MAC protocol

The abstraction provided by this design has proven very useful. Recall the example of a CSMA node having both a timed-out DATA and an ACK packet to transmit. In our design, each packet is stored in a dedicated buffer, never having to be re-located. The MAC selects the packet to transmit with a single memory access (updating the transmit buffer index register). Consider a multi-hop MAC, which re-transmits received payloads. This can be realized in our design without having to ever copy the packet payload. After receiving the packet, the MAC simply swaps the Tx/Rx buffer indices, then initiates the PHY transmission. The latency reduction here proves critical in a cooperative system, where a decode-and-forward relay implements exactly this sequence of operations. In extending the transceiver to support cooperative modes we reduce this latency further by automating the buffer index switching in



hardware via an auto-response subsystem (see Section 4.2).

A final component of this design is mapping the whole 64 kB memory block into the address space of the MAC processor. This allows MAC code to access any packet (header and payload) and communicate packets via Ethernet (also attached to the MAC processor's memory bus). The block memory primitives in the FPGA provide two independent ports, both with read/write access to the full memory. This suits our architecture perfectly; one port is attached to the processor bus, the other to glue logic in the PHY which multiplexes between the address generators in the transmit and receive pipelines.

#### 4.1.3 Frame Format

Our frame format is loosely based on that of IEEE 802.11a. The basic format is illustrated in Figure 4.5 and is discussed below.

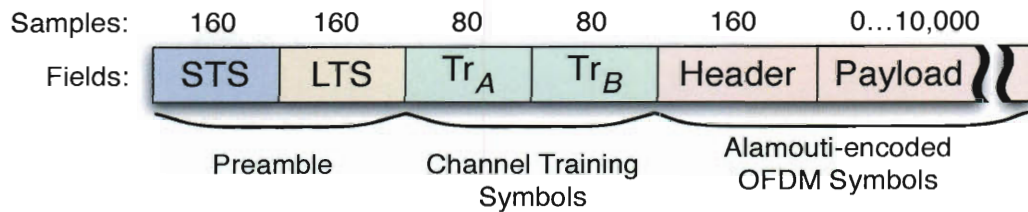


Figure 4.5 : Field descriptions and durations for OFDM frames.

**Preamble:** The preamble is a hard-coded 320-sample sequence pre-pended to every packet transmission. The preamble consists of two 160-sample sections. The first is a repetition of 10 16-sample sequences, called short training symbols (STS). The STS facilitate RSSI-based energy detection, AGC convergence and DC offset correction. The second half is 2.5 repetitions of a 64-sample sequence, called the long training symbol (LTS). The LTS are used for carrier frequency offset estimation and symbol



timing estimation.

For  $2 \times 1$  Alamouti and cooperative transmissions, both spatial streams transmit a full preamble. The antenna or node sending stream B cyclicly shifts the STS and LTS sections by three samples. This shift helps avoid unintentional destructive interference of the preamble waveforms at the receiver. This design for overlapping preamble transmissions has consequences for the packet timing correlator; this is discussed in Section 4.1.6.

**Channel Training:** Following the preamble are the channel training symbols. These are full OFDM symbols, filled with a hard-coded sequence of BPSK modulated symbols. The receiver uses the training symbols to calculate a channel estimate for each subcarrier.

In Alamouti mode every packet has two training symbols, one per spatial stream. Each stream transmits one training symbol and transmits nothing during the other training symbol period (i.e. we orthogonalize training in time). The transceiver design supports an arbitrary number of training symbols per packet. The receiver calculates the average of all symbols transmitted for a given stream. The number of symbols is configured at run time in software, but every node in a network must use the same configuration.

**Header:** The first data-bearing symbols convey the packet header. These symbols are filled with data modulated with BPSK or QPSK. Every node in the network must agree a-priori (i.e. at compile time) on the header length and modulation rate. The receiver uses two fields in the header during reception: payload length and payload modulation rate. The rest of the header is available for use by the MAC. The final

16 bits of the header are a checksum which allows the receiver to confirm an error-free header before using the modulation and length fields to process the payload that follows. The checksum is calculated in hardware by the transmitter and inserted automatically.

**Payload:** The rest of the packet is dedicated to payload symbols. The duration of this section depends on the payload length and modulation rate, both configurable per-packet by the MAC. The transceiver design supports arbitrarily long packets; in practice, we are usually limited to 1500 bytes (an Ethernet MTU).

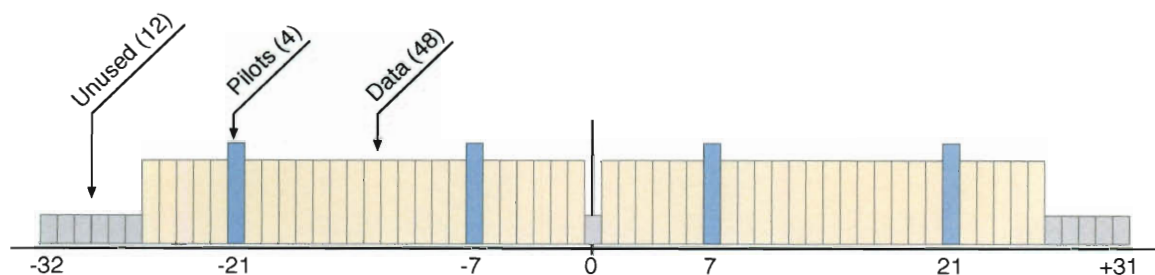


Figure 4.6 : Mapping of data symbols and pilot tones across subcarriers.

Our OFDM design divides the 10 MHz bandwidth into 64 subcarriers and supports any mapping of modulation rates to subcarriers, requiring only that the DC subcarrier be empty and that four subcarriers be allocated for pilot tones. We generally use the same subcarrier mapping as IEEE 802.11 a/g. This mapping dedicates four subcarriers for use as pilot tones to provide the receiver **phase error estimates** with every OFDM symbol. It also leaves 12 subcarriers empty, composed of the DC and 11 highest frequency subcarriers.

#### 4.1.4 Energy Detector

In a random access network each node must always be ready to receive a packet from any other node. When node is idle (not transmitting or processing a reception), it must be monitoring its RF receiver in search of new receptions. These receptions may come from nearby nodes, resulting in reception of a very strong RF signal (up to -15 dBm). They may also come from distant nodes, delivering only a weak signal (below -75 dBm, for example).

In the search for signals over this wide range of received power levels, we use a feature of the MAX2829 RF transceiver called received signal strength indicator (RSSI).

The MAX2829 receive path implements an RF power detector with an analog RSSI output. The analog signal is proportional to the log of the instantaneous received power in a  $\approx 6$  MHz band around the radio's current center frequency. This proportionality is maintained over the full range of possible receive powers ( $\approx [-85, -15]$  dBm). The RSSI signal is fed to a dedicated ADC on the WARP Radio board, which provides a 10-bit RSSI sample every 100 ns for use by the physical layer receiver.

Our OFDM receiver uses this digital RSSI to calculate a 16-sample average for use in energy detection. This subsystem declares an energy detection event when the average RSSI exceeds a threshold for a minimum duration. Both the threshold and duration are programmable at runtime. The minimum duration requirement is designed to reject short-lived energy events, like interference from frequency hopping devices. The average calculation and threshold checking are executed in hardware with every new RSSI sample. An energy detection event advances the OFDM receiver's state machine to begin searching for the structure of an actual packet, as described below.

### 4.1.5 Receiver State Machine

The OFDM receiver implements a state machine which updates with every clock cycle. The receiver progresses through the states with every packet reception. The state machine is illustrated in Figure 4.7 and described below.

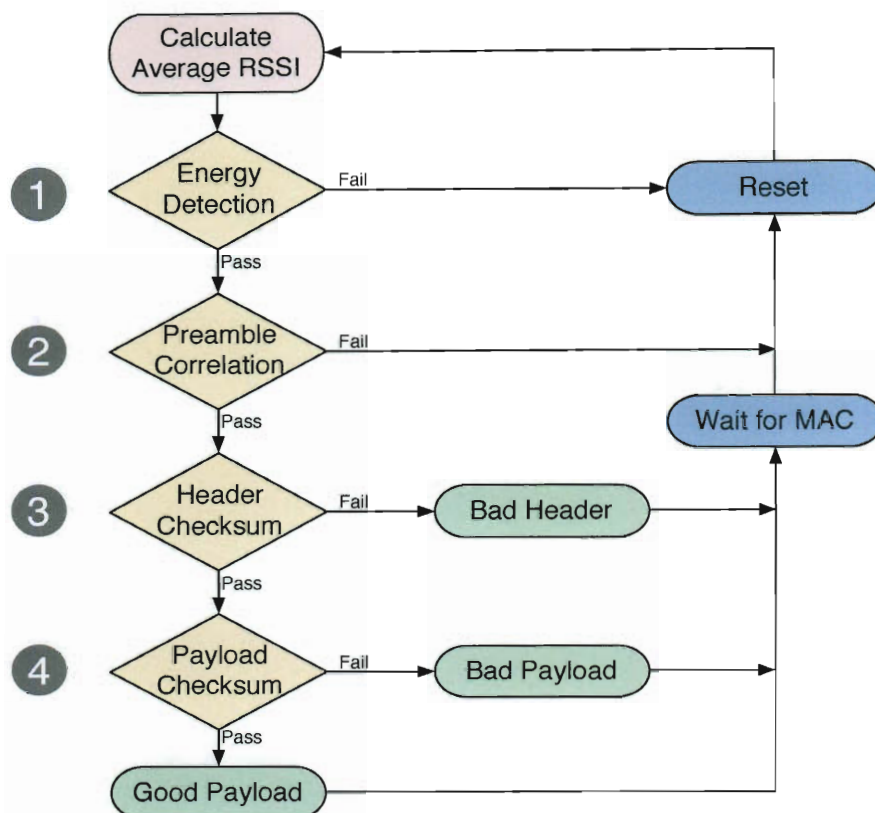


Figure 4.7 : OFDM receiver flow chart

1. Every clock cycle, the energy detector updates its running average of RSSI samples and compares the average against the thresholds for minimum energy duration and level.
2. After an energy detection event occurs, the OFDM receiver begins monitoring the output of a cross-correlator (discussed in Section 4.1.6), searching for the

long training symbols in the latter half of each packet's preamble. If the correlation exceeds a programmable threshold within a fixed window (defined by the known duration of the preamble), the receiver continues processing the incoming packet. Otherwise the receiver resets and once again begins monitoring the RSSI.

3. If the correlator indicates a valid preamble, the OFDM receiver begins taking FFTs of the incoming sample stream, starting with a sample offset indicated by the timing of the correlation event. The FFT outputs are fed into the channel estimator, equalizer and detector, which attempt to decode the packet's header using the header's 16-bit checksum to detect errors. If an error is detected, the receiver raises a **Bad Header** event and halts until the event is cleared by the MAC. Otherwise a **Good Header** event is raised and the PHY proceeds processing the packet payload (if the packet has one). The modulation rate and length of the payload are defined per-packet by fields in the header. For header-only packets (zero payload length), the PHY halts with either event until cleared by the MAC.
4. For packets with payloads, the receiver continues processing until the final payload byte is detected. Then the receiver uses the packet's 32-bit checksum to detect errors. If no errors are detected, the PHY raises a **Good Payload** event. Otherwise it raises a **Bad Payload** event. The receiver remains halted until the MAC handles either event.

#### 4.1.6 Packet Timing Correlator

The RSSI-based packet detector provides a very coarse estimate of packet timing. Depending on the power of a received packet, the energy detector can assert anywhere in a 50 sample window. This uncertainty must be reduced before samples are fed into the receiver's FFT. This process is sometimes called block boundary detection or symbol timing estimation.

The goal is to identify the last sample of the preamble in order to establish timing for the OFDM symbols which will input to the FFT. We accomplish this performing cross-correlation against the long training symbols (LTS) in the packet preamble. We use the same LTS as in the IEEE 802.11a standard. This sequence is designed to have very good auto-correlation properties. The LTS is defined as 64 complex samples; the magnitude of the sequence's auto-correlation is shown in Figure 4.8. Note that the prominent spike is exactly one sample wide.

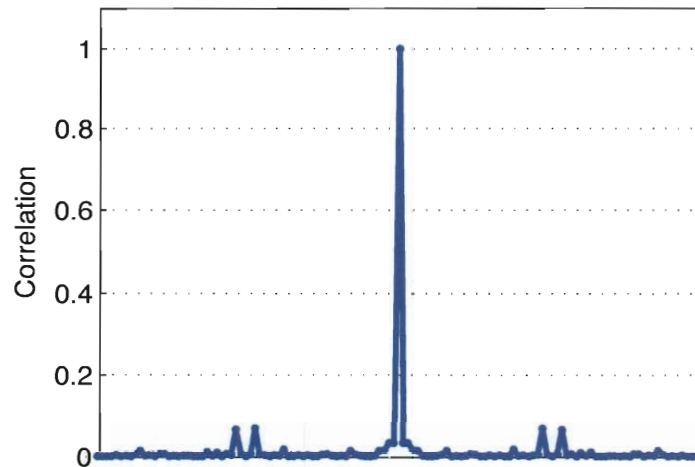


Figure 4.8 : Auto-correlation of the preamble's long training symbol (LTS).

Our OFDM receiver computes a 64-point cross-correlation in every sample period,

searching for peaks corresponding to the arrival of a valid preamble. As both arguments to this correlation (input samples and local copy of an LTS) are complex valued, this is a resource-intensive computation. If calculated at full precision, it would require hundreds of multiplications and additions per clock cycle. These requirements exceed the resources we can dedicate to the correlator. Instead, we use quantized versions of the input samples and stored LTS to reduce the resource requirements.

The architecture for this quantized correlator was originally created by Dr. Chris Dick at Xilinx and was integrated into our OFDM transceiver prior to our cooperative extensions. The original design used 1-bit quantization for both the input samples and stored LTS, effectively comparing the signs of the two 64-length complex vectors.<sup>1</sup> This architecture replaces each multiplication with a much simpler addition/subtraction operation, significantly reducing the resource requirements in hardware. It is counter-intuitive that a 1-bit $\times$ 1-bit correlation would provide sufficient precision to retain the good correlation properties of the LTS. But, as illustrated in Figure 4.9, this approach works remarkably well. These figures show the correlator output when a full preamble (10 STS plus 2.5 LTS) is input for both a full-precision simulation and the 1-bit version. The single-sample spikes corresponding to the preamble's two LTS are prominent in both plots. A slightly higher "noise floor" is present in the quantized version, as expected.

This architecture provides two key benefits:

- Low complexity: replacing each multiplication with addition/subtraction enables a very efficient implementation. The full correlator (64-point complex cross correlation) occupies less than 1% of the logic in the WARP hardware's

---

<sup>1</sup>Technically two bits are required to encode  $\pm 1$  in two's complement fixed point notation; however, the correlator architecture forces input samples to either -1 or +1, even if represented as two-bit integers

V2P70 FPGA. A full precision implementation would consume far more (256 of the 320 available multipliers, for example).

- Amplitude independence: by quantizing the received samples to  $\pm 1$ , the height of the correlator's peaks are independent of the received signal amplitude. This allows use of a simple static threshold for all received powers, a much more efficient approach than adjusting the threshold on the fly.

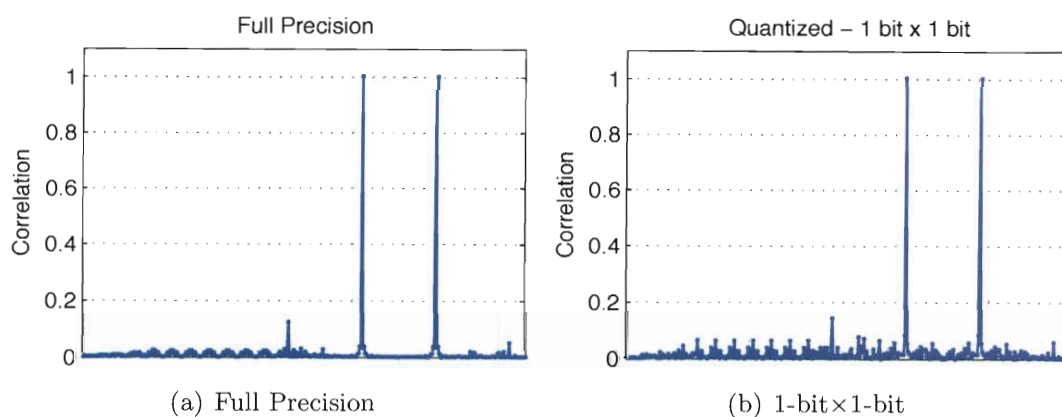


Figure 4.9 : Packet timing correlator output for full preamble, comparing full precision calculation (a) to 1-bit  $\times$  1-bit quantized version (b).

### Correlation for Cooperative Transmissions

The discussion above presents the design of the correlator used in our OFDM receiver to establish symbol timing. This correlator was designed for SISO systems and performs well in this application. However, a challenge arises when employing this correlator for symbol synchronization in a cooperative receiver.

Figure 4.10 illustrates the frame format for cooperative transmissions in our implementation. The components of each transmission are those discussed in Section 4.1.3 above. Notice that both nodes transmit full preambles simultaneously (though stream B's preamble is cyclicly shifted by 3 samples to avoid destructive combining).



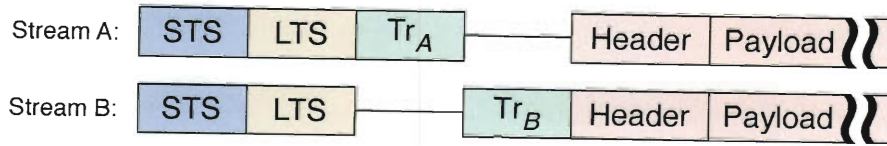


Figure 4.10 : Frame format for cooperative transmissions, where two nodes each transmit one of the streams simultaneously.

Ideally, the LTS would have perfect auto-correlation properties (zero correlation for non-zero offsets) and our correlator would operate at full precision. This would allow the best possible recovery of high spikes from the cross-correlation even when superimposed preambles (with some relative delay) are received. Of course, neither the LTS signal design nor correlator implementation are ideal. As a result, the best-case correlation values will decrease in the overlapping preamble cases.

Figure 4.11 illustrates this effect, comparing the output of a full precision correlator (a) and the 1-bit quantized version (b). The inset in (b) highlights the four correlation peaks, with neighboring peaks separated by three samples, corresponding to the cyclic shift discussed in Section 4.1.3. Note the significant degradation in the peak magnitudes, relative to Figure 4.9(b) (the figures use the same axes). This reduction constrains the selection of the correlation threshold, increasing the probability of errors during symbol timing estimation.

In our early experiments, we identified the degradation of these peaks as a dominant source of errors at high SNR. A full precision correlator would improve this, but we recognize a full precision implementation is infeasible. We address this challenge by slightly increasing the precision with which the correlator stores its copy of the LTS.

Our new design uses 3 bits per stored coefficient (verses 1 bit in the original

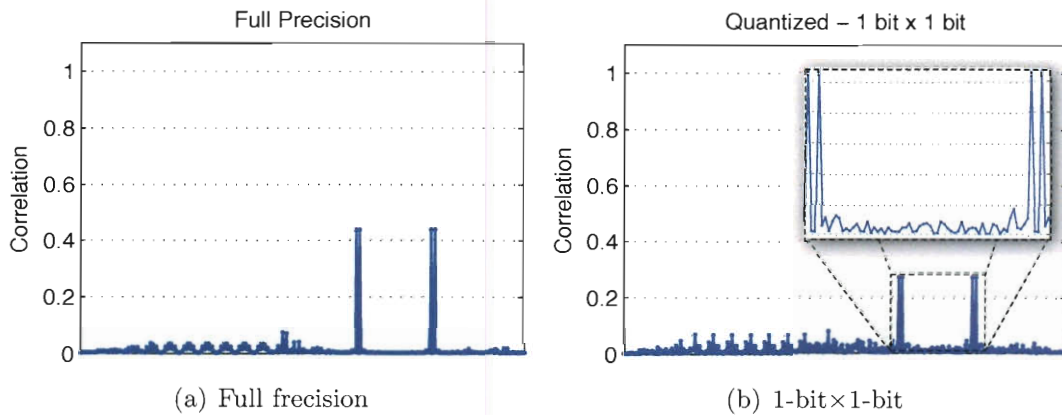


Figure 4.11 : Output of full precision (a) and 1-bit $\times$ 1-bit (b) correlators when processing overlapping preambles for 2 $\times$ 1 or cooperative transmission.

architecture). The new design still quantizes received samples to  $\pm 1$ , preserving the multiplier-free architecture and correlation values independent of received power. The results for the 1-bit $\times$ 3-bit correlator are shown in Figure 4.12, illustrating the correlator outputs for both a single transmission and two simultaneous transmissions (e.g. 2 $\times$ 1 or cooperative). The updated correlator design significantly improves the peak correlation values, nearly matching those achieved with a full-precision version. We chose 3 bits for the stored LTS coefficients to balance the correlator performance with resource usage; the updated design consumes  $\approx 3\%$  of the logic in the V2P70 FPGA.

We can further illustrate the impact of the higher precision correlator with the experimental results shown in Figure 4.13. These traces show the probability of error during packet timing estimation for both a non-cooperative and AF link using the original (1-bit $\times$ 1-bit) and updated (1-bit $\times$ 3-bit) correlators. This experiment models a cooperative topology with co-located source and relay nodes, with the destination node placed at various distances. There is a dramatic improvement in error perfor-

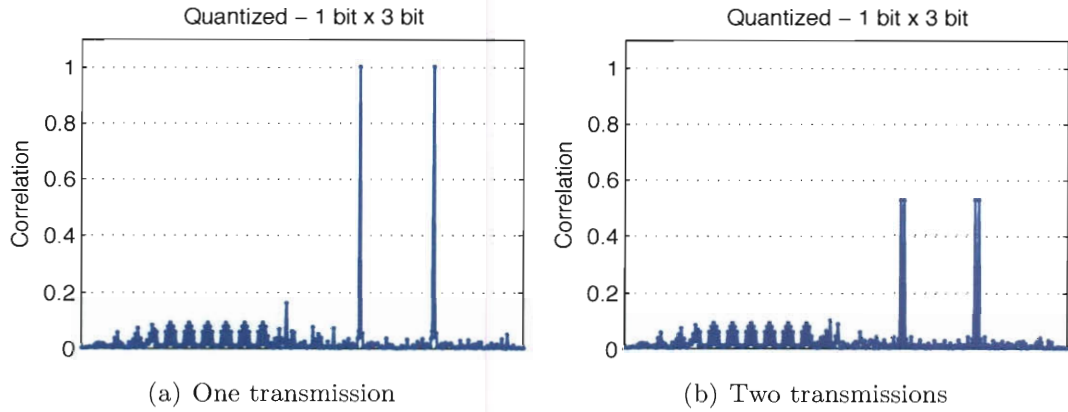


Figure 4.12 : Output of 3-bit $\times$ 1-bit correlator for a single transmission (a) and two transmissions (b).

mance using the higher-precision correlator, reducing the error rate for the highest SNR data points by nearly 100 $\times$ . These curves are produced using the experimental methodologies discussed in Section 5.

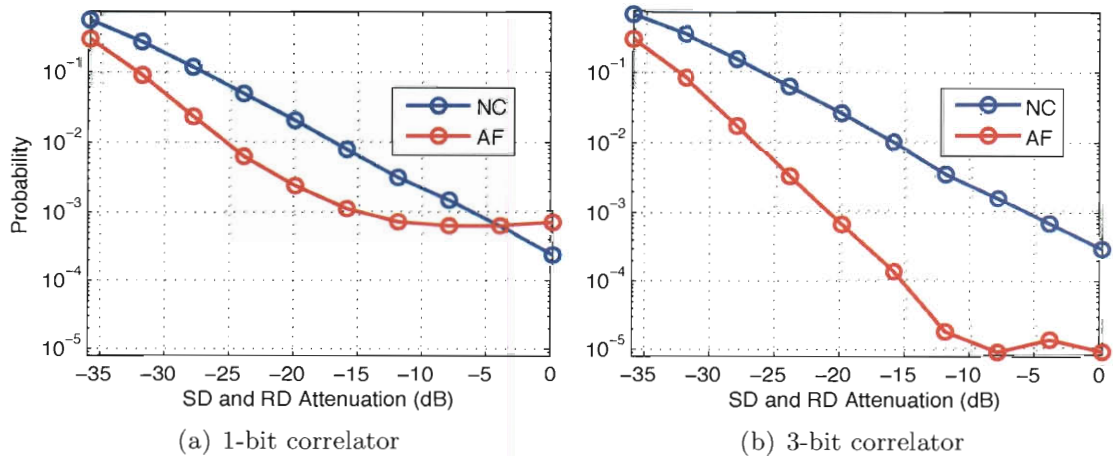


Figure 4.13 : Experimental results for probability of error during packet timing estimation for non-cooperative (NC) and amplify-and-forward (AF) links, using the original 1-bit $\times$ 1-bit (a) and new 1-bit $\times$ 3-bit (b) correlators.

#### 4.1.7 Waveform Buffer

As discussed in Chapter 2, amplify and forward (AF) relays operate by storing the raw samples corresponding to a received packet, then re-transmitting those samples without any modification. A baseband AF implementation must therefore dedicate a buffer large enough to store all the complex samples for a maximum-length packet. Our design implements this buffer using memory in the FPGA.

The memory block is constructed as a circular buffer which records I/Q samples input to the receiver at all times except when the receiver is blocked waiting for the MAC (i.e. after completing reception of a packet). The buffer captures I/Q samples at 10 MHz in the receive pipeline before any time domain processing is applied; capturing the samples before time domain CFO correction is applied is key, as explained in Section 3.7.3.

The buffer's addressing logic also has an input tied to the correlator in the symbol synchronization system discussed in Section 4.1.6. When the correlator flags the last sample of the incoming preamble, the waveform buffer records its current write address. This address is used to calculate the address of the first sample to be transmitted (when an AF transmission is triggered), by subtracting the length of the preamble.

The outputs of the I/Q buffers are routed to the final stages of the OFDM transmitter pipeline. Here the waveform is interpolated by four to match the sampling rate of the DACs. The filter outputs are then scaled to fill the dynamic range of the DACs. The AF waveform uses the same filters and scaling blocks employed for locally generated transmissions.

In our current implementation the waveform buffer can store 32k samples, more than enough to realize AF for all packet durations we use in our experiments. The

buffer consumes 52 (of 320) BRAMs in the FPGA.

## 4.2 Auto-Response System

As discussed in Chapter 1, one of the primary goals of our design is to enable cooperation in random-access networks. A core challenge imposed by this goal is the requirement that nodes be able to trigger cooperative transmissions in response to receiving data or control packets from other nodes in the network. This requirement imposes very strict synchronization tolerances on the latency between receiving a wireless packet and transmitting one in response.

This Rx→Tx turnaround is analogous to the SIFS interval in the IEEE 802.11 specifications (SIFS is the minimum inter-packet time in 802.11, occurring between DATA-ACK and RTS-CTS transmissions, for example). Before extending our OFDM transceiver to support cooperation, our primary goal was minimizing this interval, recognizing the Rx→Tx turnaround is guaranteed idle time (i.e. pure overhead). We eventually reduced it to just 24  $\mu$ s, nearly matching the very short SIFS interval in 802.11a (19  $\mu$ s).

In our original design all packet transmissions were initiated by the MAC software running in the FPGA's PowerPC core. During a reception the MAC code would continuously poll the OFDM receiver's registers waiting for either a Good Payload or Bad Payload event. Depending on the protocol implementation the MAC code could respond to these events by transmitting a packet in response (i.e. sending an ACK after receiving a good DATA packet). This software-driven approach worked well for many iterations on the OFDM design for both SISO and MIMO applications.

### 4.2.1 Rx→Tx Turnaround in a Cooperative System

A cooperative system, however, poses a new challenge. In a random access cooperative network two nodes (source and relay) may need to simultaneously initiate transmissions in response to a packet reception. Ideally the nodes' Rx→Tx turnaround intervals would be exactly the same. In practice, however, there will always be some difference. Recall from Chapter 2 that we chose OFDM as the underlying physical layer for our system in part due to its tolerance for small differences between the timing of source and relay transmissions. This tolerance is provided by the guard interval (cyclic prefix) appended to every OFDM symbol (overlapping transmissions with slight timing differences are modeled as multipath components). The guard interval in OFDM is actually intended to protect against actual multipath fading. Thus, larger source/relay timing offsets reduce the receiver's actual tolerance for multipath fading.

The 24  $\mu\text{s}$  latency stated above (for the software-driven Rx→Tx turnaround) is actually a mean value. We measured this latency in real-time and observed values  $\pm 0.8 \mu\text{s}$  around this mean. This 1.6  $\mu\text{s}$  window is a significant problem for a cooperative system, as it is equal to the full duration of the guard interval (cyclic prefix) appended to each OFDM symbol in our PHY design. With this level of variation in the timing of source and relay transmissions, the full cyclic prefix would be dedicated to synchronization tolerance, leaving none for mitigating actual multipath in the propagation environment.

To address this challenge, we conceived and implemented a new subsystem in the OFDM transceiver which manages all Tx-Rx and Rx-Tx transitions in FPGA logic, instead of in software. This subsystem has two major functions. First, it controls the pins which enable the transmit and receive paths through the WARP hardware's

radio transceiver. Second, it contains logic which can automatically initiate a packet transmission in response to a packet reception. The conditions for triggering the transmission and the contents of the response packet are programmed by the MAC software and can be changed on per-packet time scales. This subsystem essentially functions as a MAC “accelerator,” allowing protocol behaviors to be specified in C code but executed by dedicated hardware resources. The resulting turnarounds are both fast (to minimize overhead) and deterministic (to align behaviors among distributed nodes). The flexibility of this subsystem is motivated by our goal of building a transceiver which allows exploration of various protocols for triggering physical layer cooperation without having to design a custom PHY for each one.

This auto-response subsystem, depicted in Figure 4.14, is composed of three major components: header match units, actors and the header translator. Each of these is discussed in detail below.

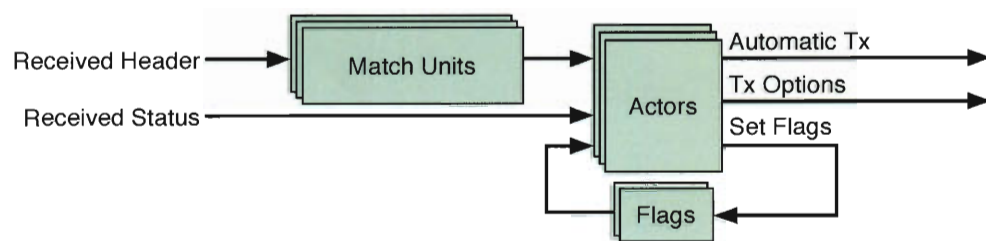


Figure 4.14 : Block diagram of OFDM receiver’s auto-responder subsystem.

#### 4.2.2 Header Match Units

The auto response match units search for user-specified patterns in received headers. There are six match units in the current transceiver, each of which is configured independently from user code.

Each match unit searches for a sequence of one, two or three bytes starting at some offset in every received header. The search values and offset are configured from user code and can be updated at runtime. Multiple match units can be used together to search for longer sequences. Common examples of match values include searching for a particular packet type (i.e. DATA) or destination address (i.e. matching the node's own address).

### 4.2.3 Actions

When packet receptions meet the programmed **conditions**, each actor implements an **action** customized by **options**. Each of these aspects are discussed below. The current transceiver implements six independent actors whose behaviors are configured from user code.

#### Actions:

- **Do nothing:** disables the specified actor; this is the default state at boot.
- **Set Flags A/B:** asserts the value of either Flag A or B. The flag values are available to other actors as a condition for their own actions. Each flag is automatically cleared by the next packet reception (unless another actor re-asserts it). The flags allow actors to transmit in response to a sequence of two receptions (i.e. the first sets the flag, the second initiates the transmission, conditioned on the flag's state).
- **Transmit a packet:** the primary role of an actor is to initiate an automatic transmission in response to a reception meeting its programmed conditions. The available conditions and options for automatic transmissions are discussed below.



**Conditions:**

- **Match Units:** Requires that some combination of match units declare a match of their programmed search pattern in the received packet header. Any combination of the six match units can be required.
- **Good Header:** Requires reception of an error-free header, based on its 16-bit checksum.
- **Good Payload:** Requires reception of an error-free header and payload, based on the packet's 32-bit checksum.
- **Bad Payload:** Requires reception of an error-free header but payload with error.
- **Flags:** Requires Flag A or B be asserted (flags are asserted by an actor in the previous packet reception).

**Transmit Options:** When an actor is configured to transmit, its behavior is controlled by a number of parameters. Each parameter is configured independently per actor and can be changed on the fly from software.

- **Transmit packet buffer:** Selects the index of the packet to transmit. Values of 1-30 select that packet buffer for transmission. A value of 31 selects the amplify-and-forward (AF) waveform buffer for transmission.
- **Use header translation:** Controls whether the header translator (described below) is used for the automatic transmission. This is ignored for AF transmissions.

- **Use CFO pre-correction:** Controls whether CFO pre-correction is applied for the automatic transmission (see Section 3.7.4). The CFO pre-correction value for the selected packet buffer is automatically accessed by the transmitter.
- **Swap spatial stream:** Controls which spatial stream (A or B) is transmitted from the active antenna. Enabling this option swaps whatever stream-antenna assignment is configured by default in software; the original assignment is restored after the automatic transmission.
- **Transmit delay:** Sets the delay in sample periods between the end of the packet reception and the beginning of the automatic transmission.

#### 4.2.4 Header Translation

The header translator, illustrated in Figure 4.15, is used to construct the header of an automatically transmitted packet using fields from the header of the received packet which triggered the transmission. This allows auto-transmitted packets to use values from received headers without having to access them from software before transmission. For every header byte of an automatically transmitted packet, the user can configure the packet buffer and byte indices of the byte that should be substituted.

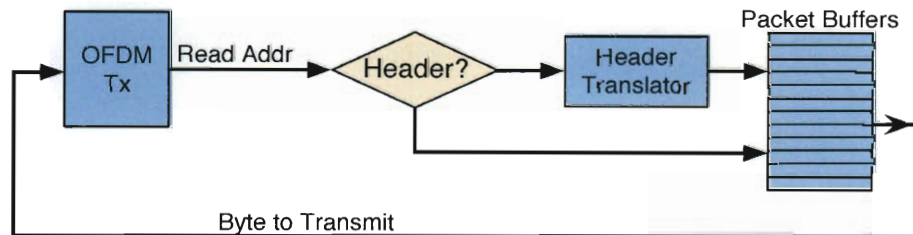


Figure 4.15 : Block diagram of OFDM transmitter's header translation subsystem.

A simple example is the transmission of an ACK in response to a received DATA

packet. At boot, the user code creates an ACK template, filled with the correct header fields for source address and packet type. The destination address and sequence number fields are left empty in the template. Then when a DATA packet is received, an auto-responder actor is configured to transmit the packet buffer containing the ACK template. During transmission, when the PHY reads the bytes corresponding to the destination address and sequence number fields, the header translation logic re-directs the memory access to the corresponding fields in the received DATA packet. This operation is totally transparent; it does not affect the ACK template, allowing it to be used for all automatic ACK transmissions without further modifications.

#### 4.2.5 Timing

The auto-response subsystem is our solution to the need for deterministic timing between packet events in a network of random-access cooperative nodes. By implementing this functionality in the FPGA fabric, the latency for an Rx→Tx turnaround is a fixed number of clock cycles every time. Two nodes which receive the same packet and are programmed to transmit a response will do so nearly simultaneously, with a maximum timing difference of just one sample period. This one sample period window is due to sampling frequency offset, as each node uses an independent oscillator for generating its sampling clock. Every node synchronizes its receiver to an incoming waveform via the correlator discussed in Section 4.1.6. Once synchronized, the receiver will operate for a fixed number of clock cycles (determined by the payload length and modulation rate). Thus, for the same received waveform, two nodes will finish reception within one sample period, the maximum phase offset between sampling clocks.

The timing of transmissions from two nodes using the auto-response system is

illustrated in Figure 4.16. This figure is a screenshot of an oscilloscope monitoring two digital signals indicating the initiation of transmissions from two WARP nodes. The scope displays a history of many transmit events and is triggered by the Relay Tx signal (hence its assertion being drawn at the same point every time). The Source Tx signal always occurs within a  $\pm 1$  sample ( $\pm 100$  ns) window around the relay transmission.

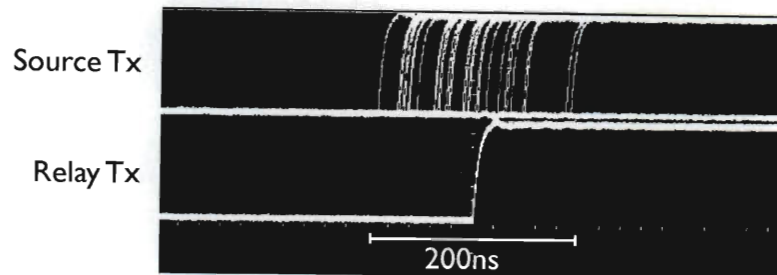


Figure 4.16 : Relative start times of two automatic transmissions from cooperating nodes, triggered by independent receptions of the same packet.

### 4.3 Designing for Characterization

In a typical depiction of a wireless networking stack, the physical layer transceiver exposes very little about its internal state to higher layers. The MAC layer, for example, is concerned primarily with the PHY's basic activity (currently transmitting or receiving) and the outcome of any reception (passing or failing checksums). This level of information is sufficient for evaluating the PHY's performance with aggregate metrics like packet error rate (PER). However, it does not provide the visibility needed to evaluate individual PHY subsystems. Our work requires this per-subsystem evaluation. As a result, our physical layer design includes a number of extra blocks dedicated for use in characterizing various aspects of the transceiver's performance.

