Using Physical Layer Emulation to Understand and Improve Wireless Networks

Glenn Judd

CMU-CS-06-164

October 2006

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Peter Steenkiste, Chair
Srinivasan Seshan
Dan Stancil
David Maltz, Microsoft
Robert Morris, Massachusetts Institute of Technology

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the NSF under award numbers CCR-0205266 and CNS-0434824. Additional support was also provided by Intel and Xilinx. Glenn Judd was supported by a National Defense Science and Engineering Graduate Fellowship and an Intel Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Wireless, Network, Emulation

Abstract

Researchers and developers have long faced a fundamental tension between the experimental realism of wireless testbeds on one hand, and the control and repeatability of simulation on the other hand. This thesis introduces physical layer wireless network emulation – a new approach to wireless network experimentation that balances the stark tradeoff of traditional alternatives by enabling both realistic and repeatable experimentation.

The design and implementation of a functional wireless emulator are presented along with a discussion of how this implementation overcomes the challenges necessary to meet operational requirements. In particular, solutions to the problems of developing a hardware architecture for emulation, and software control of that architecture will be presented.

To illustrate the power of physical layer wireless network emulation, case studies are presented. First, physical layer emulation is used to analyze several aspects of wireless LAN link-level behavior. Physical layer emulation is then used to investigate wireless LAN access point selection performance, and to develop improvements.

This thesis shows that – compared to traditional approaches – physical layer wireless network emulation provides a better understanding of real-world wireless network performance, shortens the development cycle of wireless networking software, and facilitates the deployment of research into operational wireless networks without sacrificing a controlled experimental environment.

Acknowledgements

This work would not have been possible without large amounts of assistance from several people and organizations. Peter Steenkiste, supported me in taking this project from the "crazy idea" stage through to a working system both technically, as my advisor, and practically by procuring support for this work. Dan Stancil provided a wealth of knowledge of all things RF, space and equipment with which to conduct this work, and prevented me from offending engineers with abused terminology. Dave Maltz and Srini Seshan provided much needed external perspective on this work and on useful problems to investigate in wireless networks.

The Roofnet Group led by Robert Morris and consisting of Sanjit Biswas, John Bicket, and Dan Aguayo provided a reality-based perspective and a bevy of interesting problems in need of emulation-based analysis. In addition, numerous discussions with them yielded insight into important aspects of wireless performance, and pointed me in productive directions.

In addition Dan Stancil's students: J. P. Vant Hof, Ben Henty, Ahmet Cepni, Kevin Borries, and Ratish Punnoose provided a wealth of both practical and theoretical engineering knowledge essential to accomplishing this work.

I am also grateful for the National Defense Science and Engineering Grant that supported me for several years as well as to Intel, the National Science Foundation, and Xilinx for their support.

Finally, I am extremely grateful for the support and patience of my wife and the unfailing faith and encouragement of my children.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

As wireless networks become ubiquitous, it is increasingly important to make efficient use of the finite spectrum available. Unfortunately, research aimed at evaluating and improving wireless network protocols and applications is hindered by the inability to perform repeatable and realistic experiments. Experimental techniques that have proven successful for wired networks are inadequate for wireless networks since a wireless physical layer fundamentally affects operation at all layers of the protocol stack in complex ways. Links are no longer constant, reliable, and physically isolated from each other, but are variable, error-prone, and share a single medium with each other and with external uncontrolled sources.

This thesis presents a new technique for conducting wireless experiments that simultaneously achieves large degrees of realism and repeatability. A primary goal of this work is to provide the wireless networking community with a powerful experimental methodology that can be used to understand and improve wireless networks. The benefits of the approach presented in this thesis - physical layer wireless network emulation - are not, however, limited to the networking community. The high degree of fidelity achieved by the emulation technique presented here allows this approach to be used by the telecommunications research community. This approach could be particularly useful in analyzing wireless environments with a large number of channels such as MIMO (especially multi-user MIMO.)

Moreover, this approach has a broad potential for industrial application. The development, implementation, and tuning of wireless networking protocols is hindered by the same issues afflicting the research community. In particular, the distributed, ephemeral nature of wireless signal propagation greatly complicates the development of wireless protocols, software, and hardware. Physical layer emulation increases the effectiveness of developers by giving a controlled signal environment that greatly shortens the development and debug cycle by allowing complete control and repeatability. Likewise, testing and tuning can be largely automated - providing for shorter product development cycles.

1.1.1 Trade-offs in Existing Experimental Techniques

Before discussing physical layer wireless emulation, this discussion first considers an idealized method of experimentation. An ideal method of wireless experimentation would possess the following properties: repeatability and experimental control, layer 1-4 realism, the ability to run real applications, configurability, the ability to modify wireless device behavior, automation and remote management of experiments, support for a large number of nodes, isolation from production networks, and integration with wired networks and testbeds. The remainder of this section considers how alternative methods of experimentation fare with respect to this list of desirable properties, and introduces a new approach that has several advantages compared to existing approaches.

The most direct method of addressing realism is to conduct experiments using real hardware and software in various real world environments. Unfortunately, this approach faces serious repeatability and control issues since the behavior of the physical layer is tightly coupled to the physical environment and precise conditions in which an experiment is conducted. Uncontrolled interfering radio sources, and the mobility of people and physical objects, make signal environment conditions nearly impossible to reproduce. As a result, even repeating the same experiment twice can be a daunting task; remote researchers face an even bleaker situation trying to reproduce an experiment. It is also difficult to avoid affecting colocated production networks. Moreover, configuration and management of even a small number of mobile nodes distributed in three dimensions is cumbersome.

For these reasons, many researchers have understandably embraced simulation. This approach solves the problems of repeatability, configurability, manageability, modifiability, and (potentially) integration with external networks, but faces formidable obstacles in terms of realism. Recent work [33], for example, has shown that unless a great deal of care is taken, simulation-based experiments can produce imprecise and even inaccurate results. While careful simulation setup is a necessary condition for producing valid experimental results, it is clearly not sufficient in and of itself. A more fundamental condition that must be met to achieve valid wireless networking simulation results is the use of an accurate simulator. Not only must this simulator have a correct networking protocol stack, but it must accurately reproduce wireless signal transmission, propagation, and reception. To make the problem tractable, simplifications are typically made throughout the implementation of the simulator. Even fundamental functions such as deciding what a received frame looks like [59] diverge greatly from the operation of real hardware. As wireless signal reception behavior is only completely accurate when using real hardware, wireless simulations will always be a coarse approximation. Evaluating real applications running over wireless networks is typically very difficult using a simulator.

In addition, while wireless technology is undergoing rapid advances, wireless simulators, in particular open source wireless simulators, have lagged significantly behind these advances as discussed in Section 1.3.

The aforementioned issues with simulators, and a desire to avoid long simulation times, have caused some researchers to adopt emulation as a means of evaluation. Emulation retains simulation's advantages of repeatability and manageability, while potentially mitigating the issue of realism. Unfortunately, as discussed in Section 1.3, most emulators have adopted extremely simplified MAC and physical layers. As the operation of these layers is fundamental to the operation of a wireless network, it is unclear that these emulators gain any realism over existing simulators.

1.1.2 A New Approach

This thesis presents a new approach to wireless emulation that enables both realistic and repeatable wireless experimentation by accurately emulating wireless signal propagation in a physical space. Unlike previous approaches, this technique utilizes a real MAC layer, provides a realistic physical layer, and supports real applications while avoiding adopting an uncontrollable or locale-specific architecture. The key technique used to accomplish this is digital emulation of signal propagation using a field-programmable gate array (FPGA). This approach is made possible by the confluence of advancing programmable logic technology and advancing data converter technology. Only recently have commodity FPGAs and data converters become available that allow wideband wireless LAN signals to be digitized and processed using commodity dataconverters and programmable logic. Previously, wireless signal emulation was limited to a small number of wireless channels.

Physical emulation's high degree of control and fidelity allow signal propagation to be modeled in several ways: first, widely used statistical models of signal propagation can be used; in addition, traces of observed signal propagation can be "replayed"; lastly, manual control of signal propagation can be used to analyze behavior in artificially created situations that would be difficult or impossible to reproduce in an open system. Chapter 5 will discuss signal environment emulation in more detail.

Physical layer emulation provides an attractive middle ground between pure simulation and wireless testbeds. To a large degree, this approach maintains the repeatability, configurability, isolation from production networks, and manageability of simulation while retaining the support for real applications and much of the realism of hardware testbeds. As a result, physical layer network emulation provides a superior platform for wireless experimentation in many instances.

Physical layer network emulation is not, however, a complete replacement for simulation and real world evaluation. Simulation is still useful in cases where a very large-scale experiment is needed or in certain cases where functionality not available in hardware is required (e.g. changing the radio behavior beyond what is allowed by the hardware design or vendor policies). Real world evaluation is still useful when radio channel fidelity beyond the capabilities of the emulator is required, or for verifying the operation of emulation in real-world settings.

1.2 Architecture Overview

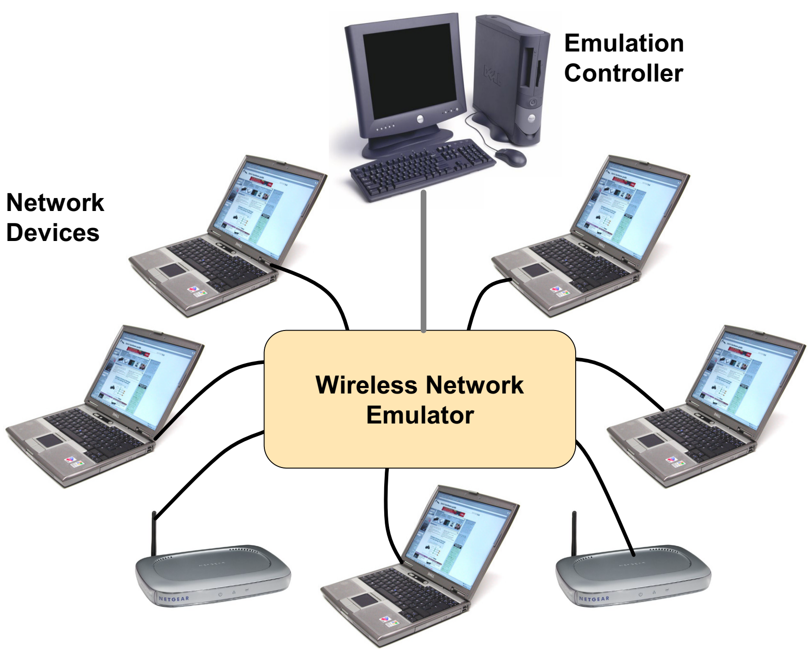

Figure 1.1: Emulator Architecture

The high-level emulator architecture is depicted in Figure 1.1. A number of "RF nodes" (e.g. laptops, access points, cordless phones, or *any* wireless device in the supported frequency range) are connected to the emulator through a cable attached to the antenna port of their wireless line cards (each RF node corresponds to a single antenna, so a single device can be represented by multiple RF nodes). Signals transmitted by each RF node are digitized, and then processed using a large FPGA to create impairments similar to those found in the real world. For each RF node, the received signal is computed by digitally combining the incoming signals from in-range RF nodes, and then converting the outgoing signal back to analog form. The RF nodes are shielded from each other so that **no communication occurs over the air**. All communication between RF nodes occurs through the signal propagation environment modeled within the emulator. That is, the emulator achieves via digital signal processing the same physical effects that occur in the real world: attenuation, multipath, interference, etc. This allows researchers to use real commercial wireless networking hardware, running on a real device, with a real OS, and real applications. Emulation is controlled by an Emulation Controller PC which models the physical environment and coordinates the movement of RF nodes in the modeled physical environment with the modeling of the signal propagation environment on the emulator hardware. Emulation occurs in real time. Thus, the emulator supports both real wireless devices and real applications operating as if they were actually in the emulated environment.

Chapter 2 discusses this architecture in more detail.

1.3 Related Work

1.3.1 Wireless Simulators

For several years now, ns-2 [38] has been the de facto standard means of experimental evaluation for the wireless networking community. Yet ns-2's wireless support has not kept pace with current technology, and is targeted towards the original 802.11 standard developed in 1997. Even this support, however, is inexact as ns-2 does not support automatic rate selection, uses a non-standard preamble, and a non-standard 802.11 ACK timeout value. In addition, ns-2's physical layer is particularly simple [59]. As a result, some researchers are opting to use commercial simulators such as QualNet [53] and OpNet [43] since they claim better support for current standards. Despite these claims, however, it is unclear how well these simulators reflect actual hardware.

More fundamentally, the real APIs, parameters, and capabilities provided by real wireless hardware are not supported by wireless simulators. Moreover, real user applications running on real operating systems are not supported using simulators.

Thus, testing simulator code in the real world frequently involves the development of two implementations: one for the simulator, and one for a real device. As a result, simulation-based research is frequently never ported to a real wireless device which hampers verification of results obtained during simulation, and lessens its impact and benefit to the community.

Some efforts have been made, however, to ameliorate this problem by creating environments that enable certain types of simulation code to be tested in real devices without modification. In particular, environments [9, 52] have been created that allow ad hoc routing protocols to be developed in ns-2 and run at user level on real hardware without code modification. Similar systems [35, 22] also allow sensor network applications to be evaluated via simulation and then deployed into sensor networks without modification. In contrast to these systems that are hand-crafted for a particular problem, physical layer emulation provides a completely general approach that is agnostic to the specifics of the system attached to it. Moreover, physical layer emulation accomplishes this without forcing users to resort to user level code and without taking shortcuts in wireless network modeling.

1.3.2 Wireless Emulators

Simulation's lack of support for real devices can be overcome by combining elements of simulation with real devices. This combination of real devices and simulation is referred to as emulation. Emulation - the combination of real hardware endpoints with simulated network communication - has proven to be a useful technique in wired networking research [65, 17, 62], and it has an even larger potential in the wireless domain.

A common approach that has been adopted for wireless emulation [42, 37, 36] is to capture the behavior of a wireless network in terms of parameters such as capacity and error rates and then use a wired network to emulate this behavior. This has the advantage of allowing the use of real endpoints running real applications in real time. The wireless MAC and physical layers, however, are only very crudely simulated. For this reason, it is unclear whether or not this approach can obtain more realistic results than pure simulation.

RAMON [23] uses three programmable attenuators to allow emulation of the signals between a single mobile node and two base stations. While useful for the intended application of mobile IP roaming investigation, the inability to independently control all signal paths severely limits this approach.

1.3.3 Wireless Testbeds

More recently, several efforts such as Emulab [64], WHYNET [61], Orbit [50], and MiNT [12, 11] have begun using controlled wireless testbeds.

The emulator's use of real devices and real applications is similar to that found these testbeds such as Emulab [65], Orbit [50], Modelnet [37], or Mint-b [11]. Like these systems, the emulator supports real devices running, and real applications.

The emulator and these systems, however, differ greatly in both their mechanism for modeling signal propagation environments and the range of environments that they are capable of modeling. Like the emulator, Emulab supports real wireless devices, but in a testbed fashion; i.e. the wireless network is hardwired to a particular physical location. Orbit is also a testbed, but attempts to emulate different physical environments through selection of nodes positioned in a grid and the injection of noise to alter signal-to-noise ratios and mimic the effect of path loss. The techniques used in this approach are very different from those that can be leveraged in the emulator. While injecting noise can provide the primitive ability to modify the signal environment, it lacks the power of physical layer emulation in directly and completely specifying the signal propagation environment. Mint-m and Emulab also use robots to introduce mobility; while this supports mobility, the emulated signal environment cannot be altered and is still bound to the room in which the robots reside. Earlier work [30] proposed using coaxial cables to hardwire a custom signal propagation environment. While useful for very limited experiments, physical layer emulation allows for vastly improved flexibility, scale, mobility and complete signal control. Modelnet, makes use of real end hosts, but does not make use of real wireless devices, preferring to simulate their behavior.

The wireless network emulation developed in this work is - in some ways - a hybrid of traditional network simulation and emulation. Like other systems, the emulator uses real devices; unlike other systems, however, the emulator is capable of full control over the physical signal propagation between the devices attached to the network. This is enabled by the unique emulator hardware which contains a powerful digital signal processing platform capable of digitizing and manipulating signals streaming

Moreover, the emulator is more general than typical wireless network simulators. ns-2's wireless support, for instance, was originally developed with the assumption that ad hoc routing was the problem of interest; the emulator makes no assumption about the problem or even the devices attached to it (as long as they fall in the supported frequency band.) Thus mixing devices of different technologies is supported with no additional effort. In contrast, a traditional packet simulator with n different types of devices will need to resolve the $O(n^2)$ possible interactions between these devices. In addition, the emulator removes the assumption that a physical world is necessarily being modeled. That is, the emulator allows users to directly control the signal propagation channels between devices without necessarily considering a physical world. This is similar to "multipath fading simulators" [15, 56] discussed below that are commercially used to evaluate RF devices.

1.3.4 Channel Emulators / Fading Simulators

The most functionally similar approach to the technique introduced in this thesis is provided by channel emulators [26, 2, 4, 6, 15, 56, 8] (also known as fading simulators.) The goal of these emulators, however, is quite different. As the name of these devices implies, rather than supporting emulation of all channels in a wireless network, commercial channel emulators [15, 56] are designed to support very fine-grained emulation of the wireless channel between either a pair of devices or between a small number of base stations and a small number of mobile devices (with the total of both typically being less than 8.)

In contrast to the telecommunications networks typically targeted by channel emulators, wireless data networks are frequently contention-based with half-duplex devices. Channel emulators lack direct support for half-duplex nodes and require external components to support half-duplex nodes. As a result, while channel emulators are very useful for equipment vendors evaluating a new device, the limited scale, lack of support for complete interaction between all nodes, and high cost make commercial channel emulators an unattractive option.

1.3.5 Analog Network Emulator

Azimuth Systems produces an analog network emulator [57] that provides very coarse grained network level emulation. In their analog emulator, RF devices are connected to programmable attenuators which are interconnected in a star topology. While this technique provides network level testing, the star topology severely limits the signal propagation topologies that can be emulated. In addition, the use of programmable attenuators precludes the support for multiple signal paths and limits the resolution at which fading can be emulated.

1.3.6 Software Radio

The physical layer wireless emulator presented here bears several similarities to (wideband) software radios [34, 5]. In both systems, wideband RF signals are digitized and then manipulated in a programmable manner and vice versa. The manner in which the signals are manipulated, however, is very different. Software radios send and recover digital data which requires a great deal of filtering to select signals in addition to modulation/demodulation logic. A network emulator, on the other hand, is agnostic to the format of signals passing through it. Thus the operations performed are more straightforward. In spite of this, however, the scale of the operations required imposes large computational demands on the emulator.

1.4 Challenges

Physical layer network emulation must overcome several difficult challenges in order to be a viable experimental methodology. This section briefly discusses several of the most difficult obstacles. Subsequent chapters will discuss how these challenges are addressed by this work.

1.4.1 Bandwidth

Traditional wireless channel emulation has focused on telecommunication market where the bandwidth in use is fairly limited. Wireless data networks - in contrast - use large swaths of bandwidth. For example, the 802.11b/g band utilizes 72 MHz of bandwidth in the United States. Digitizing this entire band results in a digital data stream with a data rate of approximately 2 Gbps, and requires approximately 170 million scaling operations per second per channel.

Moreover, emulating an entire network requires not only emulating a single channel, but emulating all channels in use in that network as discussed above. For example, a ten node network has 90 channels; for the 802.11b band discussed above, this means streaming approximately 180 Gbps of data to/from the signal processing unit, and performing 15 billion scaling operations per second. Thus, signal conversion and processing must operate at a very high sample rate in order to support the emulation of an entire network.

1.4.2 Dynamic Range

Dynamic range refers to the difference - usually measured in dB - between the strongest and weakest signals that can be emulated. Commodity signal conversion components operate on digital samples of signals of n-bits resulting in 2^n represented signal strengths. The represented signal strengths are evenly spaced in volts in the case of the A/D converters and amps in the case of the current source D/A converters.

The sample size n determines the dynamic range d in dB as follows: $d = 20 \cdot log_{10}(2^n)$. Thus, in approximate terms $d = 6.02 \cdot n$. In practice, data converter imperfections result in a dynamic range less than this theoretical upper bound.

While it is desirable - from the standpoint of dynamic range - to increase the sample size, doing so imposes requirements on the A/D converters, the D/A converters, and the digital signal processing system. In particular, the resolution of available A/D converters drops rapidly as sampling rate increases. Thus, there is a tradeoff between supported bandwidth and dynamic range.

1.4.3 Signal Integrity

Processing analog signals inevitably degrades them in some undesired manner. There are numerous factors that may harm signal integrity. Several of the most important factors are briefly discussed here. These are covered in more detail in Chapter 6.

Noise. Every component that touches an analog signal introduces some noise into it. In addition, quantization introduces signal noise.

Intermodulation. Several stages of signal conversion introduce undesired intermodulation products where signals mix either with the local oscillator (LO) or each other in undesired ways.

Phase Noise/Jitter. Jitter in the digital system clock, jitter in the LOs, plus data converter aperture uncertainty produces phase noise in the signals processed by the emulator. In certain circumstances - such as undersampling - these can place an upper bound on the achievable signal-to-noise ratio.

1.4.4 Latency

In the real world, physical distance introduces latency in signal propagation. In certain situations, it is useful to emulate this latency. This would allow the investigation of effects related to time of propagation. This is challenging since it requires delaying the signal in a controlled fashion over a potentially wide range of values. Given the large data rate of each data stream, the data storage requirements for latency emulation make it difficult to support large delays.

Moreover, latency in the signal processing unit must be kept to a minimum since the minimum latency required for the emulator to receive a signal from an RF Node, process the signal, and send it to its destination RF Node imposes an effective “minimum distance” on time of flight emulation. That is, even if nodes are co-located in an emulated physical environment, there will be an artificial latency added to transmissions as if the nodes were actually located at a greater distance from each other. For many wireless protocols, however, the artificial latency introduced by emulation is not significant.

1.4.5 Scale

Arguably the most difficult challenge facing physical layer network emulation is that of scale. Physical layer network emulation fundamentally requires physical network devices. In the simplest implementation, one physical device must be present for every emulated device. Under this approach, the scale of the emulated network is limited by the number of devices that can be procured and included in the emulator. This straightforward one-to-one mapping of physical and emulated devices is the approach used in this thesis; it is possible, however, to employ various techniques to allow one physical device to support multiple emulated devices.

A much more significant scaling challenge arises when all-to-all connectivity needs to be emulated. The real-world interaction of n devices that are in communication range of each other grows with complexity $O(n^2)$: specifically, for n devices in communication range of each other $n \cdot (n-1)$ channels must be emulated. Even modeling a single channel requires a large amount of computation; thus the number of channels required quickly limits the amount of all-to-all communication that can be emulated with a physical layer network emulator.

1.4.6 Internal Isolation

A fundamental requirement of physical layer network emulation is that no direct over-the-air communication or interference occur. This thesis refers to this requirement as "internal isolation." Without this requirement, nodes would never be out-of-range of each other, and undesired multipath effects would occur. Achieving internal isolation is difficult since receivers are designed to be sensitive to very weak signals. For instance some commodity receivers can decode signals as weak as $1 \times 10^{-13}W$. Thus achieving sufficient internal isolation is a difficult task and requires very meticulous design.

1.4.7 External Isolation

Signals generated external to the emulator should not be receivable by or affect the operation of devices inside the emulator. Similarly, signals generated inside the emulator should not affect devices outside of the emulator. This thesis collectively refers to these properties as "external isolation." Ideal external isolation allows the emulator to be co-located with a production wireless network without affecting its performance and vice versa. In practice, only a certain degree of isolation can be guaranteed. Fortunately, it is possible to guarantee enough isolation so that external isolation is achievable for most realistic scenarios. That is all devices that are not within close physical proximity to the emulator will be isolated from it.

1.4.8 Hardware

Physical layer wireless network emulation requires a powerful distributed hardware architecture that must be carefully designed and implemented. Many aspects of designing this hardware are technically challenging.

In the analog domain, interfacing with the wireless card requires the ability to manipulate microwave frequencies. Microwave circuit boards are difficult to design and implement correctly; they are not easily amenable to correction after fabrication and assembly.

In the digital realm, the most significant challenge is synchronizing the large amount of incoming data from the data converter units to a common digital clock on the central signal processing unit without introducing additional latency.

1.4.9 Software Infrastructure for Managing Experiments

The custom hardware developed for the emulator must be controlled in such a way as to mimic the effects of signal propagation in a real environment; this must be done in real time.

Moreover, the software infrastructure must be designed in a way that enables novice users to easily execute basic experiments. Advanced users must be able to control the full range of the emulator's functionality and completely control the wireless devices in synchrony with the environment emulation.

1.5 Thesis Statement

Physical layer network emulation is a powerful technique for wireless experimentation that simultaneously achieves realism from the physical layer through the application layer while providing precise physical layer repeatability. This enables fair side-by-side experimental comparisons; a capability that is extremely difficult to achieve using other realistic techniques. This technique can be feasibly implemented using commodity components.

1.6 Contributions

This thesis presents several significant contributions to the research community.

New Experimental Methodology. This most significant contribution of thesis is the introduction of a new research methodology: physical layer wireless network emulation. This technique enables wireless experiments to be conducted in a controlled and repeatable environment while maintaining much of the realism of wireless testbeds. Moreover, the complete control over the signal propagation environment enables experiments to be conducted that could not be conducted in a wireless testbed or a wireless simulator.

Hardware Architecture and Implementation for Wideband Wireless Network Emulation. This thesis presents solutions to the problems necessary to develop a practical wireless network emulator. At the hardware level, this thesis presents an architecture for performing physical layer wireless network emulation. In addition, as this work includes functional implementations for the components in this architecture that overcome key challenges such as the synchronization of distributed data conversion components.

Software Architecture for Physical Layer Emulation. At the software level, this thesis presents an architecture for controlling the hardware-based emulation in real time, as well as an implementation of that architecture.

Trace Capture and Playback. This work also introduces a simple method for recording, processing, and playing back coarse-grained channel traces using commodity wireless hardware. This technique is shown to obtain good results in low a delay-spread environment.

Functional Emulator. Moreover, this work has resulted in fully functional emulator that will continue to be used for wireless research. This emulator will also serve as a template for the development of emulators for use by other researchers.

Device and Link-level Analysis. This thesis presents an investigation into 802.11b device and link-level behavior. The results of this investigation should enable a better understanding of the behavior of 802.11b devices by replacing convention with actual measurement. This data should be particularly useful in developing more accurate wireless simulators.

Access Point Selection. Finally, to demonstrate the ability of physical layer emulation to yield insight into wireless network behavior this thesis analyzes 802.11 access point selection for a popular wireless platform: Madwifi-0.9.1. Emulation-based experiments quickly show that Madwifi-0.9.1 performs poorly in changing signal environments, and that emulation enables the rapid development of an improved driver. Moreover, observations gained during this analysis result in a proposed extension to the 802.11 protocol that improves 802.11 access point selection and roaming performance.

1.7 Scope of Work

Physical layer wireless network emulation is a broad topic with many interesting problems. This thesis addresses a specific subset of the possible problems that could be addressed. In this section, we clarify issues that - while interesting problems to solve - will not be addressed by this thesis.

1.7.1 Signal Environment Emulation

The signal environment emulation discussed Chapter 5 of this thesis employs a variety of techniques. The majority of these techniques have been known in the literature for many years and this thesis does not attempt to extend them further. In particular, this thesis does not introduce new methods for analytical wireless channel modeling. This thesis does, however, introduce a simple method for recording actual wireless channels using commodity hardware and replaying them in the emulator.

1.7.2 Architectural Enhancements and Alternatives

The emulator architecture and implementation presented in this thesis are designed to be useful in a variety of configurations. The majority of this thesis will, however, use only a single configuration of the emulator. Chapter 2 discusses how the emulator's architecture and implementation support additional configurations, but the actual use of these configurations is left as future work. In particular, this thesis will not examine:

Implementations Using Multiple DSP Engines. Multiple DSP engines can be employed to increase the number of emulated channels and nodes that can be supported. The implementation presented in this thesis is designed to allow two DSP engines to be used. The architecture's support for multiple DSP engines is discussed; this thesis does not, however, evaluate actual implementations using multiple DSP engines.

I/Q Implementations. It is frequently useful to break an ordinary scalar signal up into two parts "I" (cosine) and "Q" (sine). The signal processing system then operates on a vector signal representation. The architecture and implementation discussed here are designed to support this, but I/Q implementations are not evaluated as part of this thesis.

Time Multiplexing. One approach to scaling the number of devices attached to the emulator is to multiplex signal processing in time. This technique is particularly useful for low bandwidth devices such as sensor networks. This technique is not considered in this thesis.

Additional Architectural Alternatives. There are numerous alternative emulator architectures that may be useful beyond those listed above which will not be addressed other than to present the rationale behind current design decisions.

1.8 Outline

The remainder of this thesis proceeds as follows. Chapter 2 presents an architecture for physical layer wireless emulation; Chapter 3 then discusses the implementation of that architecture developed in this thesis. Chapter 4 discusses the control software used to coordinate and manage the emulator. Chapter 5 discusses techniques for emulating signal propagation environments in a physical layer emulator. Chapter 6 presents tests verifying and quantifying the emulation integrity of the implementation presented in this thesis. Chapter 7 uses physical layer emulation to measure 802.11b link and device behavior. Chapter 8 uses physical layer network emulation to analyze the problem of 802.11 access point selection and roaming. Chapter 9 concludes this thesis.

Chapter 2

Architecture

Section 1.2 briefly sketched a high-level view of the architecture of the emulator. This chapter discusses this architecture in detail.

2.1 Requirements

Before discussing the architecture of the emulator, this section first considers the requirements that the emulator is designed to meet. Several of these were touched on earlier in Section 1.4. In this section, specific targets are defined for many of the items discussed earlier.

Real-time Digital Signal Processing. In order to support real devices, the emulator must digitize incoming signals from devices, process the resulting signals, and convert these signals back to analog in real time.

Low Latency. Moreover, signals must be processed with as little latency as possible since latency incurred while processing signals imposes a minimum latency on the signal propagation between devices attached to the emulator. In the real world latency is directly proportional to distance between devices. Any latency added by the emulator creates an artificial minimum "distance" between devices. Thus, we desire to keep latency as low as possible.

Note that typical protocol level behavior - e.g. 802.11 ACK timeout - is not affected until tens of microseconds of latency are added. Physical layer emulators should be able to attain a minimum latency of under 100 nanoseconds, thus protocol-level artifacts should not result due to increased latency.

In certain cases, the absolute latency requirements discussed above are less important than the relative latency between devices. For example, consider two channels *A to C* and *B to C* involving three devices A, B, and C. The relative difference in the latency between channels $A\ to\ C$ and $B\ to\ C$ may be more important than the absolute latencies. In cases such as this, the relative latency can be modeled without encountering the fundamental artifacts that are introduced into absolute latency modeling.

Internal Isolation. Devices attached to the emulator must be isolated from each other. The targetted class of devices has transmit power as high as 19 dBm and receive sensitivity as low as -95 dBm. Hence at least 114 dB of internal isolation is required between nodes. An additional margin of 7 dB, however, is required to avoid leaking signals under the reception threshold from interfering with weak desired signals. Thus, the emulator requires 121 dB of internal isolation.

External Isolation. External isolation with respect to typical ISM band devices located in close proximity to the emulator is necessary to avoid uncontrolled transmissions from affecting emulator operation. The desired target, in this case, is to achieve isolation from 19 dBm devices - a somewhat strong WLAN transmitter - located at least 5 meters away from the emulator and separated by one 10 dB partition (a wall.) The intuition behind the 5 meter goal is that devices closer than that should be located in the same room as the emulator and hence under the control of emulator operators. Assuming a receive sensitivity of -95 dBm and a desired margin of 7 dB beyond that yields a required maximum received signal strength of -102 dBm.

Now the required isolation i is: $i = 19dBm - -102dBm$ or 121 dB. To determine the path loss naturally present in the target environment the log distance path loss model in [49] is used: $PL(d) = PL(d_0) + 10nlog(d/d_0)$. Adding 10 dB for the partition loss, using a typical value for $PL(d_0)$ at $d_0 = 1m$ of 40 dB, and assuming free space loss where $n = 2$ gives $PL(d) = 40dB + 10 * 2 * log(5/1)$ Thus $PL(d) = 54dB$; i.e. the targeted environment provides 54 dB of isolation. To compute the additional isolation that must be provided by the emulator packaging we subtract the isolation provided by the environment from the total isolation required. Hence $requiredisolation = 121dB - 54dB$ or 67 dB of additional isolation is required to achieve the external isolation goal

Half-duplex Device Support. Most wireless LAN devices are half-duplex. Transmission and reception cannot occur simultaneously. Traditional channel emulators must employ external components to support half-duplex devices. This approach can limit the performance of the emulator as these external components have limited isolation which results in a restricted dynamic range. As most devices attached to the emulator are likely to be half-duplex, these devices must be supported without the use of external components and degradation of performance.

Wide Bandwidth. Many wireless networks utilize multiple channels, and many interesting experiments require multiple channel support. Thus, the emulator must support a wide enough bandwidth to allow for multiple-channel experiments. The goal for this thesis is to support emulation of all US 802.11b channels. These channels range from 2.401 GHz to 2.473 GHz. Hence, the emulator must support 72 MHz of bandwidth.

Scale. The initial target in this thesis is to produce an architecture capable of supporting 15 devices with provisions for higher numbers of devices. The implementation discussed in this thesis consists of 8 devices.

2.2 System Architecture

Developing a physical layer wireless network emulator is a difficult task requiring careful design. The massive computation, low latency, high isolation, and support for diverse experimental setups require a power, flexible architecture.

Traditional channel emulators have utilized a monolithic ASIC-based design. In these designs analog signals from transmitters are routed via coaxial cable to a central emulator unit. Inside this unit the signals are digitized, channel emulation occurs, and then the resulting channel analog output signals are routed out via coaxial cables to receivers. Some advanced emulators provide digital inputs and outputs for use with digital baseband devices.

At an abstract level discussed in Chapter 1 and shown in Figure 1.1, the architecture of the wireless network emulator is similar to the traditional monolithic approach. Wireless devices - “RF nodes” - are connected to the emulator; all communication between RF nodes occurs through the signal propagation environment modeled within the emulator. While sufficient for channel emulation, this traditional architecture is less than ideal for network-wide emulation. Routing a large number of analog signals to a small physical space makes the problem of isolating these signals from each other extremely difficult.

Instead, this work presents a distributed emulator architecture, as shown in Figure 2.1. On transmit, the RF signal from a given RF node is passed into the RF Front End where it is shifted down to a lower frequency. This lower frequency signal is then digitized by the Signal Conversion Module, and forwarded in digital form into a central DSP Engine that is built around one or more FPGAs. By digitizing signals in a distributed fashion, there is no centralized isolation problem to solve in the DSP Engine. All signals exiting the Signal Conversion Modules for transmission to the DSP Engine (and vice versa) are digital.

The RF Front End is distinct from the Signal Conversion Module since RF components are frequency specific. Keeping the RF Front End separate allows for different RF bands to be supported by swapping RF Front End cards without the need to construct an expensive multi-band RF Front End (unless desired.)

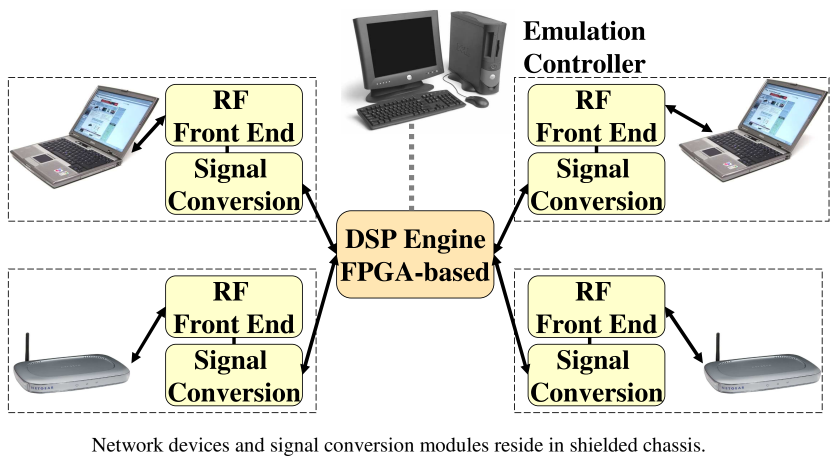

Figure 2.1: Detailed Emulator Architecture

The DSP Engine models the effects of signal propagation (e.g. large-scale attenuation and small-scale fading) on each signal path between each RF node. For each RF node, the DSP Engine combines the processed input signals from all the other RF nodes. For each RF node, the resulting signal is then sent out to the signal conversion module which converts the digital signal back to a radio signal. It is then sent to the wireless line card through the antenna port. Using multiple DSP Engines, larger systems can be built. Using an FPGA-based architecture yields the massive signal processing power necessary to perform network-wide emulation. At the same time, this programmable logic provides flexibility in the allocation of resources.

The Emulation Controller controls RF node movement in the emulated physical environment as well as application behavior on the end hosts. Movement within the physical environment is emulated in real time; the controller coordinates this movement with the real-time modeling of the signal propagation environment. Each RF node runs a small daemon that allows the Emulation Controller to control its operation via a wired network. A proxy can be used for nodes that cannot run the daemon themselves.

Connecting the Emulation Controller to an external network allows remote management of the emulator. In addition, individual nodes in the emulator may be connected to external networks in order to allow emulator nodes access to the Internet at large or to allow the emulator to be used in conjunction with testbeds such as PlanetLab [46], Emulab [65], or Orbit [50].

2.2.1 Provisions for Future Improvements

The architecture discussed here, and the implementation discussed in the next Chapter, have been enhanced to provide provisions for future modifications. This section discusses two of these provisions that utilize multiple DSP Engines: in the first case to enable improved scale, and in the second to provide additional signal processing power.

Improving Scale with Multiple DSP Engines

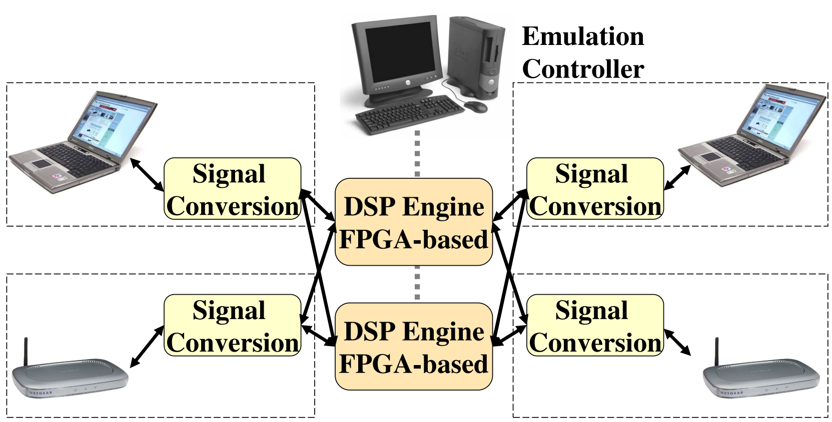

Figure 2.2: Modified Architecture for Increased Scale

The basic architecture presented above is limited by the internal resources of the FPGA in the DSP Engine. In most circumstances, the scale of all-to-all connectivity that can be supported is limited by the number of multipliers on the FPGA since one multiplier is typically required per signal path. In the current DSP Engine implementation, a total of 15 wideband RF nodes can be supported.

To grow beyond the limits of a single DSP Engine, the modified architecture shown in Figure 2.2 may be used (in this figure and in the following figure the RF Front End is not shown since it is co-located with the Signal Conversion Module). In this architecture, each Signal Conversion Module is connected to multiple DSP Engines. The figure depicts the case where the number of DSP Engines is two; for simplicity, four RF Nodes are shown.

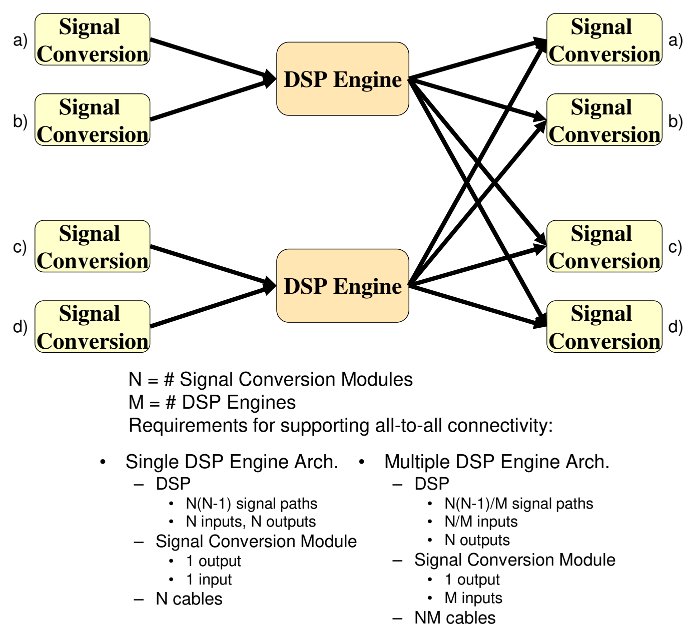

Figure 2.3: Multiple DSP Logical Architecture

Consider Figure 2.3 which shows the logical flow of signals in a multiple DSP Engine Architecture (again a two-DSP Engine, four RF Node case is shown) from source Signal Conversion Modules on the left to destination Signal Conversion Modules on the right. Each physical Signal Conversion Module appears once on the left-hand side as a source and once on the right-hand side as a destination. In this architecture, the output from each Signal Conversion Module is connected to only a single DSP Engine. An input into each Signal Conversion Module, however, is required for each DSP Engine.

The key benefit of using multiple DSP Engines is an increase in the number of paths supported that is directly proportional to the number of DSP Engines used. In a single DSP Engine implementation, if N RF Nodes are attached to the system, the number of paths that must be provided by the DSP Engine in order to support all-to-all connectivity is $N \cdot (N-1)$. Now in the multiple DSP Engine case, if the number of RF Nodes attached to the system is N and the number of DSP Engines utilized is M, then the number of paths that each DSP Engine must support is then $N \cdot (N-1)/M$. In other words, the number of paths available in the system scales linearly with the number of DSP Engines used.

The fundamental complexity of all-to-all connectivity, however, is manifested in increased requirements for DSP Engine I/O, increased numbers of cables, and increased Signal Conversion Module I/O. These requirements limit the multiple DSP Engine approach to a few DSP Engines. Nevertheless, while this approach is not a panacea, it can yield a useful increase in system scale.

Interconnecting Multiple DSP Engines

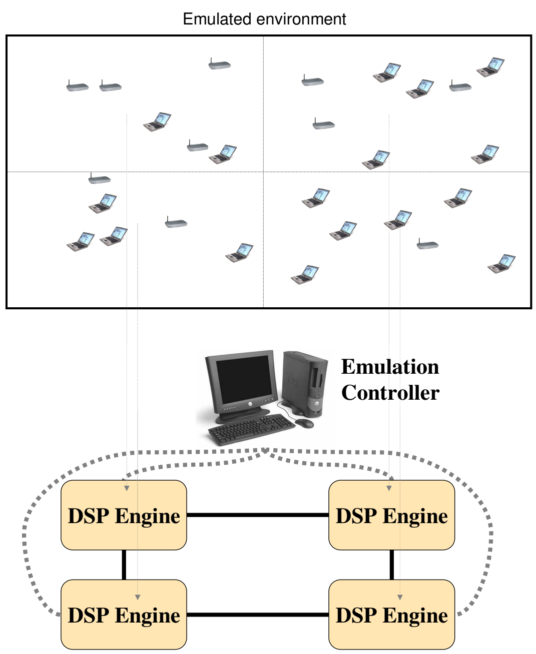

Figure 2.4: Connecting Multiple DSP Engines

In many circumstances, nodes in wireless networks are physically separated to the extent that they are not all in communication range of each other. It is possible to leverage this physical distribution to allow the emulator to scale beyond the limits of a single DSP Engine.

Figure 2.4 depicts how a two dimensional distribution of wireless nodes could be leveraged by an emulator using multiple DSP Engines. In this architecture, each DSP Engine is responsible for fine-grained emulation of nodes within a unique region of physical space. DSP Engines communicate with neighbors to handle borderline cases or distant emulation at a coarser scale.

Quadrature Architecture

The current architecture and implementation operate on scalar signals in order to ease Signal Conversion Module design and improve scale. Radios typically process signals by creating a phase shifted copy of the signal. This is referred to as I/Q (cosine-sine) or quadrature signal processing. Processing signals in I/Q form is simpler than scaler form for many operations.

In addition, if I/Q conversion is done in the analog domain, using I/Q signals allows the signals to be digitized at baseband instead of at a low IF as is done in the standard emulator architecture. Thus, an analog I/Q architecture effectively doubles the emulator's supported bandwidth at the expense of doubling the required signal paths.

There are a variety of ways in which I/Q modulation can be supported. The current implementation can implement I/Q modulation by digitally decomposing and recovering the signal into I/Q either on the Signal Conversion Module or the DSP Engine. This has the advantage of requiring no changes in the current implementation, but consumes more resources since the number of signal paths doubles.

More scale can be achieved using a modified architecture to better support I/Q modulation as shown in Figure 2.5. In this architecture, two DSP Engines are used: one for I and one for Q (this provides twice the computational resources compared to handling both I and Q in a single FPGA.) These DSP Engines operate in symmetry, but on different portions of the signal. The current DSP Engine can be used in this case. The I/Q signals can be created and recovered either digitally as mentioned above, or in the analog realm by modifying the RF Front End and using two Signal Conversion Modules per RF Node.

Multiple Antenna Support

In the architecture discussed in this chapter, each Signal Conversion Module is connected to a single device antenna port via an RF Front End. Supporting true antenna diversity under this architecture requires using two Signal Conversion Modules. This approach works without requiring any modification to the emulator hardware or software, but sacrifices scale. In the implementation discussed in the next chapter, true antenna diversity is not directly supported in favor of increasing emulator scale. The effects of antenna diversity, however, could be emulated by altering channel modeling to consider diversity. In this approach, each device would have two (or more) virtual antennas for each actual antenna port. The channel modeling would then be altered - e.g. by reducing the depth of fading - to model the impact of antenna diversity.

Antenna diversity support could be added at the hardware level - without requiring

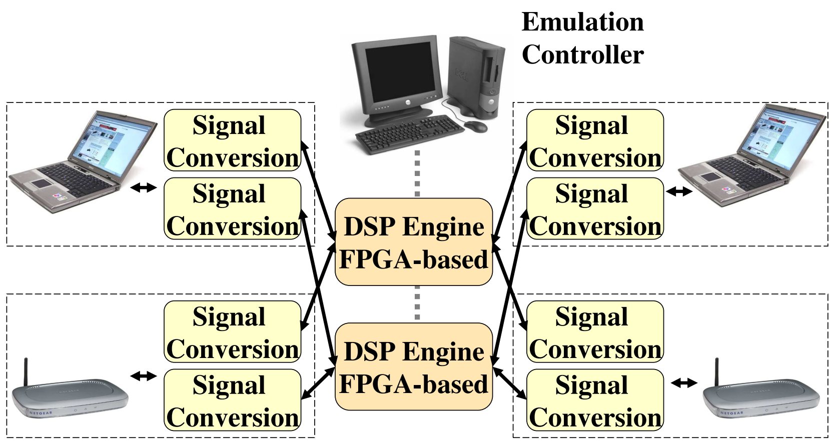

Figure 2.5: Modified Architecture for I/Q Signal Processing

one Signal Conversion Module per antenna - by altering the Signal Conversion Module to support multiple RF Front Ends per Signal Conversion Module. A single digital data stream to and from the DSP Engine would still be used in order to maintain scalability. For receive diversity, signals outgoing to each element would be altered to mimic the effects of diversity in the real-world. Thus, the hardware diversity circuitry would still be exercised, but scaling would not be any different than the non-diversity case. Transmit diversity could be handled in a similar fashion by altering incoming data streams.

Emerging and future networks will increase throughput by using multiple antennas to support techniques such as steerable antennas, MIMO [19], and "time reversal" [18]. Like antenna diversity, these technologies can be supported through either the brute force approach of using one Signal Conversion Module/RF Front End per antenna, or by modifying the Signal Conversion Module to support multiple RF Front Ends. In the long run, the latter technique of developing Signal Conversion Modules with multiple RF Front End support appears to be the most promising approach.

2.3 Summary

This chapter has presented an architecture that enables physical layer wireless network emulation, discussed this architecture's benefits, and outlined its provisions for growth. The following chapter discusses the hardware implementation of this architecture, and illustrates concretely how system requirements are met.

Chapter 3

Implementation

The previous chapter discussed the emulator architecture and implementation at a high level. This chapter now discusses each component of the emulator in detail. For each component, this chapter first discusses the requirements that it must meet; the implementation of the component is then discussed in detail.

3.1 Implementation Versions

The work presented in this thesis was conducted using three increasingly capable implementations. These implementations will be referred to as "Prototype", "Version 1", and "Version 2." When not otherwise specified, the implementation referred to is Version 2 which is the version described in this chapter. The architecture is the same for all three versions.

3.2 RF Front End

3.2.1 Requirements

The RF Front End converts the high frequency signals transmitted by the device to a low frequency that can be digitized by the Signal Conversion Module. When the device is not transmitting (the emulator targets half-duplex devices), signals from other devices in the emulated environment must be converted from the low frequency signals produced by the Signal Conversion Module to the high frequencies that the signals were originally transmitted at.

In addition, the RF Front End must meet the following requirements:

Signal Scale. The RF Front End must ensure that transmitted signals are presented to the Signal Conversion Module at an ideal signal strength: as close as possible to the maximum signal strength supported by the Signal Conversion Module, but not greater.

Similarly, received signals must be attenuated in order to place the received signal at the desired range of signal strengths.

Flatness. At each frequency supported by the RF Front End, the scaling introduced by the RF Front End should be as close to equal as possible. That is, the transmit frequency response should be flat as should the receive frequency response.

Noise. The RF Front End should introduce as little noise as possible into the signal.

Spurious Signals. Finally, the RF Front End must produce as little spurious signal energy as possible.

3.2.2 Implementation

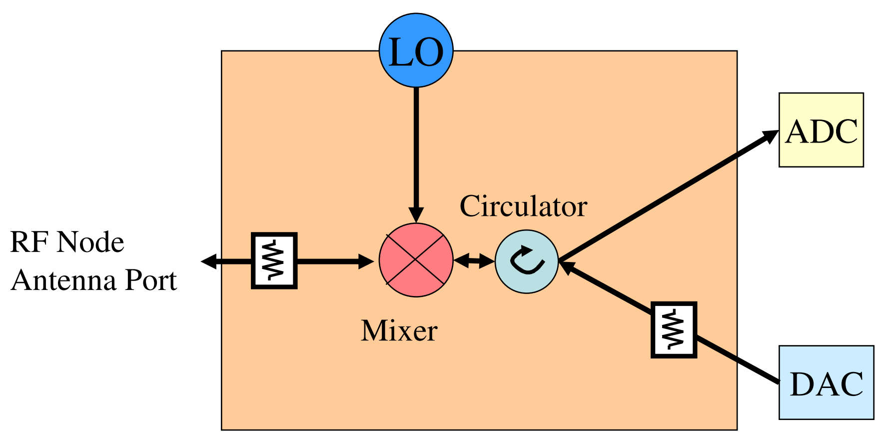

Figure 3.1: Strawman RF Front End Implementation

Strawman Design.

The main signal paths in the RF Front End are 1) from the RF port of the RF Node to the A/D port of the Signal Conversion Module, and 2) from the D/A port of the Signal Conversion Module to the RF port of the RF Node. Conceptually, the RF Front End could be implemented as shown in Figure 3.1. This figure shows the RF

Front End in a shaded box; the RF Front End connects the RF Node's antenna port to the ADC and DAC which are located in the Signal Conversion Module. An off-board local oscillator (LO) signal source is used. In this implementation a single "mixer" is used. On downconversion, the mixer shifts the high-frequency signal generated by the RF Node down to a lower frequency as required by the Signal Conversion Module. On upconversion, the mixer shifts the low-frequency signal generated by the Signal Conversion Module up to the high-frequency that it was originally transmitted at. As the RF Node may be half-duplex, these two paths must share the same RF Port. The architecture in Figure 3.1 solves this problem using a "circulator." This device sends signals transmitted by the RF Node to the downconversion path; signals received from the Signal Conversion Module are sent to the RF Node. In principle, no signal would leak between the two signal paths.

Unfortunately, this design has shortcomings.

- Isolation - the limited isolation of the circulator means that some signals will go in undesired directions. For example, some signal coming into the upconversion path from the Signal Conversion Module will leak through the circulator onto the downconversion path. This signal will then enter the Signal Conversion Module. The net effect during emulator operation would be to make signals being *received* by one RF Node appear as if there were being *transmitted* - weakly - from that RF Node.

  In addition, the LO distribution system can be an additional source of signal leakage. Some of the signal being transmitted will leak into the LO distribution system and arrive at another RF Front End where it will be sent for reception.

- Lack of undersampling support - the use of a single LO for both upconversion and downconversion in this design effectively precludes the use of undersampling. The use of undersampling is discussed further below.

- Signal Matching/LO Strength - The downconversion output signal strength should match the ADC input range. In practice, however, the input signal to the mixer has a limit which prevents a good match. Using a higher powered LO can achieve a better match, but such a high powered LO can be difficult to achieve.

- Spurs - All mixers produce spurious signals known as "mixer spurs." These become more severe as the mixer input signal strength approaches the mixer signal limit. In the strawman design, the input signal needs to be as close to the limit as possible to allow for a good ADC input range match. Thus, this design cannot simultaneously achieve good isolation from spurs and good ADC input range match.

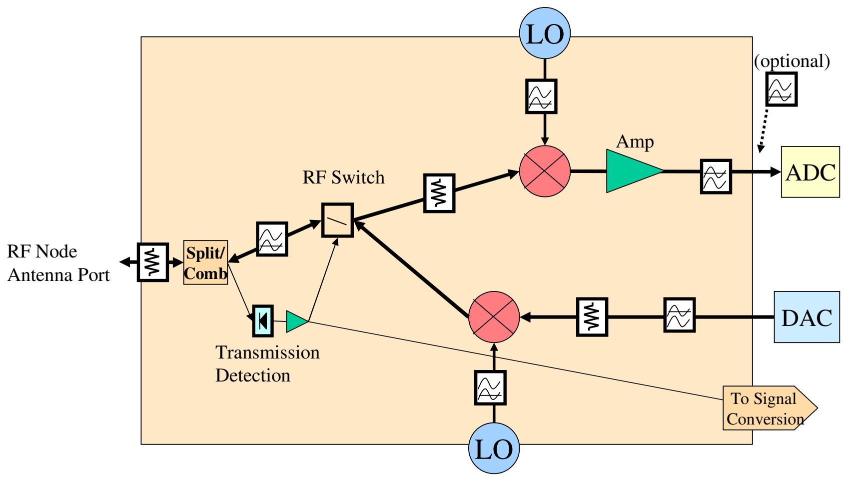

Figure 3.2: Actual RF Front End Implementation

Actual Implementation.

Developing an RF Front End to overcome the shortcomings of the simple design discussed above is a challenging problem, with a large design space in which to look for possible solutions. This discussion presents an attractive solution to this problem - shown in Figures 3.2 and 3.3 - that meets the design requirements, and provides good performance. Other solutions are, however, possible. For each of the shortcomings discussed previously, this discussion will elaborate on how the design in Figures 3.2 and 3.3 overcomes the challenge, and why each particular design element was selected.

Improving Isolation. A key problem in the simplistic design is the lack of isolation, particularly in the circulator. As the emulator targets half-duplex devices, isolation between upconversion and downconversion paths can be greatly improved by using an RF switch instead of a circulator. (Full-duplex devices can still be supported using two RF Front Ends with one dedicated to each direction.) The RF switch must be very fast to detect transmissions so as to truncate as little of the transmitted signal as possible; it also must recognize the end of a transmission quickly, though this is not as critical as recognizing the start of a transmission. The use of an RF switch requires a transmit detection circuit that controls the switch. This detection circuit must also be fast for the same reasons that the switch must be fast.

Leakage through the LO distribution system is limited as follows. First separate mixers and LO distribution systems are used for downconversion and upconversion. This allows undersampling to be supported as discussed below. Moreover, this

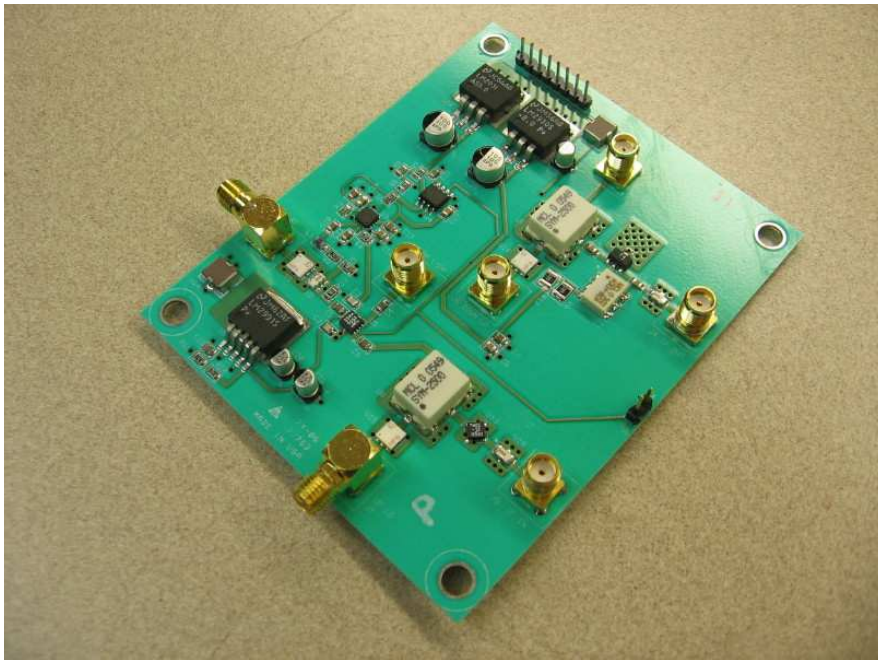

Figure 3.3: RF Front End Picture

eliminates leakage between the upconversion and downconversion LO systems. To further reduce leakage, the LO circuits contain high-pass filters that eliminate spurious low-frequency signals that result from mixing and that can be mixed back to high-frequency inadvertently.

Signal Scaling. On upconversion, the signal must be scaled into the desired dynamic range. Upconversion signal scaling can easily be accomplished by adding attenuation to the IF input port. Attenuation here affects only the upconversion path and improves spurious signal isolation.

On downconversion, in order to match the required Signal Conversion Module input level, an on-board jumper cable is added on the downconversion path. This allows variable amounts attenuation to be added in four places on the RF Front End: the RF Port, the IF input, the IF output, and the on-board jumper loop. As discussed below, rather than add attenuation to the IF output port, it is preferable to add attenuation to the loop. Hence attenuation is never added to the IF output port.

Reducing Spurs. Another important problem that must be solved is the reduction of "mixer spurs" on the downconversion path. On downconversion a mixer works by multiplying a local oscillator signal (LO) with an RF signal. This results in the desired signal of $RF - LO$ being output from the IF port. Unfortunately, however, the mixer also outputs numerous combinations of $m \cdot RF + n \cdot LO$ where m and n take on arbitrary integral values. In practice, only small values are of importance.