These blocks are designed to expose internal state without affecting the operation of the PHY. A few of these designed-for-characterization subsystems are discussed below.

### 4.3.1 Carrier Frequency Offset

As discussed in Chapter 3, a significant fraction of our effort is directed at understanding the behavior of carrier frequency offsets and investigating ways to mitigate them. Frequency offsets are usually uncontrollable, bounded only to some range by the specifications of the reference oscillators in hardware. To characterize CFO estimation performance, however, we need to control carrier frequency offset as the independent variable. We achieve this using extra logic in the OFDM transmitter which can apply arbitrary frequency offsets to its transmissions, with the offset itself controlled at run-time. This is a counter-intuitive extension; CFO is something best avoided, not intentionally induced. However, by sharing clocks between nodes (establishing zero CFO by default), then applying known frequency shifts, we can precisely control the CFO and use it as a parameter in our characterization.

### 4.3.2 Packet Detection

Our OFDM receiver detects packets in two stages: energy detection (Section 4.1.4) and symbol timing estimation (Section 4.1.6). These subsystems operate continuously, seeking to detect and establish sample-level synchronization with incoming packets. However, in order to characterize the transceiver's performance, we need to extend these subsystems in two ways.

In normal operation the energy detection block is responsible for the first detection of an incoming packet and, upon detection, advances the receiver state machine. In

order to characterize the performance of the receiver independently from that of the energy detector, we add an external input which mimics the energy detector output. This signal is connected to a top-level FPGA port, itself connected to a pin on the WARP FGPA Board. Driving this pin high “tricks” the receiver into behaving exactly as if the energy detector has asserted. We connect this top-level input to a compatible pin on the transmitting node, thereby establishing “perfect” energy detection.

The energy detector provides a second pin with the opposite effect. When asserted, this signal disables the energy detector output, allowing one node to force another to ignore a given transmission. This feature is critical in our characterization of cooperative schemes, allowing us to constrain the destination to operate only in the second time slot of each transmission.

### 4.3.3 Random Payload Generation

Our OFDM transmitter design includes a random payload generator. This block produces a sequence of pseudo-random bytes (generated by a 12-bit LFSR) which are fed into the pipeline in place of the transmit buffer contents. We use this block in experiments seeking to characterize the PHY performance independent of any MAC. Generating the payloads internally adds no latency to a transmission. Creating payloads in logic also avoids having to generate payloads in software for every packet, reducing the latency between transmissions.

Randomly generating payloads creates a challenge when conducting bit-error rate (BER) tests. In order to calculate a BER, the transmitted and received packets must be compared. We use a custom application running on a PC to perform this comparison. The PC is connected to the WARPnet Ethernet network, and requires both nodes to send via Ethernet every wireless packet transmitted or received. This is

easy at the receiver, as incoming packets are always written to a packet buffer. In the transmitter, our payload generator includes logic which writes the random bytes to a packet buffer during transmission (reversing the usual role of the transmitter reading packet buffers). Following the transmission, the full packet is sent via Ethernet for BER analysis.

#### 4.3.4 Per-Packet Measurements

In the course of processing a packet our OFDM receiver calculates a number of parameters for use in the processing pipeline. Some of these parameters are also useful for debugging and characterization, but only if they can be recorded by the receiver for offline analysis. A few examples are discussed below.

**CFO estimates:** The receiver's carrier frequency offset estimators provide values used elsewhere in the transceiver. The time domain estimator feeds the correction system before the receiver's FFT. The frequency domain estimator feeds the CFO pre-correction system employed by the relay for DF transmissions. While these values are used internally, exposing the estimates to software is necessary for characterization. Our receiver records both estimates to memory-mapped registers, whose values are included in packets transmitted via WARPnet for offline analysis.

**Channel estimates:** As with CFO the channel estimator generates values used internally by the receiver pipeline. In parallel with this pipeline, our receiver design records the channel estimates generated with every packet reception in a memory-mapped buffer. This allows the estimates to be included in WARPnet packets for offline analysis. By exposing the estimates, we are able to associate particular chan-

nel characteristics with reception outcomes. It turns out this association is key to understanding some of our experimental results, as discussed in Section 6.5.2.

**Transceiver state:** The OFDM receiver is controlled by its own state machine which does not report internal state to user code until a reception is complete (at which point it reports good/bad header/payload). However, knowing the internal state of the receiver in real-time is immensely useful for debugging both MAC and PHY behaviors. Our receiver design includes a number of top-level outputs tied to internal state variables. These outputs are tied to pins on the FPGA board which can be probed with an oscilloscope in real-time.

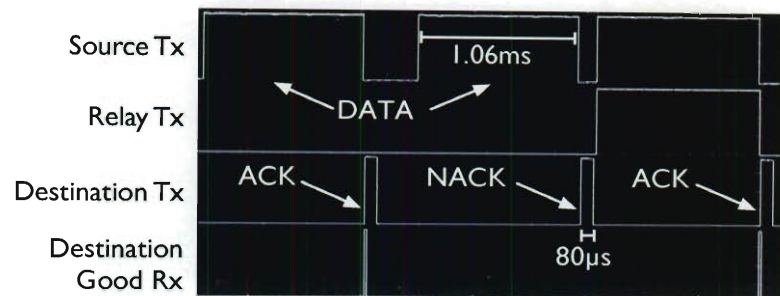


Figure 4.17 : Example observation of MAC and PHY behaviors, captured by observing state signals from multiple nodes in real-time on an oscilloscope.

This led to an unexpected observation about the utility of an oscilloscope for debugging MAC behaviors in real-time. By capturing the transmitter and receiver states at multiple nodes simultaneously (via a multi-channel scope), we can directly observe complicated state transitions and the stimuli that cause them, all in real-time. An example of this is shown in Figure 4.17, which illustrates two packet exchanges, DATA-ACK and DATA-NACK-DATA, among three nodes in a cooperative network (running the DOC protocol discussed in Section 7.1.1). Observing these signals does



not affect the PHY behavior in any way; the FPGA is able route copies of the internal state signals without altering the state machine itself. The WARP FPGA board provides 16 flexible I/O which can be assigned to monitor any internal FPGA signal.

## Chapter 5

### Experimental Methodologies and Metrics

As discussed in Chapter 1 two of our primary goals are the design of a cooperative physical layer transceiver (presented in Chapters 3 and 4) and a thorough evaluation of the transceiver under a variety of conditions (presented and discussed in Chapter 6). Connecting these goals is the need for experimental methodologies and experimental parameter selection to measure the performance of our transceiver implementation.

Addressing these requirements poses significant design challenges. For example, any cooperative experiment requires coordination of three nodes, acting as source, relay and destination. The experiment must account for every source transmission, relay reception, relay transmission and destination reception, and the conditions under which each takes place. Further complicating the experimental requirements is the need to control the wireless propagation environment. We need to test a variety of SNRs and fading conditions and must do so reliably and repeatably.

The sections below discuss our solutions to these challenges.

#### 5.1 Node Design

The OFDM transceiver is just part of our overall FPGA design. We use the design flow developed for the WARP OFDM Reference Design [37] to integrate the PHY model with the cores and code required to realize an autonomous wireless node on WARP. A block diagram of the overall design is shown in Figure 5.1. The top blocks

(group 1) represent the C code which executes in the FPGA's PowerPC core. This code includes both the high-level node behaviors (a MAC protocol, for example) and low-level code which configures the PHY and other cores (analogous to drivers). The middle blocks (group 2) represent the actual FPGA cores, including the OFDM PHY, packet buffers, Ethernet MAC and the radio controller. These cores are realized in the FPGA fabric and are connected to a common bus (PLB) mastered by the PowerPC core. The bottom blocks (group 3), the radio and Ethernet transceivers, represent peripherals on the WARP hardware itself.

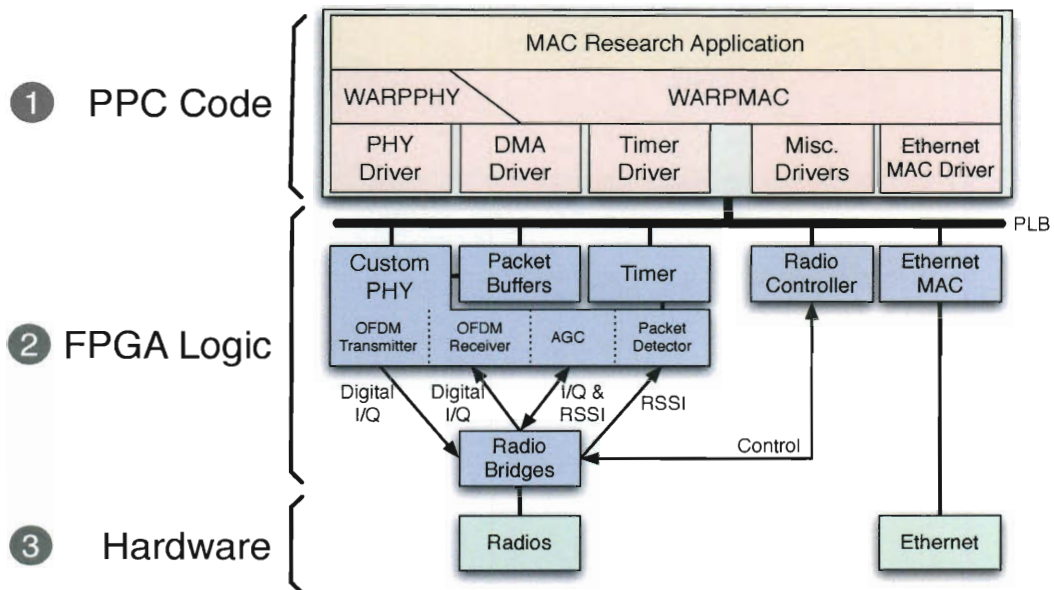


Figure 5.1 : Block diagram of the overall FPGA design for our implementation on WARP, consisting of C code for the FPGA's PowerPC (1), custom logic designs in the FPGA fabric (2) and key peripherals on the WARP hardware itself (3).

The logic designs, C code and hardware interfaces are all integrated via Xilinx Platform Studio (XPS), which serves as a front-end for both the logic synthesis flow (Xilinx ISE) and software compilation flow (built around a GCC tool chain). The output of an XPS project is a single bitstream used to configure the WARP hardware's

FPGA. We use a common bitstream for all three nodes in our experiment. The role of each node is defined by the position of a switch on the FPGA board, effectively determining the node’s MAC address at boot. This design achieves one of our key goals, in that any node can assume any role (source/relay/destination) at run-time (the PHY actually supports changing roles per-packet; the MAC is responsible for selecting the active role as needed).

We use three SISO WARP nodes in our experiments, each built from an FPGA Board (v1.2), Radio Board (v1.4) and Clock Board (v1.1). The full design is forward compatible with kits built around the newer WARP FPGA Board (v2.2) as well. In fact, a snapshot of our project, complete with cooperative extensions, was used as the basis for the release of WARP OFDM Reference Design v15. This design was posted in August 2010 with support for both versions of the WARP FPGA Board and is already in use by WARP users worldwide.

Table 5.1 lists the FPGA resource usage for both the PHY and the overall FPGA design relative to the total resources available in the FPGA (XC2VP70 on the WARP FPGA Board v1.2).

<b>FPGA Resource</b>	<b>PHY Tx</b>	<b>PHY Rx</b>	<b>Full Design</b>	<b>Available (XC2VP70)</b>
Logic Slices	4512	10360	29182	33088
Multipliers	72	83	214	328
Block RAMs	8	140	309	328
I/O	-	-	548	964

Table 5.1 : FPGA resource usage

## 5.2 Channel Emulator

One of the most challenging aspects of conducting rigorous wireless experiments is controlling the propagation environment. This holds even in an indoor laboratory setting, where both the number and arrangement of scatters are likely to change in unmeasurable ways. Interference further complicates real wireless tests. Our nodes operate in the same 2.4 GHz band as Wi-Fi devices (not to mention microwave ovens, ZigBee and Bluetooth peripherals, cordless phones, etc.). We need to vary channel properties as independent variables in our tests, which mandates these properties have known values at all times. As a result, we cannot meet this requirement with over the air tests.

For our experiments, we use an Azimuth ACE 400WB wireless channel emulator [38]. This is an instrument designed explicitly to address the shortcomings of over the air testing. The emulator interfaces to wireless devices using coaxial cables, mimicking (from the device's perspective) an actual antenna connection. The emulator accepts and generates signals at the same power levels as antennas, allowing wireless devices under test to behave exactly as if they were communicating over the air.

### 5.2.1 Connections

The Azimuth emulator is designed for MIMO applications, providing two banks of four bidirectional RF ports. This design supports tests from  $1 \times 1$  SISO (using two nodes on two ports) up to  $4 \times 4$  MIMO (using two nodes on eight ports). Each emulator output is configured independently. The configuration includes the fading channel model and the selection of inputs which contribute to each output. It also includes an output attenuation which can apply any attenuation in 0-65 dB (in 1 dB steps).

The ACE is very well designed, although our application (involving three SISO nodes in a fully connected network) deviates slightly from its intended use. Our application exposes two key constraints in the ACE design. First, every input which contributes to a given output must be subject to the same channel model. This makes sense in a MIMO configuration; neighboring antennas should see channels with different instantaneous values but with identical statistics. Further, the configurable attenuator at each emulator output affects the sum of each faded input. Thus, for a given output, the average path loss applied to each input is the same (same channel model and same attenuation). Second, the emulator can only conduct energy between ports in opposite banks; it cannot construct paths between ports in the same bank.

For our experiments, we need three paths (SR, SD, RD) with independently configured average path loss. The first constraint means we cannot use the emulator’s internal summing of faded inputs. The second constraint implies that we must connect at least one node to both banks. We overcome both issues using the connections illustrated in Figure 5.2. Note that each WARP node uses two emulator ports, one on each bank. Each node uses an external power combiner/divider (Pasternack PE2014 [39]) to connect two emulator ports to a single Radio Board. This configuration achieves three emulated channels whose properties are configured independently: SR (ports A4↔B4), SD (ports A2↔B2) and RD (ports A3↔B3).

The PE2014 is a passive device. When dividing power, each output produces  $\approx 3.3$  dB lower power than the input (0.3 dB insertion loss plus 3 dB for dividing power between outputs). When combining, each path loses  $\approx 0.3$  dB (insertion loss). Every node uses the same configuration of cables and PE2014, so these losses are uniform across all three of our emulated links. Figure 5.3 shows a photo of our hardware setup as described in Figure 5.2.

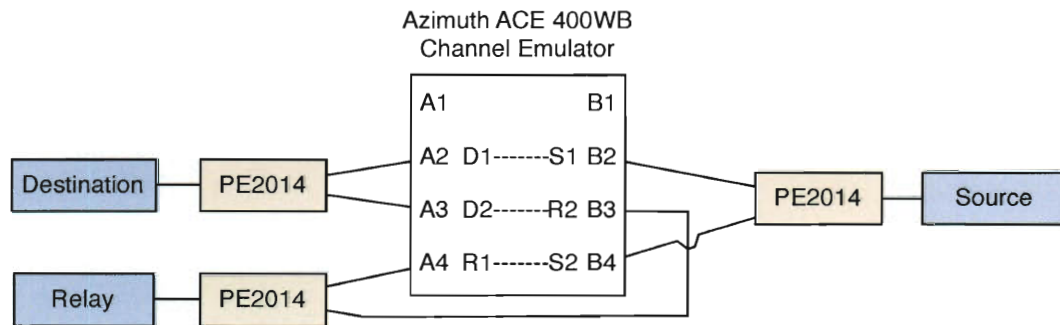


Figure 5.2 : Connections between WARP nodes and the Azimuth channel emulator for cooperative experiments.

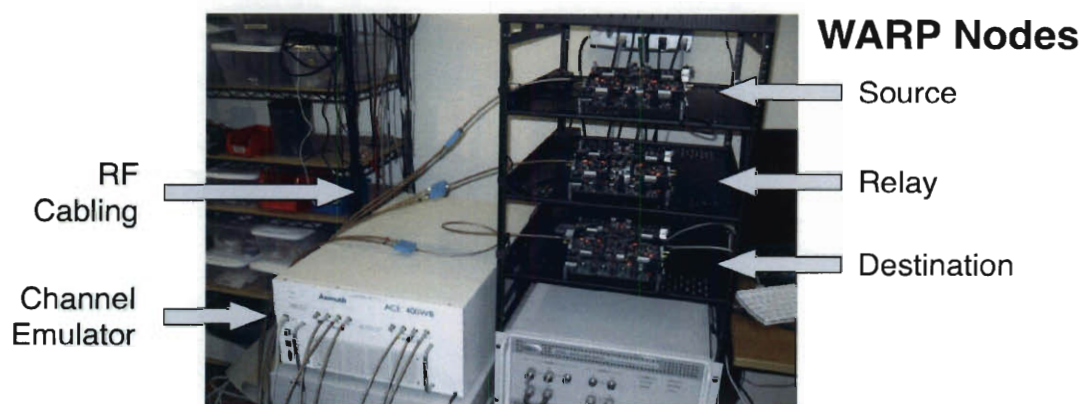


Figure 5.3 : Hardware setup for our experiments, with three WARP nodes and the Azimuth channel emulator.

### 5.2.2 Channel Models

The Azimuth emulator implements a wide array of channel models, covering a variety of propagation environments. Some model outdoor environments, like those developed by the ITU which capture pedestrian and vehicular mobility (later adopted by the WiMAX forum for conformance testing of their mobile radio standard).

We selected models from the Azimuth library originally developed by TGn Sync [40], one of the groups which merged to form the IEEE 802.11n working group. These models capture the fading properties of an indoor environment, with clusters of reflectors

and low mobility. Each model specifies multiple taps, each with a fixed delay and mean power. The power of each tap varies randomly about its mean at a rate controlled by the velocity parameter common to all taps. The delay spread of each model is determined by the distribution of delays and mean power across the model's taps. Our experiments use these models in a SISO configuration, so other MIMO-specific parameters (antenna correlation, angles-of-arrival, etc.) are not needed.

The TGn group designed six models, labeled A to F, which vary in the number and delays of taps. We use the first four of these models (A-D) for our experiments, focusing primarily on A and B. The parameters for these models are listed in Table 5.2.2, and the power delay profiles for models B-D are illustrated in Figure 5.4. Note that model A has a single tap, modeling a frequency-flat channel.

Model	RMS Delay Spread	Max Tap Delay	Num. Taps
A	0	-	1
B	15 ns	80 ns	9
C	30 ns	200 ns	14
D	50 ns	390 ns	18

Table 5.2 : TGn channel model parameters

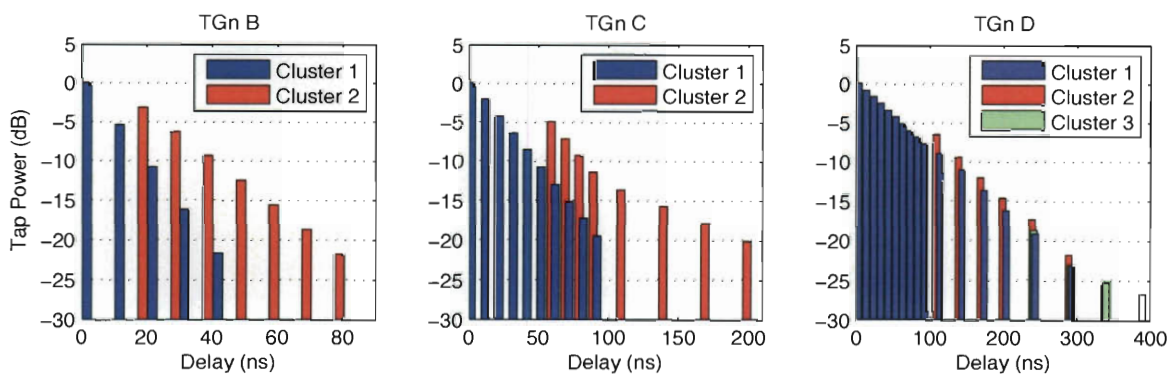


Figure 5.4 : Power delay profiles for TGn channel models B, C and D



Azimuth’s implementations of the TGn models provide two tunable parameters. The first is the emulated velocity, ranging from zero to 1.2 km/h. This parameter determines the channel coherence time by controlling the update of each tap’s value. At the maximum velocity of 1.2 km/h and carrier frequency of 2.452 GHz, the coherence time exceeds 300 msec. The maximum packet duration in our experiments is  $\approx 1$  ms (see Table 5.3). Thus, with high probability the channel coefficients will remain constant during each packet transmission, satisfying the block fading assumption inherent in our design.

### 5.3 Topologies

We emulate various topologies for our three node network by varying the average path loss along the three emulated channels. We control the average path loss using the programmable attenuators at each output of the channel emulator. By using just the attenuators to change emulated topologies we can hold constant every other parameter in the system (hardware connections, PHY configuration, channel parameters, etc.) during an experiment.

By sweeping values for the SD, SR and RD path losses, we can emulate placement of our nodes in a very large space (the attenuators are set from 0 to 65 dB in 1 dB steps). However, testing every combination of attenuations is infeasible; even for short trials, testing  $66^3$  topologies would take an unreasonably long time. Further, many topologies are not particularly interesting for studying cooperative gains. For example, a relay located far from the source will rarely transmit; a relay far from the destination can provide little gain.

Thus, we focus our experiments on the three topologies discussed below. In each topology we test both cooperative and non-cooperative transmissions to allow a fair

comparison between schemes and gauge the overall impact of cooperation.

**Co-located source/relay:** Our first topology models co-located source and relay nodes, each with equivalent path losses to the destination. As in all our experiments, the three channels employ the same fading model. The output attenuation for the source-relay channel is set to zero, realizing the minimum possible path loss ( $\approx 53$  dB) imposed by the inherent attenuation through the emulator. This loss models nodes separated by  $\approx 1.5$  meters. The source-destination and relay-destination channels have matching path losses, controlled by the output attenuators on the two emulator ports feeding the destination node. We sweep attenuations from 0 dB (matching the SR path) to 36 dB (modeling  $\approx 75$  m SD and RD separations).

This topology, with its co-located source and relay nodes, mimics an interesting usage model for cooperation. Privacy and incentivizing participation are common concerns with employing cooperation in real networks. Many researchers have explored these issues. Some propose various forms of encryption to protect payloads from untrustworthy relays [41]. Others apply techniques from game theory to construct incentive systems to encourage cooperation [42]. But these issues become much easier if a single user owns the devices participating in a cooperative transmission. For example, one person's laptop and phone (with compatible wireless interfaces) could cooperate to improve communications with a base station.

**Equidistant nodes:** Our second topology, illustrated in Figure 5.6, models source, relay and destination nodes equidistant from one another. This configuration uses a single attenuation value applied to all the outputs of the emulator. We sweep the range of attenuations from 0 to 36 dB. We chose this topology recognizing it is fre-

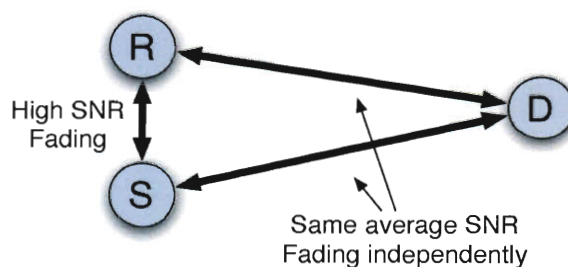


Figure 5.5 : Co-located source/relay topology, modeling a source and relay at a small, fixed distance cooperating to communicate with a distant destination.

quently used in theory-centric papers on cooperation [3], given its useful property of being parameterized by a single variable (the average path loss common to all three links).

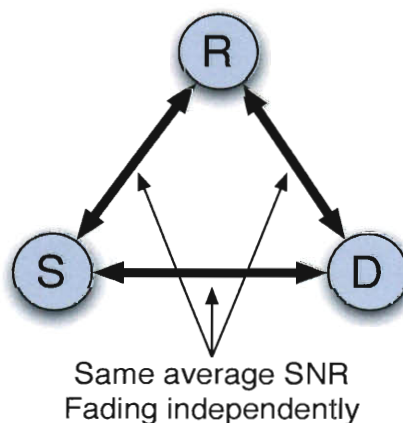


Figure 5.6 : Equidistant nodes topology, with source, relay and destination each separated by a common distance

**Linear topology:** The final topology we consider, illustrated in Figure 5.7, seeks to study the effect of relay location on the performance of a cooperative link. In this setup, the source-destination path loss is fixed, with emulator output attenuation of 18 or 23 dB (modeling distances of 10.4 m and 18 m, respectively). We vary both

the source-relay and relay-destination channels, coupling their values so the relay is always located on the line connecting the source and destination. We sweep relay locations, starting on the far side of the source, moving in between the source and destination, finally moving past the destination. We concentrate our tests on the points between the source and destination, recognizing that the relay will provide its maximum benefit somewhere in this span. It is important to note we are modeling various relay locations, not relay mobility. As in all our experiments, the average path losses along all three channels are fixed during a given trial.

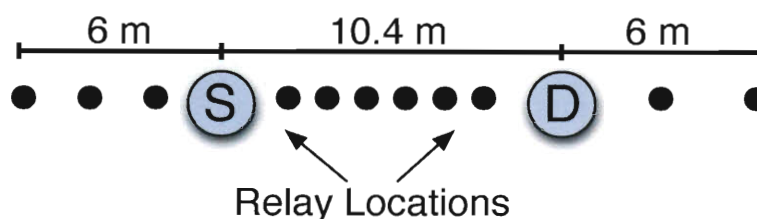


Figure 5.7 : Linear topology for experiments modeling fixed source and destination, with the relay at various points along the line connecting them.

An interesting issue arises in realizing this topology with the channel emulator. The intuitive independent variable in this topology is relay location along the SD line, achieved by setting various path losses for the SR and RD channels. However, recall from the description of the emulator in Section 5.2 above that the average path loss along a channel is configured using the programmable attenuator at each emulator output. These attenuators are set in 1 dB increments. This minimum attenuation step size of 1 dB proves problematic for realizing the linear topology.

The mapping of average path loss to distance is a straightforward calculation [43].<sup>1</sup>

<sup>1</sup>The mapping of path loss to distance requires selection of a path loss exponent. We use 2.1 (modeling indoor propagation) for all distance-path loss mappings in this work. A different exponent would change only the physical interpretation of average path losses, not our measured performance at each.

The mapping, however, is not linear. For small path losses, a 1 dB step corresponds to a small change in distance. Conversely, for large path losses a 1 dB change corresponds to a much larger displacement.

Each relay position in our linear topology defines SR and RD distances; each distance maps to a corresponding path loss. When these path losses are rounded to the nearest dB (to use the attenuators set in 1 dB steps), the effective relay location is shifted away from the SD line by an amount defined by how far each desired path loss was rounded. The result is an effective topology with the relay moving along a jagged path, instead of the desired straight line between the source and destination.

We devised a method to address this. Every Azimuth channel model has a tunable parameter which defines the average gain through the digital filters the emulator uses to apply channel responses. These gains are specified in 0.1 dB steps over a 40 dB range. Azimuth pre-calibrates these gains so that channel models with different numbers of taps can be fairly compared. The Azimuth documentation makes clear that it is important not to deviate too far from each model's calibrated gain. Too low a value induces extra quantization error, too large risks saturation. The documentation warns that the emulator does not alert the user to either condition.

Our approach uses a combination of output attenuation values and a small range of channel model gains to better emulate relay positions along the linear topology. We use 10 channel models with identical power delay profiles but with model gains spaced evenly over a 1 dB range. The right combination of model gain and output attenuation provides an effective 0.1 dB resolution on average path loss. Figure 5.8 illustrates the difference between our approach (0.1 dB steps) and using the attenuators alone (1.0 dB steps). These curves plot the difference between the ideal source-relay distance and the modeled distance, as a function of location index along the line between the

source and destination.

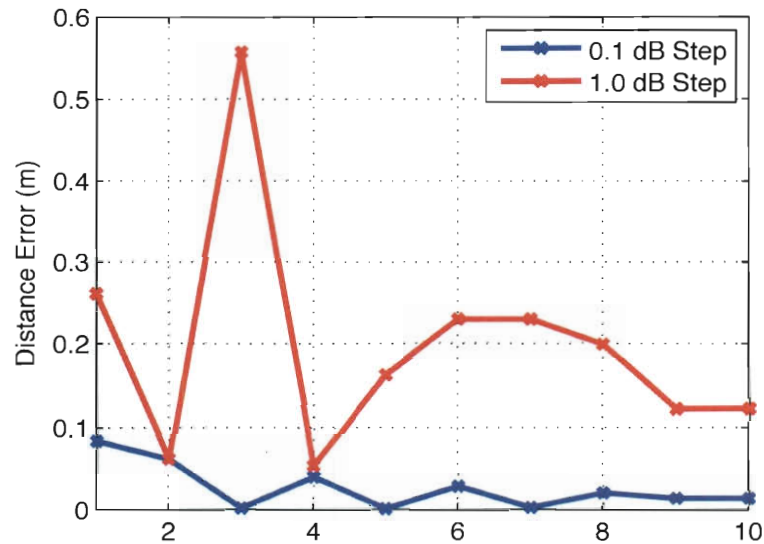


Figure 5.8 : Errors between desired and actual distances in the linear topology using only attenuators (1.0 dB step) and attenuators plus model gains (0.1 dB step).

## 5.4 Cooperative Schemes

A wide variety of cooperative schemes have been proposed in the literature. Most can be characterized by the kind of processing employed by relay nodes. At one extreme are relays which perform raw waveform capture and re-transmission. Relays in these schemes apply gain, but no other processing. At the other extreme are relays which decode the full payload before re-transmitting it, running the received waveform through a full physical layer receiver and transmitter. Some schemes operate somewhere in between, possibly not decoding the full payload but performing some other computation on the waveform (compressing [44] it, for example) and re-transmitting the result.

**Amplify and Forward:** This is a conceptually simple scheme in which a relay captures the raw waveform corresponding to a source's transmission, applies some gain, then re-transmits the waveform to the destination.

In theoretical treatments of amplify and forward [3], the gain is usually modeled as a single effective gain applied by the relay. These models apply the gain after the waveform is captured, such that the gain stage adds no noise but does amplify whatever noise is present in the received waveform.

In our implementation, the gain is actually applied in three stages. The first two are analog amplifiers in the receive path of the MAX2829 RF transceiver on the WARP Radio Board. Both are variable gain amplifiers controlled by the AGC block in our transceiver. We use the same AGC for all nodes, independent of the cooperative scheme being employed. The AGC algorithm selects the best gains for scaling the received waveform to fill the dynamic range of the ADCs. The radio amplifiers, as controlled by the AGC, effectively invert the power loss (average path loss plus fading) through the channel. It is important to note this gain is applied *before* the received waveform is digitized and recorded in the FPGA.

The final gain stage is digital, implemented as a multiplier in the FPGA which scales the captured I/Q samples before they are re-transmitted. We use a static scaling value which re-scales the digital values to fill the dynamic range of the transmit DACs. Thus, in our AF implementation the total gain applied at the relay is a function only of the instantaneous path loss between the source and relay.

In the experiments below, we evaluate two variations on amplify and forward. In the first, hereafter abbreviated **AF**, the relay re-transmits a captured waveform only if it was able to decode the full packet without header or payload errors. The second, labeled **AF-GH** below, relaxes this requirement to any packet received with a good

header, regardless of success in decoding the payload.

**Decode and Forward:** this scheme is likewise simple in concept. In this mode, the relay employs a full physical layer transceiver (same as the source and destination) and attempts to decode every transmission from the source. When it succeeds, it initiates a transmission using the received packet as the input to its own OFDM transmitter. In our implementation of decode and forward, hereafter abbreviated **DF**, the relay re-transmits only those packets it receives with zero errors in the header and payload.

**Multi-hop:** In this scheme, the relay's behavior is very similar to DF; it attempts to receive every source transmission, and re-transmits only those it can decode correctly. However, in this mode, labeled **MHOP** below, the source does not transmit in the second time slot; the destination can only receive the packet from the relay, and the relay only transmits if it receives the packet from the source. We only test this scheme in the linear topology, recognizing it provides little insight in our other topologies (where the SR and RD distances are always the same).

**Non-cooperative:** In the final scheme, labeled **NC** below, the relay is simply inactive. We use this mode to establish a baseline for comparing the performance of all schemes relative to a non-cooperative system.

## 5.5 Methodology

We seek to be as rigorous as possible in the experimental evaluation of our transceiver's performance. This requires we carefully control the conditions under which each measurement is taken. This led us to use a channel emulator, as discussed above. This



goal also requires we ensure the data we collect represent the values we are trying to measure. These requirements inspire additional techniques, discussed below.

### 5.5.1 WARPnet Framework

One of the most significant challenges in running our experiments is reliably coordinating the behaviors of three WARP nodes, the channel emulator and external support applications (like the BER calculator) over the course of experiments lasting hours, sometimes days. An additional complication is the variety of experiments we conduct, ranging from BER measurements to characterizing oscillator stability. We address these challenges using WARPnet, a testing framework designed for exactly these kinds of experiments.

The architecture for WARPnet was conceived by the members of the WARP team in early 2010 and implemented from scratch in Python by Siddharth Gupta. At its core, WARPnet is a custom Python framework which provides a level of abstraction between a high level experiment script and the low level message passing among the nodes and test instruments. The top level script (also written in Python) uses simple constructs to send messages to nodes under test. The messages can either update parameters or request results. Replies from the node are delivered back to the script to update its records of the experiment.

All node-script communication occurs via a WARPnet server. The script-server link is a standard socket connection. In the current framework, server-node messages are sent via raw Ethernet frames. This approach simplifies the design of a WARPnet-compatible node by avoiding the need for an embedded IP stack. It also emphasizes that only the WARPnet server needs to understand its connection to the nodes; new node-server connections could be designed without any modifications top level scripts.

A final key feature of WARPnet is that only the top level script and node design need to understand the custom messages used in each experiment. Specifically, adding new message types requires no code changes to the WARPnet server. This realizes one of the key tenets of the WARPnet architecture, that a researcher is responsible for their experiment and node implementation, but that the framework can facilitate all communication between the experimental script and the nodes under test.

The full WARPnet framework is open-source [45] and has already been adopted by other WARP users outside Rice to conduct their own experiments.

```
#####
### Block 1: Init ###
#####
#Load the WARPnet frameworks
from warpnet_framework.warpnet_client import *

#Enumerate the WARP nodes under test
createNode(nodes, Node(0, NODE_PCAP))
createNode(nodes, Node(1, NODE_PCAP))

#####
### Block 2: Msgs ###
#####
#Define message templates, and associate them with the nodes
cmdStructStart = CommandStruct(COMMANDID_STARTTRIAL, 0)
nodes[0].addStruct("cmdStructStart", cmdStructStart)

cmdStructStop = CommandStruct(COMMANDID_STOPTRIAL, 0)
nodes[0].addStruct("cmdStructStop", cmdStructStop)

perStruct0 = ObservePERStruct()
perStruct1 = ObservePERStruct()
nodes[0].addStruct("perStruct", perStruct0)

#Register the nodes and message templates with the WARPnet server
sendRegistrations(nodes)

#####
### Block 3: Loop ###
#####
#Define the attenuations to test
attens = range(0, 37, 4)

#Run a trial for each attenuation
for i, attn in enumerate(attens):
```

```

#Update the SD attenuation in the emulator
emulator("link_atten", "D1", "S1", attn)

#Run a 20 second trial
nodes[0].sendToNode("cmdStructStart")
time.sleep(20)
nodes[0].sendToNode("cmdStructStop")

#Request Tx/Rx packet counts from each node
nodes[0].sendToNode("perStruct")
nodes[1].sendToNode("perStruct")

#Log the results as M-code for plotting in MATLAB
log("sd_atten(%d)=%d;" % (i, attn))
log("n0_tx(%d)=%d;" % (i, perStruct0.numTx))
log("n1_rxGood(%d)=%d;" % (i, perStruct1.numRx_good))
log("n1_rxGhBp(%d)=%d;" % (i, perStruct1.numRx_GhBp))
log("n1_rxBadHdr(%d)=%d;" % (i, perStruct1.numRx_badHdr))

```

Listing 5.1: Simple WARPnet script for testing PER vs. SNR

Listing 5.1 shows an example WARPnet script. This script consists of three sections:

1. **Initialization:** Block 1 loads the WARPnet frameworks and enumerates the WARP nodes under test
2. **Messages:** Block 2 specifies the messages to be sent and received to each WARP node. Instances of two example messages are used here: `CommandStruct` (used to start and stop experimental traffic generation) and `ObservePERStruct` (used to report transmit and receive packet counts)
3. **Experiment Loop:** Block 3 defines a range of attenuations for application by the channel emulator, then conducts a separate 20 second trial for each attenuation value. At the end of each trial the Tx/Rx packet counts are requested from each node and recorded in a log file.

This WARPnet script illustrates the basic flow of all our experiments. The WARPnet scripts for our experiments are considerably more complicated, looping over inter-

node attenuations, channel models, cooperative schemes, modulation rates and packet lengths. But even our longest scripts exercise the same primitives (initialization, message definition and experiment loops) exercised in the simple example above.

Our suite of WARPnet scripts for characterizing the cooperative PHY will be available open-source in the WARP repository.

### 5.5.2 Node Behaviors Across Time Slots

All of the cooperative schemes we consider operate in two time slots. In the first time slot the source attempts to deliver its payload to the relay. In the second slot the source and relay cooperate to deliver the payload to the destination.

At many points in the topologies under test, the source and destination may be able to communicate directly. In a deployed network of cooperative nodes, both direct and cooperative communication would be employed, with a cooperation-aware MAC protocol deciding which nodes participate in each packet exchange. We discuss early results from our work on one such protocol in Section 7.1.1. In this section, however, we focus on physical layer performance when cooperation is employed, not on the protocol which decides when to cooperate. Thus, we gather results only in the second time slot of each cooperative transmission, ignoring any potential receptions at the destination in the first slot.