For example, using an LO of 2.396 GHz, an RF signal of 2.412 GHz mixing with the LO would produce the desired signal of 16 MHz, but also the undesired signals of 32 MHz (the “first mixer spur”) and 64 MHz (the “second mixer spur.”) (The first and second mixer spurs are the most difficult to deal with hence this discussion is restricted to the reduction of these mixer spurs.) If these mixer spurs are sufficiently weaker than the desired signal, they can be ignored.

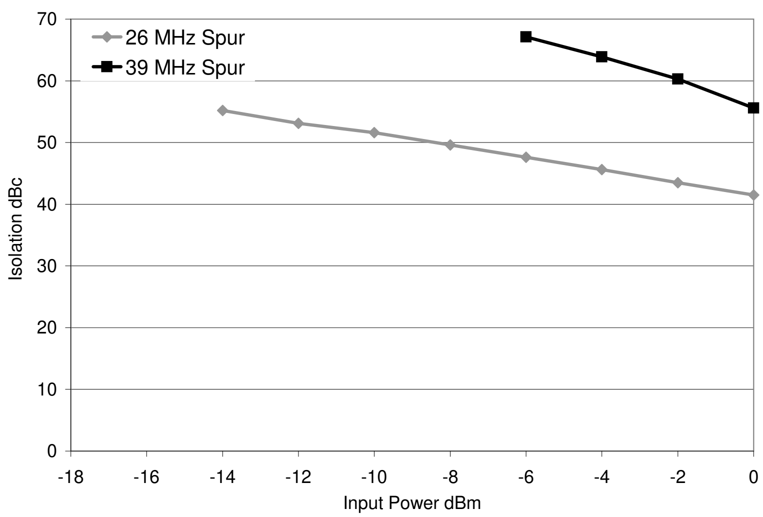

Figure 3.4: MAMXES0042 Spur Isolation

Figure 3.4 shows the measured performance of a medium power (13 dBm LO) mixer with respect to first and second mixer spur isolation. For this measurement a 13 dBm, 2.449 GHz LO was connected to the LO port of a M/A-com MAMXES0042 mixer; this mixer has very good spurious isolation. The x-axis shows the strength of the RF signal connected to the mixer's RF port. The output from the mixer's IF port was sent to a spectrum analyzer, and isolation of the desired signal with respect to spurs was measured. The y-axis shows the measured degree of spurious signal isolation. Higher values indicate more isolation, and hence are better. This figure shows that at 0 dBm, the isolation achieved with respect to the first mixer spur is 41.5 dBc. Hence, using this mixer and a 0 dBm RF input signal strength would result in a spurious free dynamic range (SFDR) of 41.5 dBc which is good performance for a mixer, but still less than ideal for the RF Front End.

This can be addressed in a few ways. Using a higher power mixer (i.e. a mixer that uses a higher power LO) produces greater isolation with respect to spurs. Using a higher power LO, can be inconvenient as the LO signal source must be strong enough to drive multiple RF Front Ends. More importantly, the best SFDR achieved after testing several mixers using this technique was 41.8 dBc which is still less than desired. Moreover, this value was achieved with an input signal strength of 0 dBm. The output signal from the mixer will experience additional signal loss of around 7 dB. Hence the mixer output signal that achieves 41.8 dBc of isolation is -7 dBm. The signal conversion module, however, requires +8 dBm in order to match the full range of the A/D converter. As a result, 15 dB of dynamic range would be lost simply due to this mismatch.

A key observation in Figure 3.4 is that isolation - and hence SFDR - increases as the RF signal incident to the mixer becomes weaker. Hence, excellent SFDR can be obtained if a sufficiently weak RF signal is used. Unfortunately, doing this will further increase the mismatch of the required input signal strength to the Signal Conversion Module, thus simply reducing the RF signal strength is not feasible.

Part of the solution used in this work is to amplify the signal after it has passed through the mixer in order to meet the signal strength requirement of the Signal Conversion Module. This introduces some undesired noise into the signal. However, much of this noise is unimportant since the signal is so much stronger than the noise. Thus, this noise is "buried" in the actual noise floor received by RF nodes. Added phase noise, however, does degrade the signal somewhat.

Another alternative design would be to use an "active mixer" that amplifies as it mixes. In practice, however, active mixers were found to perform very poorly with respect to flatness over frequency due to impedance variation over frequency.

Note that on upconversion, mixer spurs are also produced. The incoming signal is weak enough, however, that mixer spurs are not an issue for upconversion.

Undersampling. Spurious signal isolation can be further improved using undersampling. In order to meet the Nyquist criteria, an A/D converter running at a sample rate of F_{sample} is typically used to sample signals where $F_{signal} < F_{sample}/2$. Signals where $F_{signal} > F_{sample}/2$ will alias into the region $F_{signal} \mod F_{sample}/2$. For this reason, digital signal processing systems typically filter signals $F_{signal} > F_{sample}/2$. It is possible, however, to leverage aliasing to perform a "digital downconversion." That is, signals $F_{signal} > F_{sample}/2$ can be intentionally aliased to the desired frequency range. Now, undersampled signals where $F_{signal} \mod F_{sample} > F_{sample}/2$ will be inverted in frequency; for this reason, undersampled signals typically are sampled to meet the condition $F_{signal} \mod F_{sample} < F_{sample}/2$.

Undersampling is not, however, without drawbacks. The first effect is that additional noise is folded in. This effect is negligible in this case since the additional noise is so much weaker than the signal being sampled. A much more significant drawback of undersampling, in the emulator, is increased sensitivity to sampling clock phase noise - measured in the frequency domain - or sampling clock jitter - measured in the time domain.

The achievable signal-to-noise ratio of a sampled signal is limited by phase noise and A/D aperture uncertainty t_{jitter} as follows [31]: $SNR = -20log_{10}(2\pi f_{analog} t_{jitter})$

Solving for t_{jitter} yields: $t_{jitter} = (10^{SNR/-20})/(2\pi f_{analog})$

With an LO of 2.396 GHz and a sampling clock running at 180 MHz, f_{analog} for 802.11b/g channel 11 is (2.462 GHz - 2.396 GHz + 180 MHz) = 246 MHz. Thus, to achieve a target dynamic range of 60 dB requires 0.65 ps jitter rms. Achieving 50 dB requires 2.0 ps jitter rms.

This is further discussed in the Signal Conversion discussion below.

Miscellaneous. In addition, the RF Front End contains a filter on the RF input/output that eliminates any low-frequency components that might be present on the RF path to or from the RF node. Filters are also placed on the IF input and output ports to filter out undesired signals. An optional high-pass filter is placed between the RF Front End IF output and the SCM IF input as shown in Figure 3.2. This filter is used when undersampling since, in this case, the signal to be sampled will typically be placed at some frequency just over F_{sample}, and frequencies less than F_{sample} should be filtered out.

The RF port also contains a coaxial connectorized attenuator. This attenuator is set at a low value (e.g. 2 dB), and reduces any impedance mismatch that might be seen by the RF node looking into the RF Front End and vice versa.

Quadrature Sampling. Note that the RF Front End does not employ quadrature sampling. This approach is taken to improve the scalability of the emulator by reducing the number of signal paths that must be handled. This approach also simplifies the design of the RF Front End.

Nevertheless, quadrature sampling would be a useful avenue for future implementations. The current design could be adapted to support quadrature sampling by using two RF Front Ends connected to the RF Node via a splitter and a 90 degree phase shifter.

Quadrature digital signal processing can, however, be supported by the signal conversion module discussed in the next section. The primary advantage of supporting this directly in the RF Front End would be to double the bandwidth that could be digitized.

3.2.3 Summary

Developing an RF Front End for network emulation is a challenging task due to the demanding constraints placed on the design. This section has presented a simplistic reference design to illustrate the basic functionality required of the RF Front End, and to show the difficulties that arise in meeting the full requirements. An improved RF Front End design was then presented that overcomes the shortcomings of the naive design.

As discussed in this section, while there are multiple approaches to designing an RF Front End for network emulation, the RF Front End implementation presented here provides very good solutions to all of the individual design constraints required of it.

3.3 Signal Conversion

3.3.1 Requirements

The primary purpose of the Signal Conversion Module is to interface the RF Front End with the DSP Engine. When an RF Node is transmitting, the signal conversion module must digitize the incoming signal and send it to the RF Front End. When an RF Node is not transmitting, the incoming signal from the DSP Engine must be converted to analog and sent to the RF Front End.

Key in the design of the Signal Conversion Module is the choice of data converters. The data converters used in the Signal Conversion Module largely determine the dynamic range, bandwidth, and signal quality of the overall system.

The Sampling Theorem states that the sampling rate must be twice the desired sampled bandwidth. 802.11b/g signals span from 2.401 GHz to 2.473 GHz for 72 MHz of bandwidth. Hence, the sampling rate must be at least 144 MHz to capture signals contained in this range.

As discussed in Chapter 1, dynamic range d of an ideal A/D converted is limited by the sample size n according to $d = 6.02 \cdot n$. The emulator target devices - for version 2 - have a minimum receive sensitivity of -95 dBm and a transmit power of 19 dBm. A 7 dB margin is needed to be able to emulate interference from nodes that are out of reception range. Thus the required minimum signal strength is -102 dBm. At 1 meter, path loss is typically about 40 dB. Hence, emulating reception from 1 meter separation to out-of-range requires 62 dB of dynamic range. More sensitive devices can require support for even weaker signals. For instance supporting devices with a minimum RSS of -100 dBm requires an additional 5 dB of dynamic range for 67 dB total. It is possible to sacrifice support for strong signals in favor of supporting weak signals if desired.

Analog to Digital Conversion. Number of bits - the discussion above leads to a minimum requirement for 11 bits of sample resolution to achieve the 60+ dB dynamic range target. Data converter imperfections, however, mean that more bits are actually required. Thus the SCM must support at least 12-bit samples.

Bandwidth - the SCM bandwidth goal is to support all 802.11b/g channels. Thus the required bandwidth is 2.401 GHz to 2.473 GHz or 72 MHz wide. The Nyquist Criteria means that the sample clock must run at least twice this rate or 144 MHz.

Digital to Analog Conversion. The D/A converter requirements track those of the A/D converter. D/A converter capabilities, however, greatly exceed those of A/D converters. Thus the A/D converter is the limiting factor in the SCM signal conversion system.

It is useful, however, to employ a D/A converter that has a higher resolution than the A/D converter. While the A/D resolution still limits the SNR of the system, increasing D/A resolution allows for a greater range of signal strengths to be supported. In particular, at the extreme ends of the supported dynamic range, the supported signal strength steps in dB can become quite large. Using additional bits on the D/A improves resolution and allows those jumps to be greatly reduced. Hence, 14 bits are required for D/A converter samples.

Synchronization. A difficult requirement for the emulator implementation to meet is the synchronization of all data and clocks on the DSP Engine. Samples generated on discrete Signal Conversion Modules must be processed on a common clock inside of the DSP Engine. The DSP Engine FPGA contains resources capable of synchronizing data and clocks, but they are limited in number. Further circuitry could be added to the DSP Engine, but this would increase the complexity of the design. To achieve scalability and simplicity, this synchronization problem must be solved off-board of the DSP Engine. Thus, this problem is best addressed in the Signal Conversion Module.

3.3.2 Implementation

Figures 3.5 and 3.6 show the implementation of the Signal Conversion Module. Incoming transmissions are first digitized by the A/D converter; digitized samples are then forwarded to an on-board FPGA: a Xilinx XC4VSX25 [69]. The FPGA converts the digitized signals into half-width double data rate (DDR) signals for transmission to the DSP Engine. The DDR format reduces the number of pins consumed on the DSP FPGA as well as the number of wires required to carry signals.

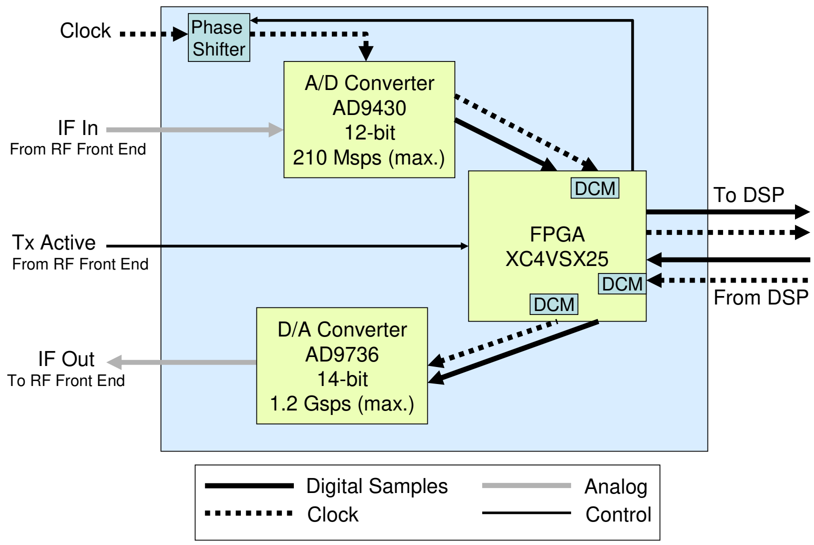

Figure 3.5: Signal Conversion Implementation

The use of an FPGA on the SCM is not strictly necessary, but it provides several benefits:

- Flexible clock and I/O systems make achieving good on and off-board clock synchronization and obtaining good digital signal integrity much easier.
- Computational resources can be used to correct signal imperfections introduced by the RF Front End.
- Some computation may be offloaded from the DSP Engine.
- The SCM can be utilized as a software radio.
- The SCM can be utilized as a standalone channel emulator.
- The SCM can be utilized as a spectrum analyzer.

On reception - i.e. whenever the RF Node is not transmitting - the node's computed signal from the DSP Engine is routed by the SCM FPGA to the D/A for conversion to analog.

A/D Converter. The only commodity A/D converter available at the commencement of this work that met the sample size and resolution criteria discussed

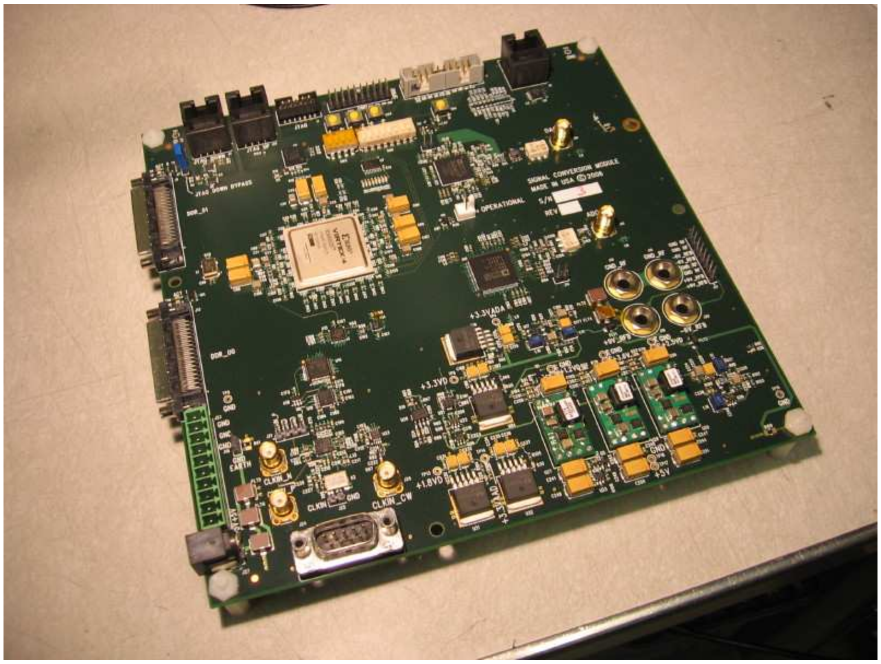

Figure 3.6: Signal Conversion Implementation Picture

earlier was the Analog Devices AD9430 [13]. This converter has a resolution of 12 bits and a maximum sample rate of 210 Msps. It has a dynamic range of 65 dBc at the 210 Msps rate.

A practical advantage of this A/D converter is its use of a low-voltage differential signaling (LVDS) data bus which greatly simplifies the integration of this A/D converter into a printed circuit board. LVDS provides increased system signal integrity and reduced electromagnetic emissions (EMI.)

D/A Converter. The D/A converter selected for the emulator was an AD9736 [14]. This converter supports 14-bit samples at up to 1.2 Gsps. It includes an integrated PLL that reduces phase noise for the output signal. Like the AD9430, this device supports an LVDS I/O interface which is important in supporting high data rates cleanly.

Synchronization. Synchronization of data when it reaches the DSP Engine is achieved using a phase shifter on the clock input line. This allows each Signal Conversion Module's clock to be shifted such that the DSP Engine can latch the incoming data on a common clock. (The common clock on the DSP Engine is generated by using a single clock forwarded from an SCM. The clocks forwarded from other SCMs are ignored.)

Data from the A/D, outgoing to the D/A, and incoming from the DSP engine uses the on-board digital clock management (DCM) resources of the XC4VSX25 to synchronize data with the accompanying clock.

3.3.3 Summary

The Signal Conversion Module discussed in this section provides a very high performance signal conversion platform that enables accurate network emulation. The key elements in this design are the use of very high performance data converters interconnected by an FPGA. While the design could have been implemented without the FPGA, using the FPGA simplifies on-board clock synchronization and enables correction of data conversion imperfections. In addition, the use of an FPGA allows the signal conversion module to be used as a powerful signal processing platform that can be the basis of a standalone channel emulator, a spectrum analyzer, or a software radio.

3.4 DSP Engine

3.4.1 Requirements

The Digital Signal Processing Engine is the heart of the emulator. It is here that signals are processed in order to mimic the effects of signal propagation in the real world. The DSP Engine must be able to process samples at the sample rate. In addition, it must be programmable in order to support different signal environment models to be used. The most difficult requirement to meet is that of scale. For n RF Nodes that are within range of each other, there exist $n \cdot (n-1)$ channels paths between these nodes. This $O(n^2)$ number of channels means that the DSP Engine can quickly become a bottleneck in system scaling. Programmability allows the DSP Engine to achieve further scale by varying signal environment modeling fidelity according to the needs of the emulation. The DSP Engine should be able work in conjunction with other DSP Engines in order to increase system scale.

Consider a system with 10 RF nodes, using the Signal Conversion Module described above, where a single signal path between all nodes is to be modeled. Each SCM generates 12-bit samples at 170 Msps for a total of 2.04 Gbps. Thus in order to support this 10 node system, the DSP Engine must be able to receive and transmit data at 20.4 Gbps. Internally, the signal conversion module must be able to support 90 signal paths with an aggregate 183.6 Gbps of data and process an aggregate 15.3 billion scaling operations per second. Moreover, the DSP Engine must process this data while introducing minimal additional latency.

3.4.2 Implementation

The only hardware capable of meeting the demands discussed above is an FPGA or ASIC based system. The massive communication, computation, and low-latency requirements are too great for a commodity general purpose or even a digital signal processor. The flexibility requirement makes an ASIC an undesirable choice. Thus the only feasible method to meet the target requirements discussed above is to use an FPGA.

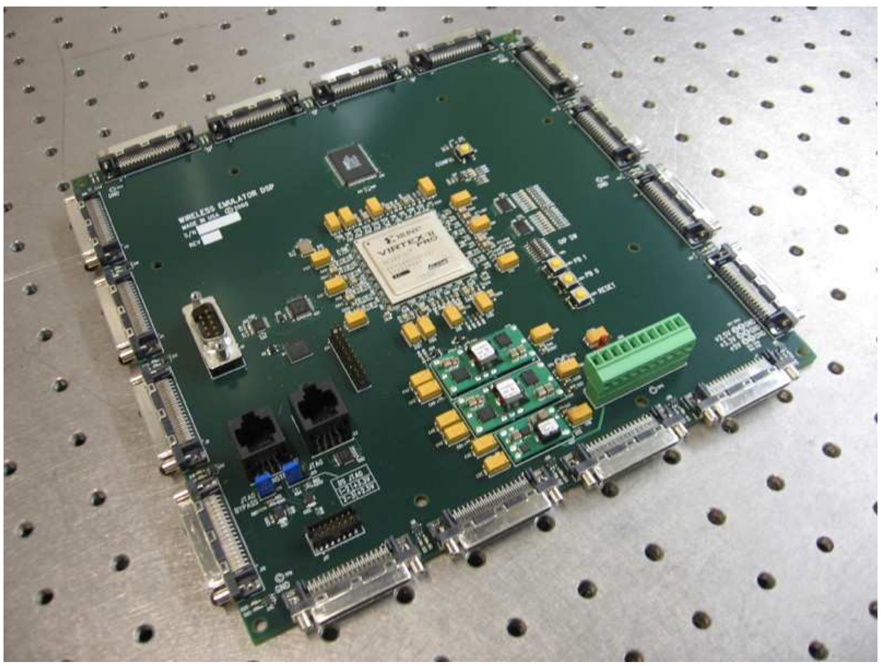

Figure 3.7: DSP Engine Picture

The DSP Engine implementation is shown in Figure 3.7. The implementation discussed in this work uses an XC2VP50 [70] with software developed using the Xilinx ISE tool [66]. This is a higher-end FPGA that contains sufficient resources to construct an emulator of several nodes. The rest of this section discusses how this FPGA is used to meet the requirements discussed earlier.

I/O. The first challenge is getting massive amounts of streaming data into and out of the DSP Engine. FPGAs come with multi-gigabit transceivers which could - in principle - be used. The high latency associated with these devices, however, makes using them less-desirable. Thus the general purpose I/O pins on the FPGA are used. The high data rate, however, makes using a single-ended I/O standard undesirable. This is overcome using the Low Voltage Differential Signalling (LVDS) standard to send and receive data. Unfortunately, this requires two I/O pins per bit. Fortunately, I/O can be sent on both the rising and falling edges of the system clock using a double data rate (DDR) approach. Using this technique is equivalent to one pin per bit without DDR. LVDS requires 100 ohm terminating resistors as close to the receiver as possible. Modern FPGAs include these resistors on-chip so that the printed circuit board (PCB) need not include them. Using these, however, requires that unidirectional (i.e. typical) LVDS be used. Thus, the number of data pins required per SCM supported is 6 for transmit plus 7 for receive for a total of 13. This is a large number of pins, and restricts the total number of devices that can be connected to the emulator.

When emulating all-to-all connectivity, however, the number of pins required is not the bottleneck in this implementation. In this case the limitation is the number of multipliers provided. Assuming one signal path per channel, the number of multipliers required is the same as the number of channels discussed earlier $n \cdot (n-1)$ where n is the number of nodes. For the XC2VP50 used in this implementation, 232 multipliers are available. As a result, the maximum number of nodes that can be supported with all-to-all connectivity for this DSP engine is 15 nodes, since 15 nodes require 210 multipliers, and 16 nodes would require 240 multipliers.

The circuit implemented inside of this FPGA scales, delays, and combines signals to implement the desired signal environment. This is discussed in Chapter 4.

3.4.3 Summary

While non-trivial to realize in hardware, the DSP Engine is a logically straightforward design that provides an extremely high power signal processing engine. This engine forms the core of the network emulator, and provides the key means of enabling signal environment emulation of sufficient scale to emulate a wireless network.

3.5 Auxiliary FPGA

The Emulation Controller and the DSP Engine must communicate via a high bandwidth connection. The implementation of the DSP Engine hardware is greatly simplified by moving as much communication logic as possible off-board. This is accomplished using an "Auxiliary FPGA" which contains the complex functionality necessary to communicate with the Emulation Controller as well as the hardware interface necessary to communicate with the DSP Engine.

3.5.1 Requirements

This design yields the following requirements:

Commodity Hardware. Using commodity hardware in the Auxiliary FPGA eliminates the need for a custom hardware element.

User-defined I/O Pins. Interfacing with the DSP Engine requires a compatible hardware interface. As a no commodity part uses the exact pin-specification required by the DSP Engine, the Auxiliary FPGA must have user defined I/O pins in order to allow custom functionality to be achieved using a commodity component.

Interface with User-level Code on Emulation Controller. Some candidate devices require a device driver to be written on the Emulation Controller in order to communicate. Providing a user-code interface on the Emulation Controller eliminates this task.

3.5.2 Implementation

These requirements are met using a commodity FPGA evaluation board: a Xilinx ML-401 [67]. Programs for this board are developed using the Xilinx Embedded Development Kit [68] (EDK.) This software makes the FPGA on the ML-401 appear as a CPU that supports user-defined devices.

Communication with the DSP Engine is accomplished by creating a user-defined device capable of communicating over the LVDS interface to the DSP Engine. The Auxiliary FPGA utilizes the exact same physical interface (a VHDCI cable and connectors) that the SCM boards use. This allows the Auxiliary FPGA to connect to the DSP Engine via the same method. (The DSP Engine has 16 VHDCI ports. 15 are used for SCM connectivity and 1 is used for the Auxiliary FPGA.) A small adapter allows the general purpose I/O pins on the ML-401 to connect to the VHDCI cables utilized by the DSP Engine.

The EDK provides a primitive embedded operating system and an embedded C development platform. Communication between the Auxiliary FPGA and the Controller utilizes the TCP/IP library provided by this environment.

Commands outbound from the Controller to the DSP Engine are sent as UDP Packets to the Auxiliary FPGA. Inside the Auxiliary FPGA, these packets are decoded and turned into low-level commands that go over the VHDCI/LVDS cable to the DSP Engine.

3.5.3 Summary

Communication between the DSP Engine and the Emulation Controller requires an interface that connects the custom interface found on the DSP Engine to a commodity PC. This section has presented a design that accomplishes this using a commodity component that is flexible enough to meet the requirements required to implement this interface.

3.6 Packaging

The large amount of hardware present in the emulator demands a modular packaging implementation to achieve manageability. The general approach taken to achieve this is the use of rack-mount packaging.

The current implementation makes use of laptop computers for RF Nodes. The stringent isolation requirements demand that the laptops, RF Front Ends, and Signal Conversion Boards Modules be housed together inside of a shielded chassis. For further modularity, the RF Front End and Signal Conversion Board are housed inside of a smaller aluminum chassis - shown in Figure 3.8 - with bulkhead adapters for all needed external coaxial connectors. The larger chassis is lined with RF absorber that greatly reduces the ability of RF energy to enter and exit the chassis. Small openings in the front panel of the large chassis allow the connection of needed cables. The large chassis are then rack-mounted as shown in Figure 3.9.

3.7 Summary

Implementing physical layer network emulation requires powerful hardware and careful coordination. This chapter has discussed the requirements of each component in the emulator, and how each component is implemented to meet its requirements.

Figure 3.8: Signal Conversion Chassis

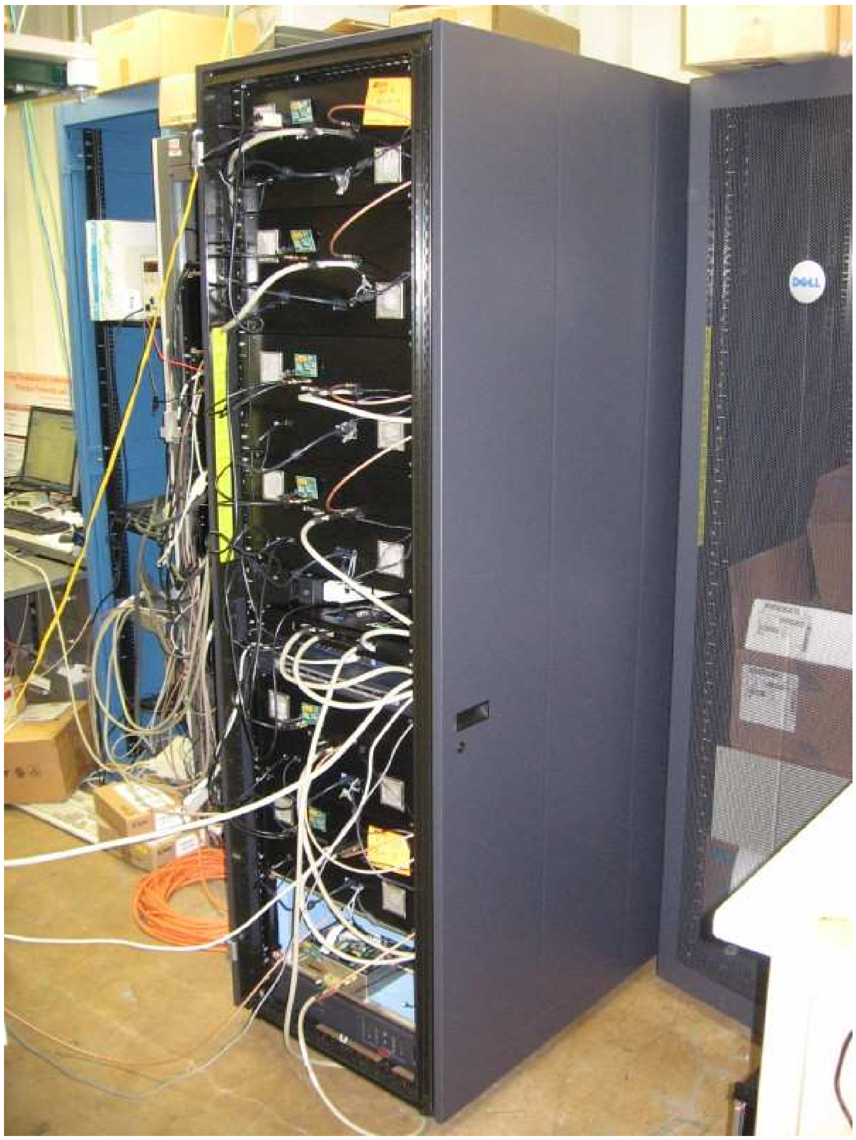

Figure 3.9: Emulator Rack

Chapter 4

Control Software

The emulator developed in this thesis consists of both custom and commodity hardware that must act in concert to enable users to accurately emulate arbitrary signal propagation environments and efficiently execute experiments. Developing software to achieve this has been a challenging task. This chapter discusses the software architecture that enables the physical layer wireless emulator to achieve accurate emulation while enabling efficient experimentation.

This discussion proceeds as follows. Section 4.1 compares and contrasts the requirements of the emulator with the capabilities of existing network simulators and emulators. Section 4.2 describes how users interact with the emulator. Section 4.3 then discusses the emulator's software architecture in detail. Section 4.4 presents several case studies of how this software architecture allows various experiments to be conducted on the emulator, and Section 4.5 summarizes this chapter's discussion.

4.1 Emulation Software Requirements

Before discussing emulator's software architecture and how users interact with the system, this discussion will briefly consider the requirements that real-time physical emulation imposes on the software that supports it. It will also consider how the capabilities of existing simulators and emulators compare to the physical layer network emulator's requirements.

At a high level, the emulator performs modeling similar to that of traditional wireless simulators. Like traditional wireless simulators, the physical layer emulator can model a physical environment with wireless devices distributed in it, and path loss between devices. Traditional simulators, however, also must model the behavior of the transmitter radio and MAC, the receiver radio and MAC, the network stack, and applications running on the network. Physical network emulation - in contrast -

removes the need to model any of these since the wireless devices are real from the application layer to the radio.

Physical layer emulation's use of real devices and real applications is similar to that found in other emulators such as Emulab [65], Orbit [50], or Modelnet [37]. Like these systems, the emulator must control real devices and the applications that run on them. Modelnet, however, does not make use of real wireless devices. Emulab, supports real wireless devices, but in a testbed fashion; i.e. the wireless network is hardwired to a particular physical location. Orbit is also a testbed, but attempts to emulate different physical environments through selection of nodes positioned in a grid and the injection of noise to alter signal-to-noise ratios and mimic the effect of path loss; the techniques used in this approach are very different from those that can be leveraged in the emulator.

The emulator requires a software infrastructure that is - in some ways - a hybrid of traditional network simulation and emulation. Unlike traditional systems, however, the emulator has complete control over the physical signal propagation between the devices attached to it. This is enabled by the emulator's unique hardware which contains a powerful digital signal processing platform capable of digitizing and manipulating signals streaming between all wireless devices attached to it.

Specifically, the emulator software infrastructure must meet the following requirements:

- Accurately emulate signal propagation - A key requirement of the emulator's software is to make use of this signal processing platform to accurately emulate signal propagation in a controlled manner. At the lowest level, this requires software to control the signal processing. In the emulator's case, this low-level software is written using a hardware description language (HDL). (As this code is reconfigurable, this discussion will refer to this HDL code as software.) No existing wireless network simulator or emulator contains software for this fine-grained level of signal processing.

- Minimal assumptions regarding problem addressed by experiments - The emulator should make no assumptions regarding what problem users are addressing. ns-2's wireless support, in contrast, was originally developed with the assumption that ad-hoc routing was the problem of interest.

- Minimal assumptions regarding hardware - Moreover, the emulator software should make no assumption about the devices attached to it (as long as they fall in the supported frequency band.) In addition, this approach removes the assumption that a physical world is necessarily being modeled. That is, users are allowed to directly control the signal propagation channels between devices without necessarily considering a physical world. This is similar to "multipath fading simulators" that are commercially used to evaluate RF devices. Fading simulators, however, are severely limited in the number of channels that they support and are not capable of modeling an entire network.

- Interactive use - The emulator should enable users to control it in an interactive fashion to mimic the unscripted experiments that are typically used for initial investigations.
- Enable simple experiments - The emulator should enable basic experiments without the need to climb a steep learning curve. This should be possible without introducing an additional programming language into the emulator (beyond Java which is used in the core software.)
- Enable advanced experiments - Advanced users should be able to use a programmatic interface to enable powerful experiments like those found in traditional network simulators.

4.2 Experimental Control

Users control the emulator via three interfaces: scripts that provide an easy-to-use method for conducting simple experiments, programs that provide more advanced users with powerful control, and a graphical user interface that enables interactive experimentation as well as visualization.

Scripts. The script interface is an easy-to-use interface for conducting basic experiments. This interface allows users to define node movement and application execution without the need to write real code. The scripting interface is deliberately simple. No looping, branching, or variables are supported. Rather scripts are defined as lists of events using an XML-based syntax. This restricted interface provides a very shallow learning curve, and makes writing basic emulator experiments an easy task.

An alternative design would be to use a scripting language such as TCL. This is avoided since 1 - it does not eliminate the need for a compiled programming interface for advanced access to the core of the emulator control code; it simply adds a new language that must be learned, and 2 - the need to access core emulator functionality from the scripting language would quickly lead to a bloated scripting interface and complex coupling with the programmatic interface. As the scripted interface does not maintain emulator state, there is virtually no coupling between user scripts and core emulator code.

Hence the emulator's design maintains the key benefit of scripting - the ability to run simple experiments - while supporting full programmatic control in a clean manner.

Programs. Experiments that have requirements exceeding the capabilities of the script interface utilize the programmatic interface. The control code running on the Emulation Controller is written in Java. Users may write Java classes that are loaded at emulation boot time and then have full access to the emulator code. Using the programmatic interface, experiments with arbitrarily complex behavior can be created in a straightforward fashion.

GUI. When conducting experiments with real hardware in a physical environment, there is frequently an exploration phase during which wireless nodes are moved and applications are run in an interactive fashion. For instance, a transmit rate selection experiment between a laptop and an access point would likely include a phase where the laptop was moved around and performance observed in an interactive fashion. After initial performance was observed, then strict procedures would be followed for formal results gathering. One unique aspect of the emulator is the ability to perform this exploratory phase without the need to actually move around the real world.

To support this type of functionality, the emulator provides an interactive GUI that can be used to move RF nodes around in an emulated physical environment and in the corresponding emulated signal propagation environment.

4.3 Software Architecture

As alluded to earlier, controlling real-time physical layer wireless network emulation is a complex task that requires careful real-time coordination of several hardware devices and the software distributed in them. At the same time, the complexity of the emulation must be shielded from users as much as possible; to accomplish this, the emulator control software presents users with a simple interface that makes the emulator appear largely like a traditional wireless network simulator. This provides a familiar experimental interface.

Figure 4.1 depicts the major systems in the software architecture of the emulator; system names are shown in shaded boxes. The software systems are grouped according to the language in which they are written; the italicized labels identify the hardware components in which the software systems reside. The remainder of this section will discuss each of the software systems depicted in the figure.

4.3.1 Execution Control

The execution control system provides three interfaces that can be used to provide experimental control as discussed earlier: a GUI for interactive experiments shown in Figure 4.2, scripts for very simple experiments, and the ability to load Java code into the core of the emulator to provide full access to emulator internal data structures

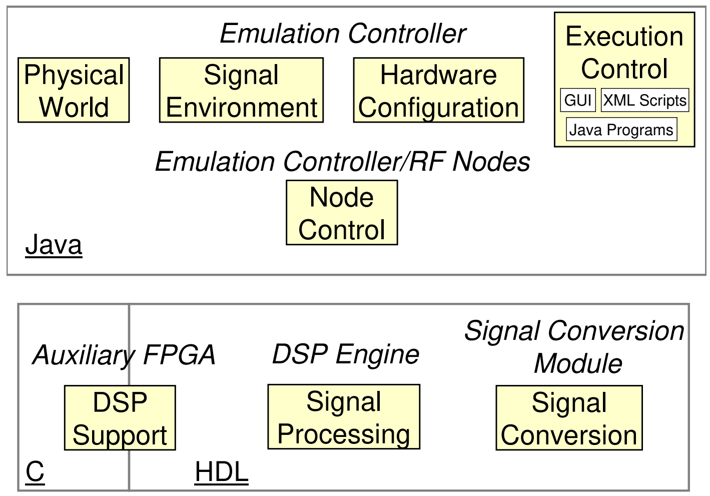

Figure 4.1: Software Architecture Overview

enabling powerful experiments to be conducted. Examples scripted and programmatic emulator control will be discussed in Section 4.4.

4.3.2 Hardware Configuration

The hardware configuration code is divided into two portions. The first portion manages information about the physical hardware setup. It keeps track of the number of emulated nodes in the system, the identities of the circuit boards used to emulate each node, etc. The second portion manages the software used for signal processing on the DSP engine. This module reads the emulation configuration file and generates HDL code to implement the channel models on the DSP engine. This code generation is necessary since the HDL code used by the DSP Engine (see below) changes based on the simulation environment specified by the user.

4.3.3 Physical World

The physical world system is responsible for managing physical objects in the emulated physical environment. In particular, the physical world controls the movement of EmuNodes which represent wireless devices.

The structure of the Physical World system is shown in Figure 4.3(a). Each EmuNode may have one or more wireless NICs, and each NIC has one or more antennas. Antennas connect the physical world with the signal propagation environment as discussed in the following section. Each antenna corresponds to a single RF node

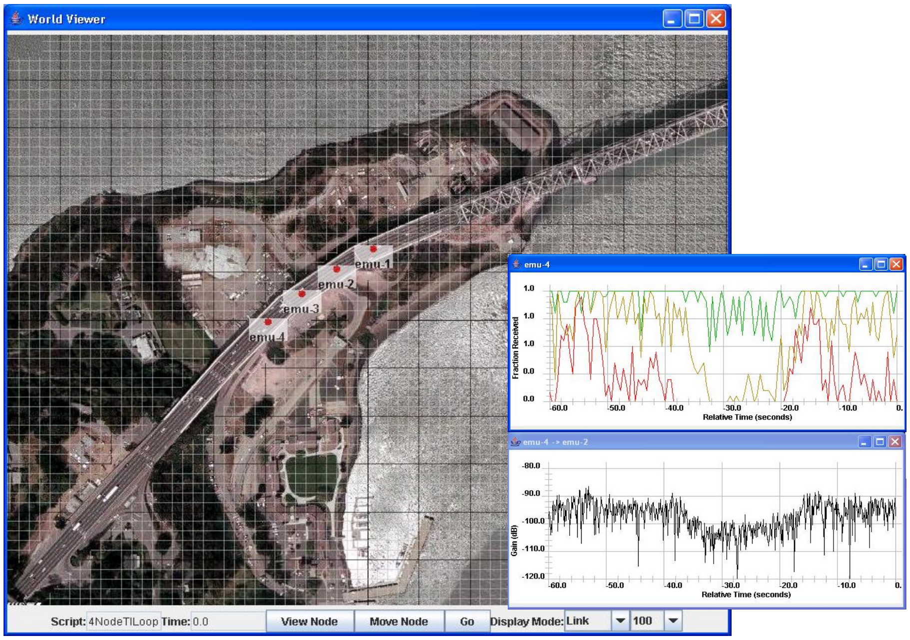

Figure 4.2: Interactive GUI

in the general architecture picture. Hence each EmuNode may be represented by multiple RF nodes.

As EmuNodes move in the physical environment, their antennas move and the corresponding signal environment modules are informed of the movement.

4.3.4 Signal Environment

Signal propagation modeling is controlled by the SignalEnvironment system which is currently structured as shown in Figure 4.3(b). The remainder of this discussion will assume this implementation, but should not be construed as implying that this is the only implementation possible. The hardware is capable of supporting many other architectures for modeling the signal environment, and the software has been designed to allow for additional signal environment models to be incorporated into it.

Between each pair of antennas in the wireless system, there exist two channels - one in each direction. Each channel consists of one or more signal paths. Each path has a propagation delay, a loss model, and a fading model associated with it. The loss model defines large-scale path loss - a fixed attenuation determined by distance - between the destination and source antennas. A typical loss-model will register itself with the physical world so that whenever the source or the destination antennas move, loss is recomputed. The fading model defines small-scale fading (rapid variation in

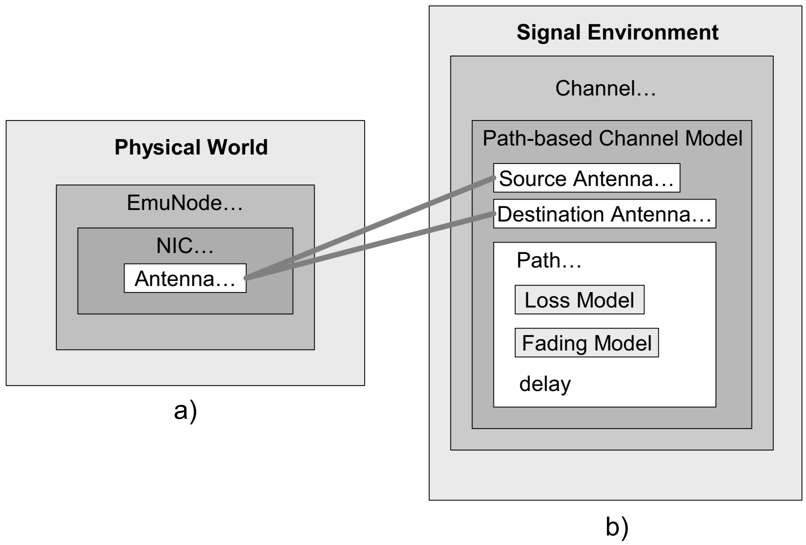

Figure 4.3: Physical World and Signal Environment

signal strength that can occur even if the device antennas are motionless) between the source and destination antennas. A typical fading model will register its interest in velocity changes of the source and destination antennas. It then uses the velocities of the two antennas to compute small-scale signal strength variations which are added on top of the large-scale path loss.

4.3.5 Signal Processing

The Signal Processing HDL code is responsible for performing the processing required by the channel modeling code discussed earlier. Figure 4.4 shows the typical operation of the Signal Processing code. Again, the programmable nature of the DSP Engine allows for alternative implementations of this code. In the typical version shown in Figure 4.4, incoming signals are first sent into a delay line where one or more copies ("taps") of the signal are pulled off after going through a programmable amount of delay. Each of these signals is then scaled by a programmable factor as determined by the signal environment emulation code. Each outgoing signal, from the FPGA to an RF node, is then computed by summing the scaled signals from the other RF nodes. These outgoing signals are then sent to the D/A board for reconstruction.

The programmable nature of this circuit allows the emulator to trade off resources such as the precise depth of the delay pipes and number of signal copies supported. Thus, users can customize the operation of the DSP Engine to the particular test being run.

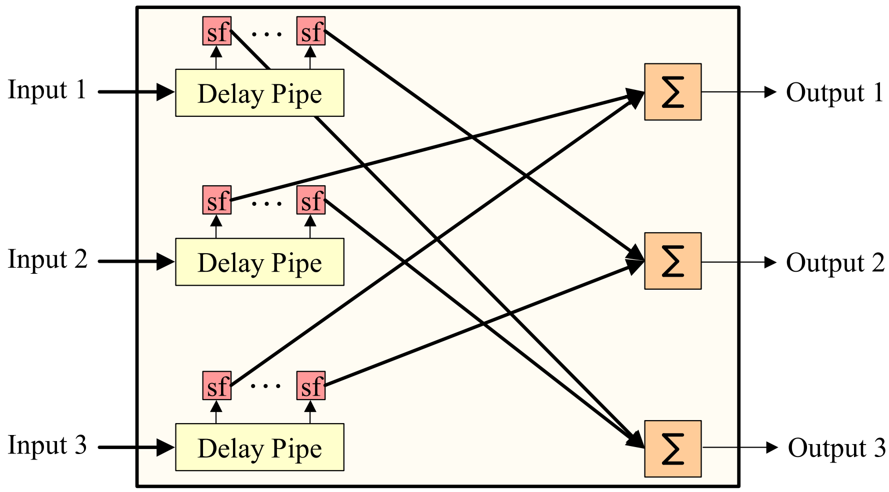

Figure 4.4: DSP Engine Operation Example

4.3.6 DSP Support

The DSP Support code acts as an intermediary between the Java-based code on the Emulation Controller and the HDL code on the DSP Engine. It translates high level requests from the Emulation Controller into into hardware instructions for the DSP Engine. Likewise, it allows the HDL code on the DSP Engine to send messages to the Java-based code on the Emulation Controller. In addition, the DSP support code provides access to an embedded processor that can be leveraged to assist both the Emulation Controller and DSP Engine in tasks such as channel modeling. This allows for more fine-grained temporal control of channel modeling.

In the current signal environment implementation, all signal environment emulation - such as large scale path loss and small scale fading - occurs on the Emulation Controller. For each signal path inside the DSP Engine, the Emulation Controller sends attenuation updates when the attenuation on a path changes. The rate at which scaling updates can be sent is limited by the Ethernet link between the Emulation Controller and the Auxilliary FPGA to approximately 3 million updates per second. Even when all signal paths must be constantly updated, the number of updates per second is approximately 14,000 per second. The corresponding minimum update period is at most 70 microseconds. Thus, the emulation controller can control fading at a very fine granularity. For instance, if the length of a signal path is changing at a speed of 60 miles per hour or 27 meters per second, the path attenuation can be updated before the path has changed by 2 mm.

Currently there is a lag of several milliseconds - maximum near 10 ms - between the time a scale factor is sent from the Emulation Controller and the attenuation is set inside of the DSP Engine. As a result, when modeling signal paths at high granularity, the Emulation Controller must work ahead of the actual time. Currently this is not completely implemented, but provisions allowing this are. To support this, scaling values sent out from the Emulation Controller can be tagged with the times at which the Auxilliary FPGA should send them out to the DSP Engine.

4.3.7 Signal Conversion Module

The signal conversion module converts between the RF signals used by the wireless devices and the digital signals processed by the emulator. On transmit, signals digitized by the A/D converter must be sent to the DSP Engine. On receive, signals from the DSP Engine must be sent to the D/A. Also, for testing and other purposes it is useful to allow signals to bypass the DSP and be sent directly from the A/D to the D/A. This functionality is achieved through the use of an FPGA on each signal conversion module. Hence, the signal conversion module software is written to accomplish this routing of signals in this FPGA. Moreover, the signal conversion software has the ability to process signals outside of the main DSP. Thus effects such as noise generation can be offloaded from the DSP Engine.

4.4 Case Studies

To illustrate how this software architecture enables users to easily conduct a broad range of experiments, a series of case studies are now presented. These examples do not cover all of the control software architecture's functionality, but they have been chosen to illustrate a range of ways in which the user is able to utilize the system.

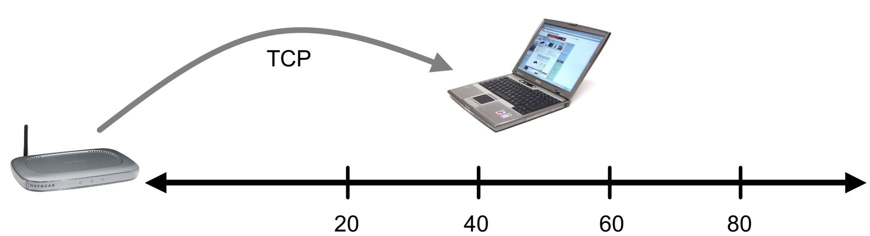

Figure 4.5: Throughput vs. Distance Topology

4.4.1 TCP Throughput vs. Distance

To demonstrate how the emulator's software architecture facilitates conducting simple experiments, consider a simple TCP throughput vs. distance experiment where TCP throughput is measured at a small number of discrete locations in an emulated environment. This is a straightforward experiment, and the emulator's script interface provides a simple means of implementing it.

For this experiment, a signal propagation environment where all channels are modeled using a single path with log-distance large-scale path loss and Ricean fading will be used. This signal propagation environment is defined as follows in the software configuration file:

```
<ChannelDef>
  <name>Default</name>
  <Channel>
    <ChannelModel type="PathBased">
      <Path>
        <LossModel type="LogDistance">
          <d0>1</d0>
          <p1d0>40.0</p1d0>
          <n>2.8</n>
        </LossModel>
        <FadingModel type="Ricean">
          <k>3.0</k>
        </FadingModel>
      </Path>
    </ChannelModel>
  </Channel>
</ChannelDef>
```

In this experiment, TCP throughput between two wireless NICs is measured at four different distances in the emulated environment. This is done as follows (pseudocode in square brackets is used for brevity):

```
<EventDef>
  <EventGroup time="0.0" concurrent="true">
    <Add>
      <node>emu-1</node>
      <pos>0.0 0.0 0.0</pos>
    </Add>
    <exec>
      <node>emu-1</node>
      <cmd>iwconfig wlan0 essid test mode ad-hoc
                    channel 6</cmd>
    </exec>
    [add node 2 and run iwconfig on it]
  </EventGroup>
  <EventGroup time="1.0" concurrent="false">
    <exec>
      <node>emu-1</node>
      <cmd>tcpThroughputServer</cmd>
```

```
    </exec>
    <exec>
      <node>emu-2</node>
      <cmd>tcpThroughputClient emu-1</cmd>
    </exec>
    <SetPos>
      <node>emu-2</node>
      <pos>40.0 0.0 0.0</pos>
    </StartRoute>
    ...
  </EventGroup>
</EventDef>
```

Application results can be either streamed to the controller or stored on emulator nodes for examination upon completion.

As shown in this example, the emulator's script control interface allows concise, easily mastered code to completely specify a signal propagation environment, and conduct simple experiments in that environment. This provides a shallow learning curve that should enable novice users to quickly begin conducting experiments using the emulator.

4.4.2 Hidden Terminal Throughput

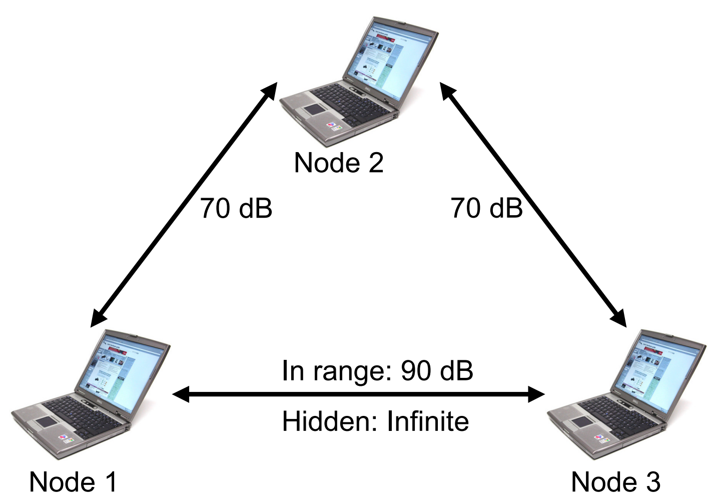

Figure 4.6: Hidden Terminal Topology

The hidden terminal problem is a well known problem in wireless networks where two stations transmitting to a common third station may interfere with each other. Figure 4.6 shows the topology of a typical hidden node scenario: two terminals - Nodes 1 and 3 - are sending data to a third terminal - Node 2. If Nodes 1 and 3 do not sense each other's transmissions, then they are said to be "hidden" from each other, and their transmissions may collide at Node 2 which may cause data loss. In the worst case, these collisions can continue and the links may become unusable.

To measure the impact of this problem on real hardware, it is useful to deliberately create a hidden node situation and run throughput tests. Despite the conceptual simplicity of this task, investigating hidden node performance with real hardware in a real physical environment can be very difficult. Unless the carrier sense mechanism can be altered (which varies by platform), determining when devices are in carrier sense range of each other requires indirect measurements such as a throughput test. Moreover, even if devices are verified to be out of carrier sense range, the signal propagation environment may change such that they are brought into carrier sense range.

The emulator's complete control over signal propagation makes this difficult problem trivial by enabling complete specification of wireless channels. Moreover, the emulator's software architecture allows this experiment to be constructed in a straightforward manner. This example illustrates the emulator's ability to directly control channel conditions without the need to worry about the physical environment. In other words, in this experiment there is no physical environment, just a manually specified signal propagation environment.

The desired signal propagation environment is created by manually specifying attenuation between RF nodes to create the in-range and hidden node topologies shown in Figure 4.6. In both cases, two nodes (1 and 3) communicate with a third node (2) over channels with a single path and 70 dB of attenuation. In the in-range case, the attenuation between 1 and 3 is set to be 90 dB.

The in-range configuration file channel signal environment definitions are as follows:

```
<ChannelDef>
  <name>Default</name>
  <Channel>
    <ChannelModel type="PathBased">
      <Path>
        <LossModel type="Manual">
          <loss>70.0</loss>
        </LossModel>
      </Path>
    </ChannelModel>
  </Channel>
</ChannelDef>

<Channel src="1" dest="3">
  <ChannelModel type="PathBased">
    <Path>
```

```
      <LossModel type="Manual">
        <loss>90.0</loss>
      </LossModel>
    </Path>
  </ChannelModel>
</Channel>
[use same definition for 3 to 1]
```

In the hidden case, the attenuation between 1 and 3 is set to be infinite. This is done by simply changing the channels between 1 and 3 as follows:

```
<Channel src="1" dest="3">
  <ChannelModel type="PathBased">
    <Path>
      <LossModel type="Manual">
        <loss>Infinity</loss>
      </LossModel>
    </Path>
  </ChannelModel>
</Channel>
[use same definition for 3 to 1]
```

Bandwidth tests may then be conducted for each of these situations in a manner similar to the TCP Throughput example.

As stated above, the key point of this example is to show how manual control of the signal propagation environment is easily specified, and can be used to obtain useful results.

4.4.3 Vehicular Convoy Channel Replay

The previous tests have illustrated modeled or manually specified signal propagation environments. An alternative to modeling wireless channel conditions is to record a real wireless channel and then to replay that channel in the emulator.

This example shows how the software infrastructure easily incorporates traces of wireless channel behavior and corresponding device location traces. Channel traces consist simply of time stamped path loss values, and can be gathered using commodity wireless hardware (see Section 5.3 and [28] for details.) A portion of an actual pathloss trace gathered by a major automobile manufacturer is shown below:

```
<LossTrace>
  <name>1-2</name>
  <duration>2700</duration>
  <sample>
    <time>38.000000</time>
    <gain>-104.000000</gain>
```

```
  </sample>
  <sample>
    <time>38.201000</time>
    <gain>-99.000000</gain>
  </sample>
  ...
</LossTrace>
```

Channel traces can be combined with node mobility traces gathered using a localization system such as GPS. The below trace is an excerpt of a location trace gathered in conjunction with the loss trace shown above:

```
<route>
  <name>1</name>
  <waypoint>
    <pos>51835.325630 4432.397624 0.000000</pos>
    <arrivalTime>38.000000</arrivalTime>
  </waypoint>
  <waypoint>
    <pos>51835.391262 4430.662439 0.000000</pos>
    <arrivalTime>38.201000</arrivalTime>
  </waypoint>
  ...
</route>
```

One application of trace record and playback is intervehicular communication. Major automobile manufacturers are working to equip vehicles with short range wireless networks. As part of this effort, they are gathering traces of intervehicular pathloss and corresponding location traces.

Replaying these path loss traces within the emulator allows researchers to conduct an experiment multiple times while varying experimental parameters - such as the routing algorithm. By replaying the exact same signal trace, researchers are able to evaluate the desired experimental aspect without the need to worry about channel variations across runs. In contrast, conducting a similar comparison using real world experiments is next to impossible since, even if the cars in the experiment were driven over the same route, large amounts of channel variation would occur across experiments.

4.4.4 Programmatic Channel Control

As a final example of how the software infrastructure enables the emulator to conduct insightful experiments, consider an experiment conducted as part of this thesis: measuring 802.11b packet capture behavior. (Packet capture is reception in spite of interfering signals. Chapter 7 will examine this problem in detail.) This test was

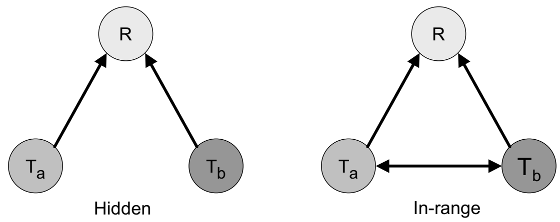

Figure 4.7: Packet Capture Topology

enabled by the emulator's manual channel control interface. The manual control interface allows the experiment to directly specify channel conditions without worrying about what physical environment might create such a channel. As the channels in this experiment required a large number of configurations, the software infrastructure's programmatic interface was leveraged.

The setup for this experiment is shown in Figure 4.7. The goal of this experiment was to establish the reception outcome given the RSS at a receiver R from transmitters Ta and Tb without controlling precisely for interference timing. That is, this experiment didn't control when packets from Ta and Tb arrived at receiver R.

In both configurations shown in Figure 4.7, the tests proceeded as follows: two transmitters Ta and Tb constantly sent broadcast packets at a very high rate to the receiver R. At first, the channels were all "turned off" in the emulator so that no packets were actually received at R. While the channels were "off" the traffic sources were started. The channels were then simultaneously turned on, and the RSS at R from each transmitter was set to the desired values by the emulator control software. In the "hidden" configuration, Ta and Tb were not allowed to hear each other's transmissions. In the "in-range" setup the RSS from Ta to Tb was set at -80 dBm and vice versa, so that Ta and Tb always heard each other's transmissions. After a fixed time interval, the channels from Ta and Tb to R were shut off. R then recorded how many packets it received from each transmitter. This test was repeated for all combinations of RSS values from Ta and Tb at R between -102 and -72 dBm in 1 dBm intervals. An excerpt from this experiment's code is shown below:

```
// set attenuation to infinity
mPathController.setAllPathsOff();

[start traffic sources]

// set attenuation from transmitters->receiver
// uplinkRSS[] is the current RSS to be used
for (int i = 0; i < uplinkRSS.length; i++) {
  mPathController.setPathRSS(txNodes[i],
          destNode, 0, txPowerdBm, txRSS[i]);
```

```
  mPathController.setPathRSS(
          destNode, txNodes[i],
          0, txPowerdBm, txRSS[i]);
}
// set attenuation between transmitters
for (int i = 0; i < txNodes.length; i++) {
  for (int j = 0; j < txNodes.length; j++) {
    if (i != j) {
      if (runningHiddenTest) {
       // turn channel off between transmitters
        mPathController.setPathRSS(
            txNodes[i], txNodes[j], 0,
            txPowerdBm, Double.NEGATIVE_INFINITY);
      } else {
       // set attenuation between transmitters
        mPathController.setPathRSS(
              txeNodes[i], txNodes[j], 0,
              txPowerdBm, interferenceRSS);
      }
    }
  }
}
// wait for traffic to be sent/received
Thread.sleep(trafficDurationMillis);

// set attenuation to infinity
mPathController.setAllPathsOff();

// gather results
```

Chapter 7 presents extensive results obtained from this experiment. A few results are briefly summarized here.