The obvious way to achieve this is to disable the S→D path through the emulator in the first time slot. Unfortunately, this is infeasible. The emulator requires its internal processing be halted before the emulated connectivity can be changed. Stopping and starting the emulator has a latency of  $\approx 15$  seconds. It is possible to change attenuation values without halting emulation. However, even this process takes a few seconds.

Instead, we achieve the necessary behavior with external control signals generated by the source node and connected to the relay and destination nodes via dedicated wires. Each signal feeds into the energy detection block in the OFDM receiver (described in Section 4.1.4). When asserted, this input inhibits reception at a node by blocking energy detection events from advancing the OFDM receiver's state machine. Blocking packet detection has the identical effect as driving the actual received power to zero (as disabling the emulator path would do).

At first glance, this wired connection between nodes seems to violate our tenet of “no cheating” on synchronization. It is important to recognize, however, that we use this back channel connection to block receiver processing, not to trigger it. When a node is allowed to receive it does so autonomously, establishing all necessary synchronization (energy detection, symbol timing, CFO, etc.) using just the received waveform. We use this technique to assure our cooperative performance measurements reflect only those transmissions where cooperation is possible.

Figure 5.9 shows the node behaviors and timing of the time slots, depicting the two slots of a single cooperative exchange. Each exchange starts when the source node periodically generates a new random packet payload. It adds a header with S/R/D addresses and a sequence number. It then transmits the packet twice with a short, fixed idle time between transmissions ( $T_{IT}$ ). This delay allows the waveform to finish propagating through the emulator (it has  $\approx 1 \mu s$  latency) and for the relay to finishing processing the received waveform. The timer which triggers the second transmission is implemented in the FPGA, assuring a deterministic interval.

The OFDM receiver at the destination is disabled during the first transmission via the control signal discussed above. The relay attempts to decode the transmission by the source in the first slot. Then, depending on programmed conditions, it may re-

transmit the packet simultaneous with the source's second transmission. The OFDM receiver at the relay is disabled in the second slot, assuring it does not misinterpret the second transmission in cases where it fails to receive the first.

The destination attempts to receive the second transmission whether or not the relay participates. The source imposes a short inter-packet interval ( $T_{IP}$ ) after the second transmission, allowing every node to update and report statistics via WARPnet. The source node increments internal counters per-scheme for every transmission. The relay and destination record the ending state (good/bad header/payload) for every reception. The source and destination nodes also report full packet payloads via WARPnet for BER calculations.

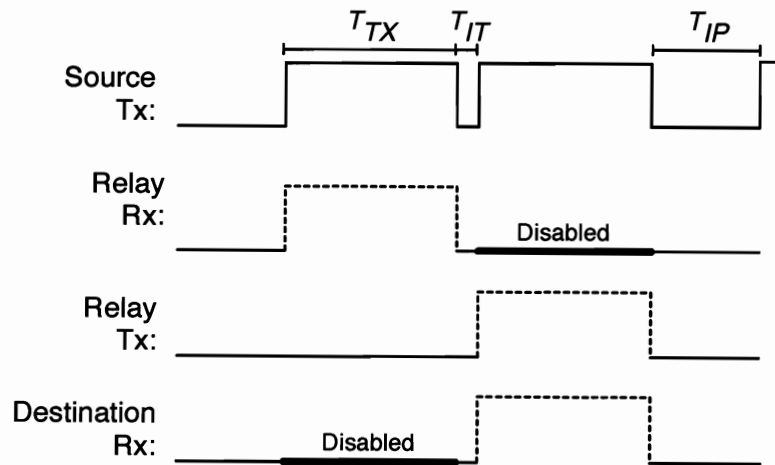


Figure 5.9 : Timing of Tx/Rx modes for cooperative experiments.

The actual transmit and idle intervals depend on the payload length and modulation rate specified at run-time. Table 5.3 summarizes the parameter combinations we used in our experiments.

Payload	data rate	$T_{TX}$	$T_{IT}$	$T_{IP}$
1412 bytes / QPSK	12 Mbps	1008 $\mu s$	30.4 $\mu s$	358 $\mu s$
1412 bytes / 16-QAM	24 Mbps	536 $\mu s$	30.4 $\mu s$	344 $\mu s$
692 bytes / QPSK	12 Mbps	528 $\mu s$	30.4 $\mu s$	358 $\mu s$
692 bytes / 16-QAM	24 Mbps	296 $\mu s$	30.4 $\mu s$	344 $\mu s$

Table 5.3 : Payload and timing parameters for cooperative experiments.

### 5.5.3 Interleaving Modes

In an ideal experimental setup, we would be able to test nodes utilizing each cooperative scheme under identical conditions. If this were possible, we could run back-to-back experiments testing each cooperative scheme in isolation. In practice, however, we cannot control every factor which may impact performance. As a result, rather than test schemes in isolate, we interleave them in time, cycling between modes with every packet transmission. With high probability this approach exposes each scheme to the same instantaneous channels by testing every scheme inside each coherence interval (assuming sufficiently long coherence times, as provided by our emulated channels). It also helps evenly distribute the impact of any uncontrolled external factors, like carrier frequency drifts, temperature changes or interference levels, which may affect results.

Interleaving schemes per-packet may seem to present significant synchronization challenges in coordinating the behavior of the source and relay, and correctly logging receptions at the destination. This approach is actually straightforward to implement using the auto-response subsystem already integrated into our transceiver. The primary goal in designing auto-responders was to reduce randomness in the timing of transmissions from cooperating nodes. However, the extensive programability of this subsystem also enables its use in defining the relay's behavior on per-packet time

scales.

As explained in Section 4.2, the conditions for triggering an automatic transmission are programmable, and include pattern matching against received headers. By using a header field to indicate the cooperation mode (analogous to a packet type), the source node can effectively program relay behaviors per packet. The relay uses an auto-responder action per cooperative mode, with each conditioned on receiving a packet of a given type. The action configurations are summarized in Table 5.4.

Table 5.4 : Relay auto-responder action configurations for cooperative tests

Scheme	Action	Conditions	Tx Options
AF	Tx AF_BUF	(GOOD_PLD & MATCH_RLYADDR & MATCH_AF)	-
AF-GH	Tx AF_BUF	(GOOD_HDR & MATCH_RLYADDR & MATCH_AFGH)	-
DF	Tx RX_BUF	(GOOD_PLD & MATCH_RLYADDR & MATCH_DF)	USE_PRECFO
NC-MHOP	Tx RX_BUF	(GOOD_PLD & MATCH_RLYADDR & MATCH_NCMHOP)	-

The abbreviations used here are defined below:

- **AF\_BUF**: ID for the AF waveform buffer
- **RX\_BUF**: ID for the current receive packet buffer
- **GOOD\_PLD**: Require the received packet (header and payload) have no errors
- **GOOD\_HDR**: Require the received header have no errors
- **MATCH\_RLYADDR**: Require the relay address field in the received header match the relay's local address
- **MATCH\_AF/MATCH\_AFGH/MATCH\_DF/MATCH\_NCMHOP**: Require the packet type field in the received header match the specified cooperative mode



#### 5.5.4 Interleaving Trials

Ideally, our hardware setup would operate entirely independent of the surrounding environment. The perfect setup would be a climate controlled anechoic test chamber. Lacking access to such a facility, we must address the impact of environmental factors in conducting our experiments.

For example, changes in temperature trigger frequency corrections in the oscillators. As explored in Section 3, our design can handle slow frequency drift, but sudden jumps can cause unrecoverable errors. Temperature changes also affect the noise characteristics of the analog ICs in the WARP hardware. Ambient interference can also contribute errors. Despite connecting our nodes to the emulator via coaxial cables, the cable connections and traces on the WARP Radio Board do admit some RF energy from the environment. Strong interference can cause symbols errors and packet detection failures, especially when testing topologies at low SNR where the radios apply the most gain. These are time-varying factors which we cannot control and whose effects we cannot directly measure. In order to distribute their impact across our results, we structure our long experiments as many iterations of short trials.

Each trial, usually 2-3 minutes long, tests a single combination of independent variables (attenuations, channel model and modulation rate). During a trial the source node generates traffic at regular intervals, cycling through cooperative modes per-packet. At the end of each trial, the transmitted and received packet counts are gathered from each node via WARPnet. The top level WARPnet script records the parameter combinations and raw packet counts per cooperative scheme per trial. Every combination of parameter values is re-tested with each iteration. When the experiment is complete, we sum the raw packet and bit counts for each trial across iterations, then calculate ratios for PER and BER.

Consider the experiment whose results are presented in Section 6.2.1. Here we tested 10 node placements (attenuation combinations) for two channel models and two payload modulation rates, resulting in 40 two minute trials per iteration. We can observe the results as the experiment progresses (the WARPnet log is written to disk with each update), gauging whether sufficient errors have been observed at every point to generate meaningful error rate plots.

## 5.6 Metrics

Our experiments are focused on characterizing the performance of our physical layer transceiver design. Two primary metrics are useful here: packet error rate and bit error rate. Our definitions for each are discussed below.

Notice that neither metric seeks to measure performance as a function of time. Such measurements (throughput, for example) are certainly important in gauging the performance of a real network. However, the true temporal cost of a given packet exchange cannot be meaningfully computed in the absence of a higher layer protocol. This is especially true for transmissions employing physical layer cooperation.

All our cooperative schemes operate across two transmissions, with the actual cooperation potentially occurring during the latter one. Any throughput measurement would need to account for both transmissions. For example, consider a MAC design which employs cooperation for every packet, guaranteeing two transmissions of each payload. The temporal cost of cooperation here is 100% (relative to a single slot, non-cooperative transmissions at the same rate). However, consider a protocol utilizing cooperation only during MAC re-transmissions, where a payload is transmitted again whether cooperation is employed or not. In such a design, the temporal cost

of cooperation would be zero.<sup>2</sup> Early results for one such protocol are discussed in Section 7.1.1.

Our work here is focused on measuring the expected outcomes of cooperative transmissions for various schemes, topologies and channel conditions. Coupled with our exploration of the requirements for employing each scheme, we anticipate these measurements will prove useful to MAC researchers seeking to incorporate physical layer cooperation into their protocol designs.

### 5.6.1 Packet Error Rate

Recall from Section 4.1.5 that our OFDM receiver progresses through a simple state machine triggered by an energy detection event. We classify a packet error as any transmission which does not end in the Good Payload state at the intended destination. This leads to our definition of packet error rate,  $(1 - \frac{N_{RxGood}}{N_{Tx}})$ , where  $N_{RxGood}$  is the number of packets received without error (i.e. ended in the Good Payload state) and  $N_{Tx}$  is the number of transmitted packets.

Figure 5.10 illustrates the OFDM receiver state transitions and outcomes for a given transmission. While the various failure states (Failed Detection, Bad Header and Bad Payload) all count as packet errors, our design tracks how many packets end in each state independently. This capability proves very useful in analyzing the overall PER and BER results for a given trial, helping identify the dominant source of errors in various regimes. The discussion of results in Section 6.5 explores this further.

---

<sup>2</sup>Cooperation here would certainly have other costs, like increased interference due to the larger spatial-spectral footprint for a given payload. Studying these costs falls far outside the scope of our work, but our bit and packet error rate measurements should prove useful in understanding the potential benefits.

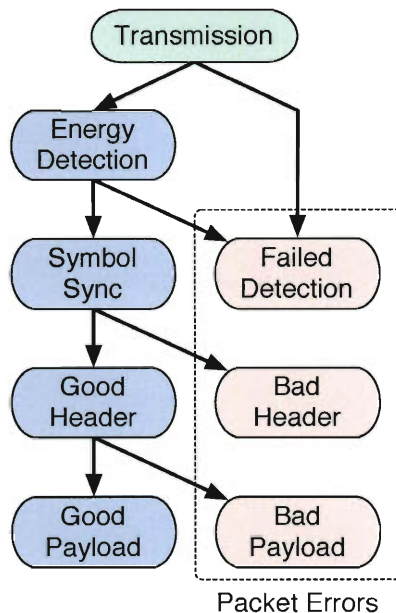


Figure 5.10 : State transitions in the OFDM receiver for a given transmission, indicating which end states are counted as packet errors.

### 5.6.2 Bit Error Rate

Bit error rate is a widely used metric for understanding PHY performance. From the MAC's perspective, any bit error is unacceptable; packets received with 1 or 5000 bit errors are equally useless for higher network layers. But knowing how many bit errors are responsible for a bad packet is very useful in gauging performance of a PHY and in designing error correcting codes.

Recall from Section 4.1.5 that our OFDM receiver interprets header fields to determine the payload length and modulation rate for each reception. If an error is detected in the header itself (the header has its own checksum) the receiver halts and does not attempt to receive the payload, being unsure of the parameters needed to process it. This is the Bad Header outcome illustrated in Figure 5.10. Thus, our design experiences payload bit errors only for receptions which end in the Bad Payload

state.

This receiver design leads to a definition of bit error rate as the ratio  $\frac{B_{Error}}{B_{Total}}$ , where  $B_{Error}$  is the number of payload bit errors and  $B_{Total}$  is the total number of payload bits processed. Note that  $B_{Total}$  includes bits from packets which end in both the Good and Bad Payload states. Header bits do not count towards either value. Packets which are not detected (Failed Detection outcome) or those received with header errors (Bad Header outcome) do not contribute to the BER calculation. As a result, our BER measurements must be considered in tandem with the corresponding packet error rates to fully understand the transceiver performance.

## Chapter 6

### Experiments with Full Cooperative Transceiver

In Chapters 3 and 4 we presented the design of our cooperative transceiver, including our solutions to numerous unique challenges encountered when implementing physical layer cooperation. In Chapter 5 we presented the design of experimental methods for rigorously characterizing the overall performance of our transceiver under a variety of topologies and channel conditions. This chapter presents the culmination of these design efforts in the form of extensive PER and BER measurements.

Section 6.1 below explains the structure and parameters common to all the experiments presented in this chapter. Sections 6.2-6.4 present the PER and BER results for the co-located source/relay, equidistant nodes and linear topologies, respectively.

Finally, Section 6.5 presents analysis of a few aspects of the overall performance results, focusing on error sources which dominate performance in specific regimes. We identify three dominant sources of errors and provide additional experimental results to isolate and explain the underlying causes.

The sections below present BER and PER results for each experiment. As discussed in Chapter 5 our methodology also records the numbers of packets which end in each error state in the receiver. Taken together, these counts compose the overall packet error rate. The complete set of plots for the probability of each packet error event is included in Appendix A.

## 6.1 Experimental Parameters

The PER/BER plots in the sections below are generated by a series of standalone experiments. Each of these experiments follows the same basic flow and was executed via a WARPnet script. Our WARPnet script has the same overall structure as the example in Section 5.5.1. The script here, however, executes four nested loops in order to sweep over every combination of the experimental parameters. The loop structure is illustrated in Figure 6.1 and is detailed below.

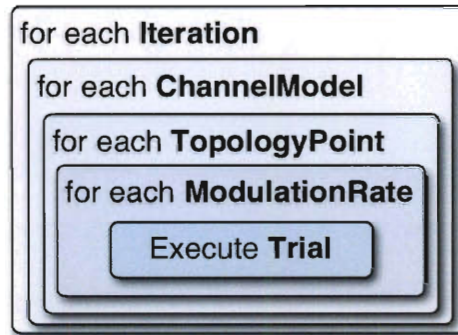


Figure 6.1 : Structure of each experiment, with four nested loops sweeping over multiple iterations of every combination of experimental parameters.

Our experiment loops execute as follows:

- **Execute Trial:** Each trial consists of 2-3 minutes of continuous traffic generation, with packet transmissions initiated at a fixed interval by the source node. Cooperative schemes are interleaved per-packet. The nodes log every packet transmission and reception and reports these counts via WARPnet after every trial. A WARPnet co-processor logs the counts of bit errors and total bits received per trial.
- **Modulation Rate:** We test payload modulation rates of **QPSK** and **16-QAM**.

Packet headers are modulated with QPSK in every experiment.

- **Topology Point:** Each point in the emulated topology is realized by setting the three channel emulator output attenuations ( $S \rightarrow D$ ,  $S \rightarrow R$ ,  $R \rightarrow D$ ), defining the average path losses along the three links. The total attenuation along each path is 53 dB higher than the configured attenuation, due to cable losses, power splitting and the emulator's inherent attenuation.
- **Channel Model:** We run every test with the **TGn A** (frequency flat fading) and **TGn B** (frequency selective fading) channel models. Both models are configured for the maximum possible fading velocity of 1.2 km/h.
- **Iteration:** Every combination of experimental parameters is tested multiple times; each iteration is a complete sweep of the parameters and is logged independently. We sum over all iterations in an experiment to generate the aggregate PER/BER results.



## 6.2 Co-located Source/Relay Topology

This section presents the PER and BER results for two experiments of the co-located source/relay topology, illustrated in Figure 6.2, testing payload lengths of 1412 and 692 bytes, respectively. The parameters common to both experiments are listed below; experiment-specific parameters are listed above each set of results.

- **Topology Points:** The SR attenuation is always zero; the SR and RD attenuations are always equal and are swept from 0 to 36 dB in 4 dB steps, giving 10 distinct topology points. The overall path losses are 53 dB higher than the attenuation settings, due to cable losses, power splitting and the emulator's inherent attenuation.

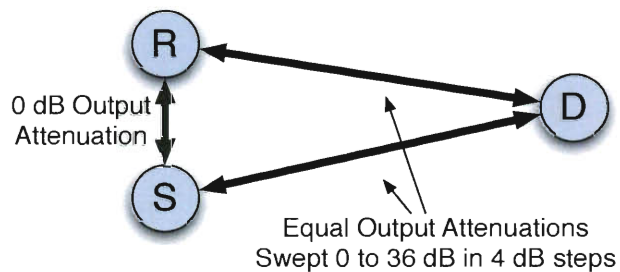


Figure 6.2 : Co-located source/relay topology.

- **Cooperative Schemes:** This experiment tests three schemes:
  - **AF:** Amplify and forward (relay re-transmits successfully decoded packets)
  - **DF:** Decode and forward
  - **NC:** Non-cooperative (no relay participation)

### 6.2.1 PER/BER with 1412 Byte Payloads

The eight plots below present the PER and BER for the equidistant nodes topology tested with 1412 byte payloads. Each plot shows three curves, one per cooperative scheme, and represents a single combination of channel model and payload modulation rate.

This experiment executed 14 iterations; each trial lasted 2.5 minutes. For QPSK modulation each data point below represents 232k packet transmissions. For 16-QAM each point represents 378k transmissions. In total this experiment tested 6.9 million packets for QPSK and 11.3 million for 16-QAM.

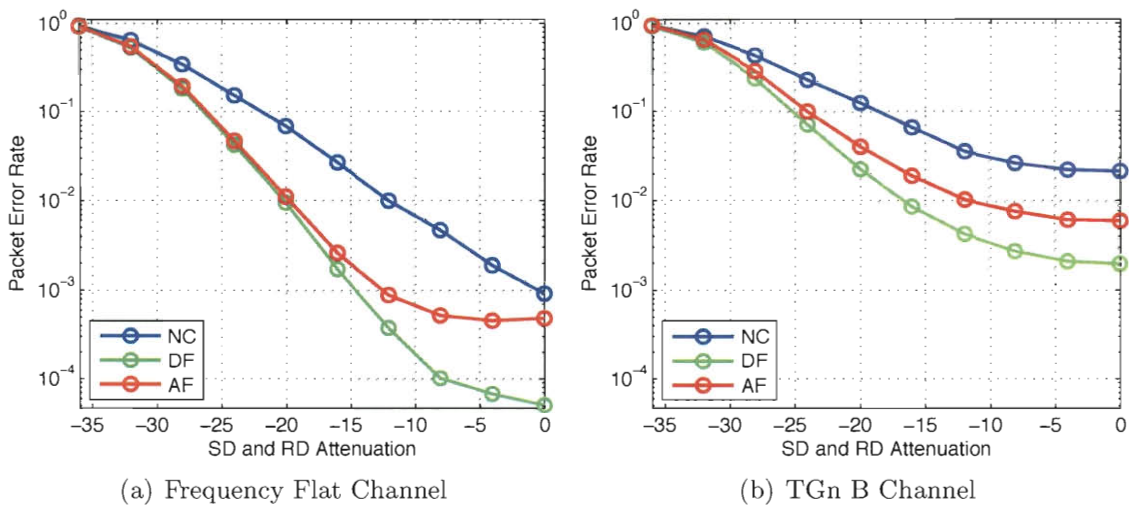


Figure 6.3 : Packet error rates for co-located source/relay topology with 1416 byte, QPSK modulated payloads.

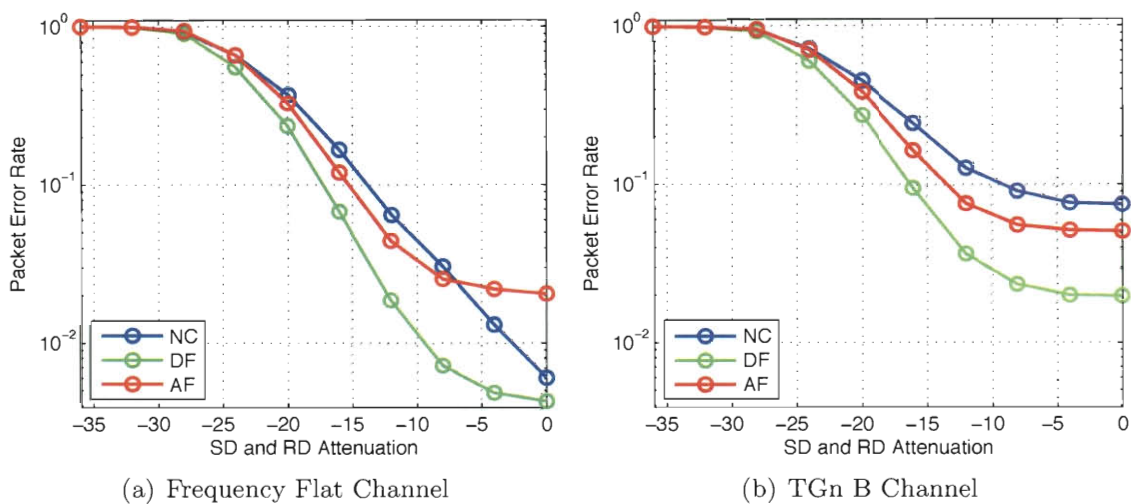


Figure 6.4 : Packet error rates for co-located source/relay topology with 1416 byte, 16-QAM modulated payloads.

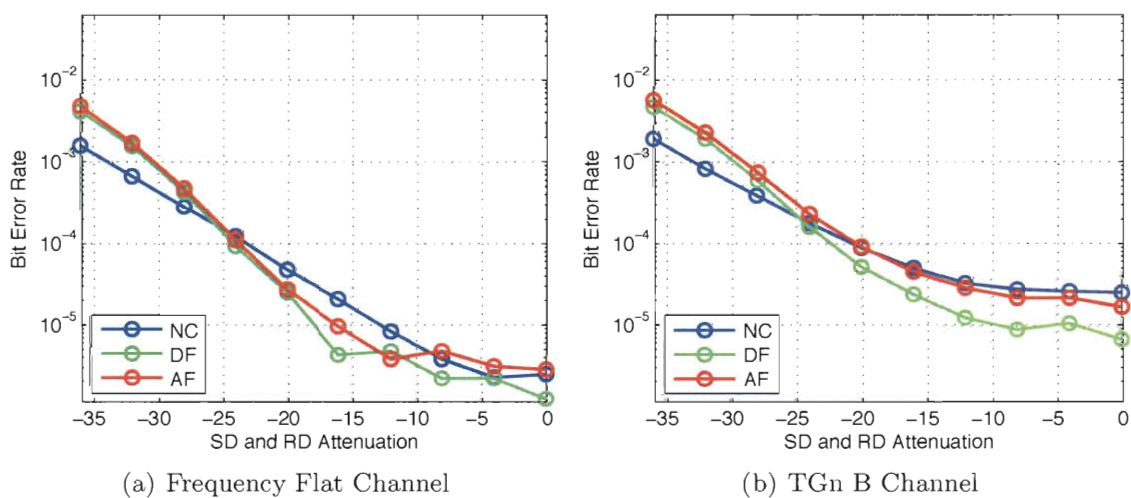


Figure 6.5 : Bit error rates for co-located source/relay topology with 1416 byte, QPSK modulated payloads.

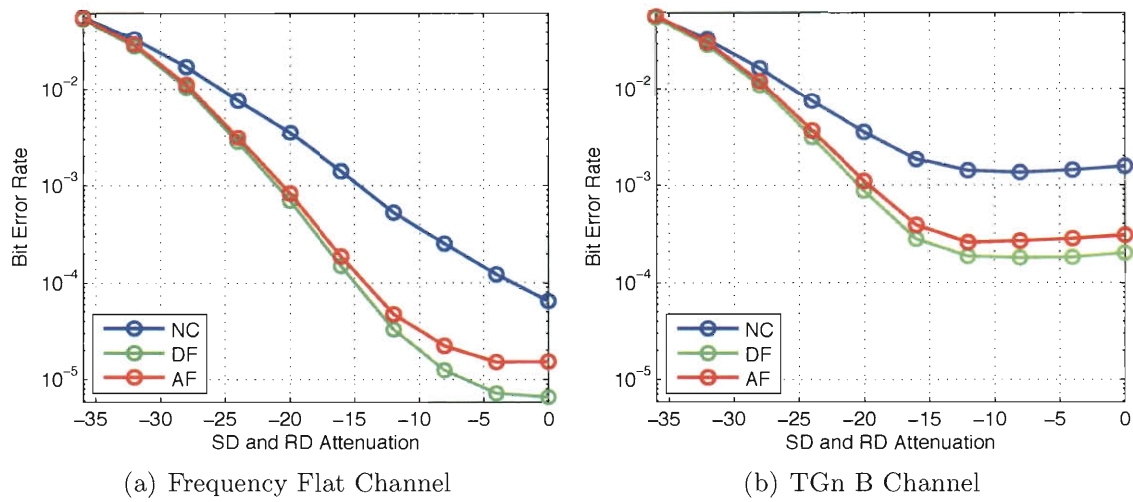


Figure 6.6 : Bit error rates for co-located source/relay topology with 1416 byte, 16-QAM modulated payloads.

### 6.2.2 PER/BER with 692 Byte Payloads

The eight plots below present the PER and BER for the equidistant nodes topology tested with 1412 byte payloads. Each plot shows four curves, one per cooperative scheme, and represents a single combination of channel model and payload modulation rate.

This experiment executed 18 iterations; each trial lasted 3 minutes. For QPSK modulation each data point below represents 316k packet transmissions. For 16-QAM each point represents 633k transmissions. In total this experiment tested 12.7 million packets for QPSK and 25.3 million for 16-QAM.

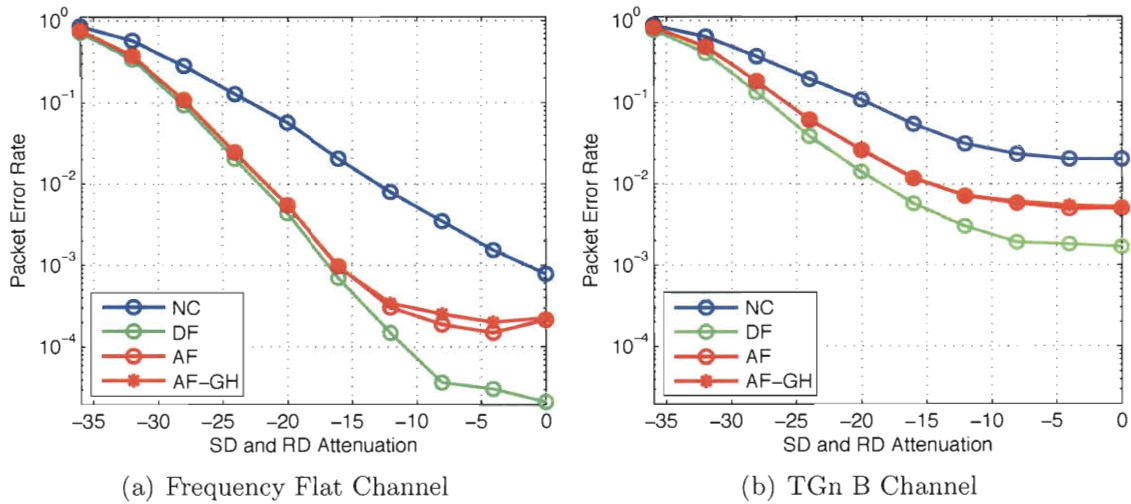


Figure 6.7 : Packet error rates for co-located source/relay topology with 692 byte, QPSK modulated payloads.

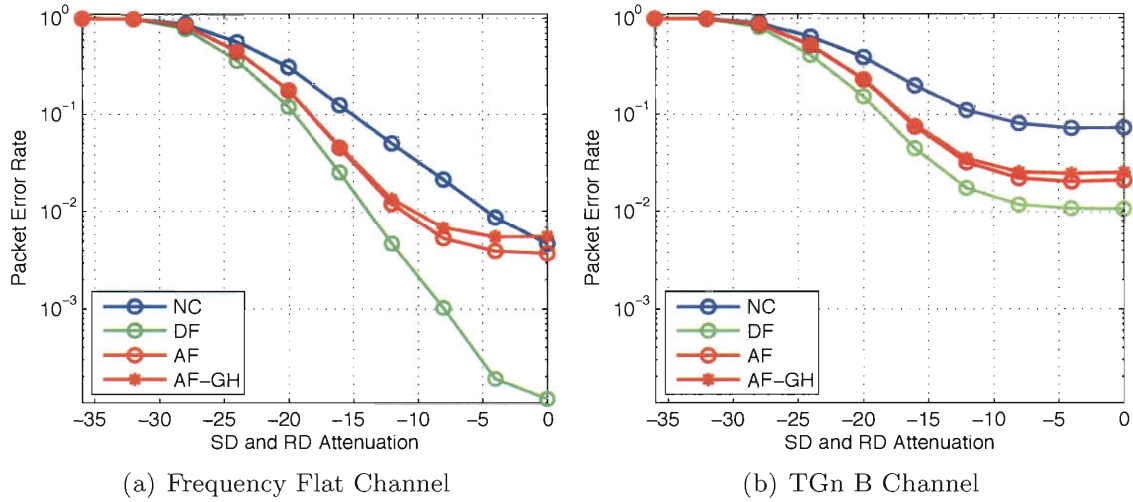


Figure 6.8 : Packet error rates for co-located source/relay topology with 692 byte, 16-QAM modulated payloads.

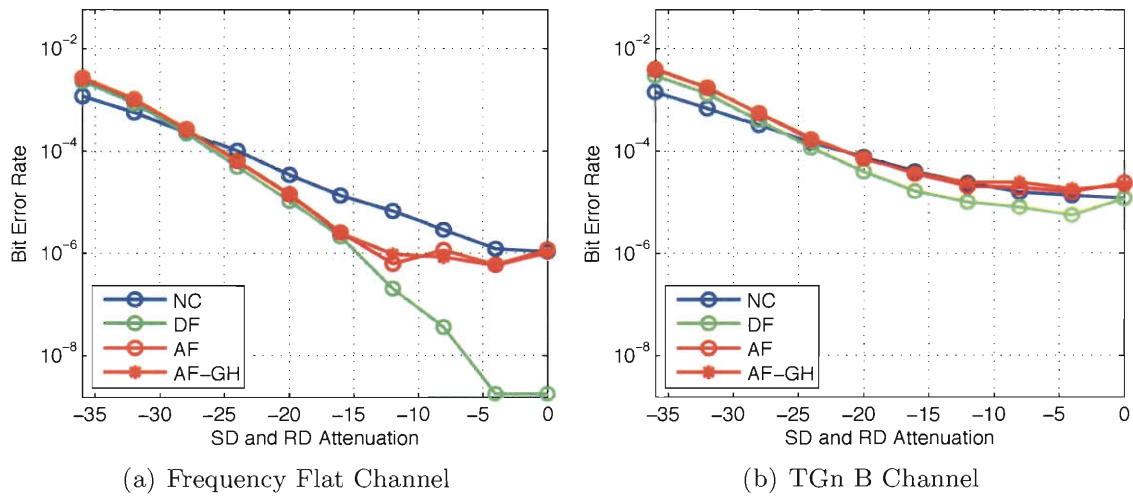


Figure 6.9 : Bit error rates for co-located source/relay topology with 692 byte, QPSK modulated payloads.

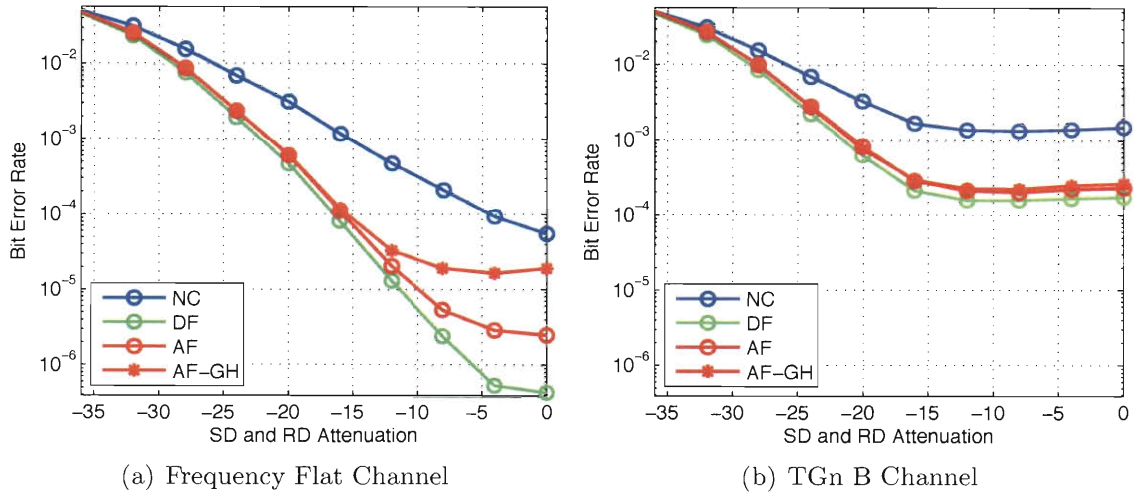


Figure 6.10 : Bit error rates for co-located source/relay topology with 692 byte, 16-QAM modulated payloads.

### 6.2.3 Observations

We can make a few observations from these results. At nearly every topology point cooperation provides a performance gain; in some cases the gain is substantial. Consider the PER for QPSK payloads in a flat fading channel, shown in Figure 6.3(a). The peak PER improvement for DF vs. NC is nearly  $45\times$  ( $1.03 \cdot 10^{-4}$  vs.  $4.6 \cdot 10^{-3}$  at -8 dB attenuation). Further, the overall shape of the cooperative curves demonstrate evidence of diversity via steeper slopes with increasing SNR than the non-cooperative curves.

It is clear that both modulation rate, channel model and packet length affect performance of every scheme. The smaller noise margin for 16-QAM and longer delay spread of the TGn B model both degrade performance. Even subject to these degradations, DF (and in some cases AF) still outperform the non-cooperative link. Shorter packets also clearly perform better.

There are a few anomalies in these results. For example, refer to the PER curves for 16-QAM in a flat fading channel, shown in Figure 6.4(a). Note how the PER for AF is actually worse than NC at high SNR. But compare this to the corresponding BER curves in Figure 6.6. Here, AF clearly outperforms NC at all SNRs. This observation is explored in detail in Section 6.5.3. Also consider all the curves for the TGn B channel model. With this model, every scheme exhibits an error floor in the high SNR regime. This observation is explored in detail in Section 6.5.2.



### 6.3 Equidistant Nodes Topology

This section presents the PER and BER results for two experiments of the equidistant nodes topology, illustrated in Figure 6.11, testing payload lengths of 1412 and 692 bytes, respectively. The parameters common to both experiments are listed below; experiment-specific parameters are listed above each set of results.

- **Topology Points:** The SD, SR and RD attenuations are always equal. The common attenuation is swept from 0 to 36 dB in 4 dB steps, giving 10 distinct topology points. The overall path losses are 53 dB higher than the attenuation setting, due to cable losses, power splitting and the emulator's inherent attenuation.

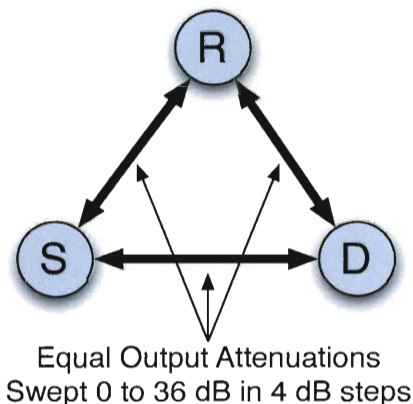


Figure 6.11 : Equidistant nodes topology.

- **Cooperative Schemes:** This experiment tests four schemes:
  - **AF:** Amplify and forward (relay re-transmits successfully decoded packets)
  - **AF-GH:** Amplify and forward - good header (relay re-transmits packets with successfully decoded headers)

- **DF**: Decode and forward
- **NC**: Non-cooperative (no relay participation)

### 6.3.1 PER/BER with 1412 Byte Payloads

The eight plots below present the PER and BER for the equidistant nodes topology tested with 1412 byte payloads. Each plot shows four curves, one per cooperative scheme, and represents a single combination of channel model and payload modulation rate.

This experiment executed 31 iterations. For QPSK modulation each data point below represents 385k packet transmissions. For 16-QAM each point represents 628k transmissions. In total this experiment tested 15.4 million packets for QPSK and 25.1 million for 16-QAM.