Figure 4.8 shows a sample set of results gathered in this experiment. In each of this figure the z-axis is the number of packets received from both Ta and Tb at R. In many RSS combinations, however, packets were actually only received from one or the other; the regions where one source or the other dominated are labeled on the plots.

In in-range case shown in this figure (where transmitters were always within reception range of each other), when the RSS at R from both Ta and Tb was high CSMA did a good job of allowing the two nodes to share the medium, and only a small number of collisions occurred. When one transmitter was out of range of R and the other was in range, the number of packets received for the in-range cases was roughly half of the channel capacity since the reception between Ta and Tb is still good, and they defer for each other's transmissions irrespective of the number of packets successfully received at R.

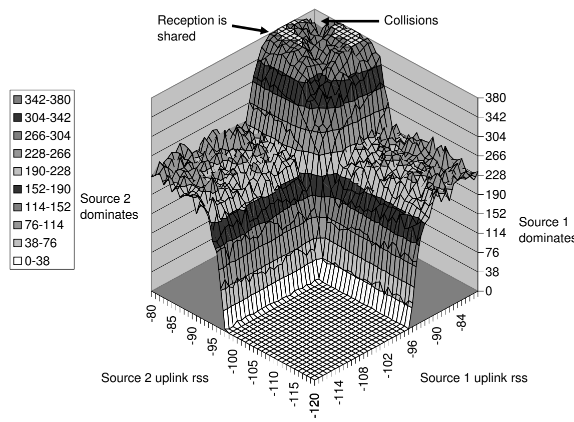

Figure 4.8: Packet Capture: In-range

An important question is what happens when transmissions from two nodes overlap in time at a single receiver. The "hidden node" configuration tests investigate this question. Full results for this case are discussed in Chapter 7. The emulator's precise programmatic control over channels was key in allowing the precise characterization of reception behavior of commodity devices and yielded insight into packet capture behavior in the presence of competing 802.11b signals.

4.5 Summary

Physical layer wireless network emulation promises to combine much of the flexibility of traditional wireless network simulation with much of the realism of testbed-based experimentation. Coordinating the operation of a physical network emulator in real-time is a complex task. This chapter has presented a software architecture that manages the complexity of physical network emulation while presenting users with a powerful yet easy-to-use interface.

This discussion has shown through a series of case studies how this software architecture allows a wide variety of wireless experiments to be conducted in an efficient manner. This architecture provides the basis for quickly conducting realistic wireless experiments in a flexible manner.

Chapter 5

Signal Environment Emulation

With the emulator's ability to completely control wireless signal propagation comes the challenge of modeling or recreating propagation in an appropriate manner for a given experiment. The goal in this work is not to develop and justify new physical models of signal propagation, but to discuss how current and future models as well as signal propagation trace playback can be used in the emulator.

Fortunately, unlike wireless simulators, the emulator is freed from the task of emulating radio behavior in conjunction with signal propagation modeling: users simply pick a suitable signal propagation model, the emulator then computes each receiver's received signal, and lets the radio decide what happens. The emulator does not need to make any assumptions regarding any radio issues such as "sensing range", "interfering range", or packet capture.

This chapter discuss several different methods of modeling wireless signal propagation in the emulator. This discussion begins with signal propagation models that require no site-specific information, and then discusses models that use increasing amounts of site-specific information. Most of these techniques are completely operational in the emulator (large-scale path loss, signal capture and replay, and small-scale fading); a couple discussed require some external tools before they can be used in the emulator (ray-tracing, channel sounding).

5.1 Modeling

The simplest approach to emulating a signal propagation environment is to employ analytical modeling. A key advantage of this technique is that it does not require any site-specific data which can be difficult to obtain. The lack of site-specific data is also this method's biggest drawback since specific environmental information is required to get realistic results for any particular environment.

5.1.1 Large-scale Path Loss

The signal propagation model most commonly used by simulators is a large-scale path loss model. Specifically, the received signal strength at each receiver (RSS) is computed as $RSS = P_t + G_t - PL + G_r$. Where P_t and G_t are the transmit power and antenna gain at the transmitter, PL is the path loss, and G_r is the antenna gain at the receiver. Large-scale path loss models simply compute PL as a function of distance between the transmitter and the receiver.

The Emulation Controller implements large-scale path loss by simply calculating the loss between nodes whenever the distance between them changes. These loss values are then sent into the emulator where they are used to control the attenuation of the signal path between two nodes. The current large-scale path loss model employed by the emulator is log-distance path loss [49] with user-defined parameters.

5.1.2 Small-scale Fading

While large-scale fading models can accurately capture the average path loss between two points, on a short time scale the path loss between these points may vary substantially.

To support this behavior, the emulator currently leverages the technique presented in [47] to incorporate the Ricean and Raleigh statistical models of small-scale fading. In this implementation, the fading parameters are computed offline, and are then loaded into the emulator's control software before emulation begins. At run time, these parameters are added to the large-scale path loss which causes short term variation with the desired statistical properties. When a measurable change in path loss occurs, the path attenuation parameters are updated in the DSP Engine. Independent use of fading parameters allows independent, on-line modification of small-scale fading for each RF node.

5.1.3 Artificial Channel Models

In some experiments it is useful to construct signal propagation environments where portions or all of the model does not correspond to any physical environment. These artificial environments are particularly useful for conducting tests measuring behavior with respect to signal strengths (relative or absolute) of the signals involved (e.g. the packet capture tests in Chapter 7), or for measuring multipath reception performance (e.g. the multipath reception tests in Chapter 7.)

5.2 Ray Tracing

The previous three methods required no site-specific information other than picking the correct path loss models and model parameters. By incorporating site-specific information, it is possible to generate more accurate signal propagation models.

One technique that can be implemented in the emulator is to leverage ray tracing techniques [25, 48]. If the motion of nodes can be pre-computed off-line, ray-tracing techniques can be used to precisely compute all rays incident on each receiver at a given point in time. If motion cannot be pre-computed, then approximations can be made.

At runtime, the pre-computed series of attenuation over time values for each signal path would then be used to set path attenuation inside the DSP Engine in much the same way that the large-scale path loss and small-scale fading cause updates to path attenuation values.

Ray tracing can produce results that are far more realistic than simple analytical modeling. Moreover, ray tracing can be performed on hypothetical environments and does not necessarily require conducting measurements in the real world. Ray tracing is, nevertheless, not a complete substitute for real-world measurements in all cases.

5.3 Trace Capture and Playback

One simple method of accurately modeling signal propagation is to measure the signal propagation in a given environment and then to replay it. This section discusses a novel method developed as part of this thesis for implementing a signal capture system using standard wireless NICs that measures path loss in a physical environment. This system works by constantly sending small packets from each transmitter to be emulated and receiving these packets on each receiver being emulated. An important part of this method is the processing of data gathered from these tests into accurate path loss measurements.

A key benefit of this approach is that it enables measurement of real signal propagation environments. The key disadvantages are that the measurements are somewhat crude: no multipath information is provided, and all-to-all connectivity cannot be measured simultaneously. Rather, measurements are made on a link-by-link basis.

This trace capture and playback method will be discussed at length to illustrate that this technique can produce accurate results in a low delay-spread environment, and to show that given an accurate model of the signal propagation environment, the wireless network emulator can produce wireless network that is very similar to the behavior that would have occurred if the experiments were conducted in the real world.

5.3.1 Trace Capture

Figure 5.1 shows this approach for gathering traces of signal strength. A transmitter constantly sends very small 802.11 broadcasts using a low modulation rate (2 Mbps for results discussed in this section). As deep signal fades may prevent sounding packets from being received successfully, packets are tagged with sequence numbers that enable the detection of packets reception failure. The receiver operates in "monitor mode." This mode gives the receiver complete 802.11 layer packet information. The receiver logs all captured packets from the transmitter including measurements of received signal strength (RSSI) and noise. This trace is then post-processed to generate a file that lists time (based on the MAC timestamp) and received signal strength. This post-processing replaces missing packets - inferred from missing sequence numbers - with low RSSI values (-1 for results discussed in this section). This system is able to record RSSI samples with a granularity of approximately 2 ms.

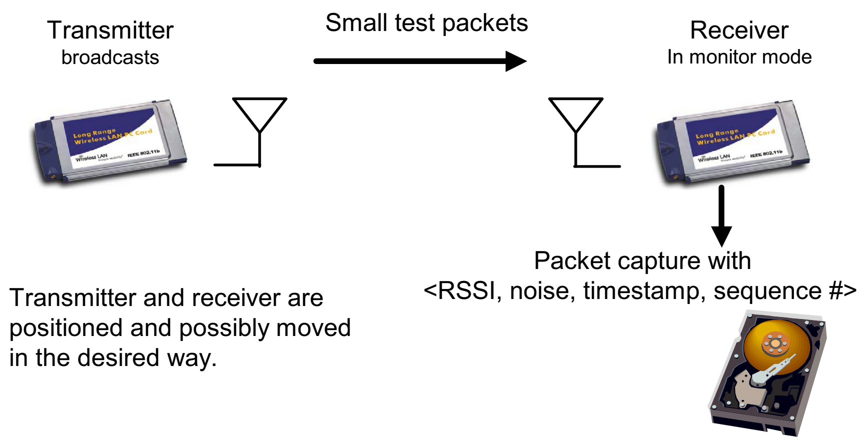

Figure 5.1: RSSI-based Channel Capture

The experiments in this section utilize Engenius NL-2511 Plus EXT2 cards based on the Prism2.5 chipset as well as an Atheros 5212 based card. The Atheros card was used for RSSI measurement only. These cards measure received signal strength at the beginning of packet acquisition, so the RSSI samples are quick samples rather than an average of RSS for the whole packet.

Figure 5.2 shows a sample signal strength trace. This particular trace was captured with the receiver antenna mounted on a car parked at the side of a freeway while the transmitter drove by at approximately 60 MPH. From this trace we see that the transmitter and receiver had a good line-of-sight connection when the cars were at their closest point. At further distances signal strength degrades and fading increases.

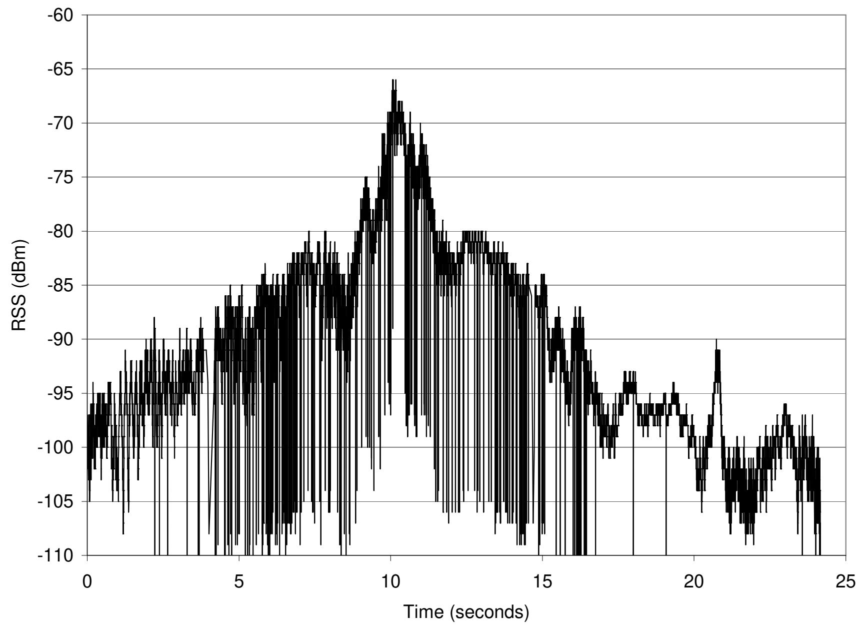

Figure 5.2: Sample Channel Trace

5.3.2 Trace Playback

Once a trace of signal strength is obtained, this trace can be replayed in the emulator. To do this, the Emulation Controller reads the trace and replays it in real time. That is, for each $< RSS, timestamp >$ pair in the sample, the Emulation Controller waits until the emulation time matches the recorded time and then commands the Emulator to set the emulated path loss to match the observed path loss as illustrated in Section 4.4.3. The temporal resolution of the channel power settings is limited by the trace recording process which is 2 ms as discussed above.

5.3.3 Limitations

This approach is attractive in that it is supported by commodity hardware currently found in wireless testbeds. Using commodity hardware, however, has limitations such as: non-linearities in RSSI measurements; bogus RSSI values; missing RSSI values in deep fades; and a lack of foreign RF interference characterization. These limitations can be worked around to some degree as discussed in Section 5.3.5. In addition there are fundamental limitations to this approach with current commodity hardware such as: lack of channel impulse response/multipath information; path loss limited by accuracy of RSSI measurement and transmit power consistency; and sounding temporal resolution limited by packet transmit rate. These fundamental issues will also be discussed further in Section 5.3.5

5.3.4 Comparison with Real-world Behavior

Methodology

The trace recording and playback method presented here is straightforward and can easily be used to gather traces from many existing wireless testbeds. An important question is how much realism is lost with respect to the real world. Clearly this technique does not completely compute the impulse response of the channel and track it over time. This would require a full-blown channel sounder. This technique does, however, track the RSS changes due to large scale path loss and small scale fading with 2 ms granularity. This section will show that in a low delay spread environment, the technique presented here is sufficient to produce link-level behavior that is quite similar to real-world behavior.

To show this, experiments were conducted that were designed to allow the simultaneously measurement of real-world link-layer performance while gathering signal strength traces. The idea is that the emulator can then replay the captured signal trace while re-running the link-layer test. Ideally similar performance will be observed. Note that this is an ambitious goal since even if the emulator could perfectly reproduce the radio channel that existed when the original link-layer test was conducted, factors outside of experimental control - such as packet transmission time variance - will lead to inevitable variance from the original test during a replay.

Figure 5.3 shows the setup for this experiment. In this experiment, two concurrent tests are run: a link-level behavior test, and a channel measurement test. Each test uses a distinct transmitter, receiver pair. To ensure that the channel is as similar as possible, both transmitters are connected to the same antenna via a splitter/combiner. Each transmitter operates on a non-overlapping 802.11 channel; this allows the link-level and channel measurement experiments to be conducted concurrently. A small amount of attenuation is introduced to further avoid interference between transmitters. The receivers are setup in a similar fashion though they require less attenuation since they will only be receiving traffic.

Note that less attenuation is used on the channel measurement receiver. This allows the channel to be measured even when no packets can be received at the receiver. In addition, the channel transmitter uses more power than the test transmitter to further increase the ability to measure the channel when the test receiver cannot receive packets. The motivation behind these measures are to increase the ability to

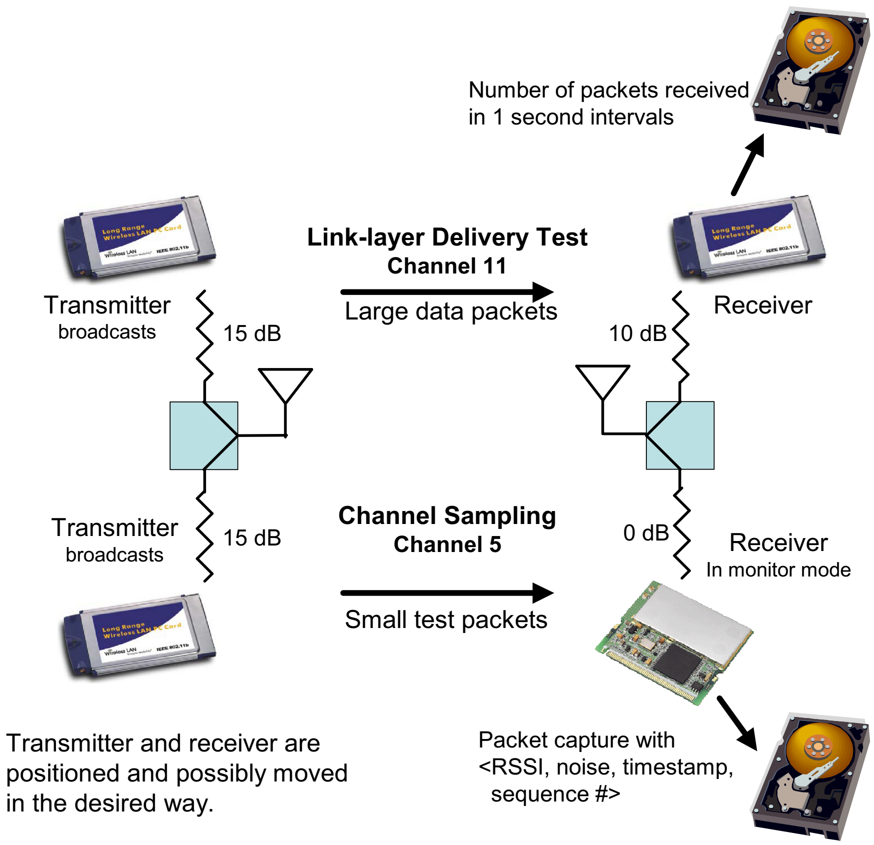

Figure 5.3: Link-layer Test/Channel Capture

measure the actual communication channel for verification purposes in this work. In ordinary channel capture where verification was not needed, a single transmitter and a single receiver would be used with no additional attenuation on any transmit or receive path.

Two-channel Measurements

While using different channels allows this setup to simultaneously run applications and gather signal strength traces, there is still likely to be some divergence between the two channels. This divergence does not affect this trace gathering and replaying approach in any manner. Rather, it only pessimistically affects the ability to verify the accuracy of this approach.

This section explores what is lost in gathering two signal strength traces simultaneously. To do this, the delivery test in Figure 5.3 is replaced with another signal strength capture. Hence, for the following tests, two channel measurements are running concurrently.

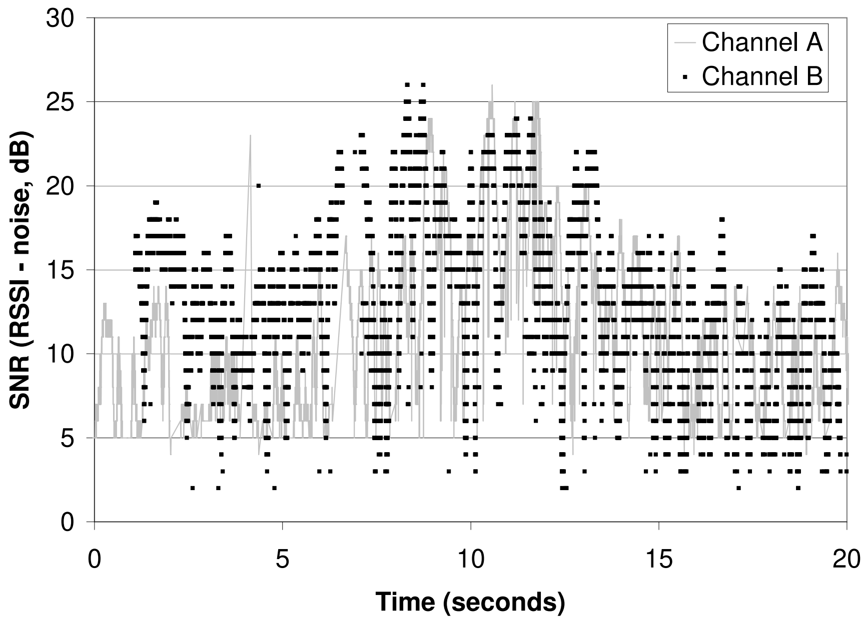

Figure 5.4: Two-channel Capture - Over-the-air

The traces from the two-channel signal strength capture test are shown in Figure 5.4. In this test, the traces are not identical for three reasons: 1 - the transmitters are not synchronized so the channel is being sampled at different times, 2 - some frequency selective fading is occurring, 3 - RSSI measurement error. Nevertheless, the traces are similar enough for validation purposes though they will introduce some divergence between the emulated results and real-world results. Hence, the comparison of real-world link-layer performance and the emulated replay will be slightly pessimistic since a single channel capture will not have this variation.

Comparison Results

This section compares the performance of real-world link-layer behavior vs. an emulated playback of this same behavior. The link layer test conducted for this comparison sent approximately 124 large (1460 bytes payload) UDP broadcast packets per second from the test transmitter to the test receiver. As previously discussed, the wireless channel was concurrently measured as shown in Figure 5.3. Approximately

2-3 channel samples were obtained per traffic test packet. This test was then replayed in the emulator for comparison.

Figure 5.5 and Figure 5.6 show the results of two separate record/ playback verification tests. With a few notable exceptions, the results are quite similar. The average packets received in the emulated replays generally closely tracks the original results. This is in spite of extraneous error introduced by the two-channel verification technique, imperfections in card characterization, variation in packet send time, and other similar factors.

For each 1 second interval in these tests, the absolute difference between the real-world throughput and the throughput in the corresponding emulated interval was computed. The CDF of these error measurements is shown as the "Atheros" plot in Figure 5.7. This figure also shows the CDF of three tests (not shown above) comparing real-world throughput vs. emulated throughput where a Prism 2.5 card was used for channel sounding instead of the Atheros card used above. In both cases the majority of time intervals were reproduced with low error. There are, however, some time intervals with significant error. As a result, it is possible to construct movement patterns where the verification tests will yield poor results.

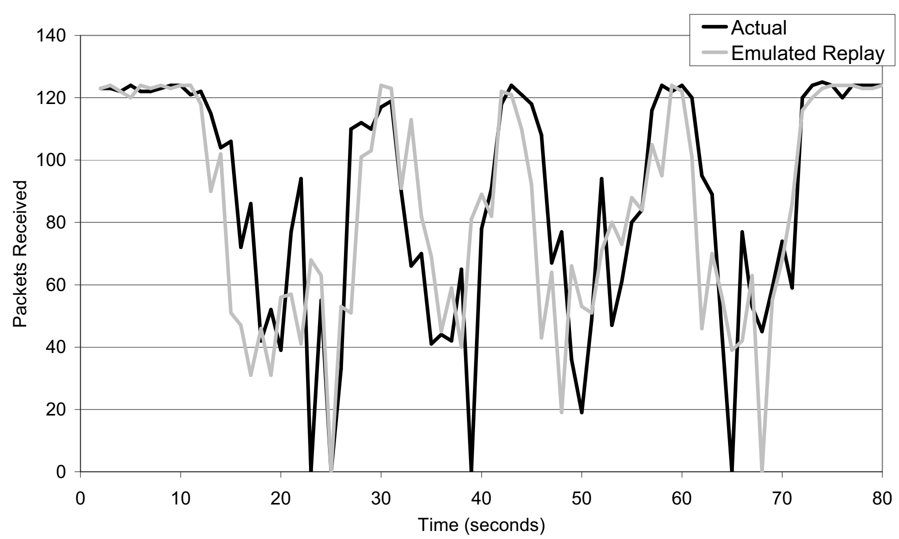

Figure 5.5: Real-world vs. Emulated Replay. Test 1.

5.3.5 Discussion

Several practical considerations must be taken into account when capturing and playing back signal environment traces. This section begins by discussing how RSSI im-

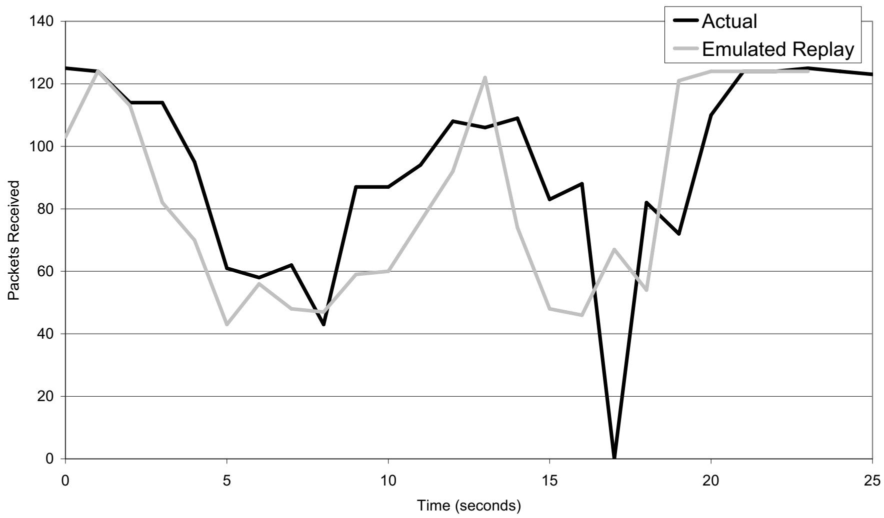

Figure 5.6: Real-world vs. Emulated Replay. Test 2.

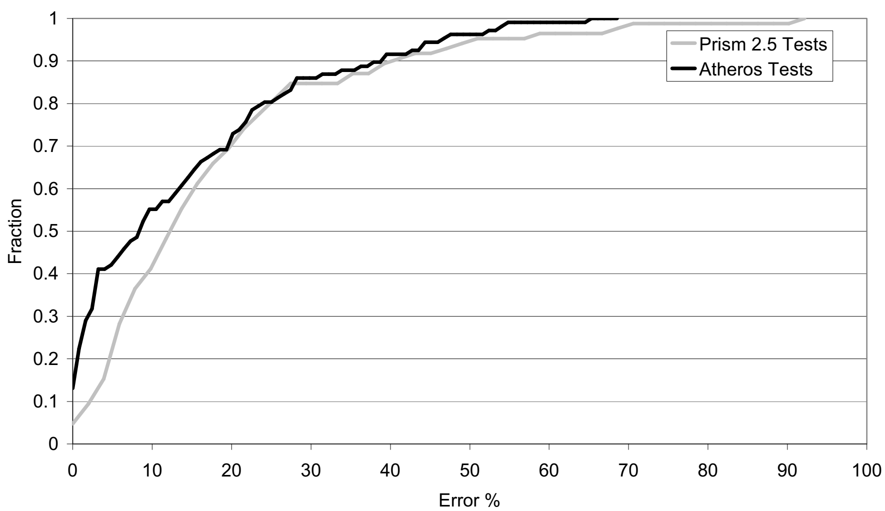

Figure 5.7: CDF of Error for All Tests

perfections can be reduced to improve the quality of path loss traces, followed by a discussion of the types of interference present in a network, and how they are supported in trace playback. Lastly, techniques for improving trace playback fidelity and extending trace recording and playback to an entire network are discussed.

RSSI Considerations

In order to effectively translate RSSI measurements into path loss measurements, the received RSSI measurements must be processed to remove imperfections in the measurements. Two classes of RSSI imperfections are particularly important: non-linearity and bogus values.

RSSI Non-linearity. As discussed in [29] on-card received signal strength measurements (RSSI) are not completely accurate even under the best circumstances. Thus, relying strictly on RSSI for trace playback without a mapping between RSSI and RSS (the actual received signal strength) will distort the replayed signal. The effect of this can be reduced by characterizing the RSSI-RSS relationship on a per-card basis. Figure 5.8 shows the mean RSSI measured by an Atheros-based card as actual RSS is varied using the emulator. Ideally each card in a testbed would be characterized in this manner. At the very least, each type of card should have an RSSI characterization performed.

An important feature of Figure 5.8 that should be taken into consideration is the fact that RSSI values near the lowest end of the card's reception range become indistinguishable. As a result, channel characterization will be less accurate in this regime.

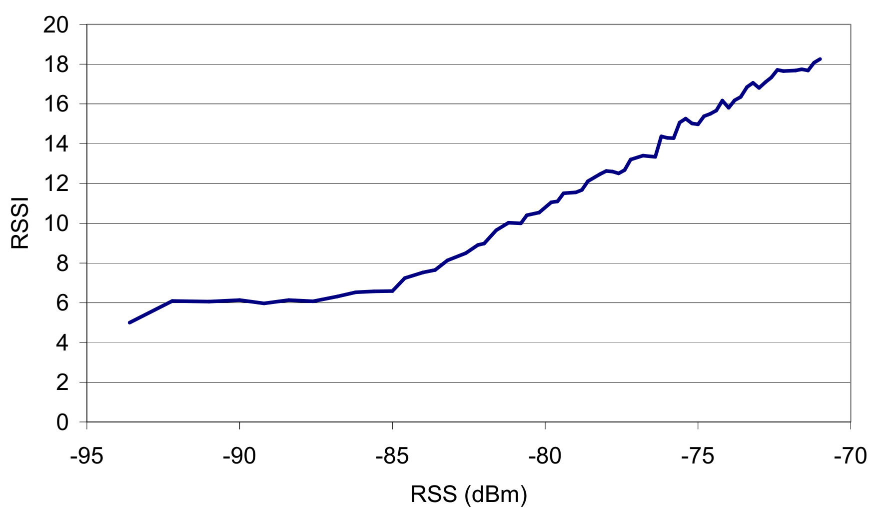

Figure 5.8: Card Characterization

"Bogus" RSSI Values. In addition to the non-linearities shown in Figure 5.8, wireless cards tend to return a certain number of RSSI values that seem to be bogus.

Both Atheros and Prism 2.5 cards were observed to occasionally return values that are around 20 dB below what seems to be the true value.

In order to get a good match between the real-world comparisons discussed earlier, it is necessary to filter out these bogus values. This was done by limiting the rate at which the signal strength was allowed to change, and interpolating between "good" values. Figure 5.9 shows the "corrected" signal trace used in the Figure 5.6 test versus its raw counterpart.

The benefit of this RSSI correction is shown by comparing the playback result of the uncorrected raw version. Figure 5.10 shows the same test as that in Figure 5.5, but with the use of raw RSSI values. Clearly eliminating these bogus RSSI values yields a significant improvement in matching the real-world measurements.