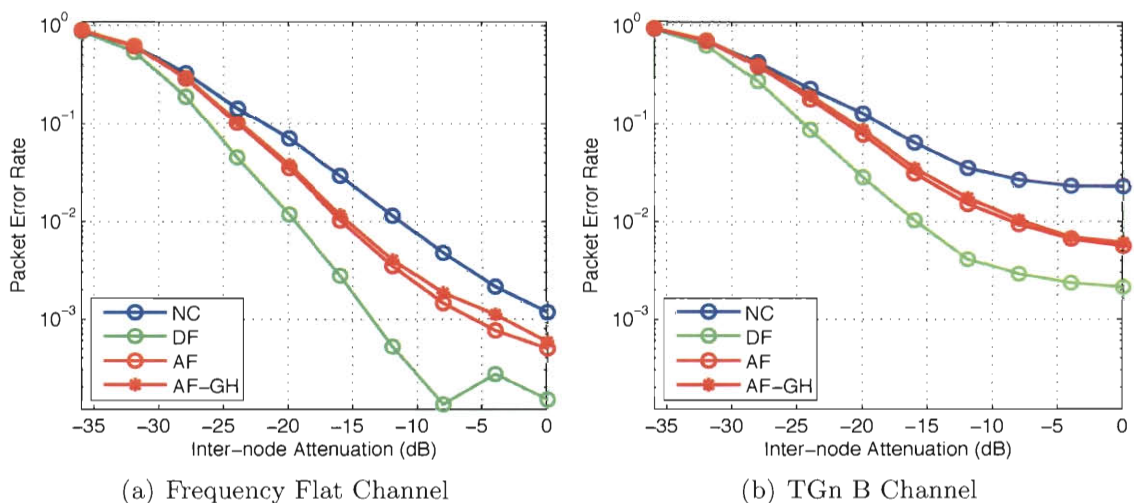


Figure 6.12 : Packet error rates for equidistant nodes topology with 1412 byte, QPSK modulated payloads.

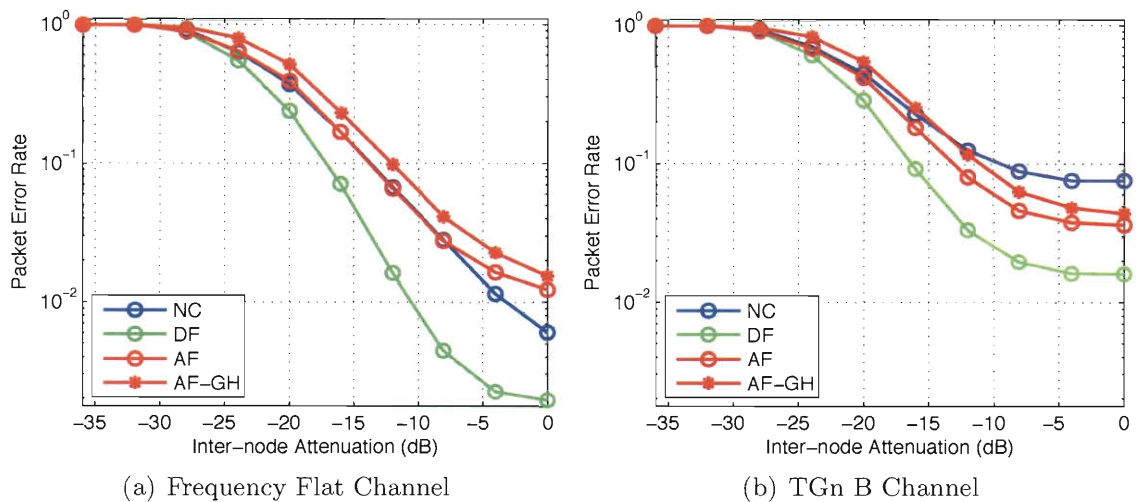


Figure 6.13 : Packet error rates for equidistant nodes topology with 1412 byte, 16-QAM modulated payloads.

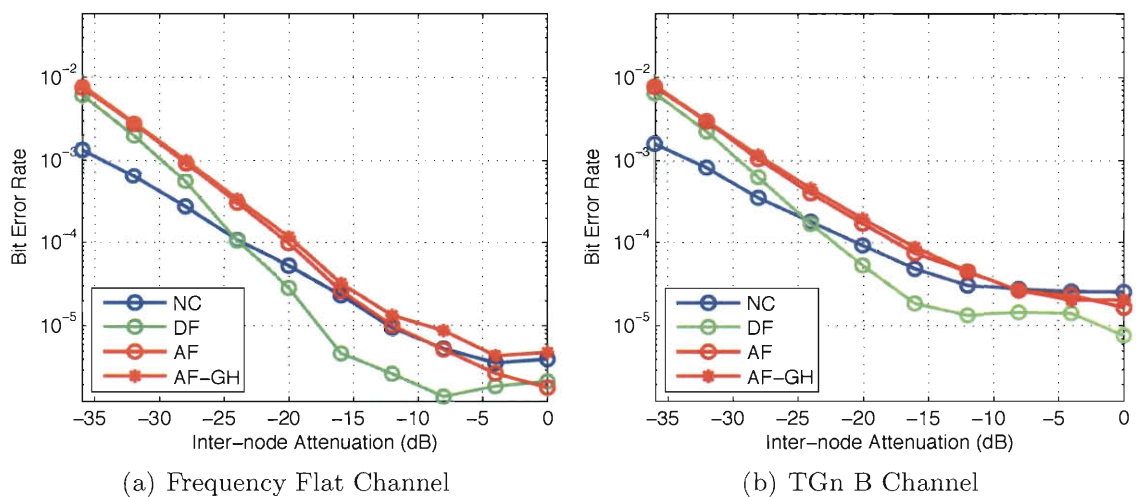


Figure 6.14 : Bit error rates for equidistant nodes topology with 1412 byte, QPSK modulated payloads.

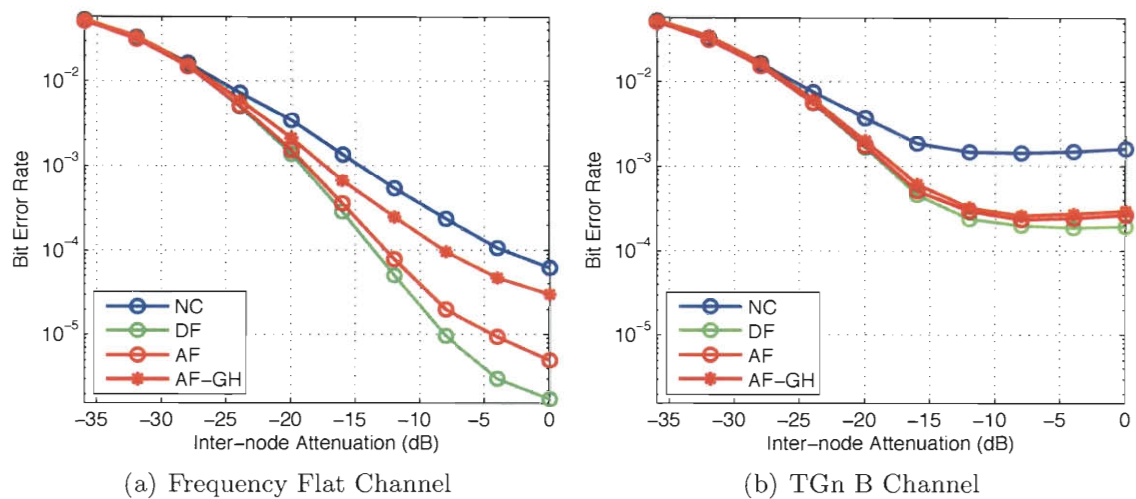


Figure 6.15 : Bit error rates for equidistant nodes topology with 1412 byte, 16-QAM modulated payloads.

### 6.3.2 PER/BER with 692 Byte Payloads

The eight plots below present the PER and BER for the equidistant nodes topology tested with 692 byte payloads. Each plot shows four curves, one per cooperative scheme, and represents a single combination of channel model and payload modulation rate.

This experiment executed 24 iterations. For QPSK modulation each data point below represents 299k packet transmissions. For 16-QAM each point represents 595k transmissions. In total this experiment tested 11.9 million packets for QPSK and 23.8 million for 16-QAM.

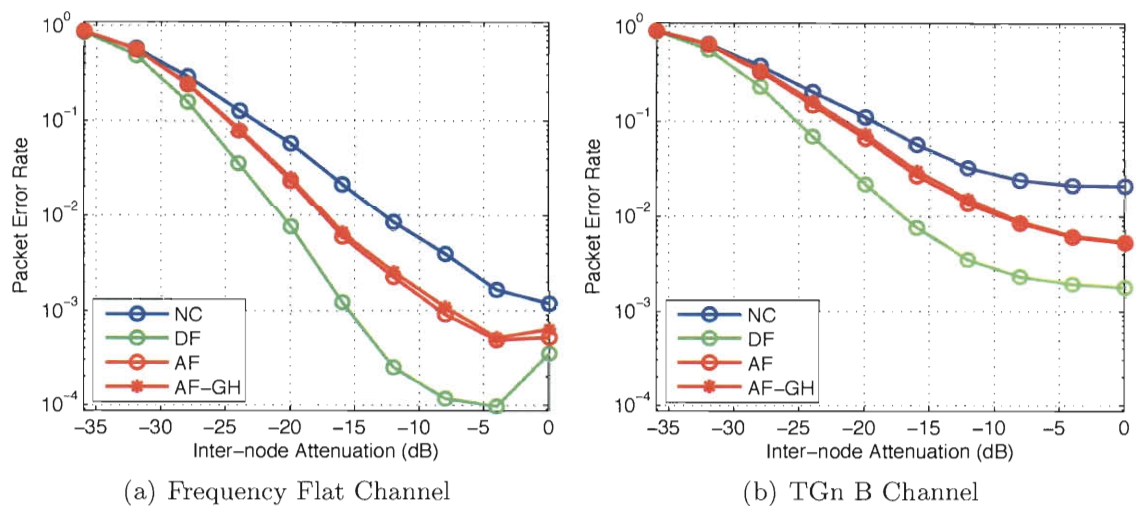


Figure 6.16 : Packet error rates for equidistant nodes topology with 692 byte, QPSK modulated payloads.

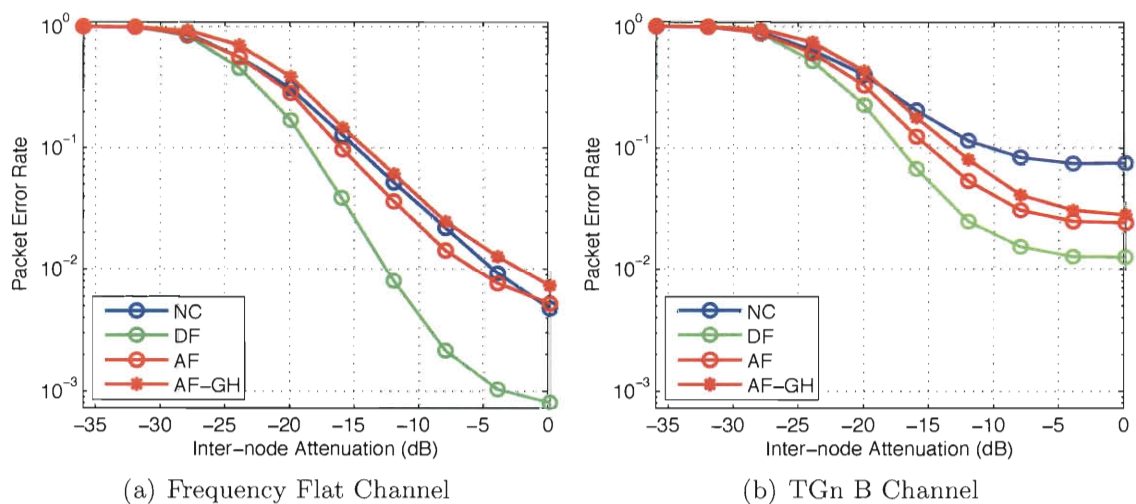


Figure 6.17 : Packet error rates for equidistant topology nodes with 692 byte, 16-QAM modulated payloads.

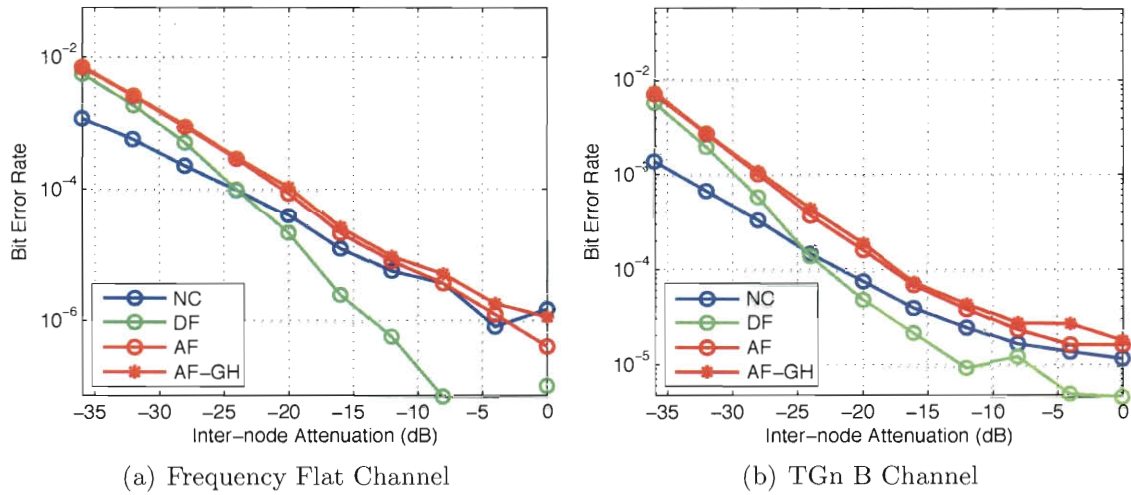


Figure 6.18 : Bit error rates for equidistant topology nodes with 692 byte, QPSK modulated payloads.

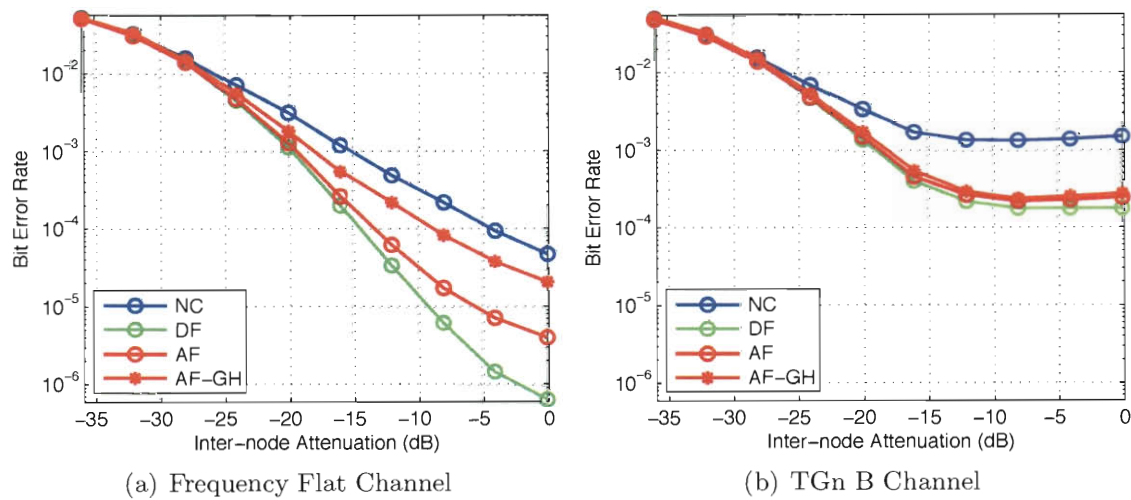


Figure 6.19 : Bit error rates for equidistant topology nodes with 692 byte, 16-QAM modulated payloads.

### 6.3.3 Observations

Overall the results for the equidistant nodes topology are consistent with those for the co-located source/relay topology. The same general trends are apparent, with DF performing better than other schemes in every test, QPSK outperforming 16-QAM and shorter packets performing better than long ones.

One key difference between the equidistant and co-located source/relay results is the performance of AF. For example, compare the DF and AF PER curves in Figure 6.3 (co-located source/relay topology) to those in Figure 6.12 (equidistant nodes topology). Both figures represent 1412 byte, QPSK payloads in frequency flat fading. The DF curves are nearly identical; there is a small decrease in performance in the equidistant nodes topology. The PER for AF, however, is much worse in the equidistant case. This clearly demonstrates the impact of noise amplification in AF. The only difference between the experimental parameters for these topologies is the attenuation of the source-relay path. As this attenuation increases the SNR of the waveform captured at the relay decreases. When this same waveform is amplified and re-transmitted the lower SNR translates into more noise in the relay's contribution to the cooperative transmission. This amplified noise clearly decreases performance at the destination.

## 6.4 Linear Topologies

This section presents the PER and BER results for two experiments of linear topologies. The parameters common to both experiments are listed below; experiment-specific parameters are listed above each set of results.

- **Topology Points:** These experiments test two linear topologies, emulating source-destination separations of 10.4 and 18 m. In both, SD attenuation is fixed, while the SR and RD attenuations are varied in tandem to emulate various relay positions along the SD line. The actual attenuations employed are listed above each set of results.
- **Cooperative Schemes:** This experiment tests five schemes:
  - **AF:** Amplify and forward (relay only re-transmits packets with successfully decoded payloads)
  - **AF-GH:** Amplify and forward - Good Header (relay only re-transmits packets with successfully decoded headers)
  - **DF:** Decode and forward
  - **MHOP:** Multi-hop (only relay transmits in second slot, and only if it decodes the payload successfully)
  - **NC:** Non-cooperative (no relay participation)

### 6.4.1 PER/BER with 10.4 m SD Separation

The eight plots below present the PER and BER for a linear topology with the source and destination nodes separated by 10.4 m. This experiment uses 1412 byte payloads. Each plot shows five curves, one per cooperative scheme, and represents a



single combination of channel model and payload modulation rate. The X-axes are all relay position along the SD line in meters.

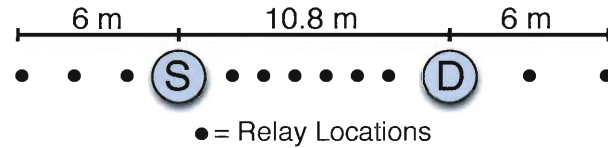


Figure 6.20 : Linear topologies with 10.4 m source/destination separation.

Table 6.1 lists the attenuations used to emulate each relay position in this topology. The source-destination attenuation is fixed at 18 dB for all trials. The actual average path loss along each link is 53 dB higher than each attenuation, due to cable losses, power splitting and the emulator's inherent attenuation.

Relay Position	SR Attenuation	RD Attenuation
-11.2 m	13.0 dB	22.1 dB
-9.2 m	9.3 dB	21.0 dB
-7.2 m	2.9 dB	19.6 dB
-3.2 m	2.9 dB	16.1 dB
-2.1 m	6.9 dB	14.8 dB
-1.0 m	9.6 dB	13.3 dB
0.1 m	11.8 dB	11.6 dB
1.2 m	13.5 dB	9.5 dB
2.2 m	14.9 dB	6.6 dB
8.2 m	20.3 dB	6.6 dB
11.1 m	22.1 dB	13.0 dB

Table 6.1 : Attenuations configured in the channel emulator for each relay position in the 10.4 m SD separation linear topology.

This experiment executed 22 iterations. For QPSK modulation each data point below represents 319k packet transmissions. For 16-QAM each point represents 536k transmissions. In total this experiment tested 17.6 million packets for QPSK and 29.5 million for 16-QAM.

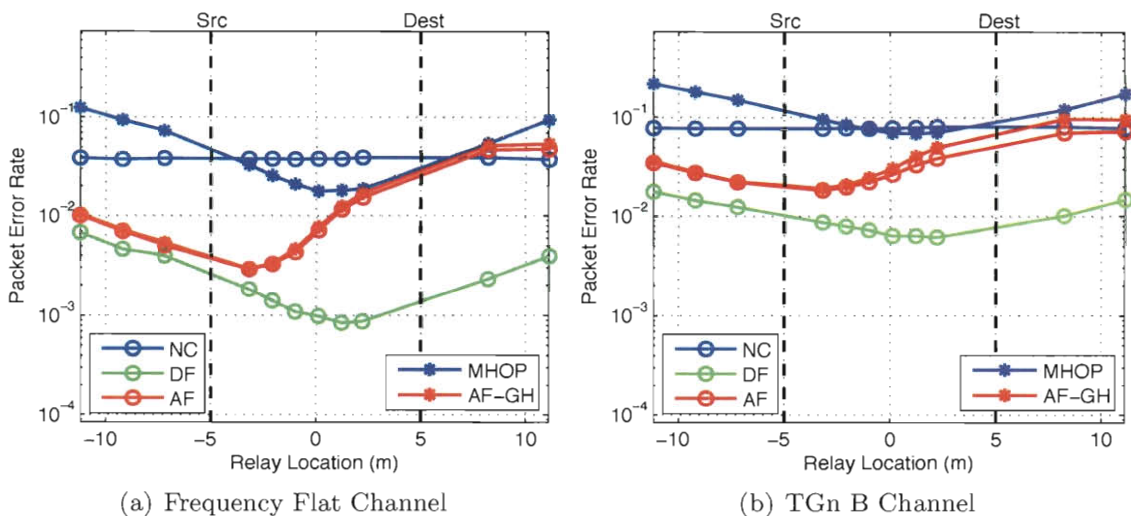


Figure 6.21 : Packet error rates for linear topology QPSK modulated payloads and 10.4 m SD separation.

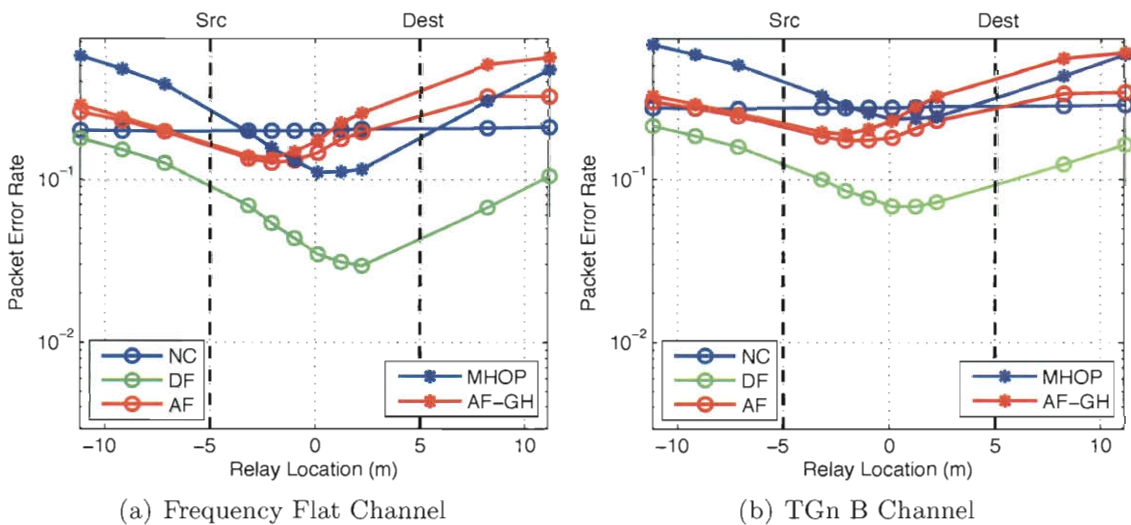


Figure 6.22 : Packet error rates for linear topology 16-QAM modulated payloads and 10.4 m SD separation.

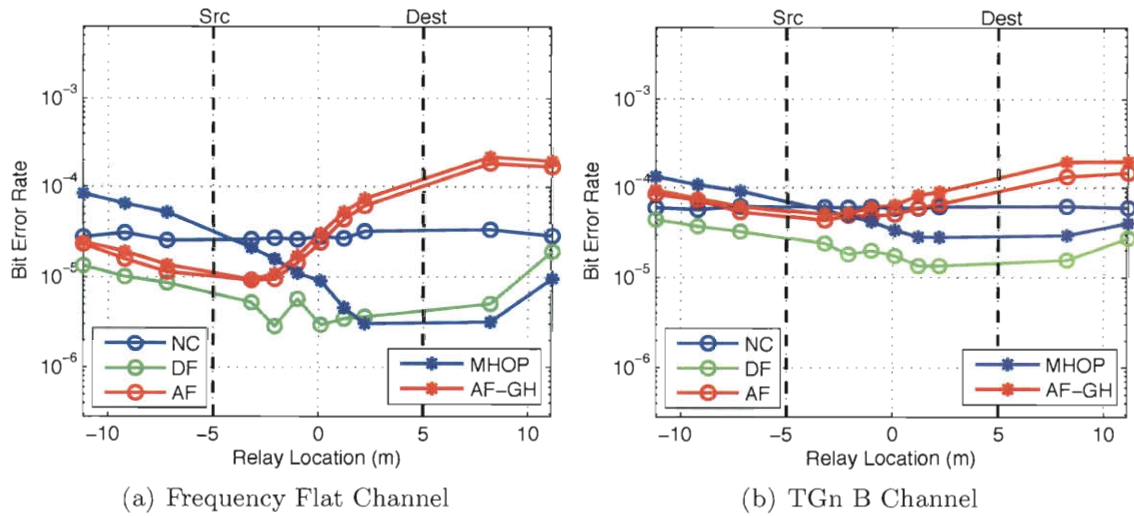


Figure 6.23 : Bit error rates for linear topology QPSK modulated payloads and 10.4 m SD separation.

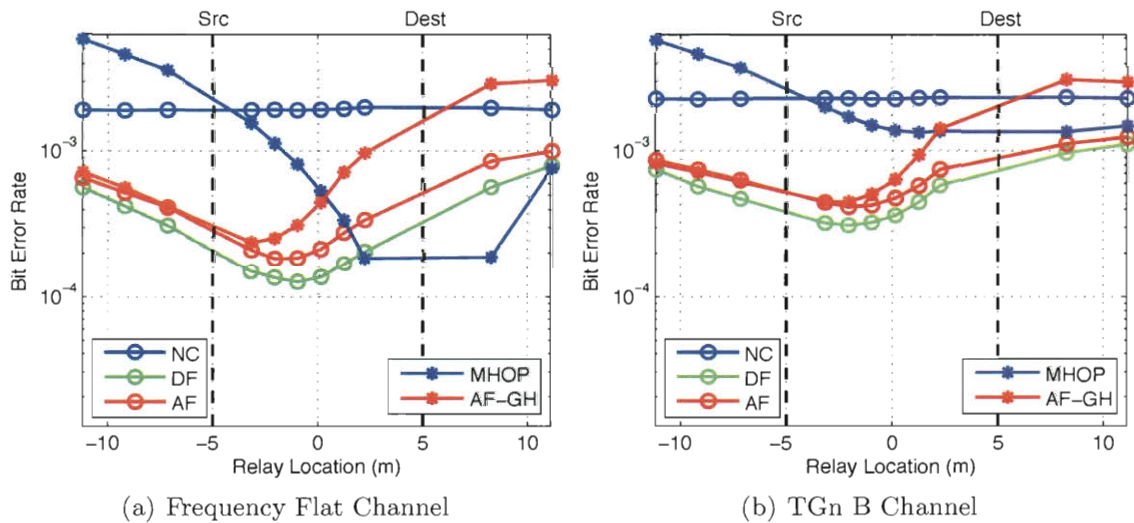


Figure 6.24 : Bit error rates for linear topology 16-QAM modulated payloads and 10.4 m SD separation.

### 6.4.2 PER/BER with 18 m SD Separation

The eight plots below present the PER and BER for a linear topology with the source and destination nodes separated by 18 m. This experiment uses 1412 byte payloads. Each plot shows five curves, one per cooperative scheme, and represents a single combination of channel model and payload modulation rate. The X-axes are all relay position along the SD line in meters.

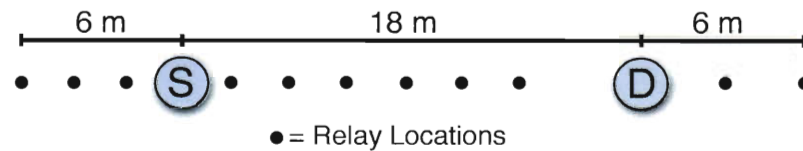


Figure 6.25 : Linear topologies with 18 m source/destination separation.

Table 6.2 lists the attenuations used to emulate each relay position in this topology. The source-destination attenuation is fixed at 23 dB for all trials. The actual average path loss along each link is 53 dB higher than each attenuation, due to cable losses, power splitting and the emulator's inherent attenuation.

This experiment executed 12 iterations. For QPSK modulation each data point below represents 174k packet transmissions. For 16-QAM each point represents 293k transmissions. In total this experiment tested 9.6 million packets for QPSK and 16.1 million for 16-QAM.

Relay Position	SR Attenuation	RD Attenuation
-15.0 m	13.0 dB	25.6 dB
-13.0 m	9.3 dB	24.8 dB
-11.0 m	2.9 dB	24.0 dB
-7.0 m	2.9 dB	21.9 dB
-4.8 m	9.7 dB	20.6 dB
-2.6 m	13.6 dB	19.0 dB
-0.3 m	16.3 dB	17.1 dB
1.8 m	18.3 dB	14.6 dB
4.0 m	20.0 dB	11.3 dB
12.0 m	24.4 dB	6.6 dB
15.0 m	25.6 dB	13.0 dB

Table 6.2 : Attenuations configured in the channel emulator for each relay position in the 18 m SD separation linear topology.

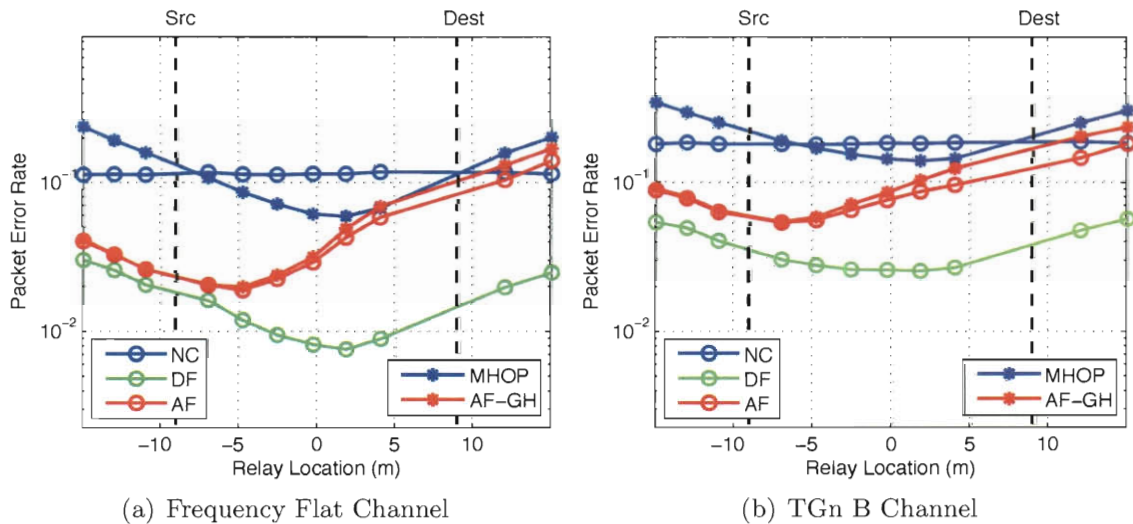


Figure 6.26 : Packet error rates for linear topology QPSK modulated payloads and 18 m SD separation.

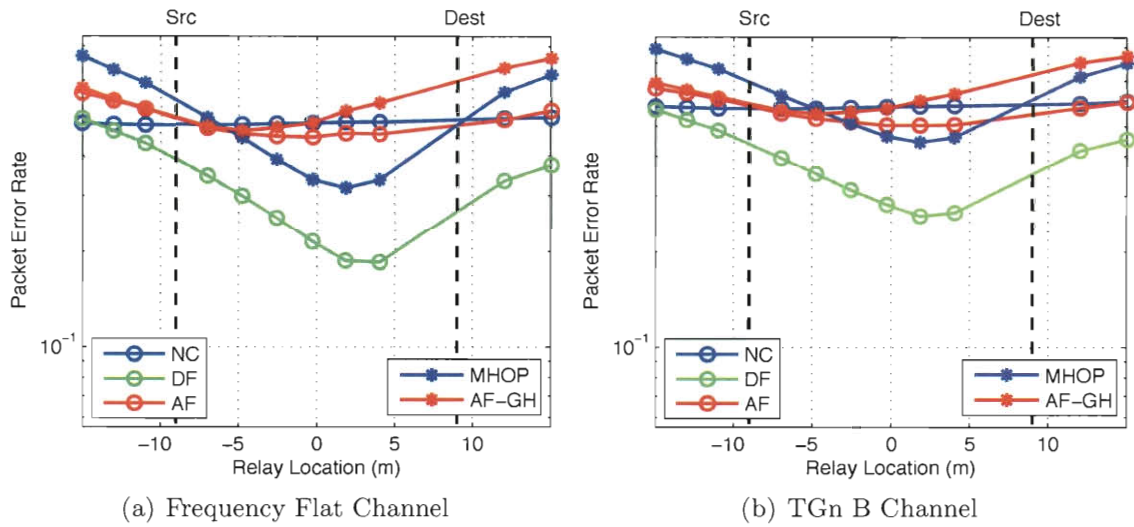


Figure 6.27 : Packet error rates for linear topology 16-QAM modulated payloads and 18 m SD separation.

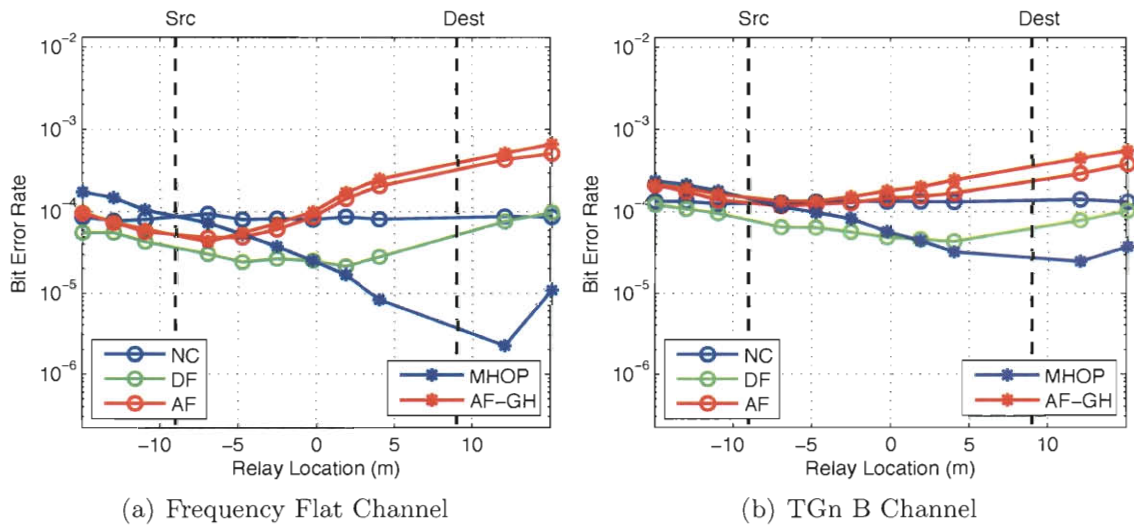


Figure 6.28 : Bit error rates for linear topology QPSK modulated payloads and 18 m SD separation.

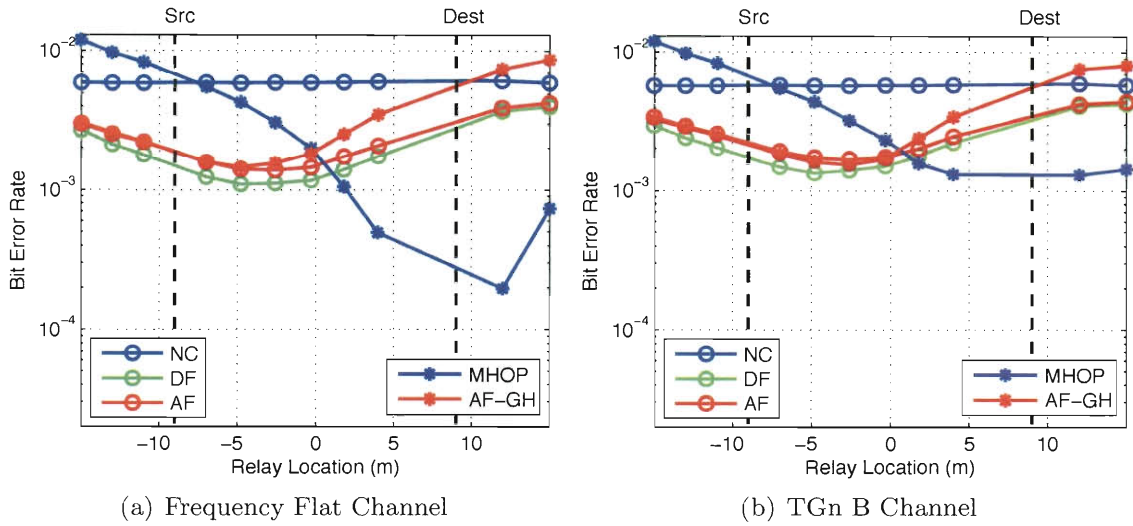


Figure 6.29 : Bit error rates for linear topology 16-QAM modulated payloads and 18 m SD separation.

### 6.4.3 Observations

We can make a few observations from these results. It is clear that the PER performance of DF exceeds that of all other schemes. This holds true for every relay position, channel model and modulation rate. The peak performance improvement with DF is significant. Consider Figure 6.21(a), which shows PER for QPSK modulated payloads in flat fading. The best PER for DF is  $8.5 \cdot 10^{-4}$  (at location 1.2 m), nearly  $45\times$  better than the corresponding PER of  $3.8 \cdot 10^{-2}$  for NC. This is a dramatic improvement. That this peak improvement occurs with the relay between the source and destination makes sense. In this regime, performance of the SR and RD links is balanced, allowing the relay to deliver the maximum possible assistance.

The AF curves are somewhat less impressive than DF, but still consistent with expectations. The linear topologies highlight an important difference between AF and DF. Both schemes use the same rule to determine when to cooperate (successful

reception of a packet). However, the amount of help each provides is different. For DF, the quality of the relay transmission is determined by the accuracy of its CFO estimate. As shown in Section 3.7.5, the performance of our estimator is very good at mid-to-high SNR, and degrades seriously only at very low SNR. For AF, the quality of its transmission is determined by the source-relay channel. A weak channel results in a noisy relay transmission. It is clear from our results that as the relay moves away from the source, the rate of degradation in captured waveform quality (for AF) significantly outpaces that of the CFO estimate (for DF).

In terms of PER, DF, and in most cases AF, outperform simple multi-hop. This is a clear demonstration of the benefits of diversity. Multi-hop provides no diversity improvement, succeeding only when two channels (SR and RD) are both able to successfully convey a packet. As expected, cooperation provides actual diversity, delivering packets when any combination of the SD and RD transmissions succeeds.

These experiments highlight the importance of interpreting BER and PER together. For example, note the unusual BER curves for multi-hop in Figure 6.29. Viewed in isolation, the BER performance of multihop dramatically exceeds all other schemes. But recall from Section 5.6 that only transmissions which end in the Bad Payload state at the destination contribute errors to the BER calculation. In other words, transmissions which are not detected or which end in the Bad Header state do not count towards BER. In the multihop scheme the destination can only receive packets from the relay, and the relay only transmits packets it receives from the source with zero errors. When the relay is far from the source it will only occasionally receive error-free packets. But if it is near the destination, it will successfully re-transmit these occasional packets with high probability. Thus, the multihop BER appears to be very good when the relay is near the destination, but the overall performance is



actually very poor. This is clearly demonstrated by the corresponding PER curves for multihop (Figure 6.26, for example) which consistently show poor multihop performance when the relay is near the destination.

Finally, the relative performance of the two linear topologies is consistent. As expected, given the longer distances involved, the overall error rates are higher in the topology with the larger source/destination separation.

## 6.5 Analysis of Performance Bottlenecks

Overall the results of our experiments are very encouraging. In every experiment, there are topologies where physical layer cooperation provides significant performance gains. The overall performance of the transceiver varies as expected with channel statistics, average SNR and modulation rate.

However, a few aspects of the overall performance results presented in Sections 6.2-6.4 merit further investigation. Specifically, we seek to understand the underlying causes for performance limitations observed in our PER and BER results. These limitations manifest as error floors, regimes where performance no longer improves with increasing SNR. This section presents discussions and additional experiments exploring three underlying causes of error floors in our results.

### 6.5.1 CFO Pre-Correction Errors

Consider the PER curves in Figure 6.3(a) for co-located source/relay and QPSK modulation. In the mid-SNR region, the cooperative schemes are improving with a slope of  $\approx 2$  (2 orders of PER improvement per 10 dB SNR increase), and the non-cooperative curve has slope  $\approx 1$ . These slopes are a clear demonstration of diversity gain in our system. At the highest SNRs, however, the AF and DF curves begin to

flatten out. This behavior for AF is explored in Section 6.5.3 below.

For the error floor in DF, recall the processes discussed in Section 3 for estimating and pre-correcting CFO at the relay. Specifically, refer to Figure 3.27, which shows the distribution of the relay’s CFO estimation error as a function of packet length and SNR. The PER curves in Figure 6.3(a) (for full-length QPSK packets) correspond to the “120 syms” error curve in Figure 3.27.

The CFO estimation error curves in Figure 3.27 were generated via the channel emulator with a static channel model. The PER curves use fading channel models. For the flat fading results (the (a) subfigures), the channel realizes a random instantaneous amplitude. Each random amplitude corresponds to some point on the X-axis in the CFO estimation plot. In other words, the quality of the relay’s CFO estimate varies along with the source-relay channel, providing higher variance estimates for lower channel magnitudes.

We verify this with an experiment similar to the one used for Figure 3.27. As in that experiment, two nodes share an RF reference clock, fixing the actual CFO at zero. The source node applies a known CFO of 305 Hz to every transmission; the destination receives each packet and records its frequency domain CFO estimate (the time domain CFO estimator is disabled). In this experiment, the emulator applies frequency flat fading.

Using WARPnet, we record the received power, CFO estimate and receiver outcome (good/bad) for every packet. The results are shown in Figure 6.30, which shows a 2-D histogram of the probability of each combinations of receive power and CFO estimation error. This plot includes only data from packets received with no errors, corresponding to those receptions which a DF relay would re-transmit. Note that the probabilities (as colors) are on a log scale. First, observe the spread of receive powers

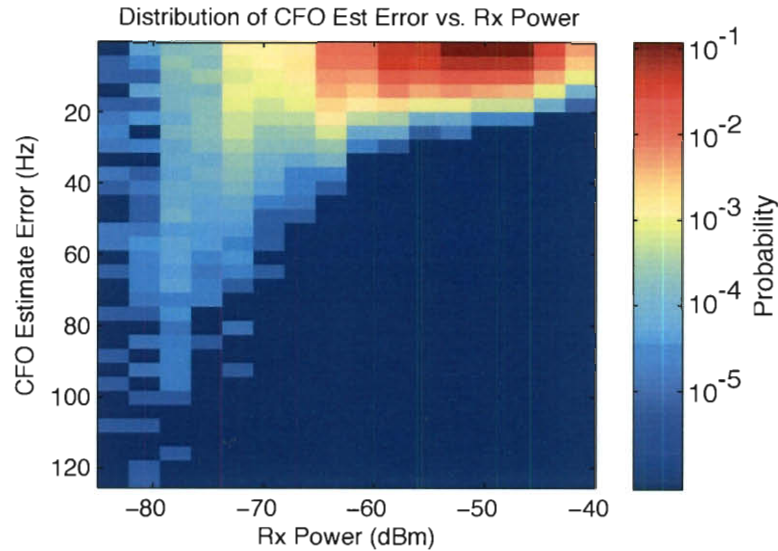


Figure 6.30 : Probability distribution of frequency domain CFO estimation error vs. Rx power, for full-length packets in a flat fading channel.

spanning 40 dB, with lower powers being less likely. This distribution represents the fading statistics of the emulator’s channel model. Second, note the range of CFO estimation errors, and compare these to the PER/BER curves in Figure 3.22. Any estimation error is bad, but errors larger than even  $\approx 50$  Hz degrade performance. It is clear from these results that our CFO estimator provides very good estimates for moderate-to-high SNRs. But at low SNR there is a higher probability of the estimator providing CFO values with errors large enough to degrade performance at the destination.

This observation presents an interesting dilemma. One of the attractive properties of decode and forward relaying is the isolation between the source-relay and relay-destination channels. In AF, for example, the re-transmitted waveform preserves whatever degradation it suffered along the SR channel. In an ideal DF relay, the SR channel would have no impact on the quality of the relay’s transmission to the

destination; the relay strips away any received noise and channel degradations by re-generating a fresh waveform for transmission.

In our implementation, however, the relay applies CFO pre-correction to its transmission, and the CFO value it uses can be degraded by the SR channel. In a sense, this process allows SR channel effects to “leak through” to the DF relay’s transmission. In the worst case, the relay would transmit with a bad CFO estimate and cause a packet error when none would otherwise have occurred. Thankfully, our experiments demonstrate this is a very rare event, as shown by the significant PER improvement with DF over NC in every topology. However, this effect will cause error floors at high average SNR, where packet losses due to rare CFO estimation errors begin to dominate.

### 6.5.2 Dynamic Range of Channel Frequency Response

In each PER and BER plot for all schemes in the two triangular topologies there is a clear floor in performance when using the TGn B channel model. The floor’s value varies with modulation rates and cooperative schemes, but the general shape is unmistakable. The same error floors are not apparent in the flat fading results which, except for the channel model, use identical experimental parameters. It is clear that some difference between the TGn A (flat fading) and TGn B channels is dominating performance at high SNR.

The only difference between the TGn A and B channels is the power delay profile of the models. The TGn A model has a single tap, resulting in a flat frequency response for all channel states, while TGn B has 9 taps spread over 80 nsec. As the instantaneous tap values vary with time, they generate frequency selective fading. In order to attribute packet errors (the different result between experiments) to fre-

quency selectivity (the only parameter difference between experiments), we need a way to associate packet errors with instantaneous channel states.

Unfortunately there is no way to extract instantaneous channel response from the emulator in real-time. It is possible to pause the emulator and see the instantaneous channel impulse response (time domain dual of the frequency response). However this display is only shown in the Azimuth GUI and not made available via the emulator API. Further, pausing and restarting the emulator requires  $\approx 5$  seconds, even when done via the API. To do this for each packet in an experiment with hundreds of thousands of transmissions is infeasible.

Instead, we created a framework for recording the OFDM receiver's channel estimates in real-time. This framework utilizes an extension to the PHY which stores a copy of the receiver's channel estimates in a memory-mapped buffer. When a packet is received, the node transmits an Ethernet packet containing the channel estimates, received state (good/bad) and other PHY data (RSSI, AGC gain selection and CFO estimates)<sup>1</sup>. This packet is received by a helper application running on a PC which writes it to a log file. The helper application is controlled via WARPnet so that the log files can also record the experimental parameters per packet.

When the experiment is complete, we load the log files into MATLAB for analysis. The logs contain frequency domain channel estimates per subcarrier for every packet received during the experiment. We start by calculating the dynamic range of the channel frequency responses across subcarriers. We define dynamic range as the difference in powers between the strongest and weakest subcarrier estimates. For NC packets, we consider only the source-destination estimates. For AF/DF packets, we

---

<sup>1</sup>This is the same framework we use for all of the experiments for CFO estimator characterization presented in Chapter 3.

first sum the powers of the estimates in each subcarrier before calculating the range. Finally, we group observations by outcome (good payload, bad payload or bad header) and channel dynamic range.

We use this framework for an experiment of the co-located source/relay topology. The results are shown in Figure 6.31. Each of the nine plots presents data from a cooperative scheme (NC/DF/AF) and attenuation (12/4/0 dB). In each plot, the X-axis is channel dynamic range. This is a linear scale where small values represent flatter channels. The Y-axis is probability. Each plot has three curves, one for each reception outcome. Each data point represents the probability of a packet being received with a given outcome and channel dynamic range.

A few trends are clear. First, for every scheme and SNR the good payload curve dominates at low dynamic ranges; flatter channels result in fewer errors. The opposite holds as well: higher dynamic ranges lead to more packet errors, with the worst outcomes (bad header) dominating at the highest dynamic ranges.

These results align perfectly with the PER/BER floors we observed earlier. In our experiments, the probability of a deep channel null (i.e. high dynamic range) is independent of average SNR. The emulator generates and applies its random tap gains before it applies the output attenuation. At low SNR, additive noise dominates the BER/PER performance. As SNR increases, the PER/BER improve until reaching a floor defined the probability of a deep null in the channel model. Beyond this point, higher SNRs will not improve performance, as the fixed probability of a channel null dominates.

To understand how a null can cause errors at high SNRs, recall the flow of a received waveform into our receiver as illustrated in Figure 6.32. The MAX2829 downconverts the RF waveform to baseband, applying gain in two stages as directed

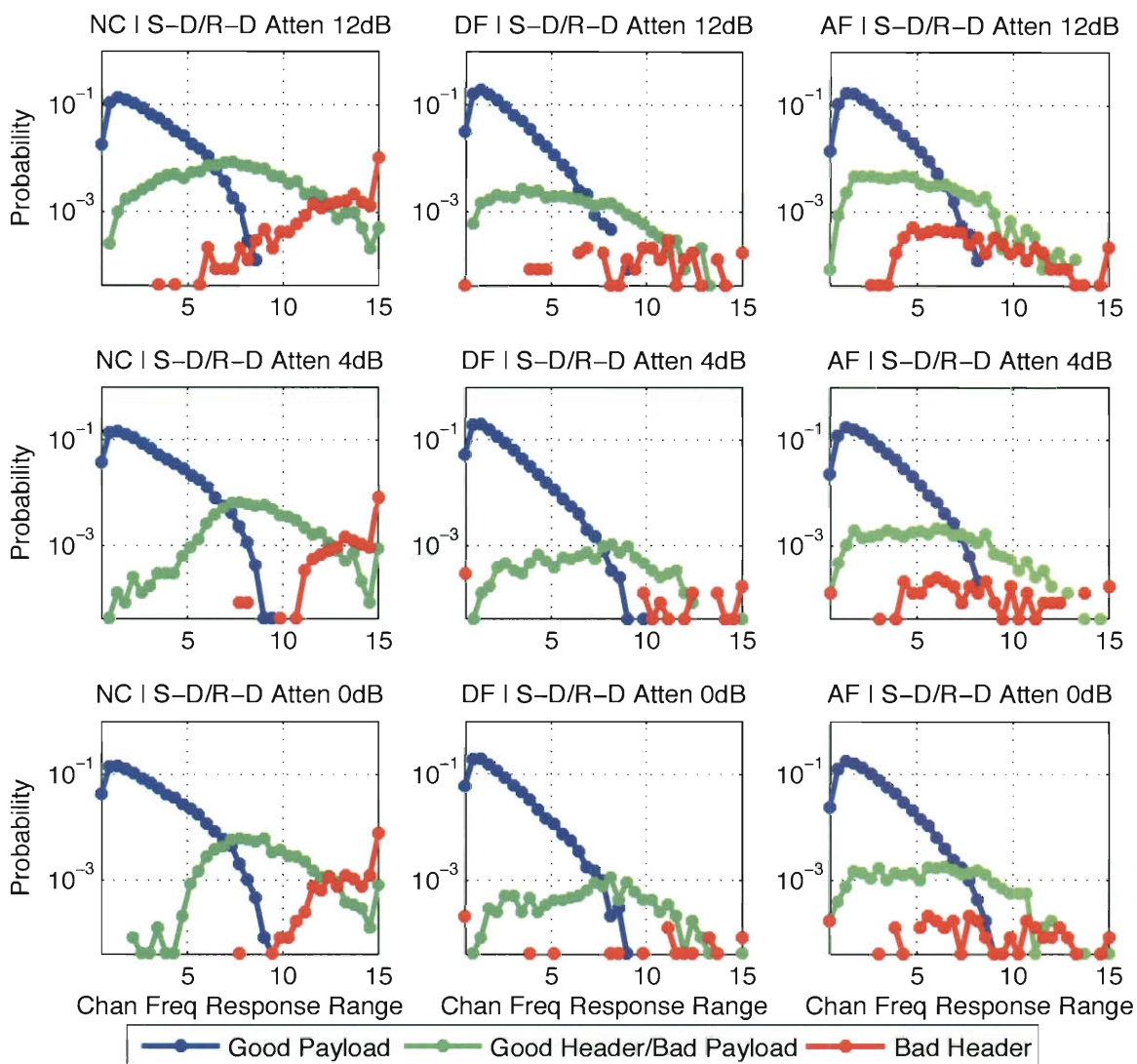


Figure 6.31 : Packet success and error rates for NC/DF/AF as a function of channel frequency response dynamic range, for full length 16-QAM payloads in the co-located source/relay topology

by the AGC core. The AGC selects gains so that the MAX2829 baseband analog outputs fill the dynamic range of the ADCs, whose digital outputs are fed into the OFDM receiver.

Our OFDM receiver is implemented entirely with fixed point arithmetic, a stan-

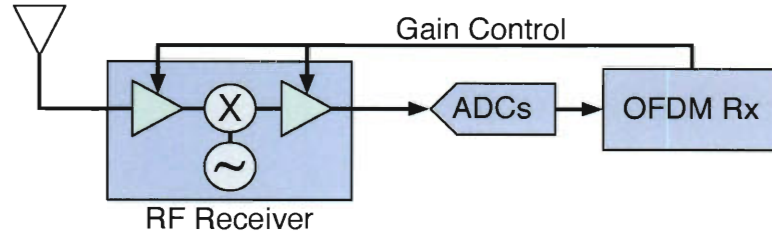


Figure 6.32 : Block diagram of the signal chain from the antenna to OFDM receiver. The RF receiver in the MAX2829 applies gain in two stages, with gains selected by the OFDM receiver's AGC block.

standard practice for FPGA designs. For our design flow, this is a requirement; Xilinx System Generator only supports fixed-point data types for hardware implementation. One limitation of using fixed vs. floating point processing is the smaller range of values which can be represented in a given datatype.

The combination of quantization, due to sampling at the ADC, and fixed-point processing in the FPGA cause the error floor difference between our frequency flat and selective fading results is tied directly to this range limitation.

Most of the processing in an OFDM receiver takes place in the frequency domain, after the received time domain waveform is passed through an FFT. It is critical that the arithmetic blocks inside the FFT not overflow. Internally, the FFT core wraps on overflow, which corrupts the full transform in progress. We set the target amplitude at the output of the AGC to fill (but not exceed) the numerical range of the FFT core I/O.

In frequency selective fading channels, the channel response will vary in magnitude across OFDM subcarriers. In the OFDM receiver, this means the magnitude of the values output from the FFT will vary according to the frequency response of the channel. In our OFDM receiver, channel estimates are calculated per-subcarrier, and symbols are equalized by inverting the channel (i.e. zero-forcing). For channels with



large amplitude variations across frequency, the channel estimates for the “weakest” channels will be very small, imprecise values. When used by the equalizer this imprecision is exaggerated, increasing the likelihood of symbol errors. Thus, in our receiver, we expect the overall performance to be dominated by the weakest subcarriers. Since the probability of a weak subcarrier (relative to other subcarriers) is independent of the average SNR, the probability of errors due to weak subcarriers is capped and, at high SNR, will dominate.

We designed an experiment to confirm this expectation. This experiment seeks to test whether a channel frequency response with large dynamic range (peak-to-peak amplitude) causes packet errors, independent of average SNR. This test uses a feature of the Azimuth emulator which allows it to “freeze” its current state. In this mode, the emulator continues applying the latest channel coefficients, but stops updating the coefficients, essentially extending its coherence time indefinitely.

In order to isolate a channel realization with high dynamic range, we used two WARP nodes connected to the emulator. The channel model was set to TGn D with the minimum velocity (0.012 km/h). This setup results in very slow changes to channel coefficients, with coherence times of multiple seconds. We then started the emulator and a one-antenna Alamouti link between the WARP nodes. By monitoring the PHY’s bad header output, we could watch for channel conditions resulting in high packet error rates. It did not take long for the emulator to impose a channel with a deep null, indicated by consistent packet errors at the destination. We froze the emulator state but allowed the nodes to continue running. Using WARPnet, we captured channel estimates for every detected packet, and checksum statuses for every detected header and payload.

Figure 6.33 shows the captured channel estimates from this test. The X-axis is

subcarrier index; only the 52 occupied subcarriers (indices  $[-26,-1]$  and  $[1,26]$ ) generate estimates. The Y-axis is the magnitude of the complex channel coefficient on a log scale. Each trace (there are thousands) is the frequency domain channel estimate extracted from a single received packet. Three distinct groupings are apparent. These are the result of the AGC choosing one of three gains, with each choice separated by 2 dB. This  $\pm 1$  gain step is standard behavior for the AGC.

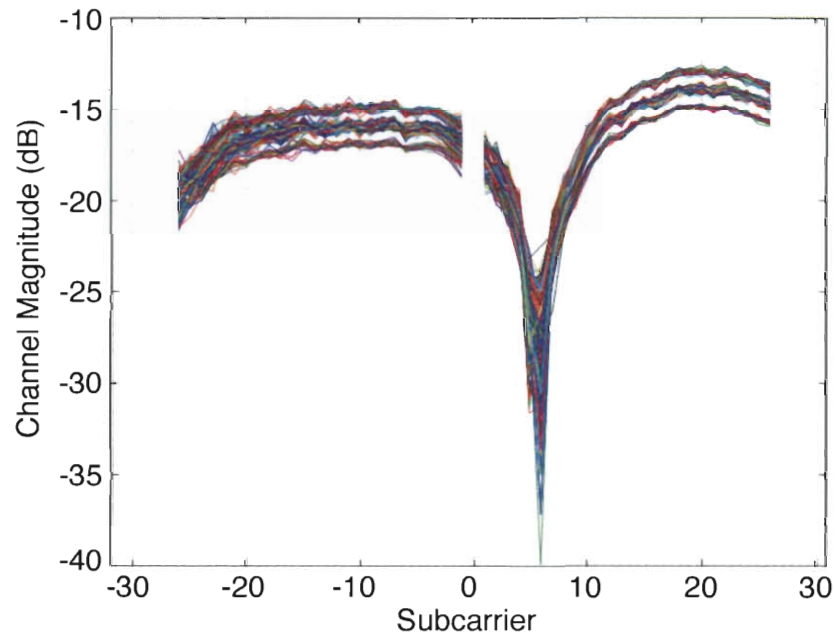


Figure 6.33 : Channel estimates for high-dynamic range channel response.

After isolating an appropriate channel response, we then swept attenuation (average SNR) between the source and destination. From the receiver's perspective, it must apply higher gain in the AGC for higher attenuations, but the impact of the channel response should be consistent across average SNRs. The test confirmed this. The PER was 100% for all average SNRs when this channel response was applied.

### 6.5.3 Bit Error Densities

Consider the PER plots for the co-located source/relay topology in Figures 6.3- 6.4. In general, the DF and AF curves show significant PER improvement over NC at nearly every point. The one deviation from this general observation is for PER of 16-QAM payloads in a flat fading channel (Figure 6.4). Notice that at the two highest SNRs (furthest to the right) the AF curve shows worse packet error rates than NC. Compare these points to the corresponding BER values (Figure 6.6). Here, both cooperative schemes significantly outperform non-cooperative. This disparity between PER and BER requires deeper investigation.

We start by analyzing curves corresponding to each kind of packet error. Recall from Section 4.1.5 that for every transmission, the OFDM receiver terminates in one of four states: no detection, bad header, bad payload or good payload. The first three count as packet errors and our experiment design allows each to be tabulated separately. Figure 6.34 presents four curves: the overall PER (a) and the three contributing sources of packet errors (b)-(d).

It is easy to spot which type of packet error dominates the PER in various regions. At low SNR (to the left), all three schemes are dominated by missed detections (d) and bad headers (c). In this region only a few packets end in the bad payload state, as the receiver terminates before attempting to decode the payloads. As the SNR increases (moving right), more bad payload events occur. At the mid-to-high SNRs the bad payload curve (b) is nearly identical to the overall PER, indicating in this region payload bit errors are the primary source of packet errors.

This observation leads us to dig deeper into the distribution of bit errors at the highest SNRs. To this end, we extend our experimental setup with a new bit error calculator. Recall from Section 5 that we use a C program running on a PC to

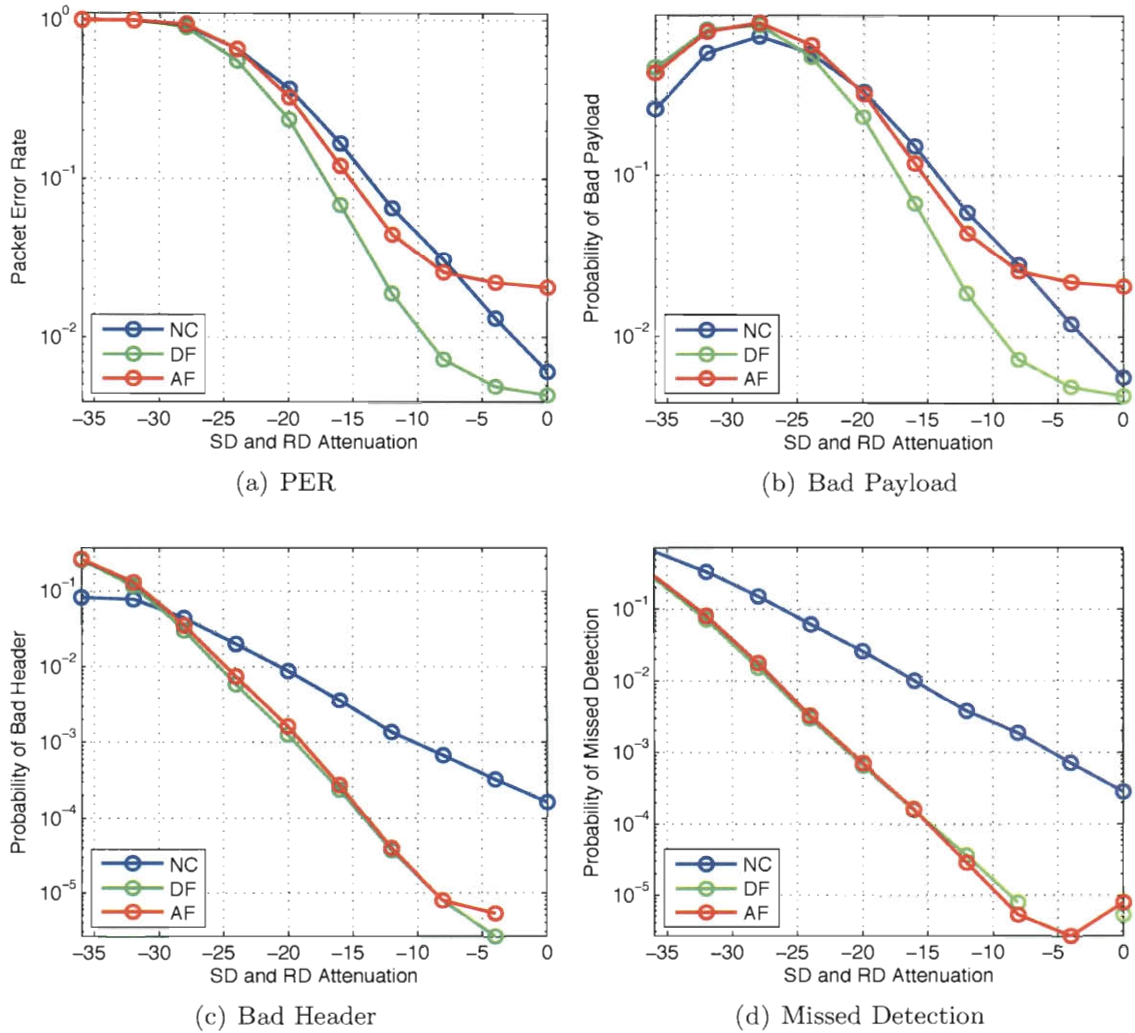


Figure 6.34 : Packet error rate (a) broken down into contributing errors (b)-(d) for full length 16-QAM payloads in the co-located source/relay topology.

calculate the BER per packet transmission. For this investigation, we use an extended version of this program which both calculates the BER and logs the index of each bit error in every packet received. The extended program still integrates with WARPnet, so we can attribute each bit error to a given combination of experimental parameters.

Using this new experimental setup, we re-test the channel/modulation/topology

combination above. We run shorter trials here, as the bit error logs grow quickly. From these logs (864 MB in total) we can extract the distribution of the number of bit errors per packet, as shown in Figure 6.35.

We illustrate this distribution two ways. The first is a standard histogram (a), with each bin showing the probability of a transmission being received with precisely that number of bit errors for each scheme (NC/DF/AF). The first bin corresponds to zero errors; the inset shows the tops of these bars. The nearly identical values align well with our overall low probability of packet errors for all three schemes. The last bin represents all packets received with 20 or more errors (essentially the sum of all bins past 19 in a full histogram).

We can make a few key observations here. First, in the rightmost bin, the distributions of errors among schemes differ significantly. A non-cooperative transmission is much more likely to experience many bit errors than with either cooperative scheme. This maps well to our intuition about diversity; a deep fade will cause many bit errors, but simultaneous deep fades on two independent channels are unlikely. Second, notice the difference between schemes in the lower bins. Here, AF shows a much higher probability of delivering packets with only a few bit errors than NC or DF.

This is especially clear in Figure 6.35(b), which shows the same distribution plotted against cumulative bit error counts per packet. Here, each X-axis value is a bit error count; each data point is the probability of a packet being received with at least that many errors. Thus, mass to the right indicates a higher probability of packets with many errors; mass to the left maps to higher probabilities of packets with just a few errors. Notice the AF and NC curves, which cross for bit error counts beyond 1. These results clearly demonstrate that the higher PER with AF at high SNR is due to bad packets which are almost good. Amplify and forward succeeds in filling

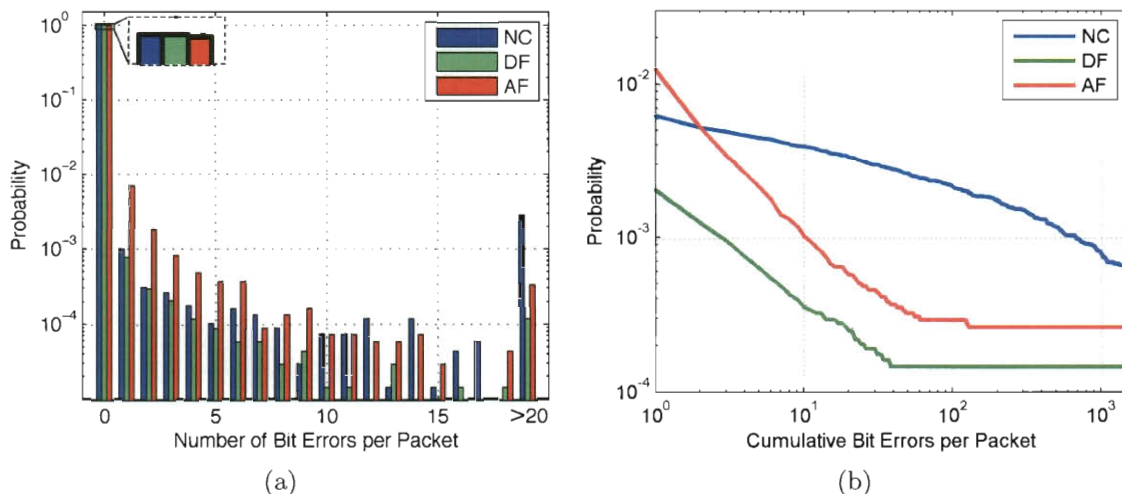


Figure 6.35 : Distribution (a) and cumulative distributions of number of bit errors per packet for co-located source/destination/relay topology and full length payloads modulated at 16-QAM.

in deep fades, but it does so with “noisy” power.

The bit error distribution curves above present data only for the highest SNR point in the co-located source/relay topology. Our experiment actually generates bit error logs for every point. An intuitive way to view all these results is as pseudo-PER curves, which represent the numbers of packets received with more than a given number of bit errors. This is equivalent to re-defining a successful packet’s threshold for bit errors to something higher than zero.

Figure 6.36 shows these curves for four bit error thresholds. Plot (a) is normal PER, where any bit error causes a packet error; plots (b)-(d) show PER for thresholds of 1, 2 and 20 bit errors per packet. Notice how the AF curve moves below the NC curve very quickly, while the NC curve improves only slightly. This reaffirms our observations above, that the PER for AF at high SNR is dominated by packets that are “almost good.”

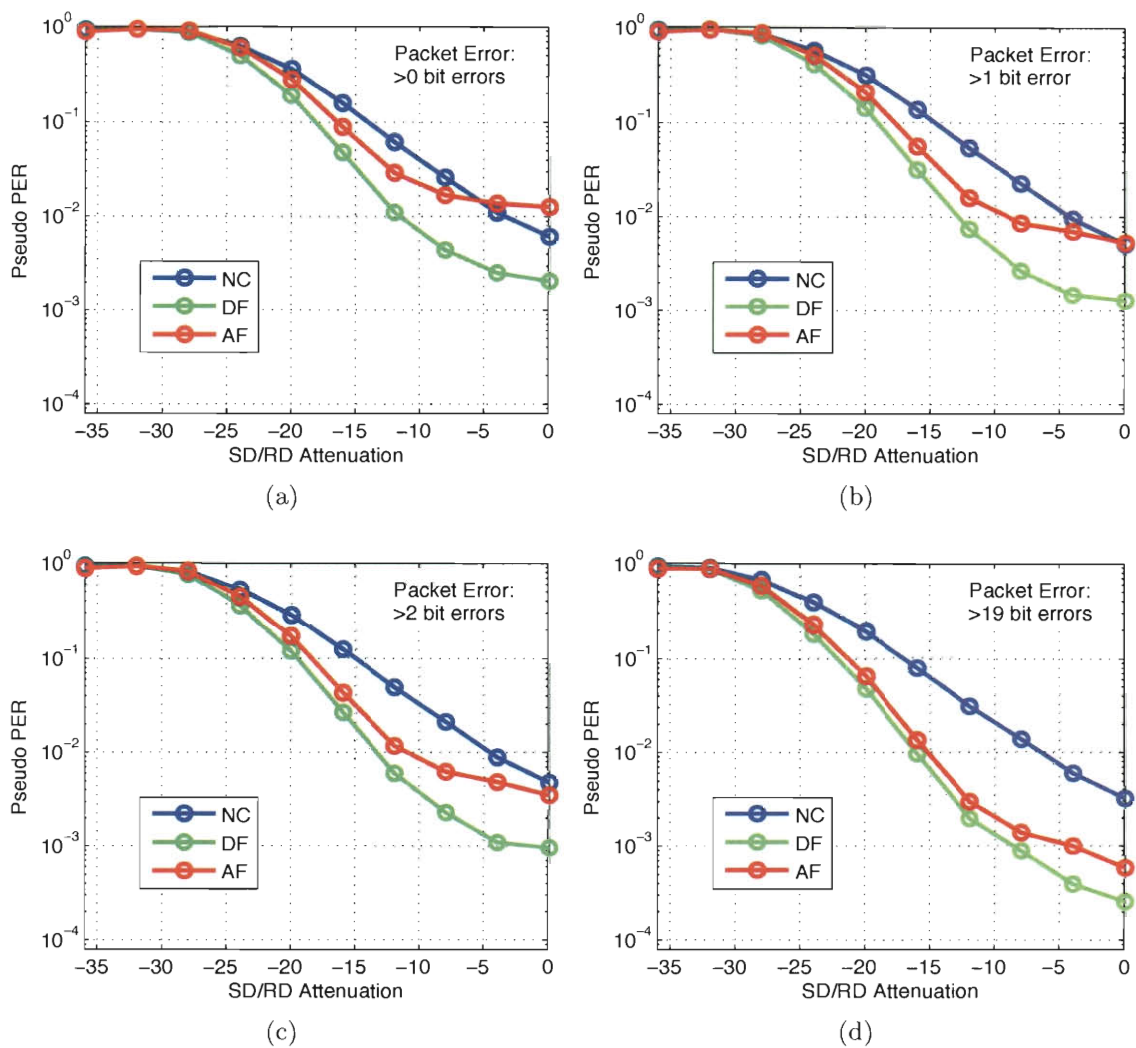


Figure 6.36 : Pseudo packet error rates for co-located source/relay topology with 1416 byte, QPSK modulated payloads, for four different bit error thresholds.

## Chapter 7

### Future Work

#### 7.1 Physical Layer Cooperation in a Network

Our work thus far has focused on a detailed performance characterization of our cooperative physical layer implementation. Our primary metrics (packet and bit error rates) are designed to capture the expected outcome of a packet transmission under various conditions. But these metrics and, our characterization in general, do not consider under what conditions a node should transmit. This determination is generally part of the Medium Access Control (MAC) layer, a rigorous study of which falls outside the scope of this work. However, as discussed below, our PHY design has already enabled some early results in studying ways to exploit physical layer cooperation at the MAC layer.

##### 7.1.1 Early MAC Results

In collaboration with fellow graduate student Chris Hunter, we designed, implemented and evaluated the Distribution On-demand Cooperation (DOC) MAC protocol.

The DOC protocol seeks to exploit physical layer cooperation to improve the reliability of MAC re-transmissions. Specifically, it aims to use cooperative re-transmissions to combat packet losses due to channel effects (as opposed to losses due to contention). The protocol is fully distributed, designed to coordinate the behavior of nodes with no central synchronization source.



The DOC protocol employs an explicit negative acknowledgement (NACK) packet, transmitted by a destination whenever channel conditions prevent the successful decoding of a packet payload. The NACK serves two purposes. First, it communicates the destination’s MAC state to the source, similar in effect to a timeout (missing ACK) in CSMA. Second, the NACK functions as a trigger for a potentially cooperative re-transmission. If the source node and a participating relay receive the NACK they will both re-transmit the original packet using either AF or DF.

The NACK serves both as the re-transmission trigger and as a means for the cooperating nodes to establish sample-level synchronization for their transmissions. Our DOC implementation employs the PHY’s auto-response subsystem discussed in Section 4.2 at both the source and relay nodes. This design provides NACK-to-re-transmission turnarounds at the source and relay that are fast and, more importantly, uniform.

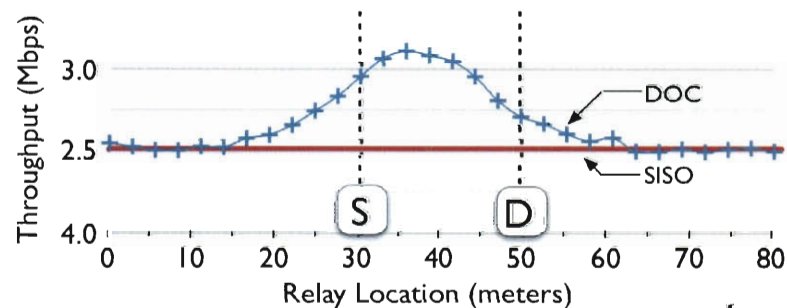


Figure 7.1 : Throughput improvement using DOC in a three node network, tested with a fixed source/destination and a relay at various points along the line connecting them.