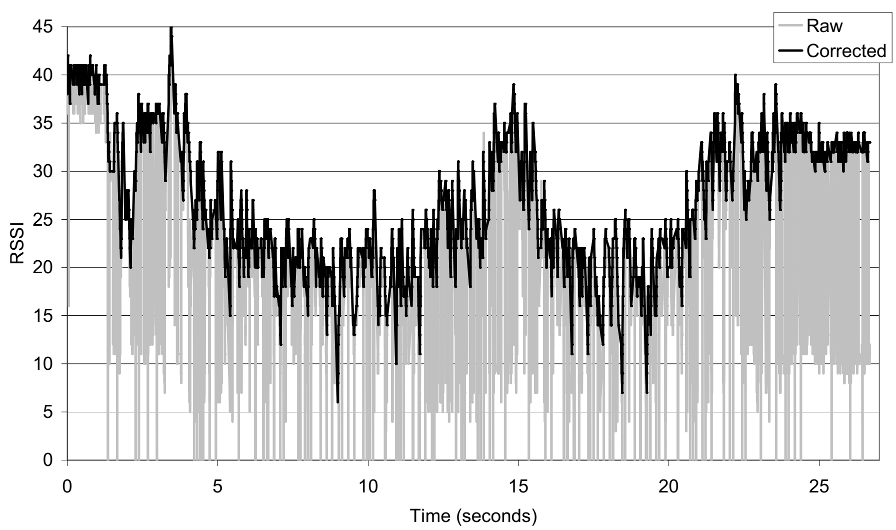

Figure 5.9: RSSI Correction

Noise

Conspicuously missing from this trace playback methodology is use of the noise values reported by the card. These noise values are the sum of both true noise as well as interference. True noise comes from both sources external to the card - most importantly thermal radiation - as well is sources internal to the card which make up the card's "noise figure". Interference that comes from sources internal to the network - "internal interference" - is simply non-captured traffic; "external interference" is from received signals from RF sources that are outside of experimental control.

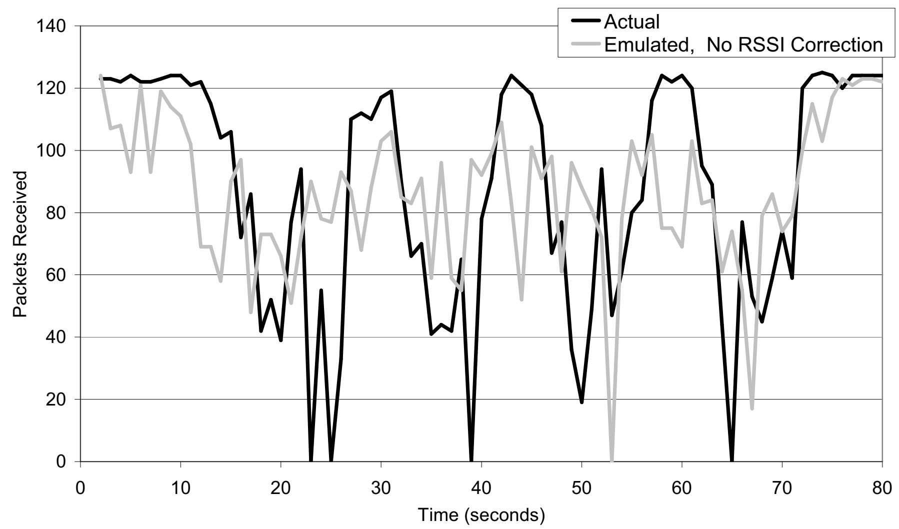

Figure 5.10: Raw RSSI Emulation Accuracy

Externally generated true noise is likely to be mostly due to thermal radiation and constant. Hence, this can usually be computed rather than measured. As the card's internal noise figure and uncaptured traffic are naturally occurring features of playback, recording them is not useful. It might be useful, however, to know levels of external interference, though this value would need to be separated out from the noise figure and internal interference. In many cases, the difficulty of this task is not worth the added fidelity that it would provide. For networks with significant external interference, however, the on-card noise measurements could potentially provide a means of emulating this interference.

Improving Channel Recording and Playback

This section now discusses a few additional sources of error in the trace recording and playback methodology presented here and how these might be addressed.

Channel Probe Granularity. The technique presented here uses simple UDP broadcast packets to probe the channel. Granularity is limited to 2 ms using this approach. Using 802.11 level packets, with a short preamble would increase trace resolution. In addition, some NICs allow the 802.11 CSMA/CA mechanism to be turned off. This could be used to greatly decrease inter-packet delay and greatly increase sampling resolution.

Multipath. Finer granularity measurements will improve the ability to capture fast fading induced by multipath effects. This technique is not, however, amenable to analyzing radio-level effects such as the efficacy of a RAKE receiver or equalizer. This level of channel modeling fundamentally requires a channel sounder that can capture the impulse response of the channel.

A related question is why multipath effects do not make this technique ineffective considering measurements [1] that show multipath can dominate RSS in certain situations. In this case, the delay spread of the network is well within the radio's capabilities. As shown in [1], multipath does not affect packet reception very much for low delay spreads. As a result, this technique should work well for environments that are within a radio's ability to mitigate multipath effects. Outside of that regime, however, this technique will be less effective. Additional work is required to quantify this technique's accuracy in higher delay spread environments.

Network Modeling

These experiments have demonstrated channel capture and playback of a single channel. This technique can be extended to an entire wireless network in several ways. First, if the channels are relatively stable or correlation between channels is low, each channel in the network can simply be captured independently in time. If channel correlation needs to be captured, these measurements must occur concurrently. In this case, transmitters that are nearby must take turns in sending probe packets; for 802.11 networks this can largely be accomplished simply by using 802.11's CSMA/CA mechanism. In some cases, it may be necessary to control the rate of probe packets in order to reduce the likelihood of collision of probe packets at distant receivers.

Once traces have been obtained for all channels in the network, playback proceeds in the same manner as before.

5.3.6 Trace Playback Summary

Accurate wireless channel modeling is an important element in physical layer wireless emulation as well as wireless simulation. This section has presented a simple method for gathering traces of wireless channel behavior, and a technique of analyzing the effectiveness of channel emulation by simultaneously recording a signal strength trace while a real application is being run on the same transmit and receive antennas. Using this technique, this section has shown that the wireless signal traces can be gathered that produce behavior that is surprisingly similar to real-world wireless behavior in spite of the simple nature of on-card channel measurements.

5.4 Channel Sounding

A more sophisticated method of measuring signal propagation in a physical environment is to use specialized hardware [16] to precisely measure the "impulse response" of the channel [49]. The impulse response provides information about multiple signal paths in the environment; this provides a much more accurate picture of the signal propagation environment. Such measurements can be difficult to obtain since they require specialized hardware.

While the channel capture technique presented in Section 5.3.6 does not provide full impulse response information, the emulator itself is capable of replaying measurements obtained by channel sounding. This is done by using one or more signal paths per channel and then setting the attenuation and delay of each signal path in the DSP Engine according to the values extracted from the channel sounding.

5.5 Discussion

Before concluding this chapter, this section briefly considers the capabilities, limitations, and applications of signal propagation modeling using the approaches discussed above, and compares these with alternative approaches.

Simulation. Many of the signal propagation models that the emulator utilizes can also be used in simulation. This superficial similarity, however, belies a massive difference in how these models are used. Computational constraints placed on a simulator, force the simulator to work at a very coarse timescale. A typical simulator will use a signal propagation model as part of its computation of an average signal-to-interference-and-noise ratio (SINR) and then make a binary reception decision. The emulator, on the other hand, uses a statistical propagation model to manipulate a real modulated signal on the timescale of 5 ns. This is then sent to a real receiver to determine the reception behavior. Accurate receiver behavior in a simulator would require transistor level simulation which is completely infeasible for the number of nodes that we are looking at. Realtime simulation of such behavior is out of the question.

Similarly, while a simulator can replay a captured channel trace, it can only do so at a very coarse timescale and with far less fidelity than a physical layer emulator.

Real-world experimentation. The ability to precisely recreate a signal propagation environment is a huge advantage compared to real-world experimentation. This power, however, comes with a price of reduced realism and scale in signal propagation.

Physical layer emulation necessarily models a wireless channel using discrete elements (e.g. one line-of-sight ray and two reflections) whereas a true wireless channel is a continuous phenomenon. Also, as the number of RF nodes attached to the DSP Engine increases, the number and length of delayed signal paths that can be implemented drops. Hence physical layer emulation is a compromise between the fidelity of the real-world and the control of simulation.

Noise. As discussed above, the term noise is frequently used to refer to both true noise (e.g. receiver noise) and interference from other wireless devices. Receiver noise is naturally present in the emulator since it uses real receivers. Interference from other wireless devices can be supported in several ways. First, if RF Node ports are free and the devices are available, these devices can simply be attached to the emulator. Secondly, it is possible to record "noise" resulting from interference and to replay this in the emulator. Third, a white noise generator can be implemented in either the DSP Engine or the Signal Conversion Module to generate noise.

Note that the emulator's effective receiver noise floor will be slightly higher than a coaxial based system since the emulator uses additional amplifiers etc. that introduce noise. This level will still be much lower than the noise floor of a true free-space wireless system.

Scale. As hardware is finite, the richness of channel modeling possible using hardware-based emulation drops as the scale of the network being emulated increases. The limiting factor is typically the number of multipliers in the DSP Engine's FPGA.

Much of this discussion has assumed the desire to support the independent pairwise emulation of all pairs of RF Nodes attached to an emulator. Clearly this approach becomes infeasible at a certain point as the complexity of pairwise interaction is order n^2.

It is important to observe, however that emulating complete interaction is not always necessary. Clearly, if nodes are out of range with respect to each other, then no emulation between them is necessary. In addition, complexity may be reduced by simplifying and aggregating the emulation of channels for distant nodes.

Multi-element Air Interface Support. Current wireless networks are pushing the limits of the throughput that are possible with a single element antenna. Future networks will increase throughput by using multiple elements to support techniques such as steerable antennas, MIMO [19], and "time reversal" [18].

The emulator can support such emerging technologies in at least two ways. First, where hardware exists, the emulator can support these multi-element experiments by simply treating each element as an independent RF node. The control software then controls these RF nodes in a coordinated fashion which also opens up some room for reducing FPGA resources consumed. Second, in certain circumstances, it may be possible for the emulator to emulate the effect of a given technology. For instance, a steerable antenna can be completely emulated without necessarily using a true steerable NIC.

5.6 Summary

This chapter has discussed several different methods of emulating signal propagation environments. Existing analytical models of path loss and fading can easily be incorporated into emulator operation. This models have the benefits of ease of computation, and the lack of any need for site-specific data. With detailed site-specific data, however, ray tracing can provide more robust models of signal propagation in an environment.

This chapter has also discussed at length how commodity wireless devices can gather simple traces of received signal strength and how these traces can be processed to produce traces of signal attenuation over time. In a low delay-spread environment these loss traces have been shown to be produce accurate link-level behavior.

Chapter 6

Validation

In order for a physical layer emulator to be effective, the emulator must digitize and reconstruct RF signals with minimal distortion. This chapter examines the performance of the emulator's implementation with respect to maintaining signal integrity, and demonstrates how low-level signal fidelity translates into accurate higher-level network behavior.

As the DSP Engine operates entirely on digital signals, the fidelity of the emulator is determined by the RF Front End and Signal Conversion Module. Hence, the fidelity of the emulator may be measured solely by measuring the fidelity of the RF Front End and Signal Conversion Module.

The configuration for the major tests in this chapter is shown in Figure 6.1. Two signal sources are used for these tests: an Agilent ESG digital signal generator and a Senao NL-5354MP ARIES2 802.11a/b/g card. Measurements are made using either a spectrum analyzer or a vector signal analyzer connected to one of the measurement ports labeled in Figure 6.1. These ports will subsequently be referred to simply by the labels shown in the figure.

This discussion will first present low-level measurements of RF Front End and Signal Conversion Module fidelity followed by tests demonstrating that this low-level fidelity translates into accurate higher-level behavior.

6.1 RF Front End

This section evaluates RF Front End downconversion and upconversion performance. A wide variety of tests can be conducted to verify fidelity, this section presents results from a few of the most important tests conducted on a single RF Front End. The behavior of other RF Front Ends is similar, but not presented here.

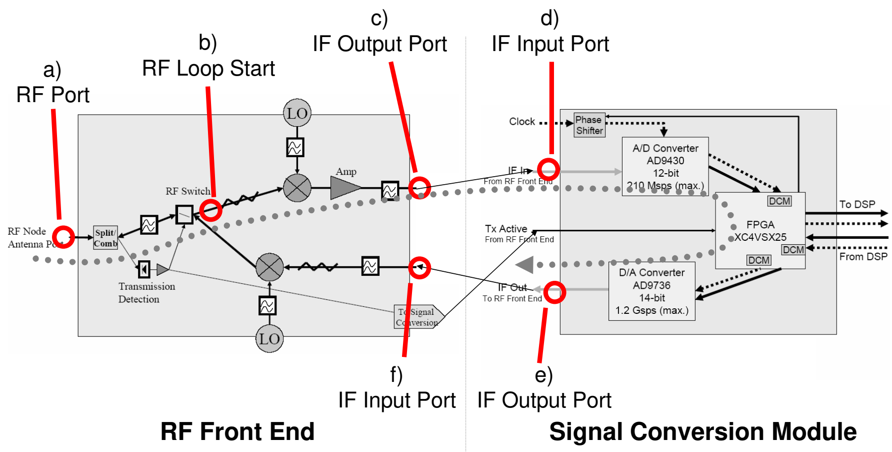

Figure 6.1: Measurement Signal Path

Some of the metrics that will be measured using these tests are:

- Gain/loss - The amount of signal strength gained or lost during signal conversion. On downconversion the final signal level should match the A/D converter's input range. On upconversion the dynamic range should be placed in the desired range (e.g. the low end should extend several dB below the receiver's minimum reception threshold.)

- Flatness - The degree to which gain/loss is constant over frequency. Under 2 dB can be tolerated in many instances without correction. Flatness variation up to approximately 10 dB can be corrected by using a digital filter inside the Signal Conversion Module or DSP Engine. Variations greater than this become difficult to correct without affecting system performance.

- Spurious Free Dynamic Range (SFDR) - The isolation between the desired signal and the closest spur. This is less important for a single channel, but is very important for wideband experiments.

6.1.1 Downconversion.

This discussion first presents tests measuring RF Front End downconversion performance. The benefits of undersampling in improving RF Front End spurious free dynamic range are examined followed by measurements of the ability of the RF Front End to eliminate spurious device output and measurements of flatness over frequency.

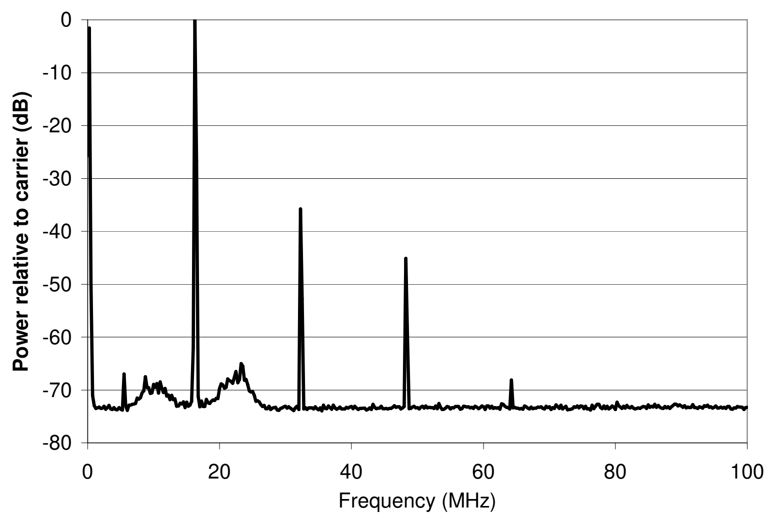

Figure 6.2: SFDR Analysis - IF Below Nyquist

Spurious Free Dynamic Range (SFDR). Spurious free dynamic range when not undersampling was measured by applying a 0 dBm 2.412 GHz signal to the RF input port (a) of the RF Front End. The LO frequency for this experiment was 2.396 GHz, and LO power was just over 7 dBm. Measurements were gathered from measurement point (c), the RF Front End's IF Output port, using a spectrum analyzer (resolution bandwidth was 30 kHz.) Figure 6.2 shows the resulting IF output between 0 and 100 MHz. The desired signal is at 16 MHz. The y-axis is normalized with respect to this signal. Two strong spurious signals can be seen at 32 MHz and 48 MHz. SFDR in this case is 37 dB.

As discussed in Section 3.2.2, undersampling can greatly improve SFDR. Assuming that the RF Front End is targeting an SCM with a sampling rate of 170 Msps, the improved SFDR (at IF output) from undersampling is demonstrated by repeating the SFDR test with an LO of 2.226 GHz and examining IF output between 170 MHz and 270 MHz. Measurements were taken from point (c) as before with the same spectrum analyzer settings except for the new frequency range. Figure 6.3 shows the result. No spurs are seen in this case. (Spurious signals near the desired signal in this case are due to the signal source.)

The emulator typically operates in undersampling mode in order to achieve improved SFDR.

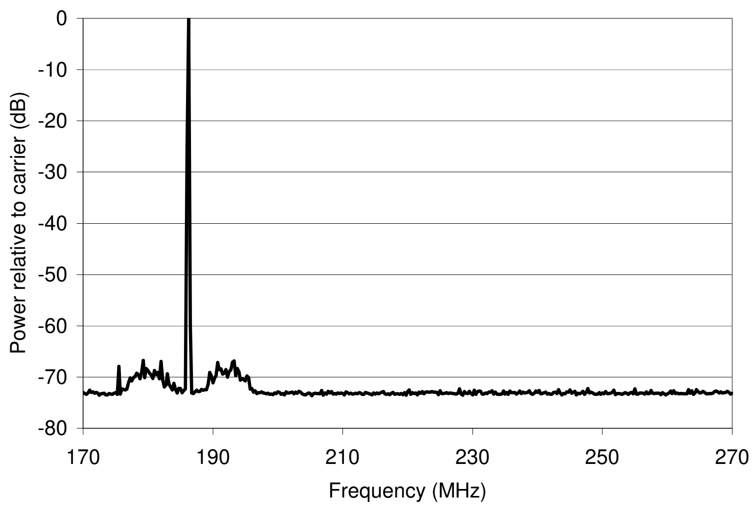

Figure 6.3: SFDR Analysis - IF Above Nyquist

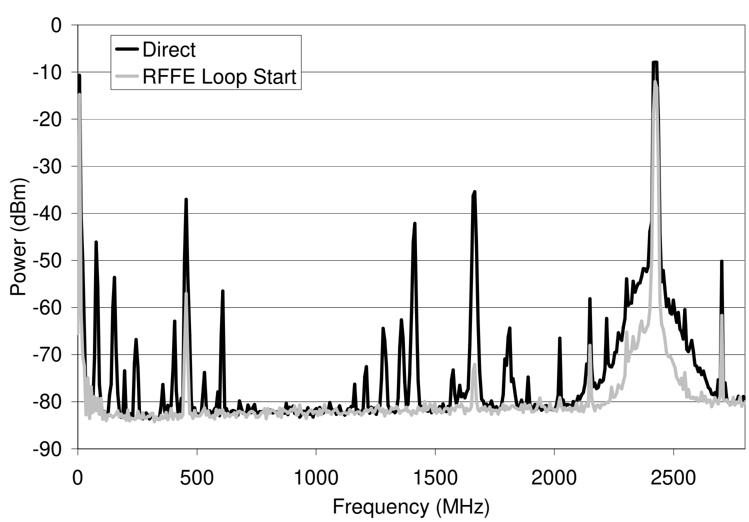

Figure 6.4: Direct Device Output vs. Output After RFFE Loop Start

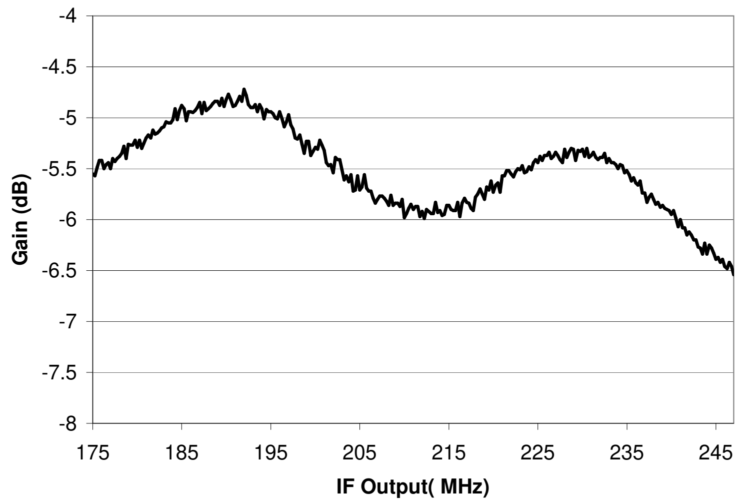

Figure 6.5: Downconversion Flatness

Eliminating Spurious Device Output. Wireless devices generate a broad range of signals in addition to the desired signal output. Unfiltered, these will be aliased into the sampled signal by the SCM. The "Direct" data series in Figure 6.4 shows the direct output of a Senao NL-5354MP ARIES2 broadcasting 1 Mbps data at 2.412 GHz. The figure shows the card's output between 0 and 2.8 GHz directly connected to a spectrum analyzer with a resolution bandwidth of 100 kHz.

The efficacy of the RF Front End at filtering out these undesired signals is shown in the RFFE Loop start series in Figure 6.4. This data was obtained by connecting the Senao NL-5354MP ARIES2 source from the previous test to the RF Port (a) and examining the output of the RF Front End downconversion loop start port (b). Thus, this measurement is after filtering, transmit detection, and transmit switching have occurred. This series shows that the RF Front End successfully filters the vast majority of spurious signals. (It also shows that a small amount of attenuation occurs between the RF input port and the loop start port as expected.)

Flatness. Downconversion performance was measured using an LO of 2.226 GHz at 7 dBm (at the mixer), and an RF signal of 0 dBm swept between 2.401 GHz and 2.473 GHz (the 802.11b/g band in the United States) sent into port (a) the RF Front End's RF Port. Figure 6.5 shows the results measured using a spectrum analyzer attached to (c) the RF Front End's IF output port (30 kHz resolution bandwidth was used.) Gain variation during the downconversion process varies by 1.7 dB. Thus, the RF Front End produces a reasonably flat frequency response.

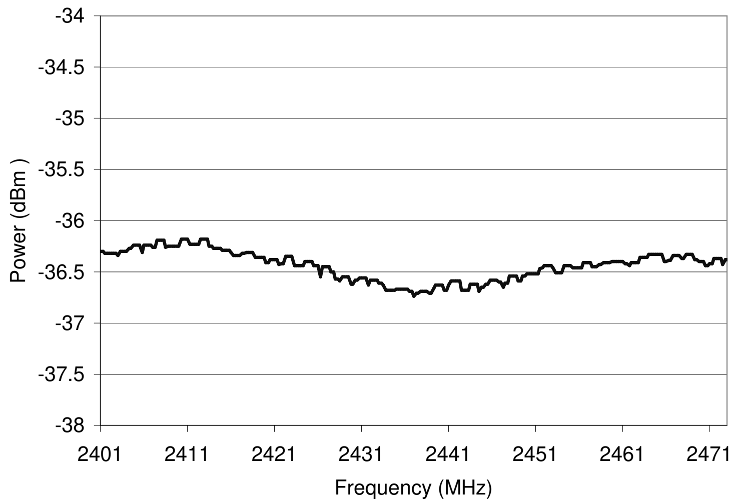

Figure 6.6: Upconversion Flatness

6.1.2 Upconversion.

Upconversion performance was then measured using an LO of 2.396 GHz at 7 dBm, and an RF signal of -10 dBm swept between 5 and 77 MHz into (f) the RF Front End's IF input port. Figure 6.6 shows the results measured using a spectrum analyzer attached to (a) (cable loss is not accounted for in these measurements.) Gain in this case varies by approximately 0.7 dB which is a very flat frequency response.

6.2 Signal Conversion

This section presents several tests to verify that good performance is obtained by the Signal Conversion Module.

Note that the Signal Conversion Module can be characterized by many of the same metrics applied to the RF Front End. The limiting factor for many metrics is the A/D converter - an Analog Devices AD9430. This section will not discuss specifications for this device as they can be found in the corresponding datasheet [13].

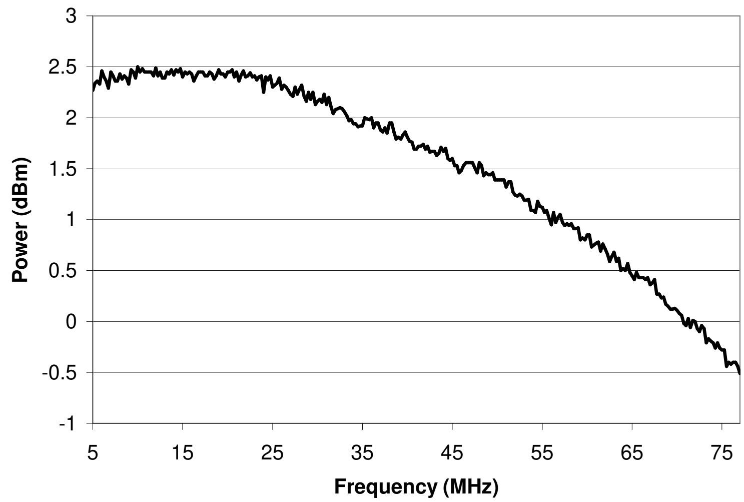

Figure 6.7: SCM D/A Conversion Flatness

6.2.1 Flatness.

The Signal Conversion Module should ideally have a flat frequency response for both D/A conversion and A/D conversion in the same way that the RF Front End would ideally have flat RF conversion frequency responses. This section measures both A/D and D/A conversion flatness.

D/A Conversion Flatness. SCM D/A conversion flatness over frequency was measured by using the SCM FPGA to digitally generate a full-scale (maximum D/A power) sine wave between 5 and 77 MHz. The output was measured using a spectrum analyzer set at 30 kHz resolution bandwidth connected to the IF output port (e). Figure 6.7 shows that, in this case, there is a noticeable lack of flatness. A technique for correcting this imperfection in the current implementation is discussed further below.

A/D D/A Composite Flatness. Composite A/D D/A performance was measured by sweeping a 0 dBm signal between 5 MHz and 77 MHz into the SCM IF input port (d) with the SCM sampling at 170 Msps. The digitized samples were forwarded to and reconstructed by the D/A converter. The output was measured using a spectrum analyzer set at 30 kHz resolution bandwidth connected to the IF output port (e).

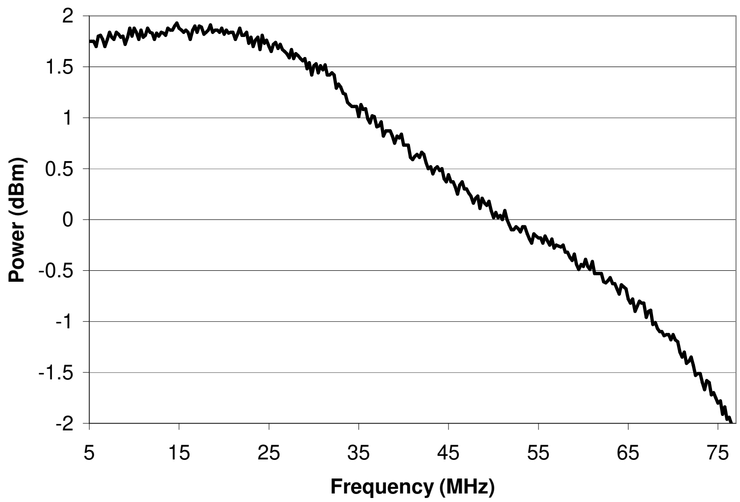

Figure 6.8: SCM A/D and D/A Conversion

Figure 6.8 shows the composite SCM A/D D/A frequency response. Again, there is a lack of flatness, though it is quite close to that of the D/A alone implying that the A/D frequency response is flat.

A/D Conversion Flatness. A/D signal conversion flatness was computed by subtracting the data in Figure 6.7 from those in Figure 6.8.) Figure 6.9 shows the resulting variation in signal conversion gain over frequency. Performance is very flat. (Loss is present due to signal loss on the PCB before the A/D die, imperfect signal conversion, and test calibration inaccuracy.)

Correcting for Lack of Flatness. While the ADC-DAC composite lack of flatness is noticeable, it can be corrected using digital filtering on board the SCM. For example, Figure 6.10 shows expected gain variation after a digital correction filter has been applied to the data in Figure 6.8. This data was generated by passing the gain values measured in Figure 6.8 through a 9 tap digital filter[1].

Error Vector Magnitude. A common technique used to measure a signal's physical layer fidelity is to compare the signal under test with an ideal signal; this is done by periodically sampling the signal and plotting the results on a polar graph as shown in Figure 6.11. This polar graph is known as the signal's "constellation". The measured constellation can then be compared against an ideal constellation.

[1]This digital correction analysis was conducted by Kevin Borries.