Our current DOC implementation is built on a previous generation of the PHY which supports only amplify and forward cooperation. Even with just AF DOC demonstrates significant performance gains. Our approach to the initial evaluation of DOC uses techniques very similar to those discussed in Section 5. Three WARP

nodes are inter-connected via the Azimuth channel emulator and are controlled by the WARPnet framework.

The primary metric in our evaluation is throughput, measured under the conditions of a fully-backlogged source node with a dedicated relay which transmits only when cooperating with the source on NACK-triggered re-transmissions. Figure 7.1 shows results from one topology, demonstrating a clear throughput improvement when using DOC.

A complete discussion of the DOC protocol and additional experimental results are provided in [46].

### 7.1.2 Rate Adaptation

In a wireless stack built on a physical layer transceiver capable of multiple transmission rates, higher network layers must select a rate for every transmission. For example, IEEE 802.11a specifies eight physical layer data rates, realized by combinations of four modulation rates and three coding rates. The standard does not specify a rate selection algorithm; the link layer is free to choose any supported rate per packet. Various rate adaptation algorithms have been proposed [47, 48]; a number of even been evaluated experimentally using an earlier generation of the WARP OFDM design (before cooperation was implemented) [49].

In a non-cooperative transceiver each data rate has some measurable performance under given propagation conditions. The link layer uses knowledge of the available rate/reliability options to make its per-packet rate decisions. With physical layer cooperation, however, different cooperative schemes can provide higher performance for a given data rate, but at the cost of employing a relay node. As discussed in Section 5.6, the actual cost of delivering a packet to the relay depends on the MAC

protocol itself. Thus, a cooperation-aware rate adaptation algorithm would be tightly coupled to the link layer protocol.

For example, consider the packet error rate curves in Figure 7.2. These plots show the PER results for the co-located source/relay topology in frequency flat (a) and frequency selective (b) fading. These curves are drawn from the same experiment as Figures 6.7-6.8). Here we show curves for just non-cooperative (NC) and DF schemes, but with results for both QPSK and 16-QAM modulation on the same axes.

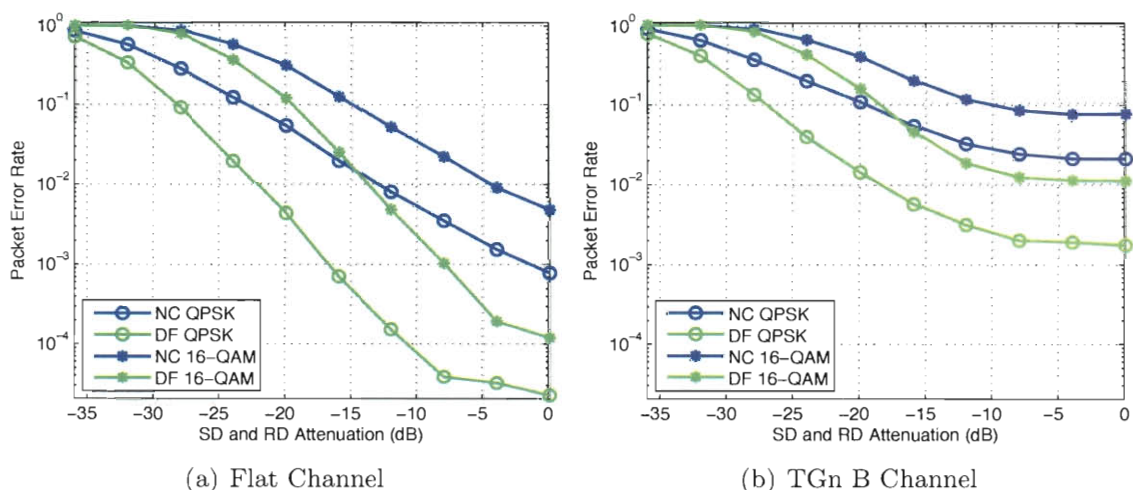


Figure 7.2 : Packet error rates for NC and DF schemes in co-located source/relay topology with 692 byte payloads modulated with QPSK and 16-QAM.

Notice that at high SNR in both fading models, the performance of DF with 16-QAM exceeds that of NC with QPSK. This presents an interesting example for rate adaptation in a network of cooperative nodes. Imagine nodes which use a MAC protocol similar to that of 802.11, but can employ physical layer cooperation for MAC re-transmissions (as in the DOC implementation discussed above). If a source node unsuccessfully attempts an initial transmission at 16-QAM, it could choose from four options: (a) re-transmit at 16-QAM, (b) fall back and re-transmit with QPSK,

(c) trigger a cooperative re-transmission using a nearby DF relay at QPSK or (d) at 16-QAM. Each option has an associated cost; re-transmission with QPSK would require more time, increasing the temporal “footprint” of the transmission. A cooperative re-transmission similarly extends the footprint spatially. The cooperation-aware link layer rate adaptation algorithm would need to balance these temporal and spatial costs with the expected performance of each option, using knowledge of the propagation environment and data like that in Figure 7.2. We expect our physical layer design, experimental methodologies and performance measurements will prove useful to researchers undertaking the design of such algorithms.

### 7.1.3 Open Questions

Our implementation of DOC is just a first step in understanding the potential benefits of employing physical layer cooperation in a real network. Future work will need to address many open questions. For example, our work on DOC demonstrates the potential gains with cooperation but does not address the costs. The use of a relay expands the footprint of a given packet transmission. In a multi-flow network, it is easy to envision topologies where a relay’s transmission helps one flow but interferes with many others. Understanding this tradeoff and designing MAC protocols to optimize it will require considerable effort.

Our cooperative physical layer is designed explicitly to enable novel MAC implementations. The WARPnet framework already supports arbitrarily large networks and our scripts could easily be adapted for over-the-air experiments with many nodes. We anticipate these results from our work will help facilitate ongoing efforts by others to study the MAC level implications of physical layer cooperation.

## 7.2 Transceiver Extensions

### 7.2.1 Temporal Combining

As discussed in Chapter 2, our cooperative implementation operates in two time slots. The source node transmits its packet to the relay in the first slot, then the source and relay cooperatively transmit the packet in the second. In this implementation the destination only receives transmissions in the second slot. As discussed in Chapter 5, our experiments focus on spatial combining in order to isolate and quantify the performance gains possible with simultaneous transmissions by cooperating nodes.

Additional performance improvements would be possible if the destination operated in both time slots. The second time slot could be avoided altogether in cases where the destination successfully receives the source's transmission in the first (this is the goal in the DOC protocol, discussed in Section 7.1.1 above). In cases where the destination cannot decode the first transmission it could still perform some sort of combining of its receptions from both slots to improve its chance of decoding the payload.

A few other cooperative implementations use temporal combining. For example, the systems described in [7, 10] only implement temporal combining; neither employs spatial combining via simultaneous source/relay transmissions.

Adding combining across time slots would clearly provide a performance gain in our design, but it poses a number of implementation challenges. The combining process would likely be implemented in the frequency domain as part of the equalization and Alamouti combining blocks. This approach would require that the receiver record the output of its equalizer (i.e. soft symbol values) in the first time slot so they could be combined symbol-by-symbol during a subsequent reception. It would also need to

store some metric of reliability per subcarrier, like a channel coefficient magnitude. The memory requirements here would be significant, comparable in size to the waveform buffer used with AF. There would also be synchronization challenges in assuring a given reception matches a previous one and should be combined with the recorded symbols and metrics. At a minimum, this synchronization would require new logic in the receiver to match header fields (source/destination addresses, sequence number, etc.) across receptions.

Each of these challenges is tractable, but non-trivial. It would be important to study the potential benefit of using spatial and temporal combining together before extending our design to support both.

### 7.2.2 Error Correcting Codes

Another useful addition to our physical layer design would be an error correcting code. Recognizing the substantial effort required to add coding to our design, we judged an uncoded PHY sufficient for our experiments. However, channel coding would clearly improve performance, as it does in all modern wireless standards. Well understood architectures exist for FPGA implementations of common codes (convolutional codes, for example). Extending our PHY with such a code is certainly tractable, but would require care to preserve the PHY features necessary for cooperation (auto-responders, frequency domain CFO estimation, etc.).

We can offer one observation relevant to coding from our results with an uncoded PHY. Recall from Section 6.5.3 our description of a “pseudo-PER” metric, based on analyses of the distribution of bit error densities in packets employing various cooperative schemes. These results demonstrated a stark difference in bit error patterns between non-cooperative and cooperative receptions in some topologies. The differ-

ence was most pronounced for AF receptions, where in certain topologies the majority of packet errors were the result of just a few bit errors (often just one).

The three pseudo-PER plots in Figure 6.36(b)-(d) can be interpreted as the PER which would result if the PHY employed hypothetical codes which could guarantee correction of 1, 2 or 20 bit errors per packet. A code which could correct 20 errors, for example, would improve the PER of all three schemes, but would provide an especially large improvement for AF and DF (see Figure 6.36(d)).

Of course, most codes cannot assure correction of  $N$  errors. However, our work would still provide a useful starting point for studying the impact of error correcting codes in a real cooperative system. We developed the bit error logging and analysis tools discussed in Section 6.5.3 to better understand the performance of our PHY. This same framework would be very useful to researchers seeking to understand the expected results of employing various coding techniques in a cooperative system.

### 7.2.3 Full Duplex

A final promising extension to our implementation would be an exploration of physical layer cooperation among full duplex wireless nodes. Most wireless systems which transmit and receive in the same band employ half duplex communication. This is due to the very high ratio of received powers between transmissions from nearby (local to the node) and remote (at another node) antennas.

However, recent results [50, 51] have demonstrated that full duplex links are feasible using commodity hardware. One of these projects is here at Rice, led by fellow graduate student Melissa Duarte, and is using WARP for the experimental evaluation of full duplex techniques. In this design, an additional WARP Radio Board is used to generate an RF signal designed to cancel the node's own transmission. This signal

is combined at RF with the signal from the receive antenna and the sum is fed to another Radio Board for downconversion. Careful design of the cancellation signal can reduce the ratio of received powers significantly, making it possible to fully cancel local interference digitally after downconversion. Early results are promising, demonstrating that a combination of RF and digital baseband cancellation can enable full duplex links in a wide range of practical topologies.

An extension of this full duplex work to a real-time, wideband system poses some very interesting opportunities for physical layer cooperation. The two time slot design of our cooperative implementation is rooted in the inability of a relay node to receive and transmit simultaneously. In theory, a full duplex relay could initiate a cooperative transmission while the source node's initial transmission is ongoing.

Realizing such a relay in practice poses a number of significant implementation challenges. For example, there is non-zero latency through the signal chain of RF downconversion→ADC→FPGA→DAC→RF upconversion. Even with no processing in the FPGA, this latency is hundreds of nanoseconds, equivalent to tens of sample periods at 10 MHz bandwidth. Occupying a substantial fraction of an OFDM symbol, this delay would preclude use of the distributed Alamouti code as currently implemented in our transceiver.

Another challenge involves calibration of the effective channel response between transmit and receive antennas for use in constructing the RF cancellation signal. Current full-duplex implementations operate with a narrow signal bandwidth. The calibration and cancellation schemes would need to be adapted for use with a wideband (possibly frequency selective) channel between the Tx/Rx antennas.

A final complication arises when considering how to adapt various cooperative schemes (AF and DF, for example) to a full duplex relay. A full duplex AF relay



is straightforward to describe. However, an implementation which requires active generation of an RF cancellation signal requires far more processing than simply buffering and retransmitting a waveform. If the cancellation signal is constructed in the frequency domain (the natural place in an OFDM system), even AF relaying would require a modified OFDM transceiver. If constructed in the time domain, the relay would need channel impulse response estimation and inversion systems not typically used in OFDM. A DF relay would have similar complications, having to generate two waveforms per reception (one for transmission, one for cancellation) via a single OFDM transmitter. Further, the cancellation waveform would require multiplying each subcarrier by a correction value derived during the Tx/Rx calibration process. Such a transmitter bears more similarity to beamforming than the Alamouti STBC and would require modifications to a large fraction of our OFDM transmitter implementation.

These are all very interesting problems whose solutions could have significant impact. None strike us as insurmountable, but each will require careful analysis, design and experimental verification to realize a real-time, full duplex cooperative system. We anticipate our current FPGA design flow, transceiver implementation and experimental framework will prove useful as these efforts are undertaken.

## Appendix A

### Additional Plots for Full Transceiver Characterization

This appendix includes additional plots from our experiments. The overall PER and BER are presented in Chapter 6. The plots below show the probabilities of each kind of packet error, which when combined form the overall PER.

Also included below are plots indicating the probability of relay participation at every point in the experimental topologies. These curves are all intuitive, indicating higher relay participation in regimes where the relay can more reliably decode transmissions from the source.

## A.1 Co-located Source/Relay Topology

### A.1.1 1412 Byte Payloads

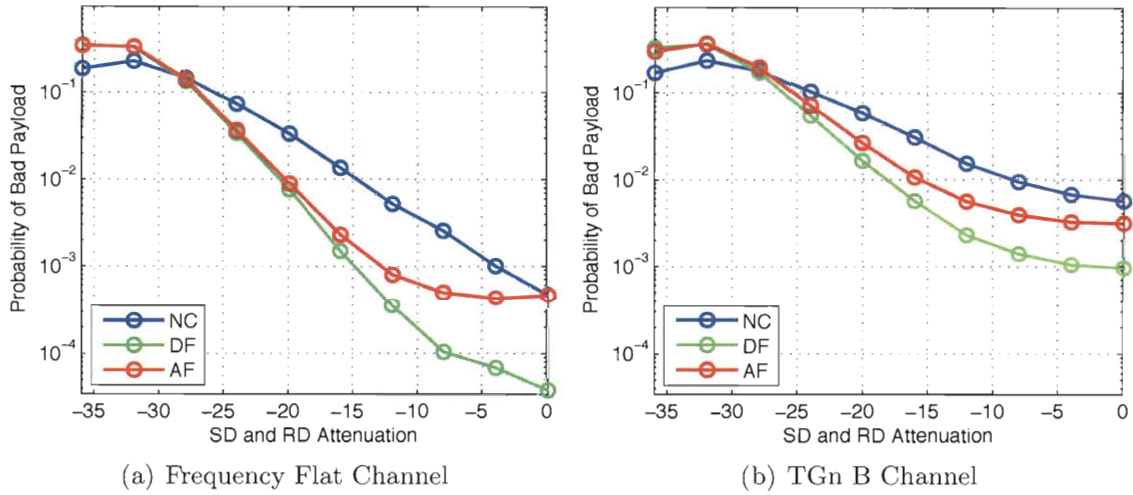


Figure A.1 : Probability of destination receiving packets with good headers but bad payloads for QPSK modulation.

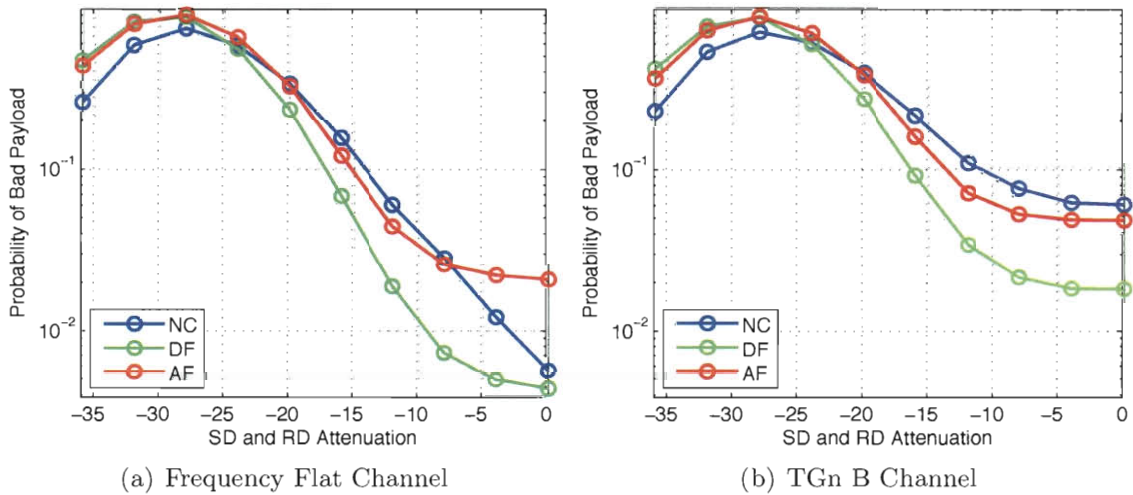


Figure A.2 : Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation.

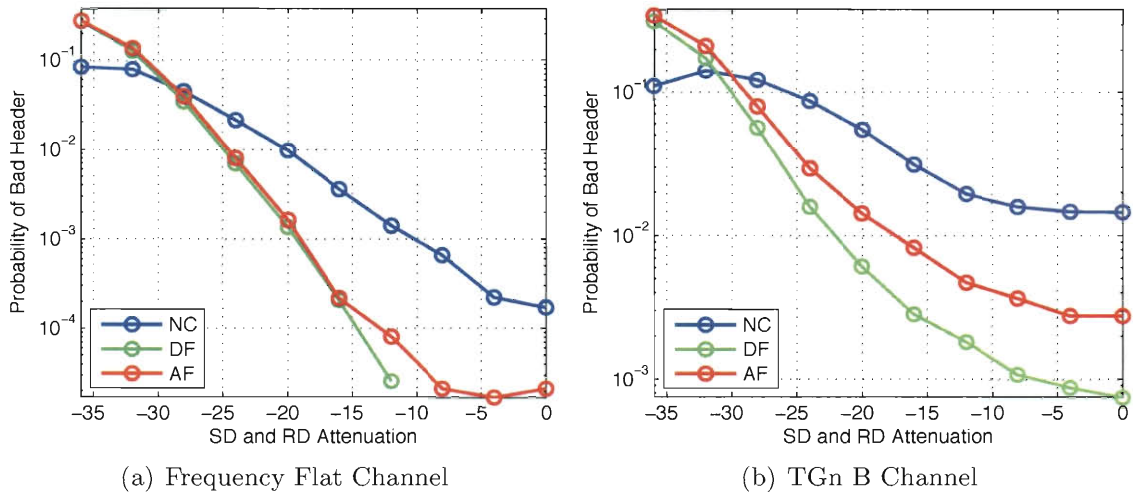


Figure A.3 : Probability of destination receiving packets with bad headers for QPSK modulation.

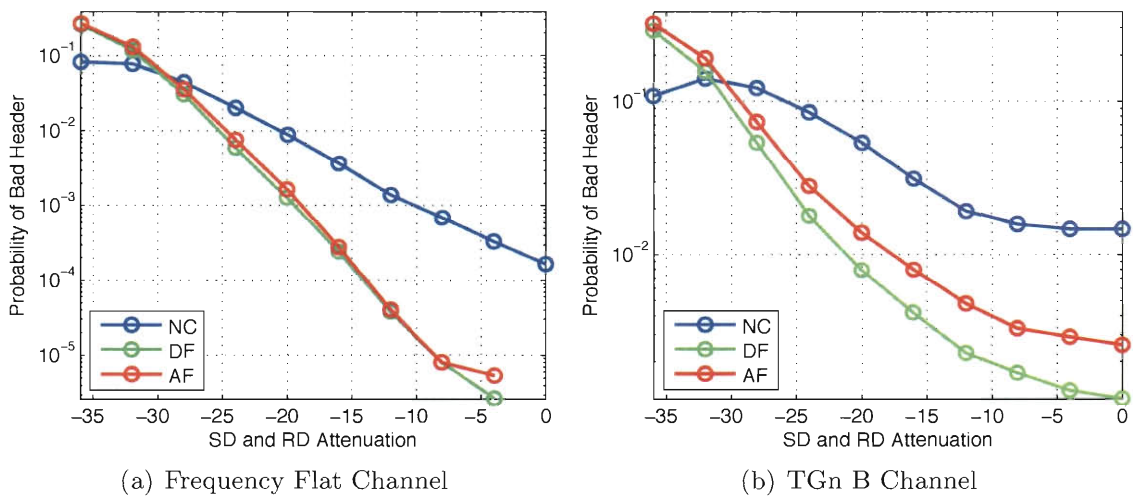


Figure A.4 : Probability of destination receiving packets with bad headers for 16-QAM modulation.

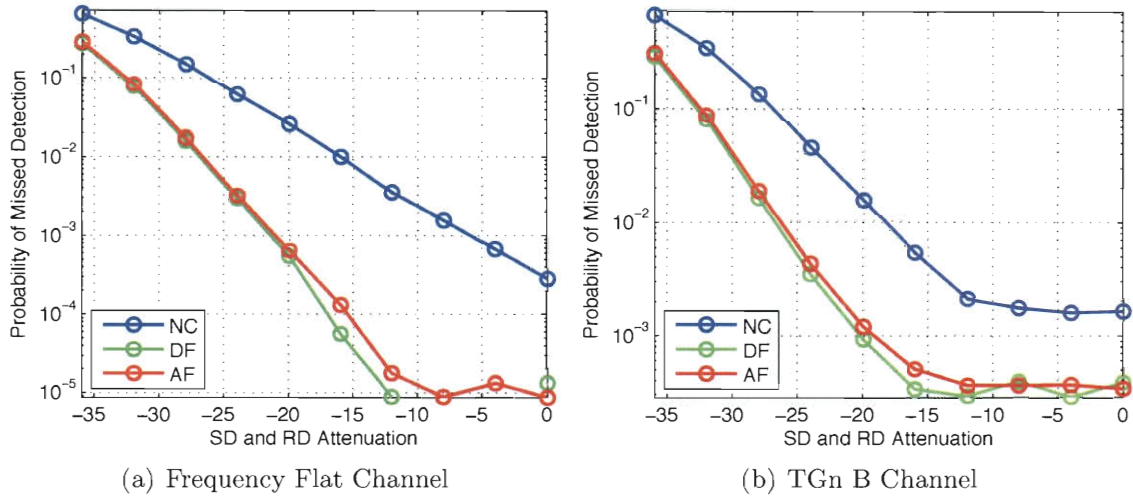


Figure A.5 : Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads.

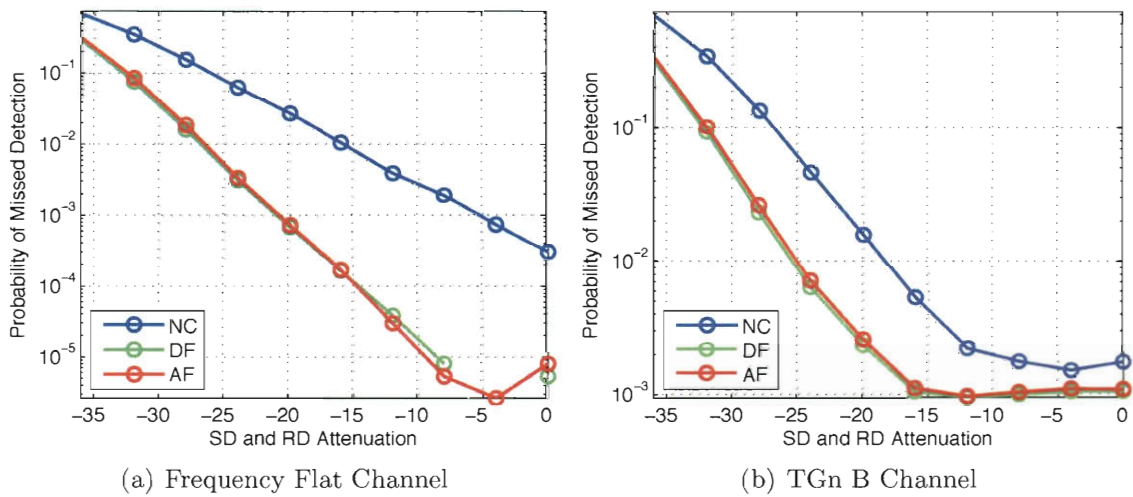


Figure A.6 : Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads.

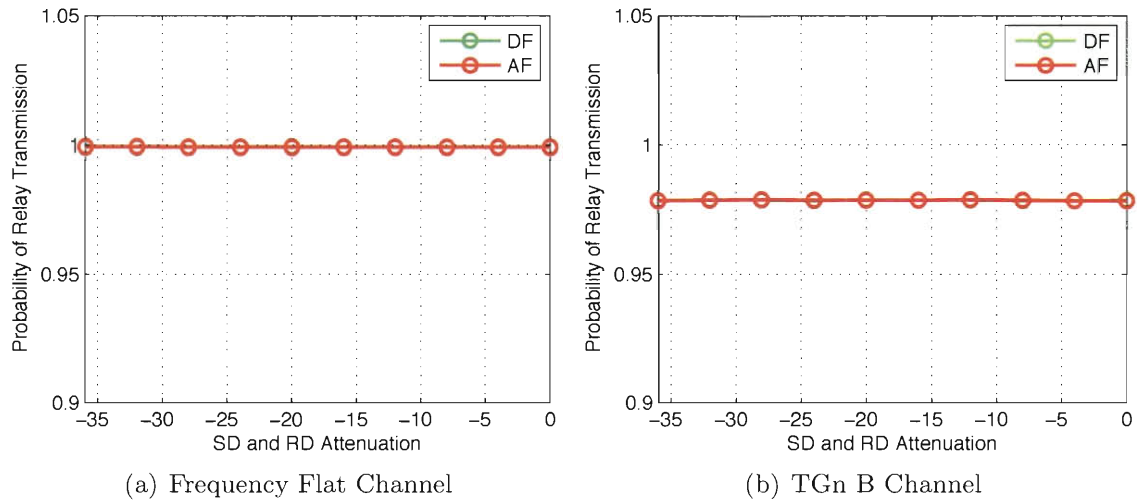


Figure A.7 : Probability of relay transmitting for QPSK payloads.

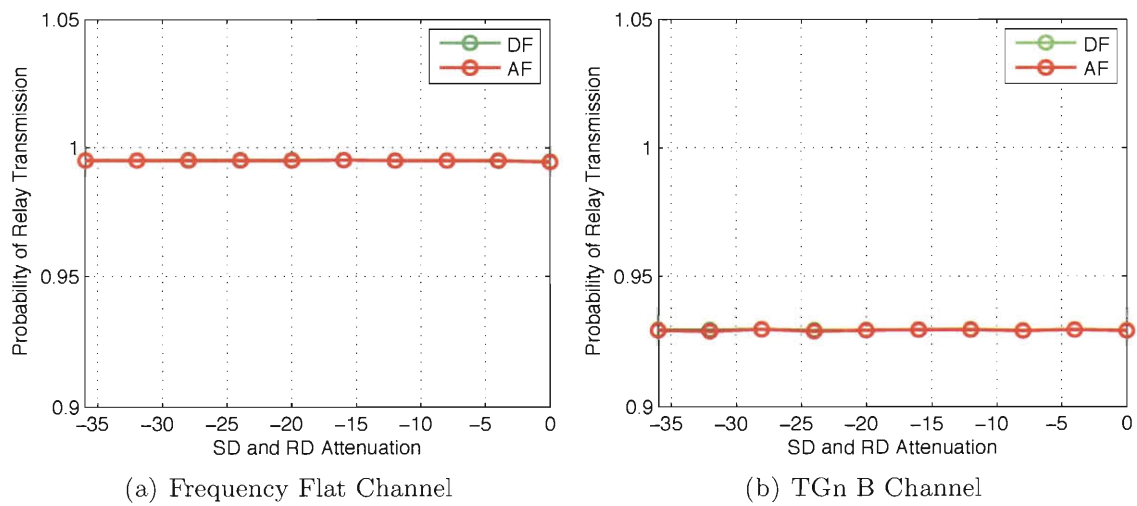


Figure A.8 : Probability of relay transmitting for 16-QAM payloads.

### A.1.2 692 Byte Payloads

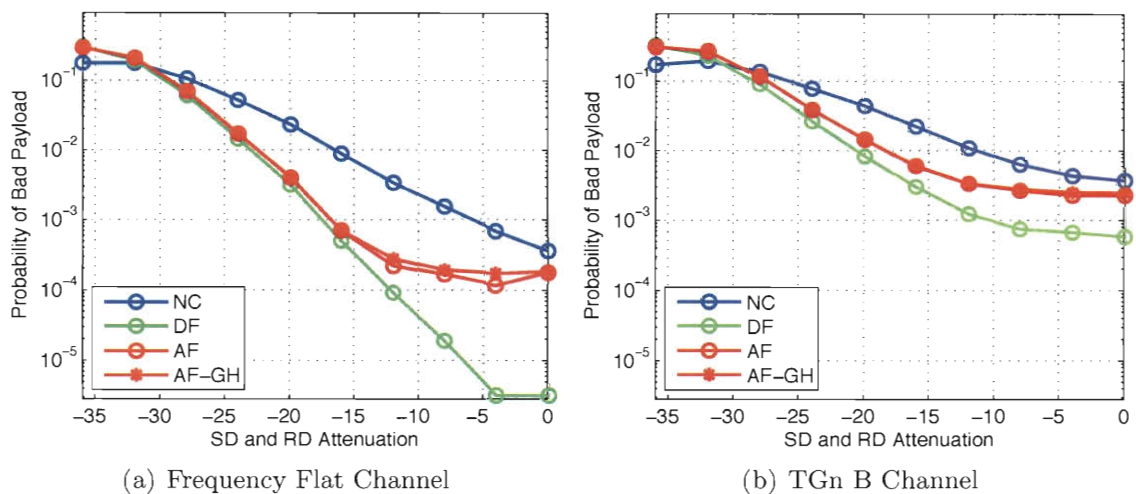


Figure A.9 : Probability of destination receiving packets with good headers but bad payloads for QPSK modulation.

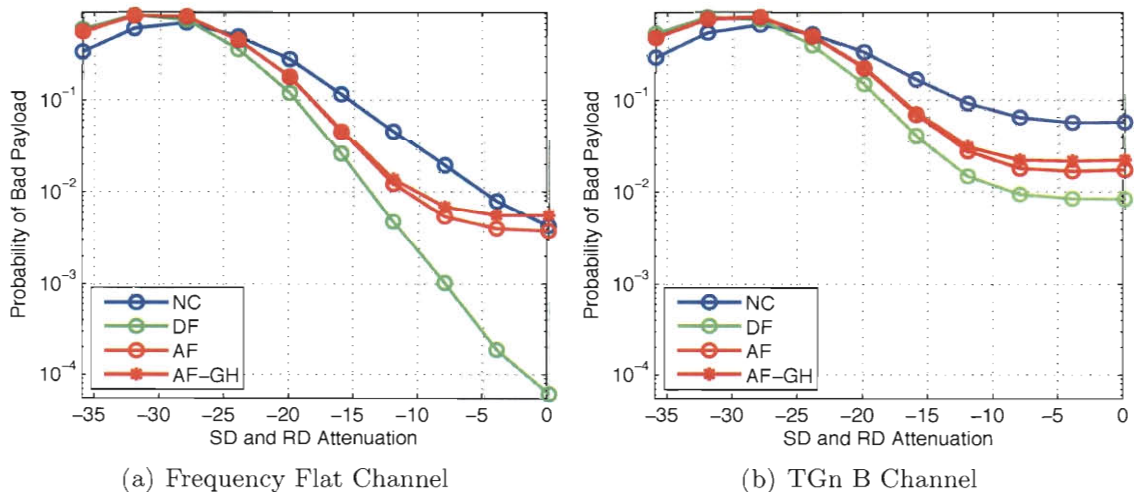


Figure A.10 : Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation.

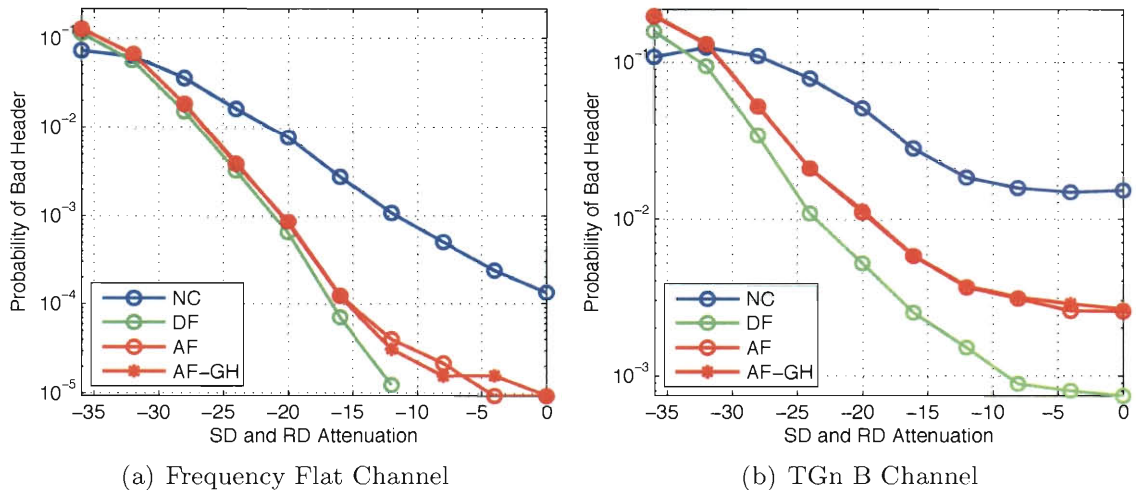


Figure A.11 : Probability of destination receiving packets with bad headers for QPSK modulation.

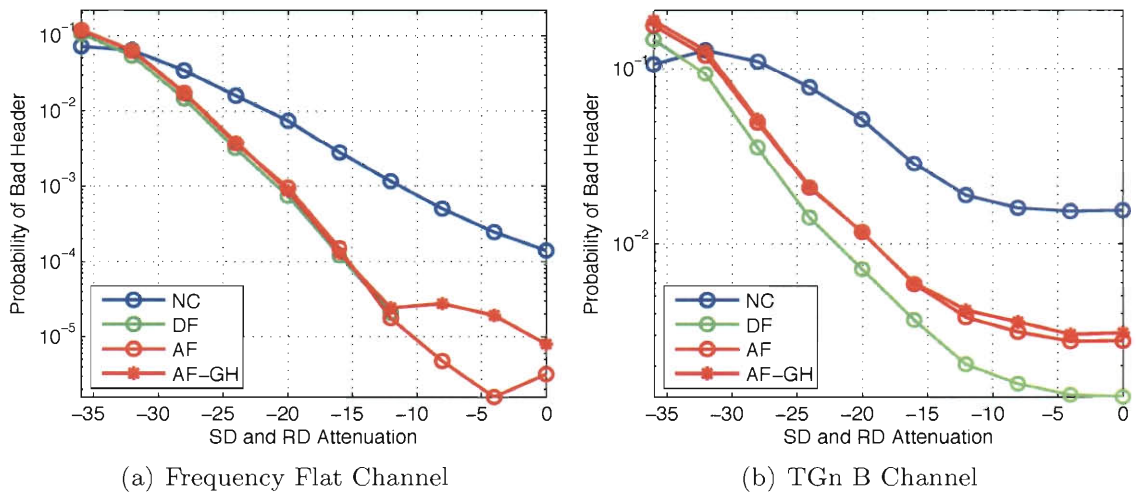


Figure A.12 : Probability of destination receiving packets with bad headers for 16-QAM modulation.



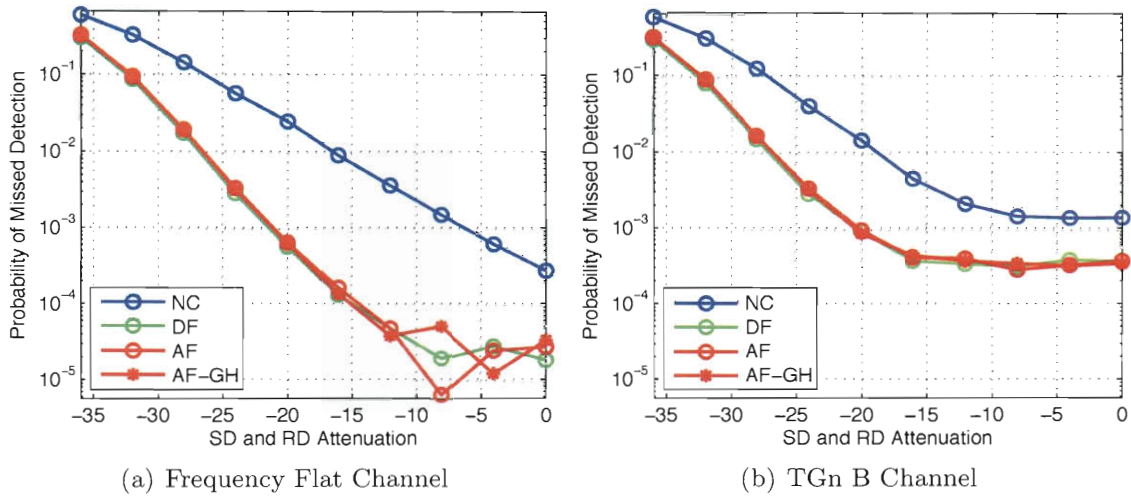


Figure A.13 : Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads.