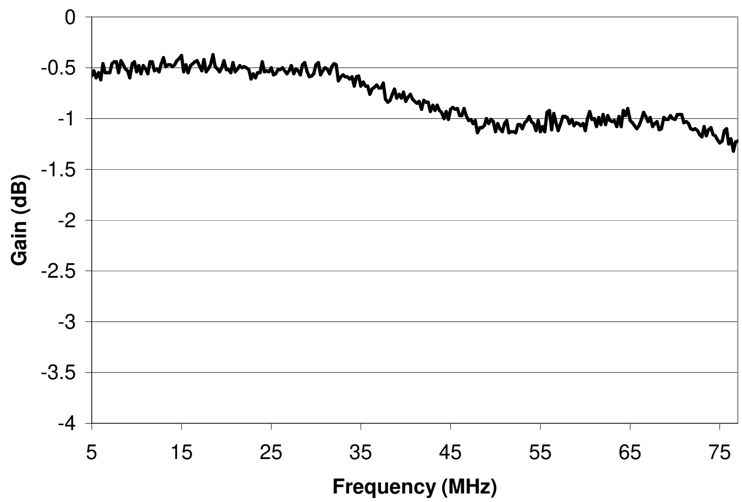

Figure 6.9: SCM A/D Conversion

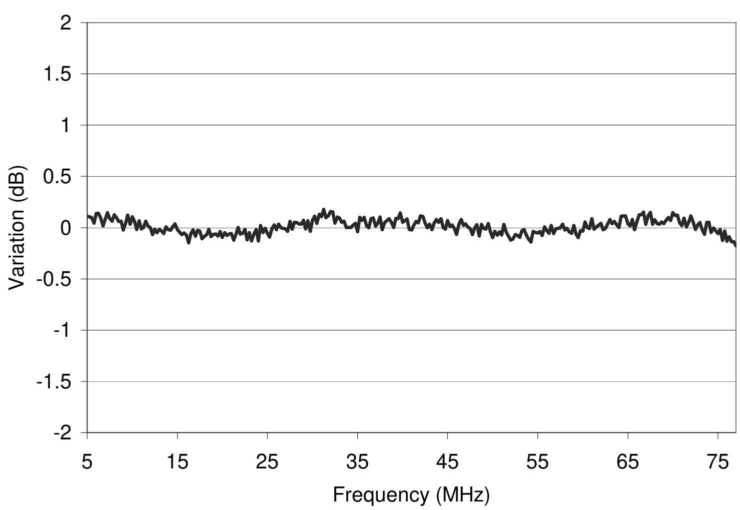

Figure 6.10: Gain Variation of Digitally Corrected A/D-D/A Signal

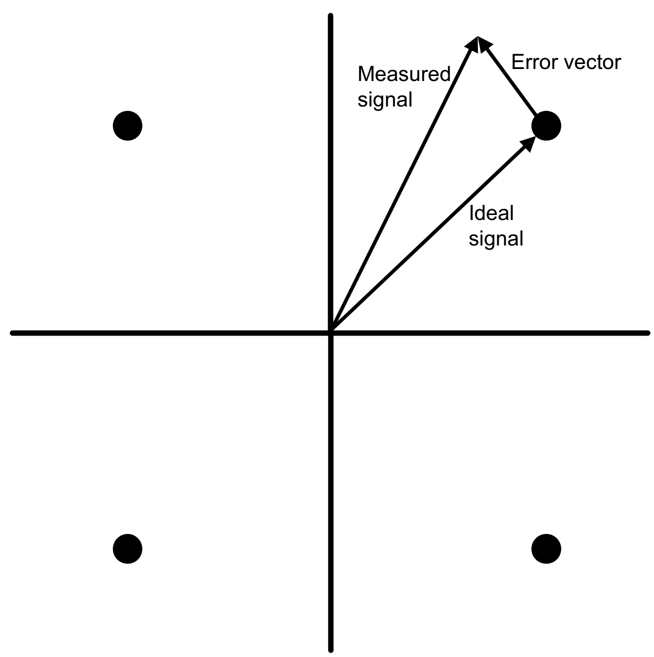

Figure 6.11: Signal Constellation and Error Vector Magnitude (EVM)

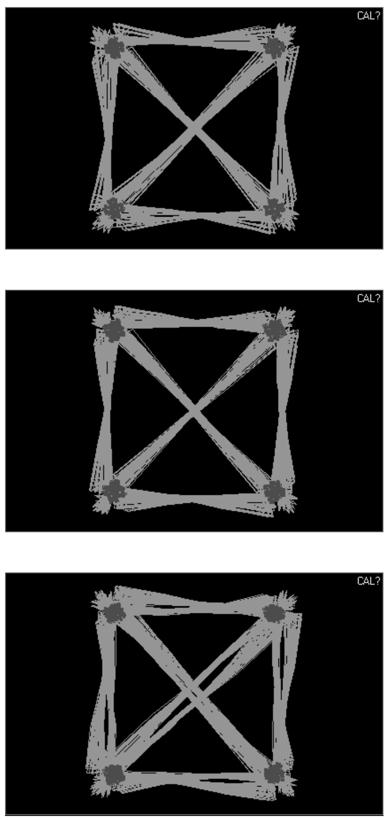

Figure 6.12: Signal Conversion Module EVM

(Figure 6.11 illustrates a quadrature phase shift keying (QPSK) constellation. This modulation represents data using four different phase shifts of the carrier signal; amplitude is held constant. The axis shown in this figure represent the decision boundaries for this modulation scheme.)

The difference between the measured signal and an ideal signal can be quantified by measuring "error vector magnitude" (EVM). EVM is the relative difference between ideal signal constellation points and observed constellation points as shown in Figure 6.11. EVM measures the average magnitude of the error vector (a vector from the ideal constellation point to the observed point) as a percentage of the ideal signal vector's magnitude. As a rough guide: an EVM of around 3% should be considered excellent while an EVM near 10% should be considered poor.

Figure 6.12 shows the results of EVM/constellation measurements for the Signal Conversion Module. In all cases the source signal was generated by a digital signal generator (an Agilent Systems ESG) modulating an 11 Msps QPSK symbol with a Gaussian filter where alpha=0.5. Figure 6.12(a) was obtained from a signal centered at 20 MHz sent directly from a signal generator (ESG) into the measurement vector signal analyzer (VSA). This measurement acts as a reference for the other two measurements; that is, ideally the other signals would be precisely the same as this signal.

Figure 6.12(b) was obtained by passing the signal in (a) through the SCM sampling - in the IF input port (d) and out the IF output port (e) at 170 Msps and then into the VSA. A small degree of degradation is evident.

Figure 6.12(c) measures the undersampling case by repeating the test in (b) with the signal centered at 190 MHz. A greater degree of degradation is evident, but the constellation and EVM are still quite clean. As mentioned earlier, the emulator typically runs in undersampling mode; thus this figure corresponds to the typical operation.

6.3 RF Front End - SCM Composite

This section examines the quality of signals sent through the RF Front End and then through the Signal Conversion Module. (As the VSA only operates at low frequency, the signals are not passed back up into the RF Front End.)

6.3.1 EVM

Figure 6.13 presents EVM/constellation measurements for this setup. As before, in all cases the source signal was generated by a digital signal generator modulating an 11

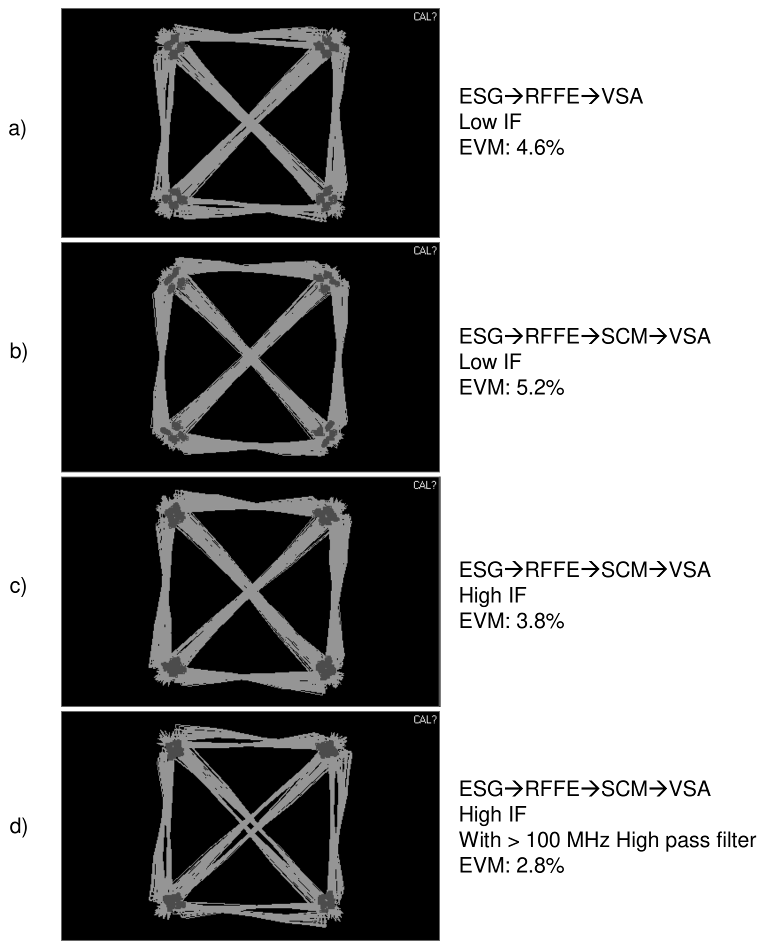

Figure 6.13: RF Front End + Signal Conversion Module EVM

Msps QPSK symbol with a Gaussian filter where alpha=0.5; the signal was centered at 2.417 GHz for all cases. Ideally a reference constellation and EVM would be measured by sending this signal from the ESG directly into the VSA. Unfortunately, the VSA can only measure frequencies less than 40 MHz. As a result, we generate a reference signal by downconverting the signal generated by the VSA using the RF Front End. Figure 6.13(a) shows the reference signal obtained by sending the signal from the ESG through the RF Front End - into the RF port (a) and out the IF output port (c) - with the LO operating at 2.396 GHz. This measurement shows that the generated and downconverted signal is very clean. Subsequent measurements would ideally be identical to this signal.

Figure 6.13(b) measures the degradation due to digital signal conversion when not undersampling. This was obtained by passing the signal in (a) through the SCM - sampling at 170 Msps and then into the VSA. I.e. the source signal enters port (a) and is measured at port (e). A small degree of degradation is evident.

Figure 6.13(c) measures the undersampling case by repeating the test in (b) with LO set to 2.226 GHz which results in undersampling by the SCM A/D converter. A greater degree of degradation is evident in the constellation, and the EVM has been altered from the original signal.

Now when undersampling with a 170 Msps sample rate, all signals should be greater than 170 MHz; thus, any signal under 170 MHz is undesired. Measurements have shown that intermodulation products can be produced in this range particularly near 0 MHz. By placing a filter on the RF Front End IF output, these undesired signals can be removed. These cannot be removed when not undersampling since they are co-located with the desired signals.

Figure 6.13(d) then shows the case with the same setup as in (c) except that the RF Front End is followed by a high pass filter designed to filter out frequencies under 100 MHz before the signal enters the SCM. In this case, some improvement is seen over (c) as these undesired signals are removed.

6.3.2 Spectrum Analysis

The quality of the digitized signal can also be measured by comparing the digitized spectrum with that of a the signal sent directly out from the wireless device. Figure 6.14 compares the spectrum obtained by directly connecting an Atheros-based device - a Senao NL-5354MP ARIES2 - to a spectrum analyzer with that of a signal that has passed through the same setup in Figure 6.13(d). I.e. in the RF Front End's RF Port (a) and out the Signal Conversion Module's IF port (e) with a high pass filter in place before the VSA. Since the signal levels and frequencies are different, the results have been normalized to allow accurate comparison. Figure 6.14 shows

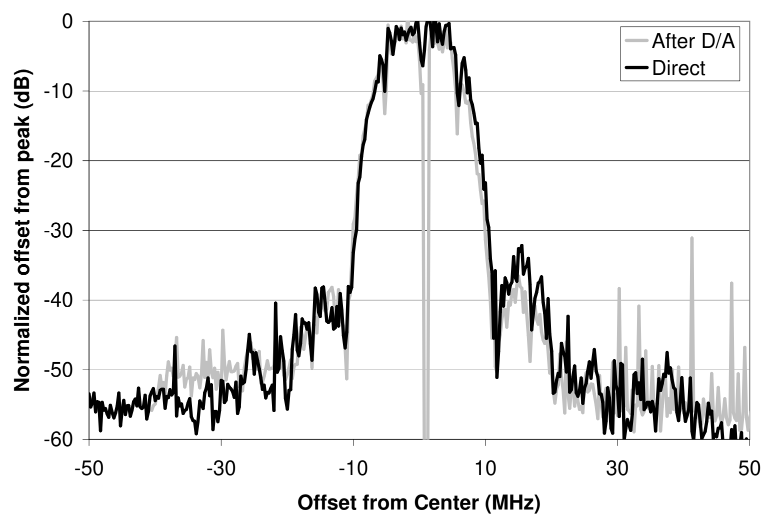

Figure 6.14: Spectral Comparison - Direct Device Output vs. Output after SCM

that the signals are quite similar with the exception of a few spurious signals toward the higher end of the digitized signal. These are almost entirely out of the supported frequency band (which ranges from -36 MHz to +36 MHz in the figure), and hence are largely unimportant. (The large downward spike at the center of the figure is an artifact due to interframe spacing, and can be ignored.)

6.4 Transport Level Fidelity

Figure 6.15 demonstrates that physical and link layer fidelity translate into transport level fidelity by comparing the TCP throughput for two laptops connected via coaxial cable and discrete attenuators versus two laptops connected via Version 1 of the emulator.

(This version used the same A/D converter, a different D/A converter (12-bits), and had somewhat inferior signal integrity. Thus the results obtained for the Version 2 discussed by the majority of this thesis should be somewhat better than those obtained with Version 1.)

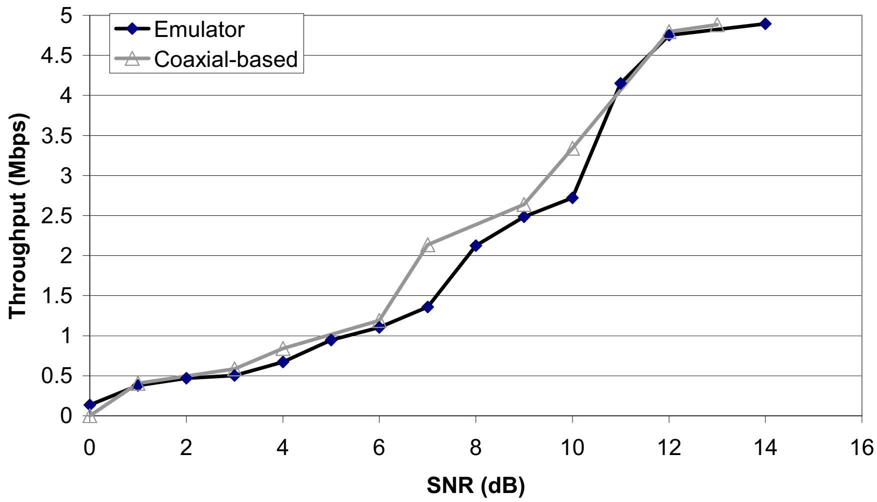

Figure 6.15: Transport Layer Fidelity

Each data point is an average of 20 trials measuring one-way TCP throughput for approximately 5 seconds. Confidence intervals are omitted since they are tight, and the SNR measurement error is dominant (about 1 dB). The results match quite closely and are within the measurement error of the experiment.

6.5 Isolation.

An important benefit of the emulator is the ability to conduct experiments in isolation from external sources of interference. Measuring isolation into the emulator is (almost entirely[2]) reciprocal to measuring isolation out of the receiver. Thus isolation was measured by configuring an emulator node as access point transmitting at 19 dBm and then using an external wireless card attached to a 5.5 dBi antenna to listen for the beacon packets. Beacon packets were captured by configuring the receiver in monitor mode and capturing packets in the immediate vicinity of the emulator. Figure 6.16 shows the results of this test. Two regions of isolation are shown: a moderate isolation region where roughly more than 20% of beacons were received, and a high isolation region shown where roughly less than 20% of beacons were received. The moderate isolation region is depicted as a circle and the high isolation region is depicted as an oval. Outside of these regions, no beacons were received. Thus the current emulator

[2]The RF Front End has separate signal paths for transmit and receive which may radiate and receive in somewhat different ways. The amount of signal radiating from these paths, however, should be reasonably small.

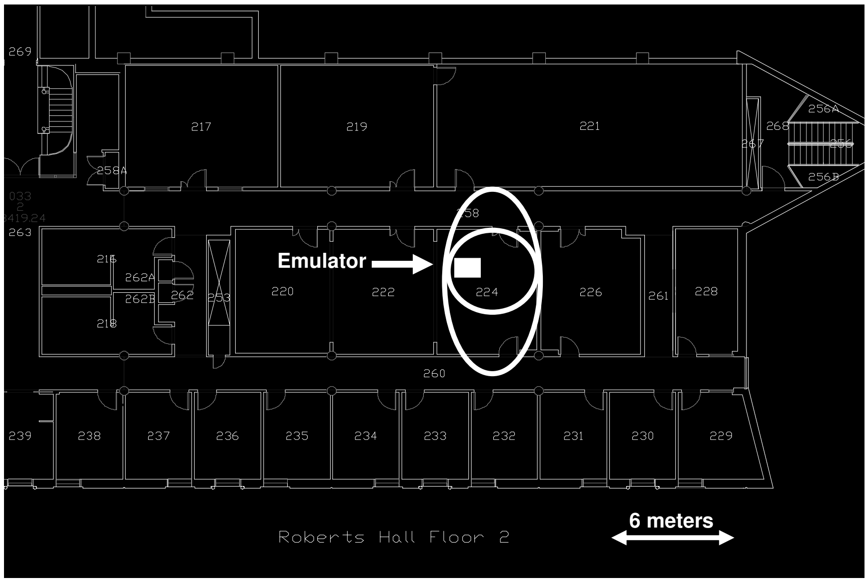

Figure 6.16: External Isolation

requires approximately 6 meters and a separation barrier (e.g. a wall) to achieve good isolation in its current environment. The main limitation on this isolation is the need to sacrifice perfect shielding in order to allow the RF nodes to be cooled. Additional work - such as adding external shielding - should cut the interfering range down to a few meters even for strong transmitters.

Building a large setup requires placing RF nodes in close proximity to each other. To allow for this while maintaining internal isolation, each emulator node is mounted inside of a shielded rack-mount chassis. By altering the external isolation test to measure internal isolation, the nodes attached to the emulator were verified to be effectively isolated against undesired transmission to each other despite their close proximity (8.75 inches). This was measured by conducting two reception rate vs. signal strength tests. Three RF Nodes were used: a transmitter, a receiver, and an interferer. In the first test, the transmitter sends broadcast packets to the receiver through the emulator over a series of signal strengths down to the minimum receivable signal strength (where interference causes the most harm) and the receiver records the number of packets received; the interferer is silent for this test. In the second test, the same experiment is conducted except that the interferer constantly blasts large broadcast packets; the emulator is configured to disallow any signal from the interferer to reach any other node. In both cases the reception rate was the same, thus the interferer was effectively isolated in this case and isolation is achieved.

6.6 Summary

In order to provide effective emulation the emulator must accurately digitize and reconstruct RF signals. This chapter examined the signal integrity performance of the emulator's implementation with using a variety of signal integrity tests. This implementation was shown to provide good performance across a variety of metrics. The weakest point of the current implementation is a lack of flatness over frequency. While not yet implemented, this discussion showed the emulator can digitally correct much of this imperfection.

Chapter 7

Link and Device Characterization

As discussed in Chapter 1, while hardware-based experimentation clearly achieves the most physical layer realism, practical considerations such as ease of development, control, and experimental repeatability have made simulation the dominant experimental technique.

Recent work [33], however, has shown that unless a great deal of care is taken, simulation based-research can produce imprecise results. While careful simulation setup is a necessary condition for producing valid experimental results, it is clearly not sufficient in and of itself. A more fundamental condition that must be met to achieve valid wireless networking simulation results is the use of an accurate simulator. Not only must this simulator have a correct networking protocol stack, but it must accurately reproduce wireless signal transmission, propagation, and reception. As wireless signal propagation is only completely accurate when using real hardware, wireless simulations will always be a coarse approximation. It is critical, nevertheless, that wireless device behavior be modeled as accurately as possible.

Despite the massive amount of research relying on simulation, relatively little work has been done to validate the accuracy of these simulators. Initial work [71, 21, 45] has shown that wireless network behavior is complex and that low level behavior has a strong impact on higher layer results. Additional work [32] has shown that the most commonly used simulator - ns-2 - produces results that vary significantly from real-world experiments.

Moreover, real-world measurements [1] [44] show that wireless networks exhibit a variety of behaviors that are difficult to understand - such as link asymmetry - and are simply not recreated in current simulators. A controlled investigation of link-level and device behavior will enable a better understanding of these difficult-to-understand phenomena.

This chapter leverages physical layer wireless network emulation to undertake a detailed analysis of 802.11 device and link-level behavior using real wireless hardware. This analysis illustrates the power that the emulator can bring to bear on wireless network link-level and device behavior. The emulator enables complete control over signal propagation between the devices under examination, and enables the detailed analysis presented here. For many items discussed, conducting these experiments using other techniques would be either impractical or impossible.

This chapter also presents a series of case studies illustrating how the emulator can be used to conduct link level related experiments that illustrate how link-level insight can be gained into a variety of issues. Finally, a small number of case studies are presented to illustrate some of the range of experiments that it is possible for users to conduct.

7.1 Link Behavior

This chapter begins with an analysis of link level network behavior conducted as a result of discussions with the Roofnet research group. Roofnet is a large-scale outdoor wireless mesh network. Analysis of link-level Roofnet behavior [1] revealed many difficult to understand phenomena that did not correspond to the typical link-level behavior assumed by simulators. The object of the work in this section was to develop a detailed understanding of link behavior in a controlled environment using emulation based experiments. The results of these experiments formed a reference which the actual behavior observed in Roofnet was then compared against [1].

This section will present the results of these controlled experiments beginning with a look at AWGN reception behavior, followed by experiments investigating hidden and exposed nodes, packet capture behavior of two competing transmitters, off-channel reception behavior, off-channel interference, and the causes of link-asymmetry. In addition, this section uses insights gained here to look at the impact of link-level behavior on WLAN performance. This analysis should contribute to a better understanding of the link-level behavior of 802.11 hardware by replacing conventional assumptions (e.g. interference range is twice the reception range) and possible misconceptions with actual recorded behavior. Moreover, this work should enable the development of more accurate wireless network simulators. Section 7.2 will use the results obtained in this section to analyze 802.11 uplink behavior in a wireless WLAN setting.

7.1.1 Experimental Setup

Wireless link behavior is characterized through a series of experiments using three wireless NICs. In some experiments, there is an implicit fourth receiver for which characterization is not necessary. All experiments in this chapter, utilizes Senao 2511CD Plus Ext2 NICs. These cards were chosen since they are based on the Prism 2.5 chipset which is in widespread use in the research community and is one of the most popular 802.11b chipsets. The Senao cards are a mature design with very good performance. While the precise values reported here are specific to these cards, the observations made should apply to many other hardware configurations. For instance, the robustness of 802.11b's 1 Mbps spread spectrum modulation to interference is a fundamental characteristic of the standard, and all standard compliant hardware should have this feature.

Fine grained characterization of wireless link-level behavior requires tight control over signal propagation between the transmitters and receivers. To obtain this control, two techniques are used: digital wireless network emulation as discussed in previous chapters, and communication via coaxial cable. In addition, standard RF measurement devices are utilized to measure specific device characteristics.

Version 1 of the emulator [29] is used to conduct emulation-based experiments. For the work discussed in this section four laptops are connected to the emulator via coaxial cable. Inside this emulator, the RF signals from each laptop are digitized as discussed earlier. The signals are then digitally attenuated and combined between laptops. This enables complete control over the signal paths between any pair of laptops. Hence, for any signal path from a transmitter to a receiver, the emulator can explicitly set the received signal strength. This allows the construction of arbitrary network topologies for test purposes.

7.1.2 Clear-channel Reception

The first test conducted was to measure clear-channel reception behavior. In this test, a single transmitter and a single receiver were used. The emulator was used to vary the RSS (received signal strength) at the receiver from -102 dBm to -80 dBm in 1 dB increments. For each RSS value, the transmitter sent 200 broadcast packets to the receiver. The receiver then recorded the number of successful packets. As broadcast packets do not use link-level retries, this experiment measured packet delivery rate as a function of RSS. This test was repeated for each of the four 802.11b transmission rates. In all cases, the same transmitter and receiver were used. The results are shown in Figure 7.1; note that different pairs of wireless transmitters and receivers will have results that vary slightly from the results shown in this graph (see Figure 7.24.) These results indicate that this receiver is quite sensitive (the noise floor and carrier sense

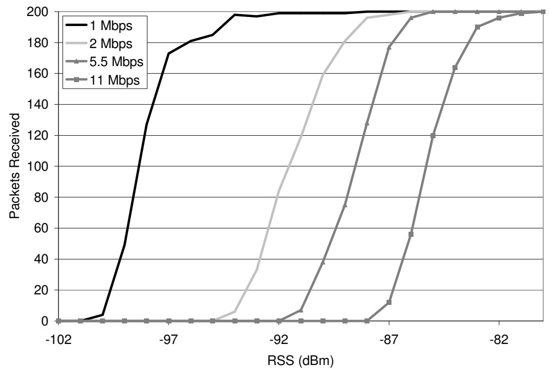

Figure 7.1: Clear Channel Reception

of the Senao cards used in this section is approximately -99 dBm.) As expected, increasing transmission rates required larger signal strengths in order to be received successfully.

7.1.3 Capture and Acquisition Under Delayed Interference

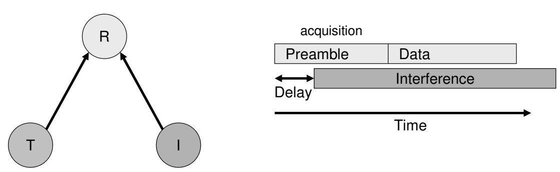

Figure 7.2: Capture Under Delayed Interference

An essential element in understanding and modeling wireless packet reception is understanding what happens when two competing signals arrive at a receiver. Is a packet received? Is there a collision? Can a receiver begin acquiring a new packet in the middle of receiving a weaker packet? Simulators have made contradictory assumptions, but little data exists on the behavior of real receivers.

This section quantifies the effects of timing and received signal strength on a receiver's ability to capture a single desired signal in the presence of an undesired interfering signal. The subsequent section will discuss the effects of received signal strength on the outcome of two competing desirable signals.

Figure 7.2 shows the setup for these experiments. In this case, a transmitter T sends traffic to a receiver R. A second transmitter I plays the role of the interferer. T and I are hidden [60] [20] and cannot hear each other's transmissions. I constantly

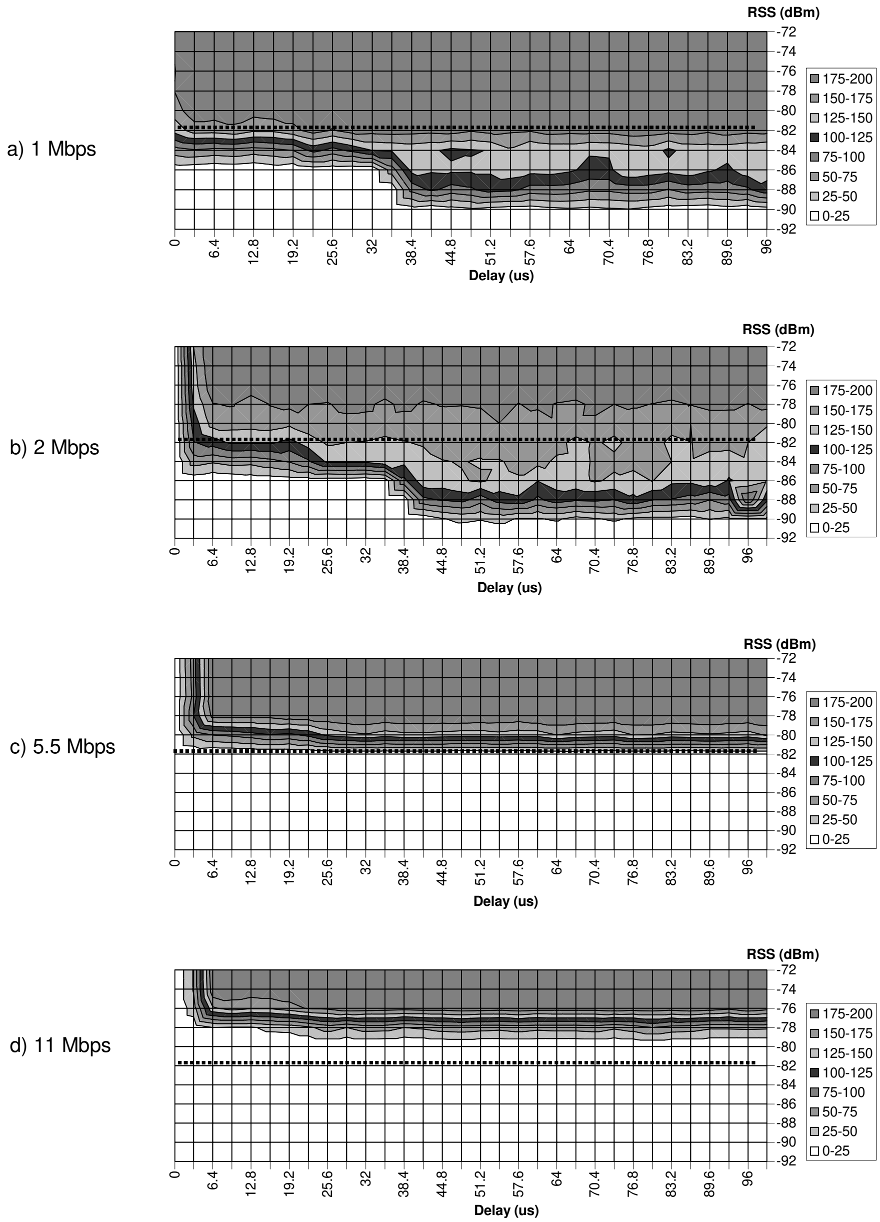

Figure 7.3: Capture Under Delayed Interference

sends interfering 1 Mbps 1500 byte broadcast packets at -82 dBm. The emulator's default behavior is modified for this experiment to allow R to hear transmissions from I only if: 1) T was actively transmitting, and 2) T's current transmission had been active for a specified delay. In this way, the effect of interference timing on packet reception was investigated by varying the delay of I with respect to T's transmission in 3.2 microsecond increments between 0 and 96 microseconds. The strength of the RSS of T at R was also varied in 1 dB increments between -72 dBm and -92 dBm in order to see how delay and relative received signal strength interact.