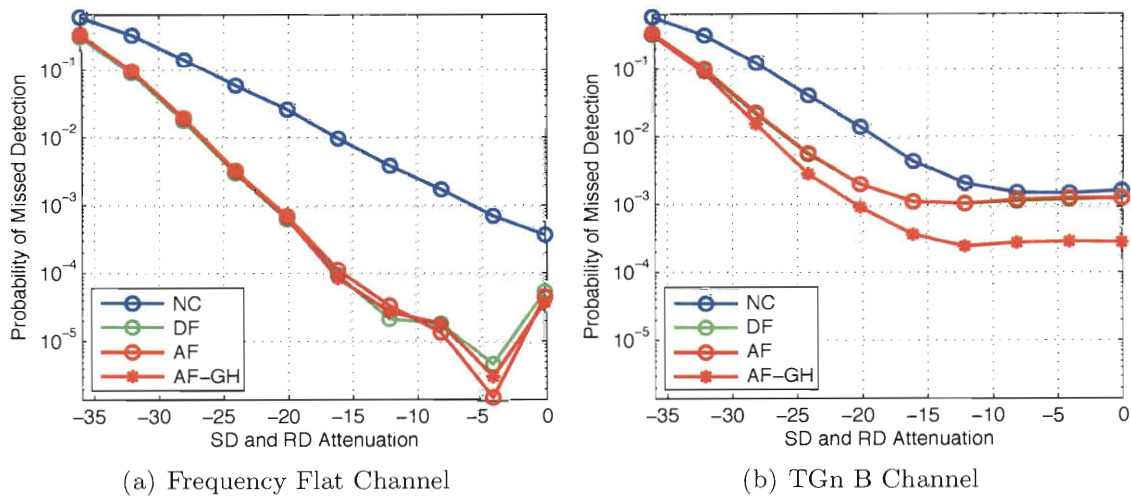


Figure A.14 : Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads.

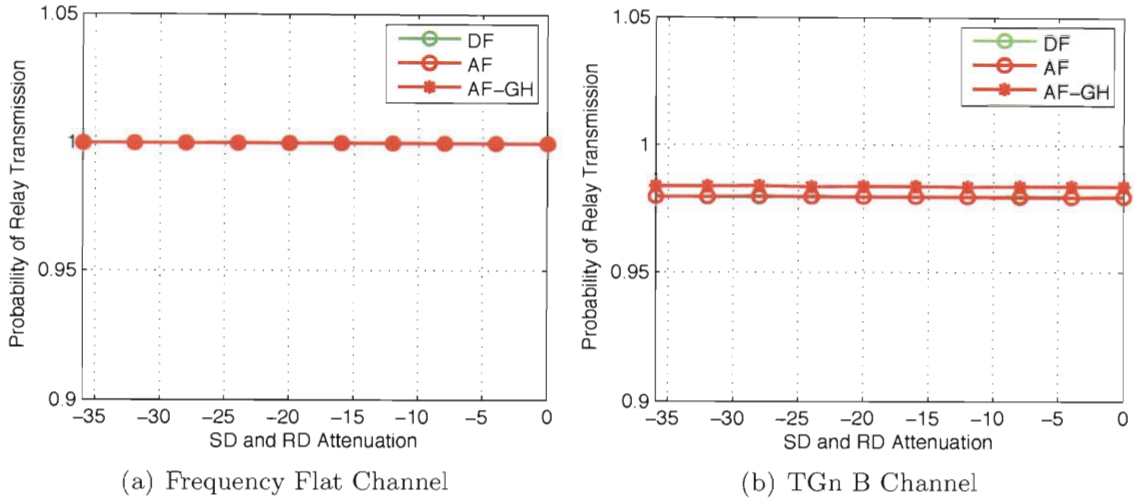


Figure A.15 : Probability of relay transmitting for QPSK payloads.

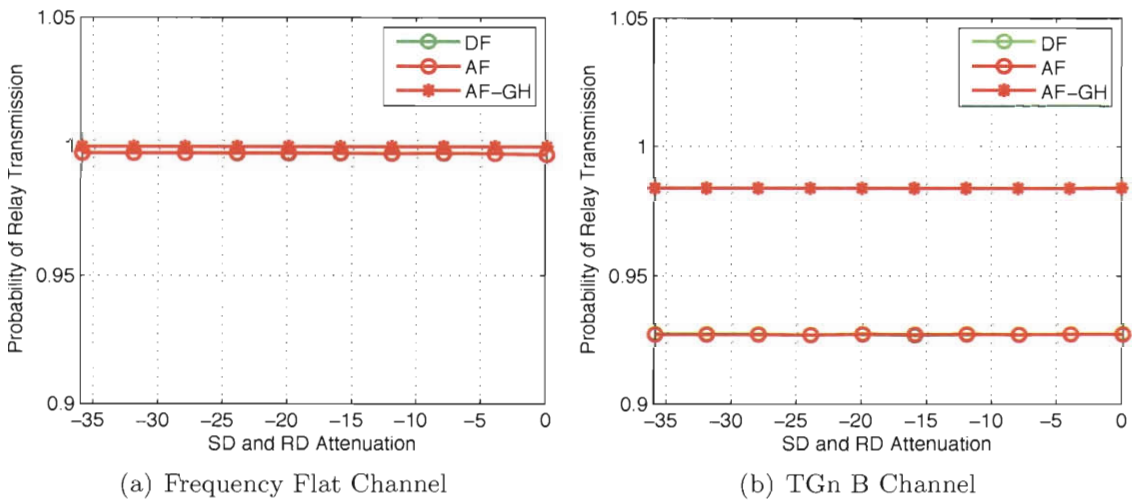


Figure A.16 : Probability of relay transmitting for 16-QAM payloads.

## A.2 Equidistant Nodes Topology

### A.2.1 1412 Byte Payloads

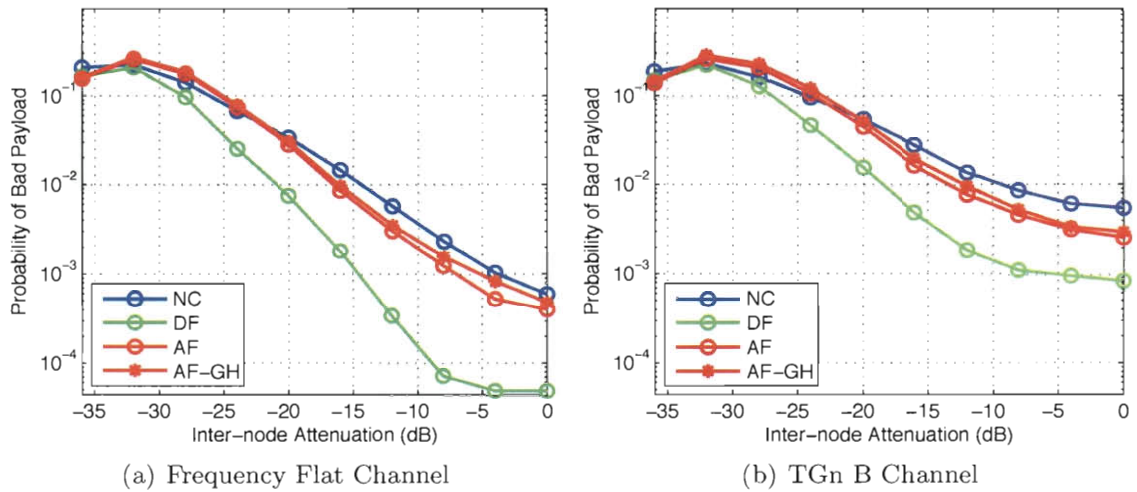


Figure A.17 : Probability of destination receiving packets with good headers but bad payloads for QPSK modulation.

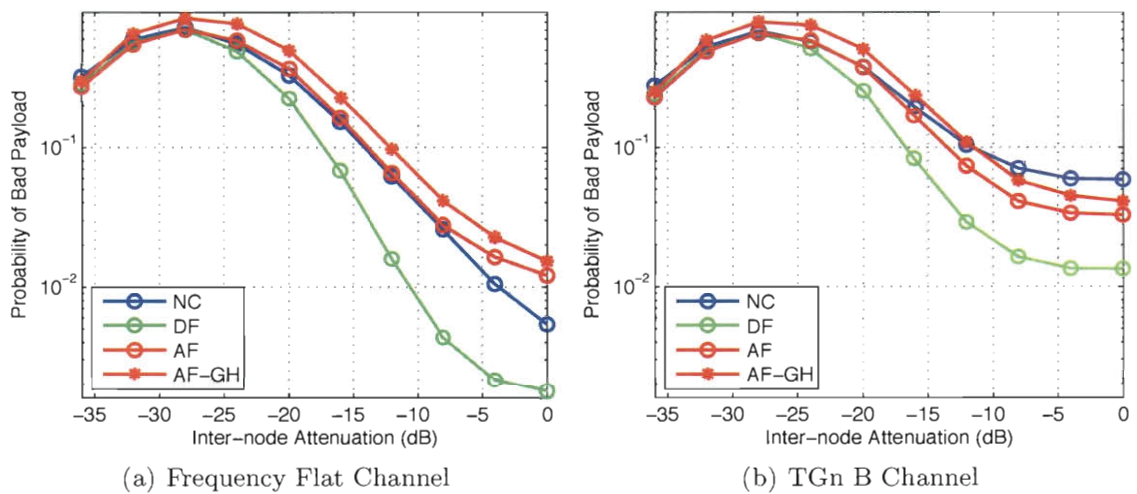


Figure A.18 : Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation.

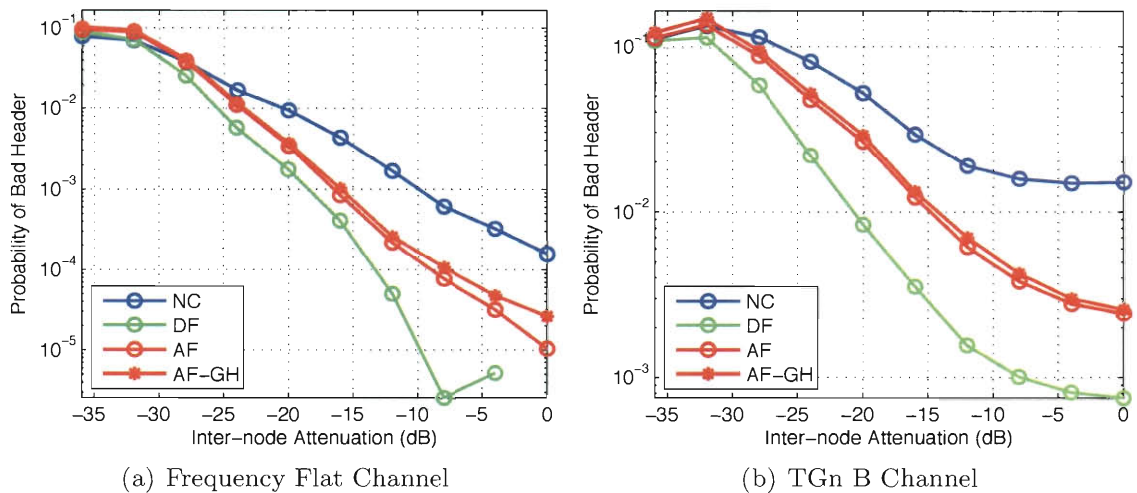


Figure A.19 : Probability of destination receiving packets with bad headers for QPSK modulation.

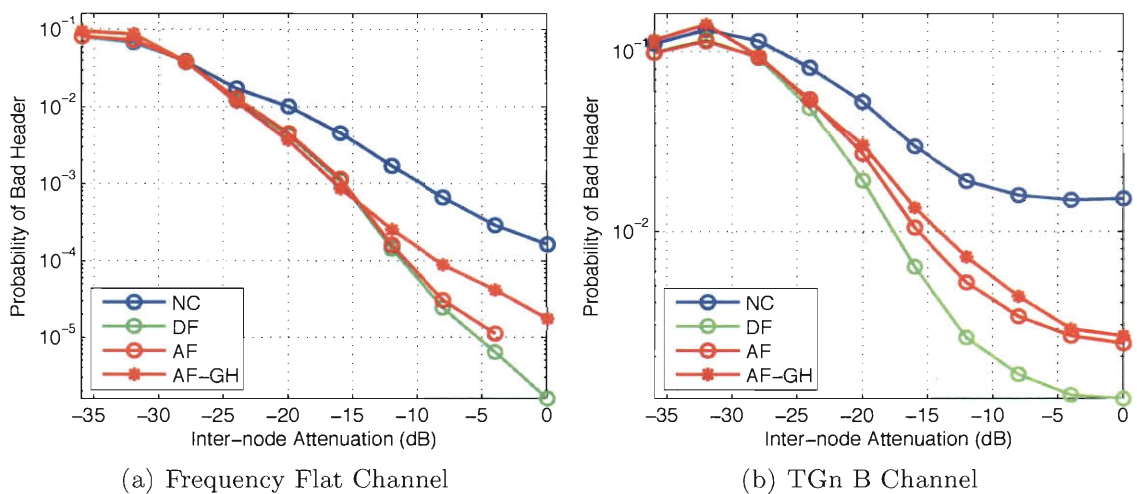


Figure A.20 : Probability of destination receiving packets with bad headers for 16-QAM modulation.

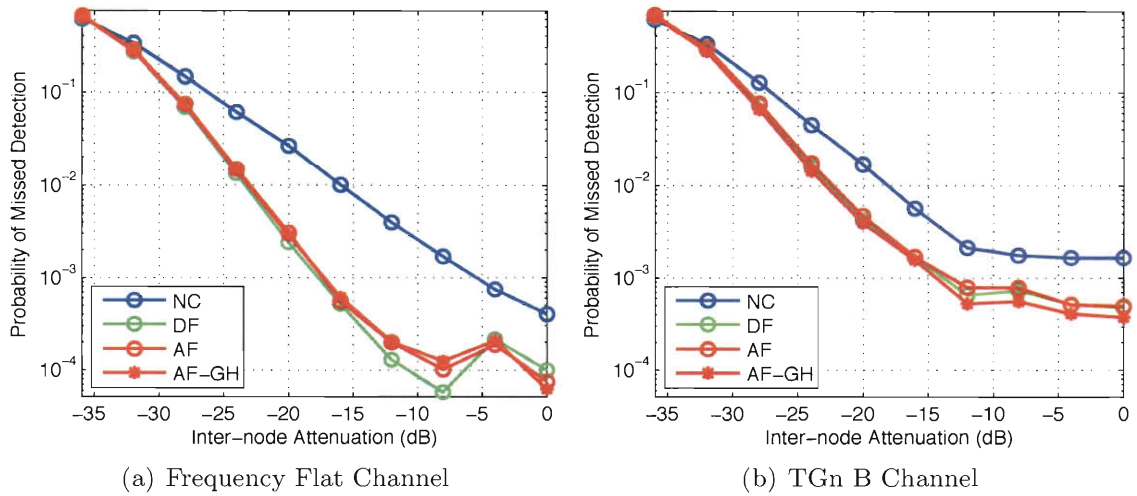


Figure A.21 : Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads.

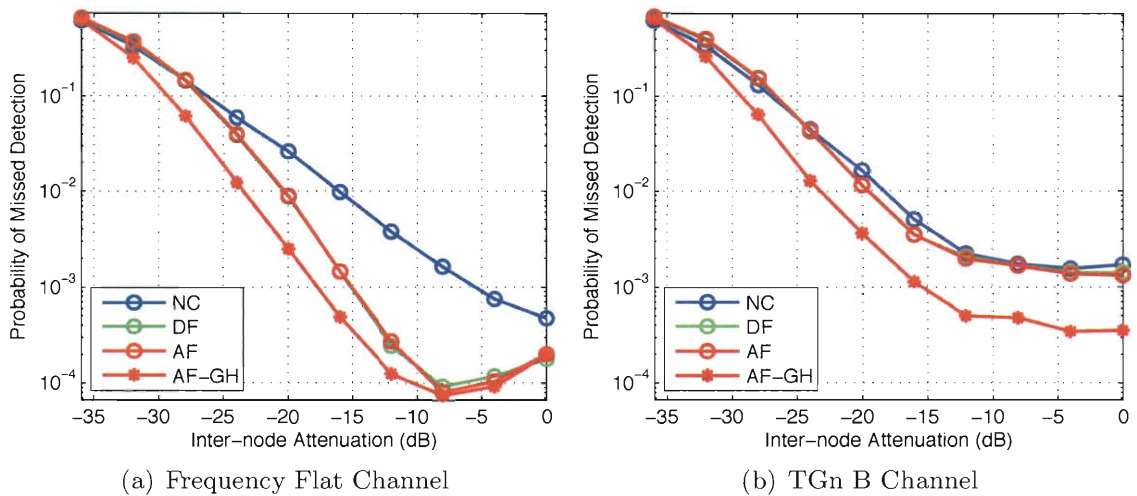


Figure A.22 : Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads.

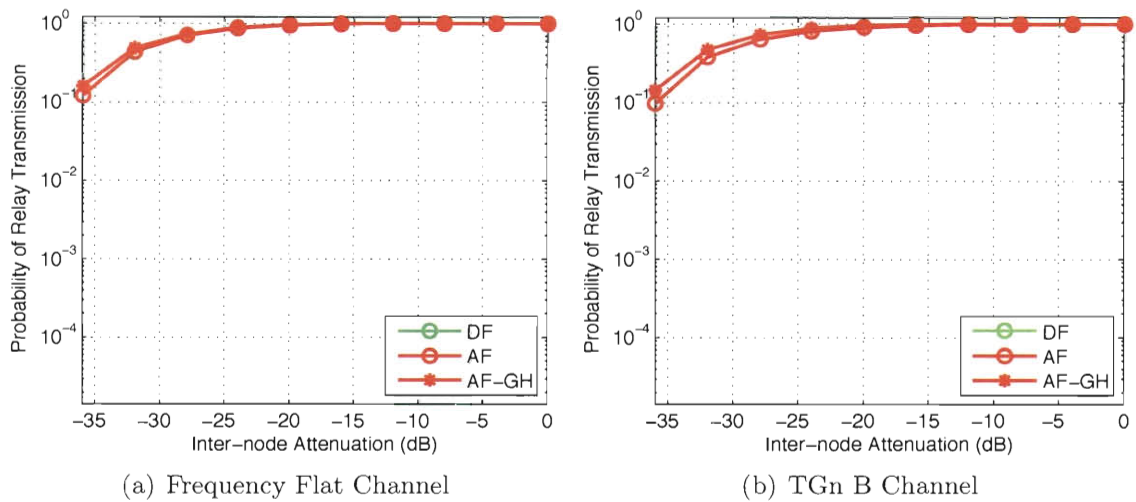


Figure A.23 : Probability of relay transmitting for QPSK payloads.

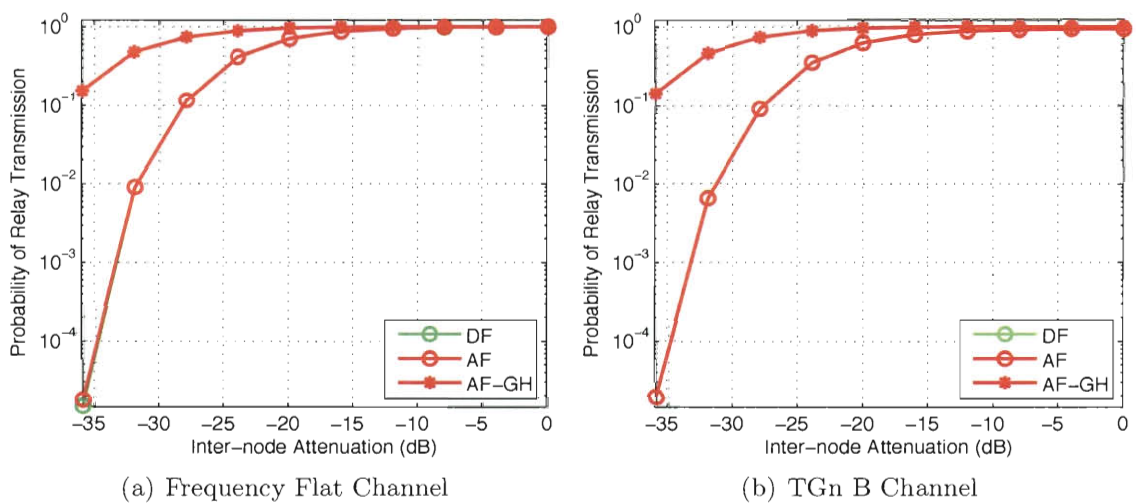


Figure A.24 : Probability of relay transmitting for 16-QAM payloads.

### A.2.2 692 Byte Payloads

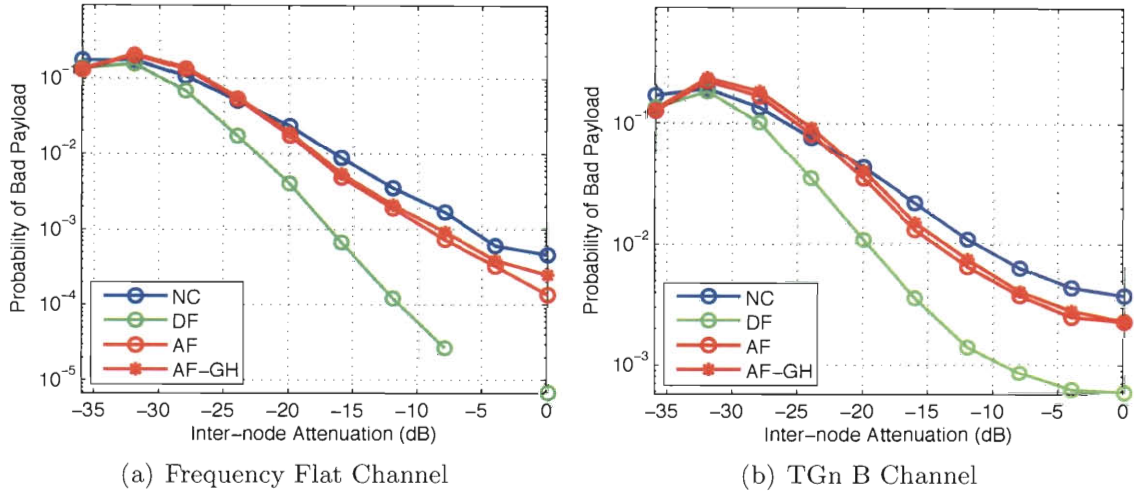


Figure A.25 : Probability of destination receiving packets with good headers but bad payloads for QPSK modulation.

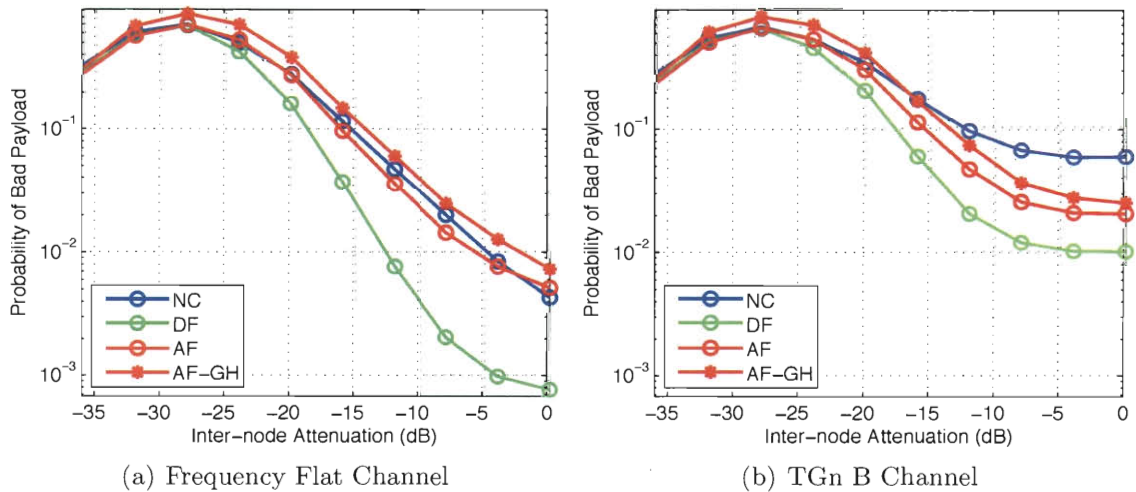


Figure A.26 : Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation.



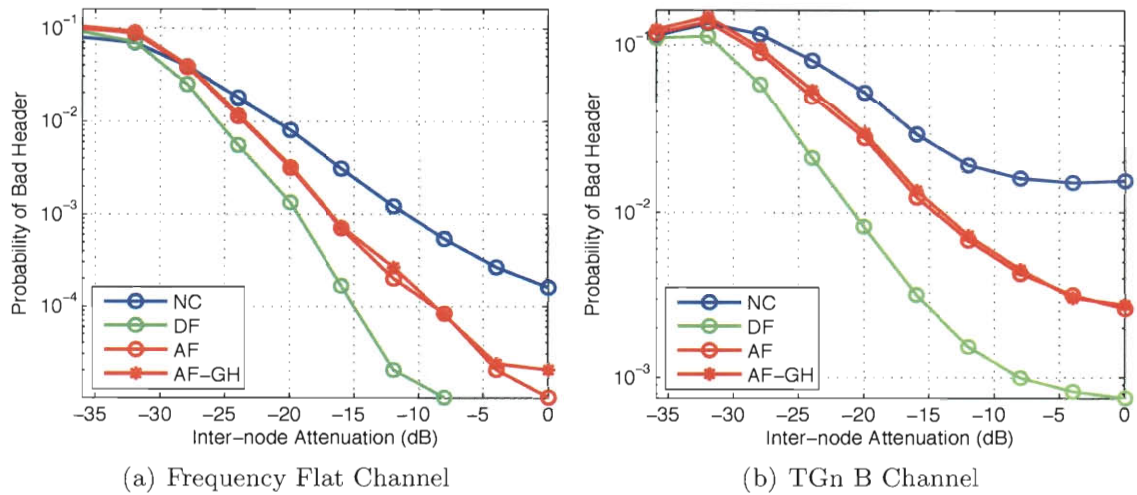


Figure A.27 : Probability of destination receiving packets with bad headers for QPSK modulation.

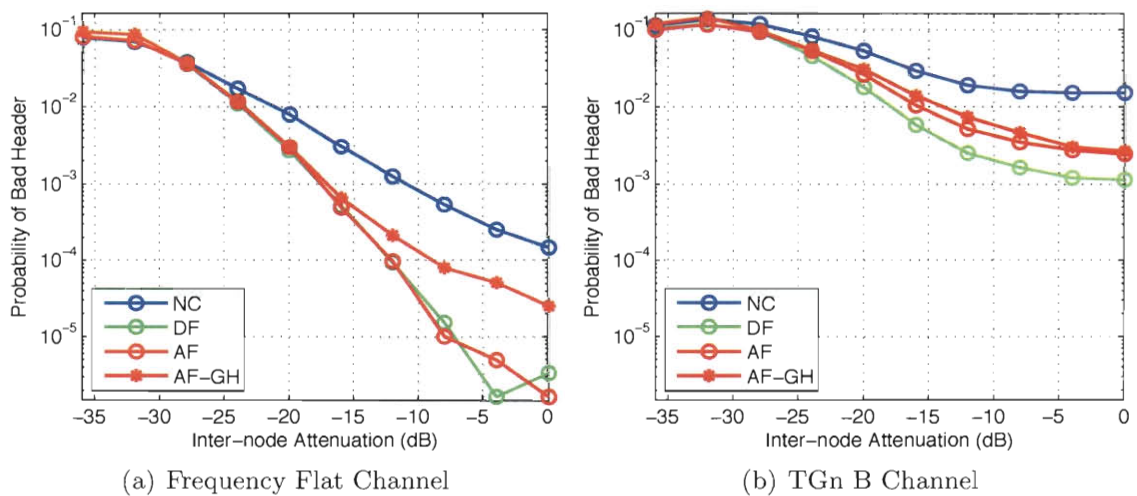


Figure A.28 : Probability of destination receiving packets with bad headers for 16-QAM modulation.



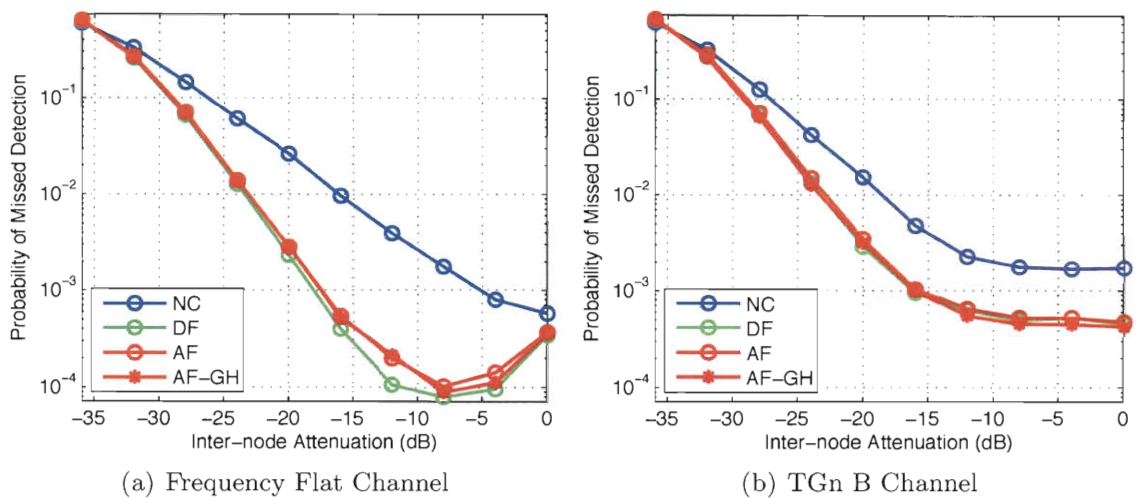


Figure A.29 : Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads.

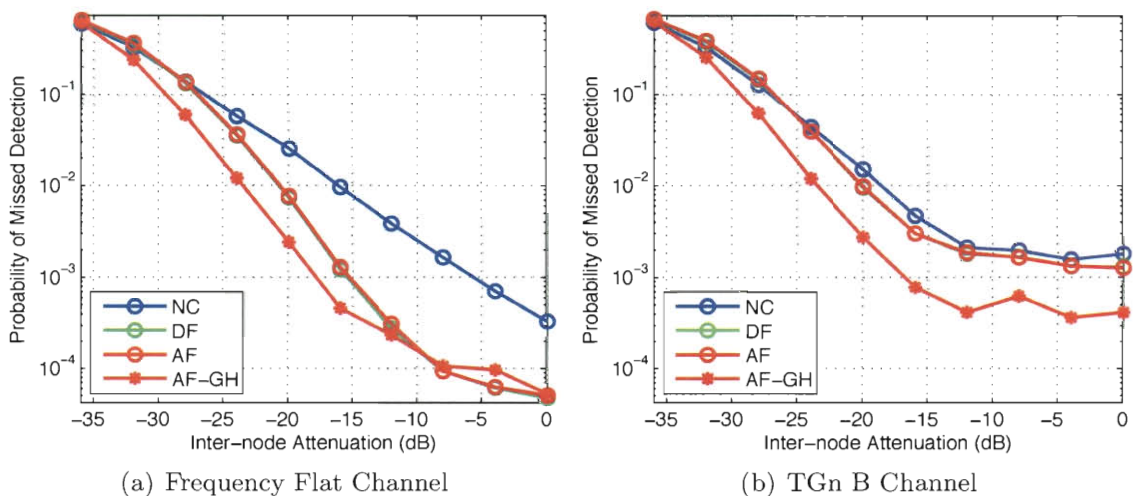


Figure A.30 : Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads.

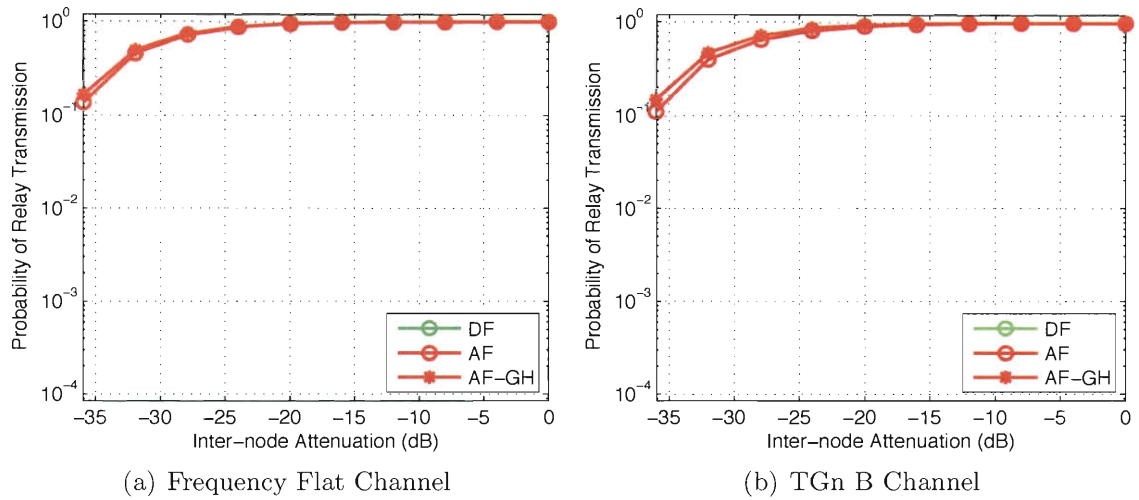


Figure A.31 : Probability of relay transmitting for QPSK payloads.

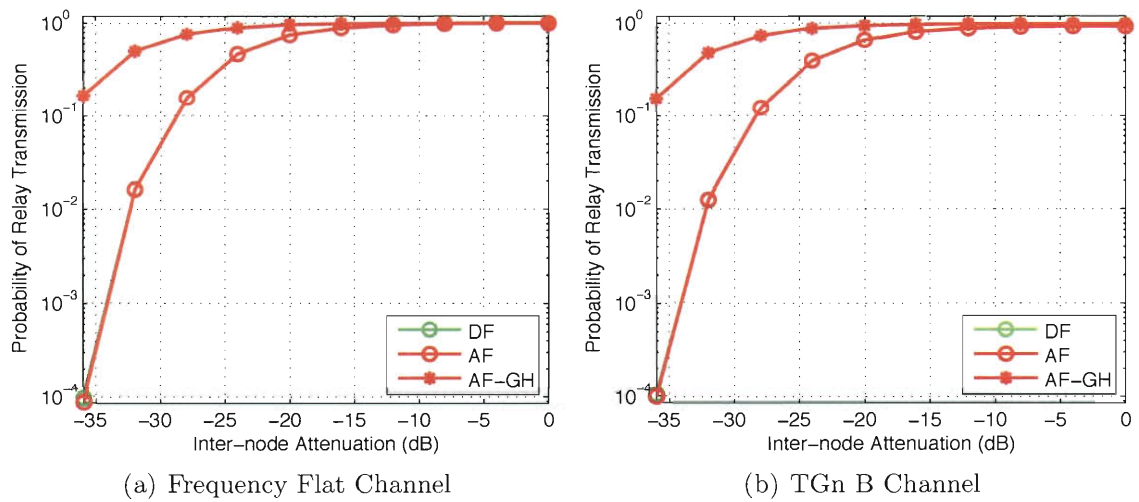


Figure A.32 : Probability of relay transmitting for 16-QAM payloads.

## A.3 Linear Topologies

### A.3.1 10.4 m SD Separation

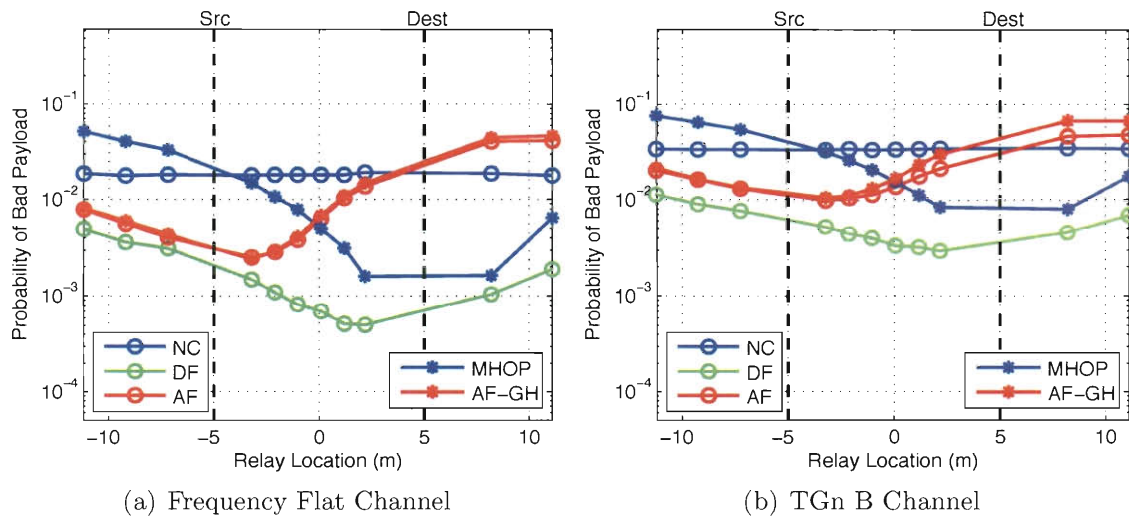


Figure A.33 : Probability of destination receiving packets with good headers but bad payloads for QPSK modulation.

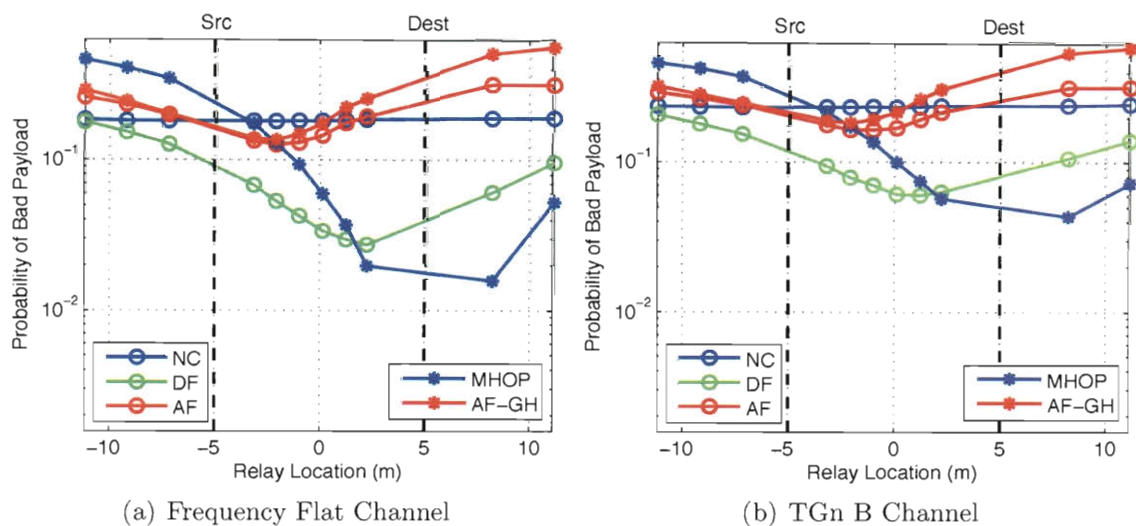


Figure A.34 : Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation.

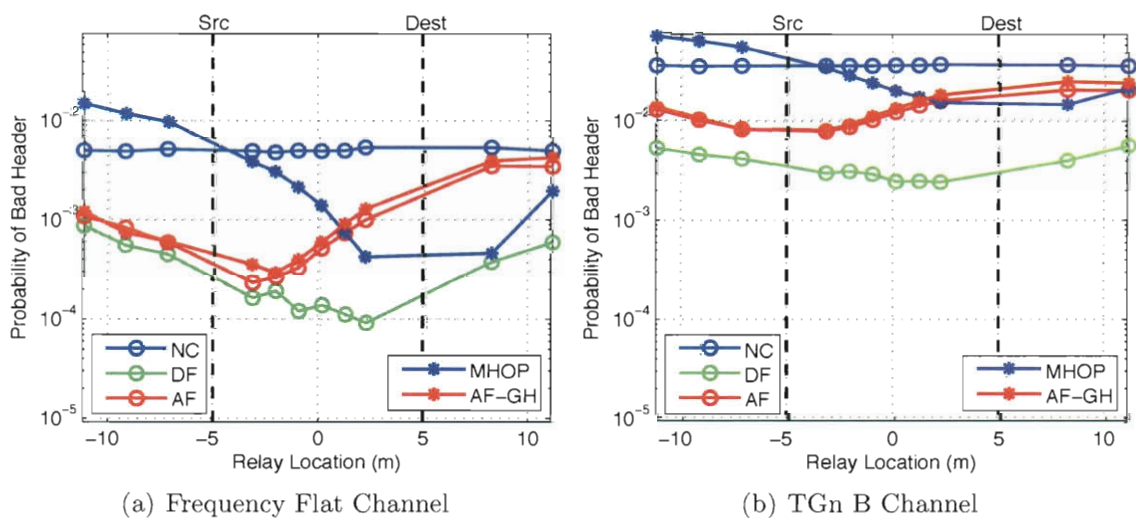


Figure A.35 : Probability of destination receiving packets with bad headers for QPSK modulation.

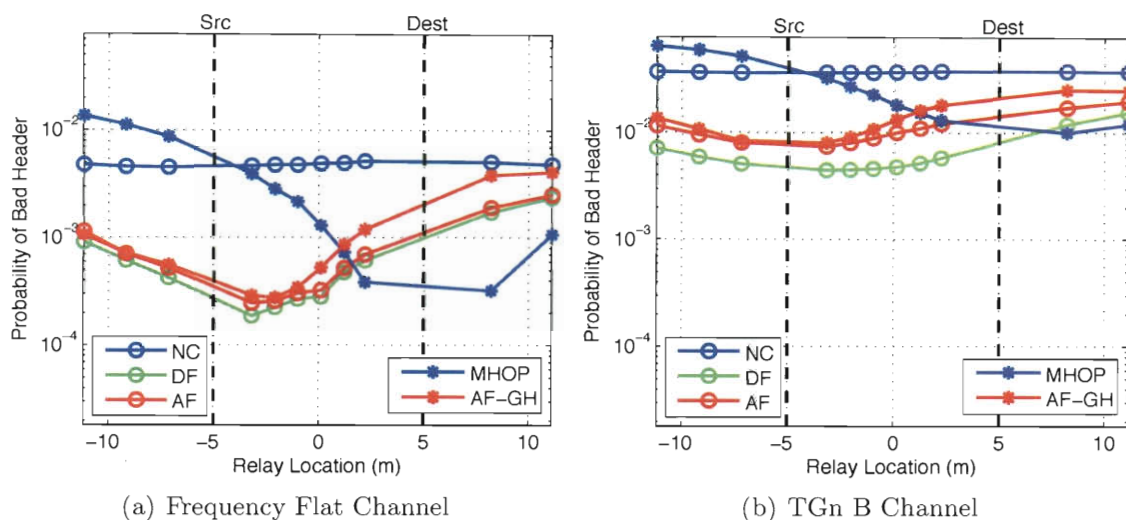


Figure A.36 : Probability of destination receiving packets with bad headers for 16-QAM modulation.

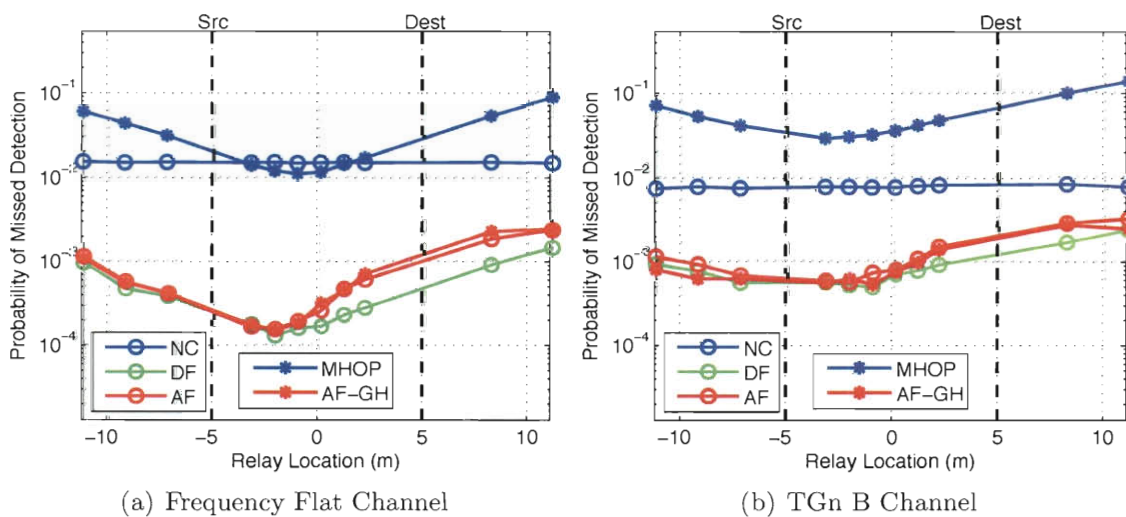


Figure A.37 : Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads.

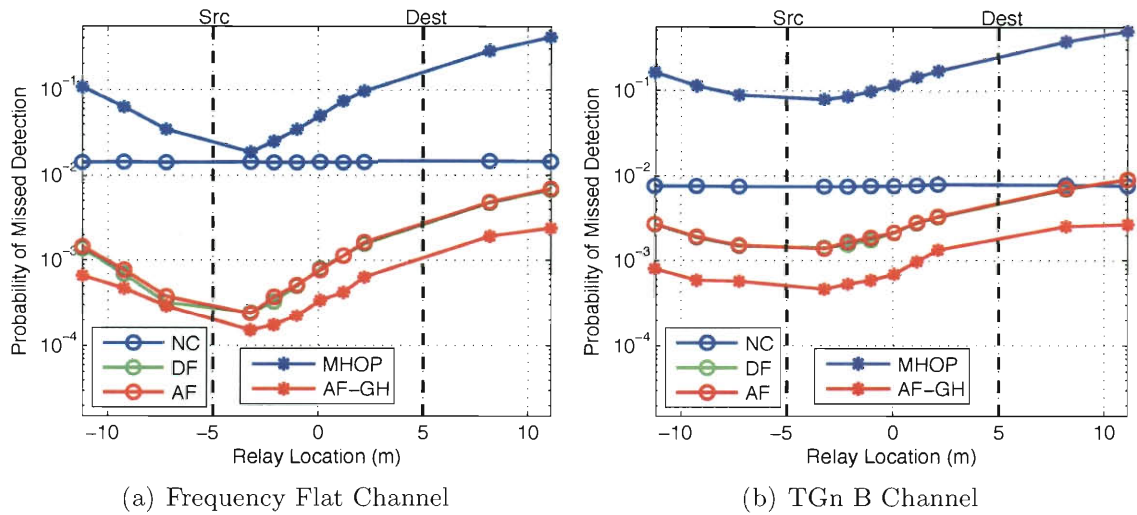


Figure A.38 : Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads.

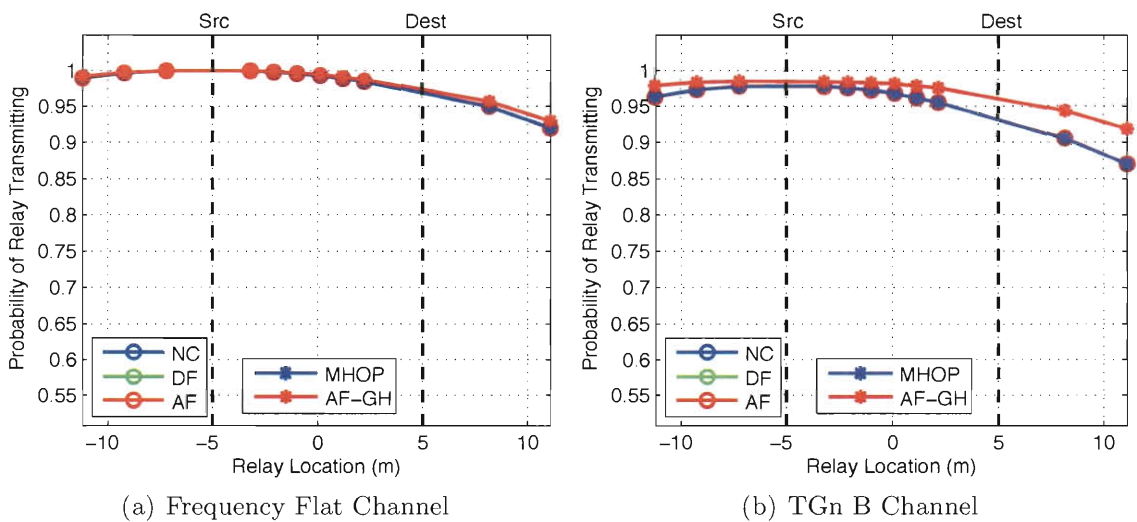


Figure A.39 : Probability of relay transmitting for QPSK payloads.

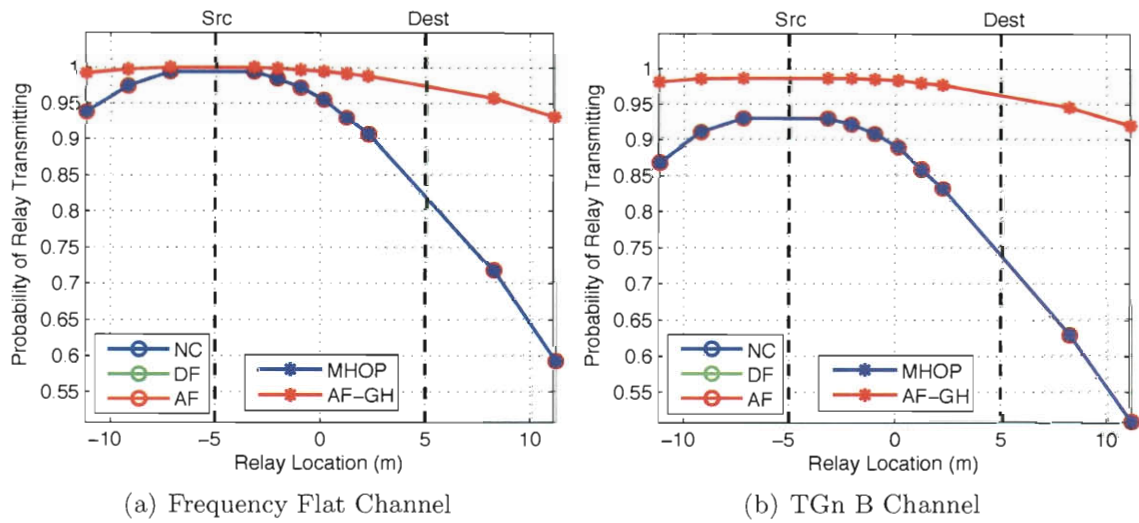


Figure A.40 : Probability of relay transmitting for 16-QAM payloads.



### A.3.2 18 m SD Separation

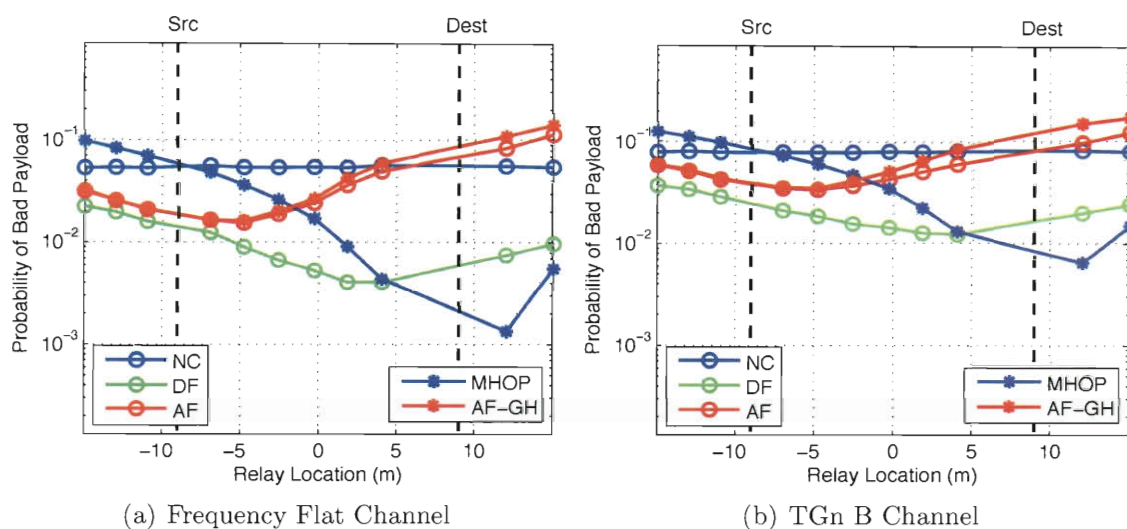


Figure A.41 : Probability of destination receiving packets with good headers but bad payloads for QPSK modulation.

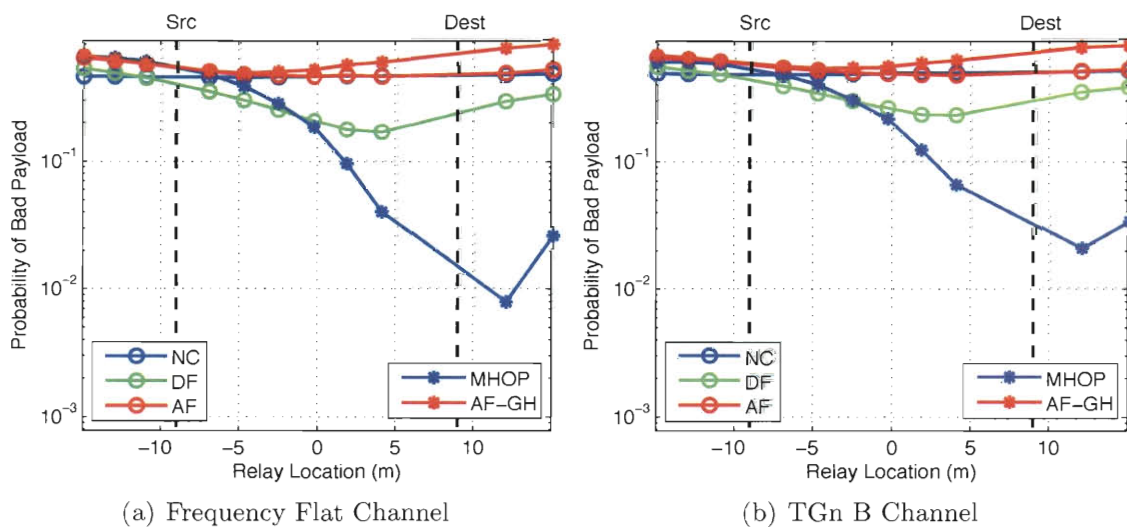


Figure A.42 : Probability of destination receiving packets with good headers but bad payloads for 16-QAM modulation.



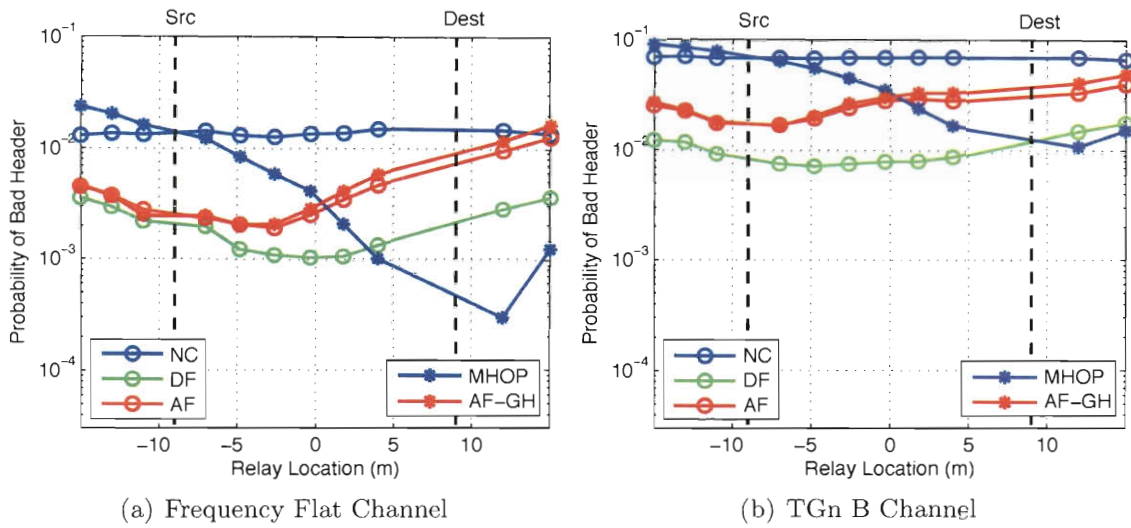


Figure A.43 : Probability of destination receiving packets with bad headers for QPSK modulation.

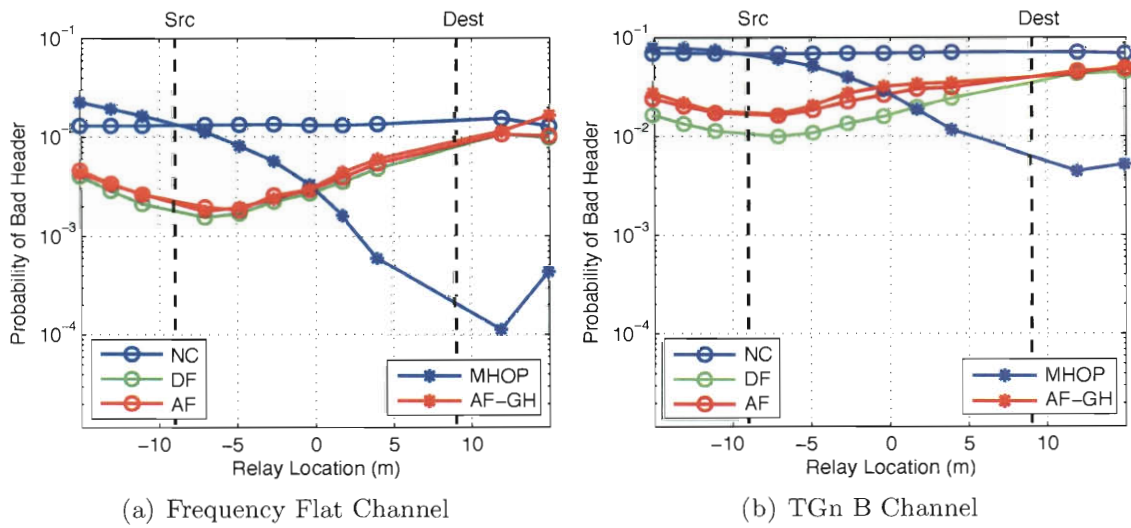


Figure A.44 : Probability of destination receiving packets with bad headers for 16-QAM modulation.

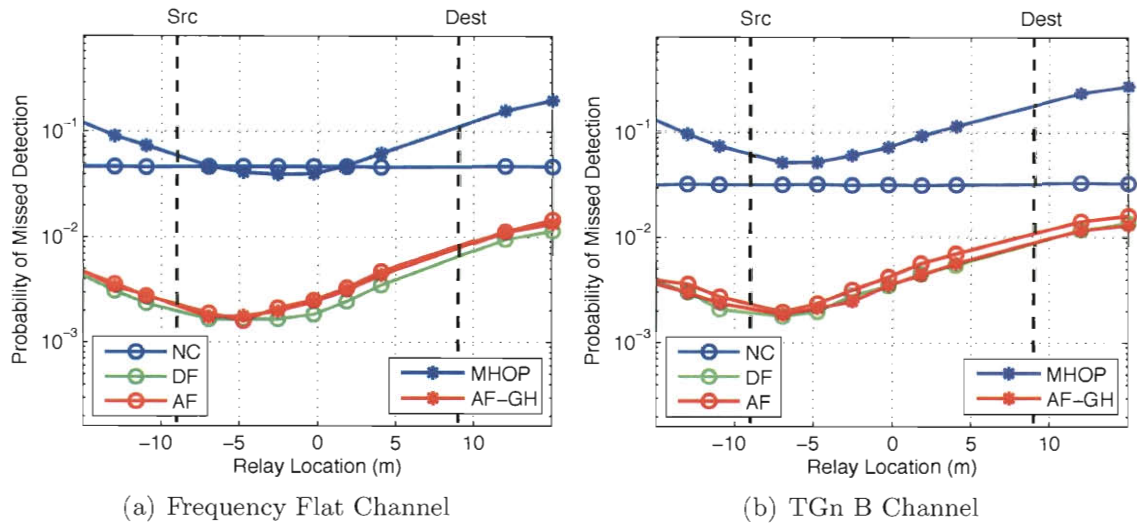


Figure A.45 : Probability of missed detection (no energy detection or preamble correlation) at the destination for QPSK payloads.

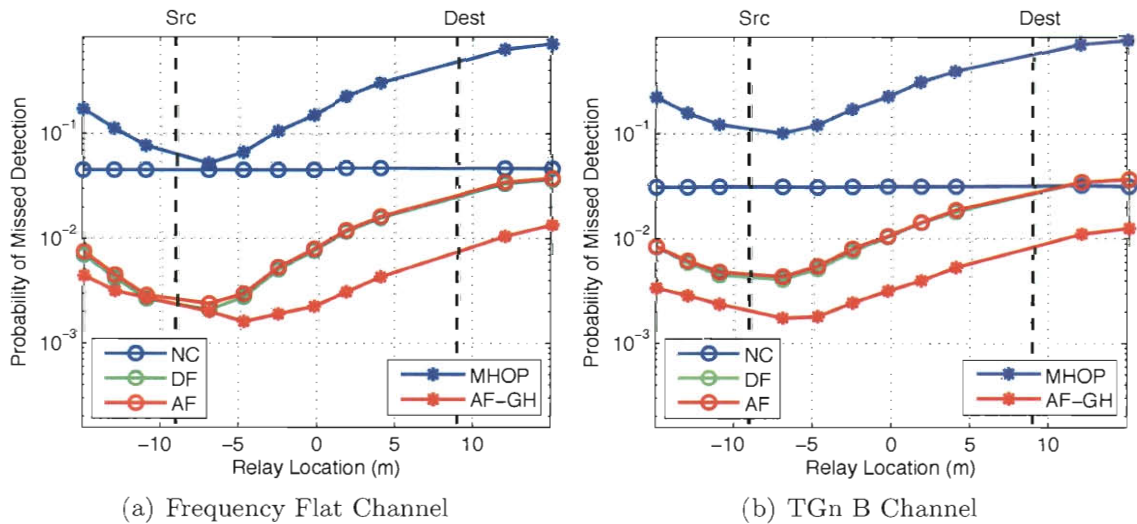


Figure A.46 : Probability of missed detection (no energy detection or preamble correlation) at the destination for 16-QAM payloads.

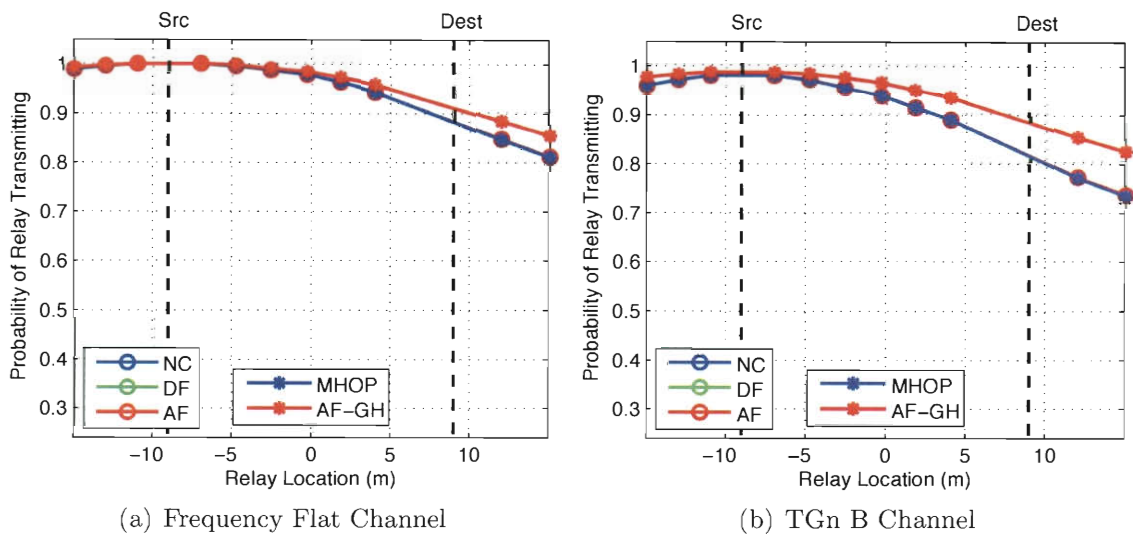


Figure A.47 : Probability of relay transmitting for QPSK payloads.

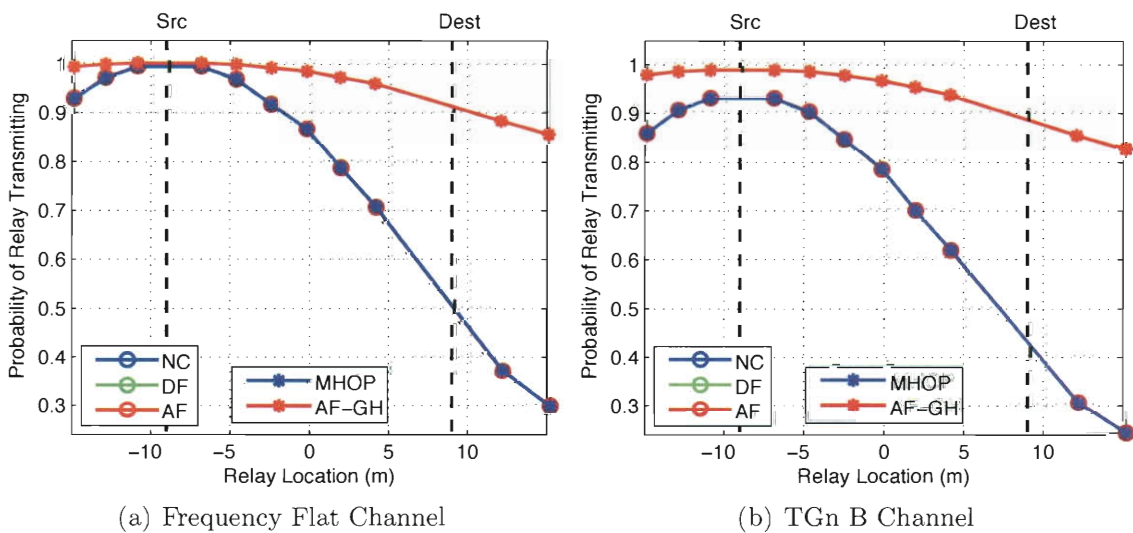


Figure A.48 : Probability of relay transmitting for 16-QAM payloads.

## Bibliography

- [1] A. Sendonaris, E. Erkip, and B. Aazhang, “User cooperation diversity. Part I: System description,” *IEEE Trans. Comm*, vol. 51, no. 11, pp. 1927–1938, Nov. 2003.
- [2] ———, “User cooperation diversity. Part II: Implementation aspects and performance analysis,” *IEEE Trans. Comm*, vol. 51, no. 11, pp. 1939–1948, Nov. 2003.
- [3] J. Laneman, D. Tse, and G. Wornell, “Cooperative diversity in wireless networks: Efficient protocols and outage behavior,” *IEEE Trans. Information Theory*, vol. 50, no. 12, 2004.
- [4] J. Laneman and G. Wornell, “Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks,” *Information Theory, IEEE Transactions on*, vol. 49, no. 10, pp. 2415–2425, 2003.
- [5] G. Kramer, I. Marić, and R. Yates, “Cooperative communications,” *Foundations and Trends in Networking*, vol. 1, no. 3, 2006.
- [6] G. Bradford and J. Laneman, “A survey of implementation efforts and experimental design for cooperative communications,” in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2010, pp. 5602–5605.

- [7] G. Bradford and J. N. Laneman, “An experimental framework for the evaluation of cooperative diversity,” in *Proceedings of CISS*, 2009.
- [8] S. Berger and A. Wittneben, “Experimental performance evaluation of multiuser zero forcing relaying in indoor scenarios,” in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 2. IEEE, 2005, pp. 1101–1105.
- [9] P. Murphy, A. Sabharwal, and B. Aazhang, “On building a cooperative communication system: Testbed implementation and first results,” *EURASIP Journal on Wireless Communications and Networking*, 2009 (in press).
- [10] T. Korakis, M. Knox, E. Erkip, and S. Panwar, “Cooperative network implementation using open-source platforms,” *Communications Magazine, IEEE*, vol. 47, no. 2, pp. 134–141, 2009.
- [11] “WARP Community Publications.” [Online]. Available: <http://warp.rice.edu/papers>
- [12] “Rice University WARP Project.” [Online]. Available: <http://warp.rice.edu/>
- [13] “GNU Radio.” [Online]. Available: <http://gnuradio.org/>
- [14] “Ettus Research.” [Online]. Available: <http://ettus.com/>
- [15] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, “Enabling MAC protocol implementations on software-defined radios,” in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*. USENIX Association, 2009, pp. 91–105.

- [16] O. Shin, A. Chan, H. Kung, and V. Tarokh, "Design of an OFDM cooperative space-time diversity system," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 4, 2007.
- [17] S. Yiu, R. Schober, and L. Lampe, "Distributed space-time block coding," *Communications, IEEE Transactions on*, vol. 54, no. 7, pp. 1195–1206, 2006.
- [18] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Comm.*, vol. 16, no. 8, 1998.
- [19] T. Pollet, M. Van Bladel, and M. Moeneclaey, "BER sensitivity of OFDM systems to carrier frequency offset and Wiener phase noise," *IEEE Trans. on Communications*, vol. 43, no. 234, 1995.
- [20] J. Armstrong, "Analysis of new and existing methods of reducing intercarrier interference due to carrier frequency offset in ofdm," *Communications, IEEE Transactions on*, vol. 47, no. 3, pp. 365–369, mar. 1999.
- [21] K. Sathananthan and C. Tellambura, "Probability of error calculation of OFDM systems with frequency offset," *Communications, IEEE Transactions on*, vol. 49, no. 11, pp. 1884–1888, 2002.
- [22] Y. Mostofi and D. Cox, "ICI mitigation for pilot-aided OFDM mobile systems," *Wireless Communications, IEEE Transactions on*, vol. 4, no. 2, pp. 765–774, 2005.
- [23] "WARP Clock Board." [Online]. Available: [http://warp.rice.edu/w/HardwareUsersGuides/ClockBoard\\_v1.1](http://warp.rice.edu/w/HardwareUsersGuides/ClockBoard_v1.1)

- [24] [Online]. Available: <http://www.crystekcrystals.com/crystal/spec-sheets/tcxo/CVT32.pdf>
- [25] “WARP Radio Board.” [Online]. Available: [http://warp.rice.edu/w/HardwareUsersGuides/RadioBoard\\_v1.4](http://warp.rice.edu/w/HardwareUsersGuides/RadioBoard_v1.4)
- [26] Y. Yao and G. Giannakis, “Blind carrier frequency offset estimation in SISO, MIMO, and multiuser OFDM systems,” *Communications, IEEE Transactions on*, vol. 53, no. 1, pp. 173–183, 2005.
- [27] J. Li, G. Liu, and G. Giannakis, “Carrier frequency offset estimation for OFDM-based WLANs,” *Signal Processing Letters, IEEE*, vol. 8, no. 3, pp. 80–82, 2002.
- [28] T. Schmidl and D. Cox, “Robust frequency and timing synchronization for ofdm,” *Communications, IEEE Transactions on*, vol. 45, no. 12, pp. 1613–1621, dec. 1997.
- [29] “WARPLab.” [Online]. Available: <http://warp.rice.edu/w/WARPLab>
- [30] L. Brötje, S. Vogeler, K. Kammeyer, R. Rückriem, and S. Fechtel, “On carrier frequency offsets in Alamouti-coded OFDM systems similar to IEEE 802.11,” in *Proc. 8th International OFDM Workshop (InOWo03)*, August.
- [31] M. Krondorf and G. Fettweis, “Numerical performance evaluation for Alamouti space time coded OFDM under receiver impairments,” *Wireless Communications, IEEE Transactions on*, vol. 8, no. 3, pp. 1446–1455, 2009.
- [32] R. Sakata, K. Akita, and K. Sato, “Real-time phase tracking method for IEEE 802.11 a/g/n receiver under phase noise condition,” in *IEEE Vehicular Technology Conference*, vol. 4, 2006.

- [33] Q. Huang, M. Ghogho, and J. Wei, "Data detection in cooperative STBC-OFDM systems with multiple frequency offsets," *Signal Processing Letters, IEEE*, vol. 16, no. 7, pp. 600–603, 2009.
- [34] X. Li, F. Ng, and T. Han, "Carrier frequency offset mitigation in asynchronous cooperative OFDM transmissions," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 675–685, 2008.
- [35] J. Mietzner, J. Eick, and P. Hoeher, "On distributed space-time coding techniques for cooperative wireless networks and their sensitivity to frequency offsets," in *Smart Antennas, 2004. ITG Workshop on*. IEEE, 2005, pp. 114–121.
- [36] "Xilinx System Generator." [Online]. Available: <http://xilinx.com/sysgen>
- [37] "WARP OFDM Reference Design." [Online]. Available: <http://warp.rice.edu/w/OFDMReferenceDesign>
- [38] "Azimuth ACE 400WB." [Online]. Available: <http://www.azimuthsystems.com/platforms-channel-400wb.htm>
- [39] "Pasternack PE2014." [Online]. Available: <http://search.pasternack.com/Search.aspx?Query=pe2014>
- [40] V. Erceg, et al., "TGn channel models," IEEE 802.11 document 03/940r4.
- [41] S. Makda, A. Choudhary, N. Raman, T. Korakis, Z. Tao, and S. Panwar, "Security implications of cooperative communications in wireless networks," in *Sarnoff Symposium, 2008 IEEE*, 2008, pp. 1–6.
- [42] B. Wang, Z. Han, and K. Liu, "Distributed relay selection and power control for multiuser cooperative communication networks using buyer/seller game," in



- INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, May 2007, pp. 544–552.
- [43] T. Rappaport, *Wireless communications: principles and practice*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2001.
- [44] G. Kramer, M. Gastpar, and P. Gupta, “Cooperative strategies and capacity theorems for relay networks,” *IEEE Transactions on Information Theory*, vol. 51, no. 9, pp. 3037–3063, 2005.
- [45] “Rice University WARP Project.” [Online]. Available: <http://warp.rice.edu/w/WARPnet>
- [46] C. Hunter, P. Murphy, and A. Sabharwal, “Real-time testbed implementation of a distributed cooperative MAC and PHY,” in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*. IEEE, 2010, pp. 1–6.
- [47] M. Lacage, M. Manshaei, and T. Turetli, “IEEE 802.11 rate adaptation: a practical approach,” in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2004, pp. 126–134.
- [48] S. Wong, H. Yang, S. Lu, and V. Bharghavan, “Robust rate adaptation for 802.11 wireless networks,” in *Proceedings of the 12th annual international conference on Mobile computing and networking*. ACM, 2006, pp. 146–157.
- [49] J. Camp and E. Knightly, “Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation,” in *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 2008, pp. 315–326.

- [50] M. Duarte and A. Sabharwal, “Full-duplex wireless communications using off-the-shelf radios: Feasibility and first results,” in *Asilomar Conference on Signals, Systems and Computers*, 2010.
- [51] J. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, “Achieving single channel, full duplex wireless communication,” in *Proceedings of Mobicom*, 2010.