Figure 7.3 shows the results of these experiments. The x-axis shows the delay in microseconds, and the y-axis shows the RSS of T at R. For each delay, RSS combination, 200 packets were sent from T to R and number received was recorded as shown in the figures.

Figure 7.3(a) shows capture performance for 1 Mbps. Three regions of performance with respect to delay can be seen: 0 microseconds, (0-37] microseconds, and $>$ 37 microseconds. As expected, reception is worst when the interference has no delay, though packet reception is still possible even when interference arrives at the same time as the desired transmission. If the interference is delayed by at least 3.2 microseconds, a noticeable improvement in performance is seen due to the fact that the receiver has begun acquisition of the desired signal. At delays greater than approximately 32 microseconds, there is a further improvement of approximately 4 dB in reception behavior. This improvement is due to the receiver having acquired the transmission. Interference that occurs before this point may prevent signal acquisition. After this point, the interference must be strong enough to overcome the demodulation that is occurring. Of particular note is the fact that after signal acquisition, interference can be rejected even if it is stronger than the transmission.

At 2 Mbps (Figure 7.3(b)), delayed capture behavior is similar to 1 Mbps, though somewhat worse, as expected. At 5.5 (Figure 7.3(c)) and 11 Mbps (Figure 7.3(d)), however, reception is no longer possible when the signal is weaker than the interfering signal.

The results discussed above have important ramifications on MAC design and performance. 802.11's carrier sense mechanism operates without respect to the cell in which a station resides. Not only will this cause transmitters to needlessly defer (an exposed node situation), but since carrier sense works without respect to the cell in which transmitters reside, transmitters in different cells (i.e. with different receivers) will tend to synchronize their attempted transmissions. The idea is to limit the time that the medium is experiencing collisions [51].

As the results discussed above have shown, this is the worst possible timing for packet capture since the very start of a frame is the most vulnerable. A more sophisticated MAC could take packet capture into consideration and avoid needlessly synchronizing transmitters in different cells. A shift of a few microseconds would greatly improve capture performance, and would have negligible impact on the time that the medium might experience collisions. In addition, a capture aware MAC could calculate the degree of interference likely to be present and realize when deferring was necessary due to possible poor signal-to-interference and noise ratio at the receiver.

7.1.4 Capture with Competing Transmitters

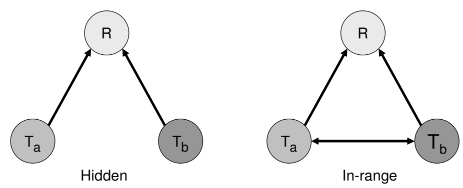

Figure 7.4: Capture

The previous section discussed the effects of timing and received signal strength on a receiver's ability to capture a single desired signal in the presence of an undesired interfering signal. This section now discusses the effects of received signal strength on the outcome of two competing desirable signals.

This capture behavior is quantified using a series of experiments with the configurations shown in Figure 7.4. The goal of these experiments is to establish the reception outcome given the RSS at R from T_a and T_b without controlling precisely for interference timing. That is, this experiment doesn't control when packets from T_a and T_b arrive at receiver R.

In both configurations shown in Figure 7.4, these tests proceeded as follows: two transmitters T_a and T_b constantly send broadcast packets at a very high rate to the receiver R. At first, the channels are all "turned off" in the emulator so that no packets are actually received at R. The emulator then simultaneously turns on the channels and sets the RSS at R from each transmitter to the desired values. In the "hidden" configuration, the emulator does not allow T_a and T_b to hear each other's transmissions. In the "in-range" setup the emulator sets the RSS from T_a to T_b at -80 dBm and vice versa, so that T_a and T_b will always hear each other's transmissions. After a fixed time interval, the emulator shuts off the channels from T_a and T_b to R. R then records how many packets it received from each transmitter. This experiment measured all combinations of RSS values from T_a and T_b at R between -102 and -72 dBm in 1 dBm intervals. This test was repeated for all 802.11b transmission rates.

Figure 7.5 shows the results. In each of these figures the z-axis is the number of packets received from both T_a and T_b at R. In many RSS combinations, however,

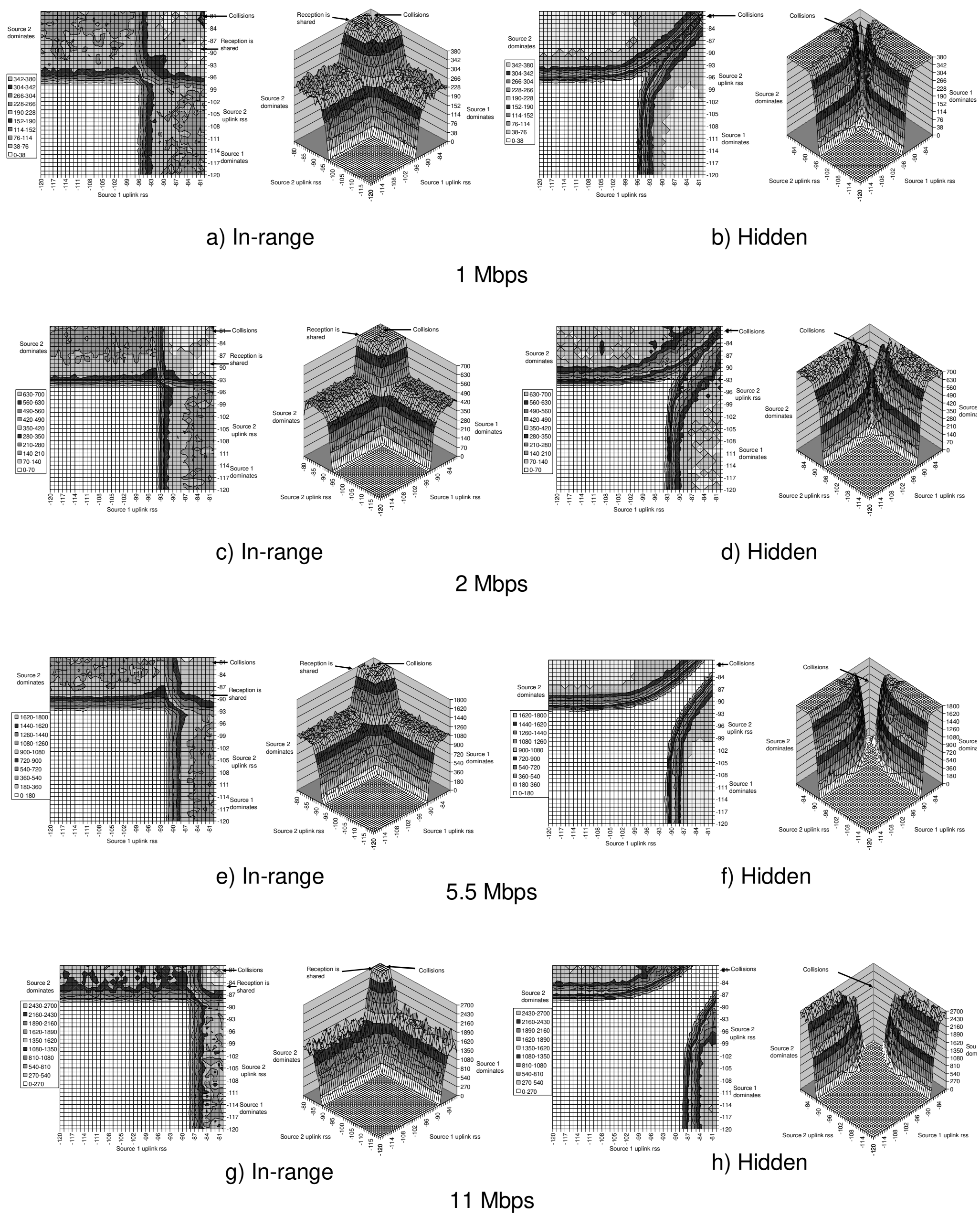

Figure 7.5: Packet Capture Results

packets were actually only received from one or the other; the regions where one source or the other dominated are labeled on the plots.

In all in-range cases, these results show that when the RSS at R from both T_a and T_b was high CSMA did a good job of allowing the two nodes to share the medium, and only a small number of collisions occurred. As expected when one transmitter was out of range of R and the other was in range, the number of packets received for the in-range cases was roughly half of the channel capacity since the reception between T_a and T_b is still good, and they defer for each other's transmissions irrespective of the number of packets successfully received at R. In an actual network, this would only occur when the out-of-range node was sending to a receiver other than R (or broadcasting) since unicast communication requires acknowledgement of successful reception. For these "exposed node" cases, the in-range node may be needlessly deferring since the out-of-range node isn't communicating with the same receiver.

An important question is what happens when transmissions from two nodes overlap in time at a single receiver. The "hidden node" configuration tests investigate this question. In hidden node situations, T_a and T_b send at full rate since they are out of carrier sense range. Looking at the 1 Mbps results, collisions only occur for a very narrow range of signal strengths where the RSS at R is nearly identical from both T_a and T_b. Hence, in the exposed node situation, waiting is likely unnecessary if the transmission of the in-range node is 1 Mbps. At higher transmission rates, the range over which collisions occur grows especially for 5.5 and 11 Mbps rates. Hence, higher transmission rates require a larger signal to interference and noise ratio (SINR) in order to be captured successfully.

In summary, these tests have shown that for low transmission rates, collisions occur only when the received signal strengths of the competing signals at a receiver are nearly equal. Hence, packets sent at low rates - in particular management and control packets such as beacons, RTS, CTS, and ACK - are very robust to interference. At higher modulation rates, however, a broader range of received signal strengths will interfere. Nevertheless, even high modulation rates will very often capture packets in spite of interference. Hence, deferring transmission due to an interfering source below the capture threshold is not necessary and hurts network performance. Section 7.2 will use this data to illustrate the occurrence of this inefficiency in 802.11 networks. Finally, an important corollary of these results is that recreating realistic capture behavior in simulators requires more than the fixed threshold that is commonly used, but a realistic model such as the data gathered in these tests.

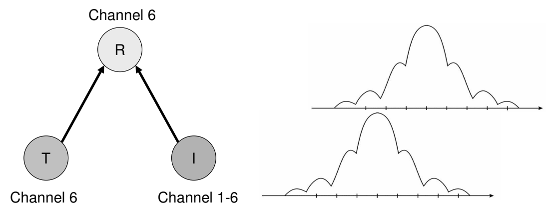

Figure 7.6: Off-channel Interference

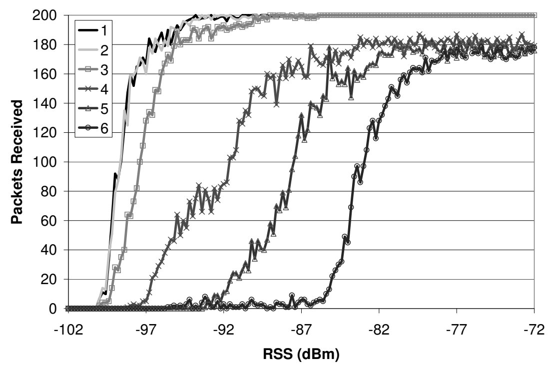

Figure 7.7: Off-channel Interference, 1Mbps, No Delay

7.1.5 Off-channel Behavior

Off-channel Interference

In the United States, eleven 802.11b channels are available in 5 MHz increments from 2.412-2.462 GHz. Each 802.11b channel is designed to have 22 MHz occupied band-

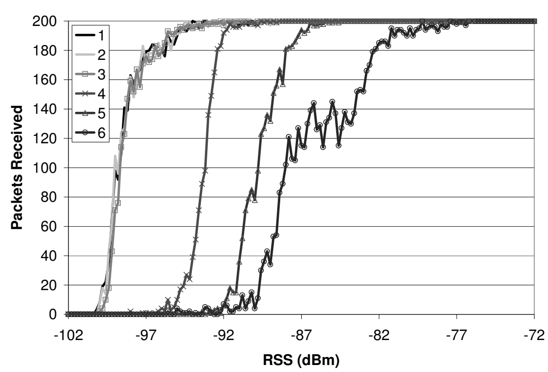

Figure 7.8: Off-channel Interference, 1Mbps, Large Delay

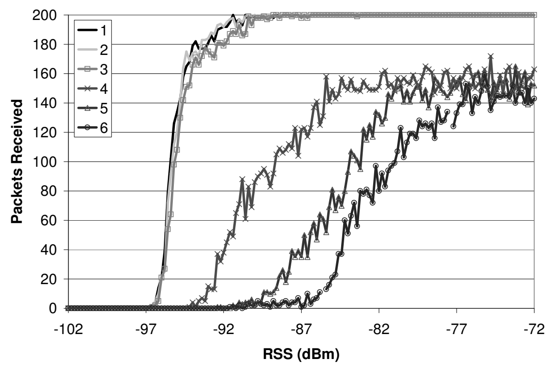

Figure 7.9: Off-channel Interference, 2Mbps, No Delay

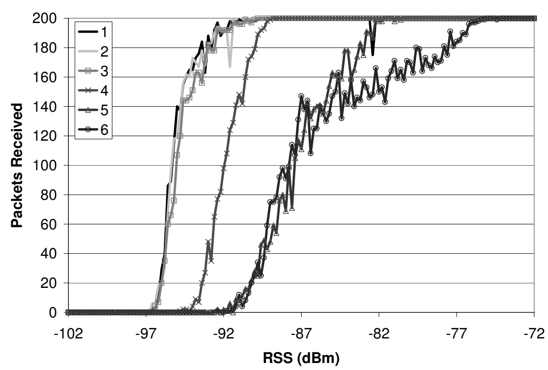

Figure 7.10: Off-channel Interference, 2Mbps, Large Delay

width which implies that a total of three 802.11b signals may coexist - on channels 1, 6, and 11 - without interfering. Ideally, adjacent 802.11b cells would utilize different channels. Unfortunately, it is frequently impossible to deploy an 802.11b network without placing some adjacent cells on the same frequency. For this reason, some have advocated using four channels [39] despite the fact that there would be some signal overlap.

While there is some evidence to support this idea, there has not been a tightly controlled measurement of the impact on real hardware.

In order to quantify the viability of this 4-channel proposal and, more importantly, to understand the impact of off-channel interference on successful packet capture, this section presents measurements of the impact of off-channel interference on packet reception using the setup shown in Figure 7.6.

In this experiment there are two transmitters T and I and a single receiver R. Both T and R are on channel 6; I plays the role of an off-channel interferer on channels

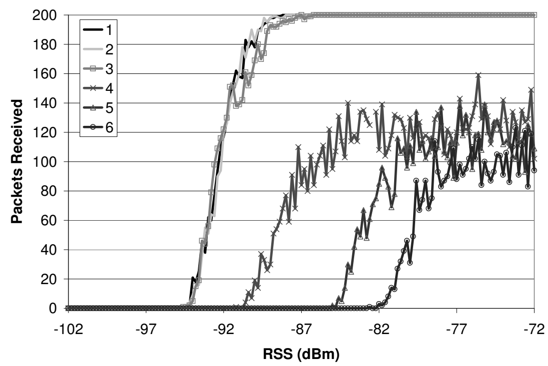

Figure 7.11: Off-channel Interference, 5.5Mbps, No Delay

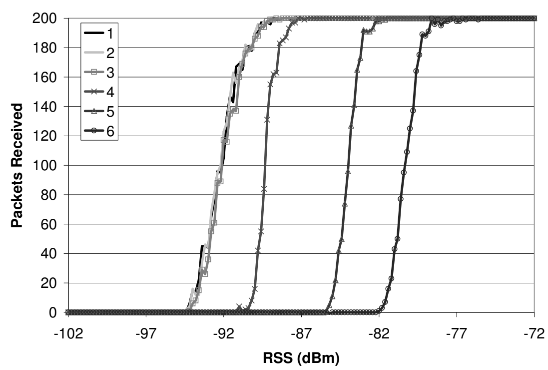

Figure 7.12: Off-channel Interference, 5.5Mbps, Large Delay

1-6. As in the delayed capture test discussed in Section 7.1.3, the interference from I is controlled so R only hears the signal from I some specified delay after R begins to hear a packet from T. For this test, two delay values are used: 0 and 384 microseconds i.e. immediately, or well after packet acquisition. For each channel that I is placed on, the RSS at R from I has held constant at -82 dBm while the RSS at R from T is varied between -72 and -102 dBm. For each channel, RSS combination T sends a series of packets to R and R records how many made it through successfully. Packets were broadcast, so no retries took place. This test was repeated for all four 802.11b transmission rates.

Results are shown in Figures 7.7- 7.14. For all tests where interference was prevented until well after packet acquisition, the impact of interference from channels 1, 2, and 3 was low and virtually identical. Channel 4 degraded performance approximately 4 dB, while channels 5 and 6 degraded performance more significantly.

For tests where interference was allowed to occur at the start of packet reception,

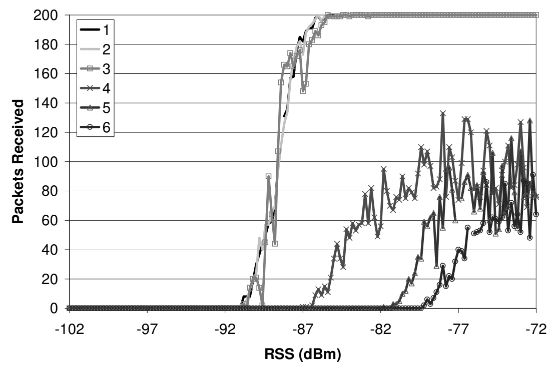

Figure 7.13: Off-channel Interference, 11Mbps, No Delay

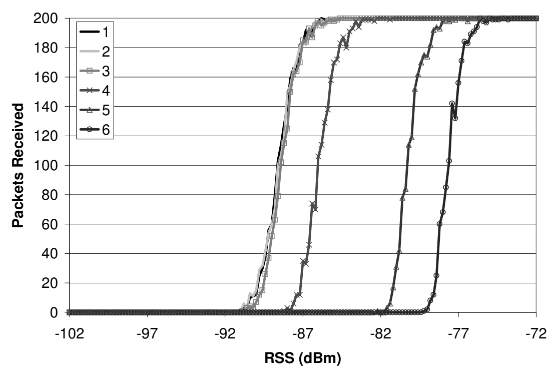

Figure 7.14: Off-channel Interference, 11Mbps, Large Delay

the interference of channels 1, 2, and 3 was still nearly identical though channel 3 was slightly worse in some cases. Interference from channels 4-6 was much more significant in this case.

To investigate the effect of stronger interference, the 11 Mbps large delay tests were repeated, but with an interference of -72 dBm. As shown in Figure 7.15, the larger interference has a strong impact when the interferer is on channels 4-6. When the interferer is on channels 1-3, interference impact is approximately 2 dB stronger than it was with -82 dBm of interference.

These tests show that a well-designed receiver can cope quite well with off-channel interference that is at least 3 channels away. This is an important result as it demonstrates that the 802.11b five channel separation that is typically used is overly conservative for well-designed receivers. Thus, these tests have shown that 802.11b networks can safely use four channels in place of the typical three and reap nearly a 33% improvement.

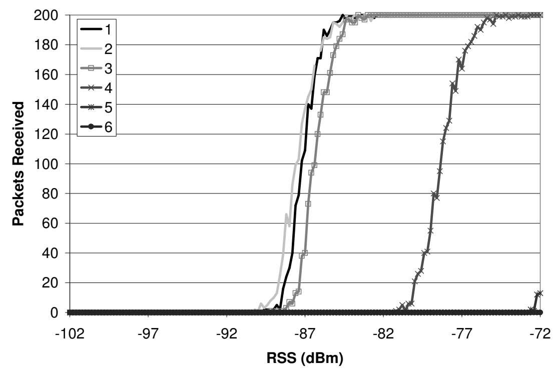

Figure 7.15: Off-channel Interference, 11Mbps, large delay, -72 dBm Interference

Off-channel Reception

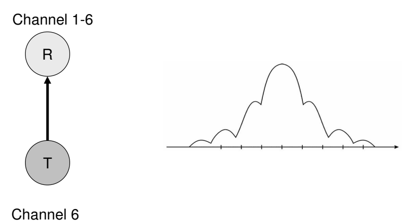

Figure 7.16: Off Channel Reception

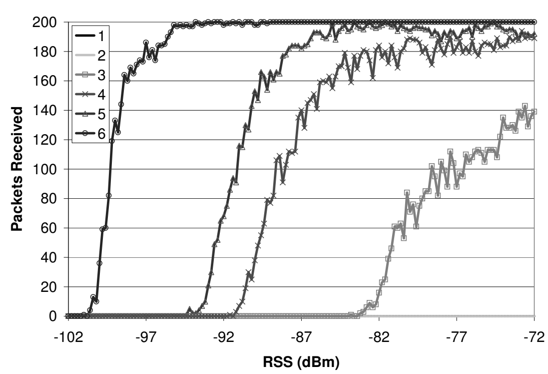

Figure 7.17: Off-channel Reception, 1 Mbps

Recently the observation that some off-channel packets can be received has lead to an ambitious proposal for leveraging off-channel communication for purposes such as bridging between channel regions in multi-hop networks. The utility of this proposal,

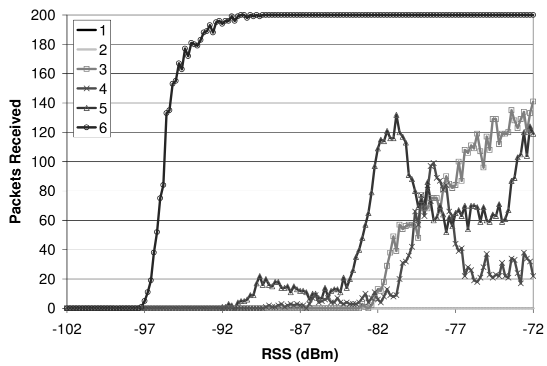

Figure 7.18: Off-channel Reception, 2 Mbps

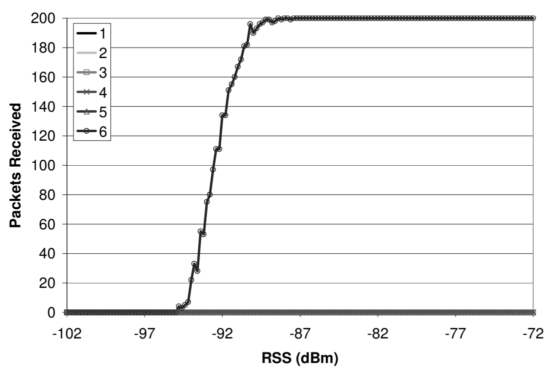

Figure 7.19: Off-channel Reception, 5.5 Mbps

however, clearly relies on the efficacy of off-channel communication which has not been analyzed in a controlled manner.

To remedy this lack of knowledge of off-channel reception behavior, this section characterizes off-channel reception as shown in Figure 7.16. In this test, there is a single transmitter receiver pair. The receiver is varied from channels 1-6 while the transmitter remains on channel 6. (Note that there is no interference at all in this test.)

For each receiver channel, the RSS at the receiver from the transmitter was varied between -102.0 and -72.0 dBm. For each channel, RSS pair the transmitter sent 200 broadcast packets to the receiver and measured how many were received. This test was repeated for all 802.11b transmission rates.

Figure 7.17 shows the results of this test for 1 Mbps. At 1 Mbps, off-channel communication appears to work as anticipated by providing increasing isolation as the channel separation increases.

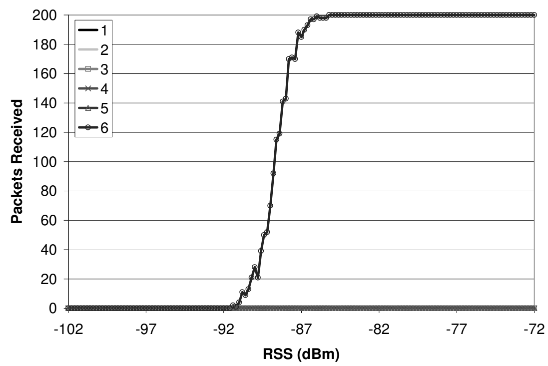

Figure 7.20: Off-channel Reception, 11Mbps

At 2 Mbps, however, this scheme begins to break down as shown in Figure 7.18. For this modulation, reception is possible, but only when the signal is strong and even then it's imperfect. Also, the fact that packet delivery rate is not monotonically increasing with RSS hints that signal distortion may be occurring. At 5.5 and 11 Mbps, things are even worse. Up through -72.0 dBm, the receiver received no off-channel packets as shown in Figures 7.19 and 7.20.

The trouble with off-channel reception likely lies with several elements of the receiver. For instance, when the receiver filter is applied off-center with respect to the modulated signal's center frequency, the result is distortion of the signal in time. Also, the receivers acquisition circuitry may be unable to acquire the signal. The 1 Mbps transmission rate uses BPSK modulation which is somewhat robust to these effects. All remaining bit-rates use QPSK modulation which is much more susceptible to these effects.

Unlike the previous section where emulator-based measurements confirmed the efficacy off previously proposed off-channel techniques, the results presented here have shown that the efficacy of off-channel reception is limited. In particular, off-channel reception is only effective at the lowest transmission rate of 1 Mbps or when the received signal is extremely strong. Thus, while this technique may prove useful in some unique circumstances, it is unlikely to be broadly applicable since alternative designs may provide better performance.

7.1.6 Multipath Performance

Link behavior in the presence multipath was then measured. To do this the emulator was configured to emulate the signal propagation environment shown Figure 7.21 using three different primary ray strengths (-70 dBm, -90 dBm, and -95 dBm). For

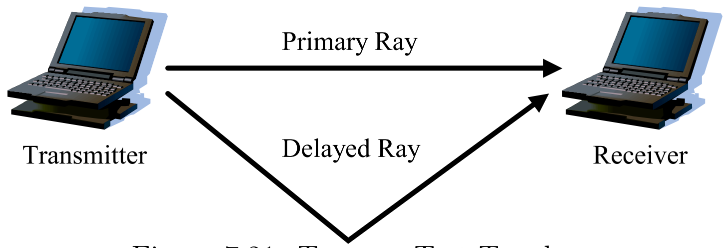

Figure 7.21: Two-ray Test Topology

each primary ray strength, a delayed ray was emulated at all 2 dB increments of attenuation between the primary ray strength and -100 dBm. For each primary ray, secondary ray signal strength combination, the secondary ray's delay was varied between 0 and 2.22 us in 0.0185 us increments. For each of these combinations, a test was conducted by transmitting 500 packets, of 1500-bytes each, from the sender. The receiver then measured the packet delivery rate and other on-card statistics such as signal and noise measurements (for successfully received frames).

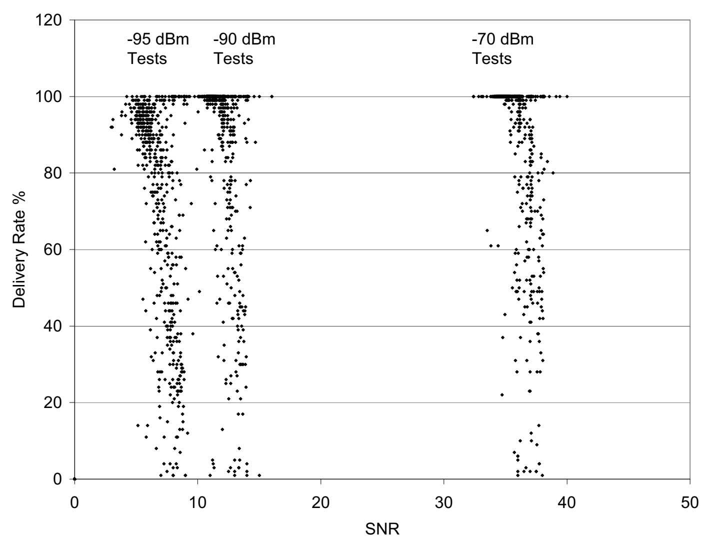

Figure 7.22: Two-ray Delivery Rate vs. SNR

As seen in Figure 7.22, the delivery rate exhibited large variation for different delay spread, delayed signal strength combinations (each point represents a the delivery rate for one primary ray strength, delayed ray strength, delay spread combination). Hence, SNR may be a very poor indicator of packet delivery rate when significant multipath is present.

7.1.7 Link Asymmetry

Several research groups have independently observed asymmetric wireless link behavior. In particular several instances of asymmetric packet delivery rate have been observed; i.e. cases where for two wireless nodes A and B the packet delivery rate from A to B is not the same as the packet delivery rate from B to A. As this is an important observation, this section now discusses why such behavior might exist.

7.1.8 Causes of Link Asymmetry

There are a variety of factors that can - and many that cannot - contribute to link asymmetry such as:

- Asymmetric signal propagation (not actually possible)
- Transmit power variation
- Antenna diversity
- Transmit modulation quality variation
- Receive noise floor variation
- Receive quality variation
- Interference variation

Asymmetric signal propagation. This is actually a "non-cause". That is, asymmetric signal propagation is physically impossible according to the reciprocity theorem [58] which states that *if the role of the transmitter and the receiver are interchanged, the instantaneous signal transfer function between the two remains unchanged.* As asymmetric signal propagation has frequently been posited as an explanation for link asymmetry, it is listed here to discourage its consideration.

Antenna Diversity. Antenna diversity could result in different transmit and/or receive antennas being used on one or both ends of the link. In these situations, some degree of link asymmetry could arise. Nevertheless, asymmetry has been observed even in cases where no antenna diversity exists.

Transmit power variation. Clearly using asymmetric transmit power on a link can cause asymmetric packet delivery rates due to the disparity in received signal strength. As real wireless networks are typically composed of a heterogeneous mix of devices, real networks will likely have asymmetric links due to asymmetric transmit power. Link asymmetry has been observed, however, even when the same model card is used. The transmit power of 11 different Senao cards was measured using a spectrum analyzer in order to see how much transmit power variation was present. 23 individual measurements were made for each card. 0.5 dB was added to the measurements to account for pigtail loss (an estimate). Figure 7.23 shows the results and computed 95% confidence intervals.

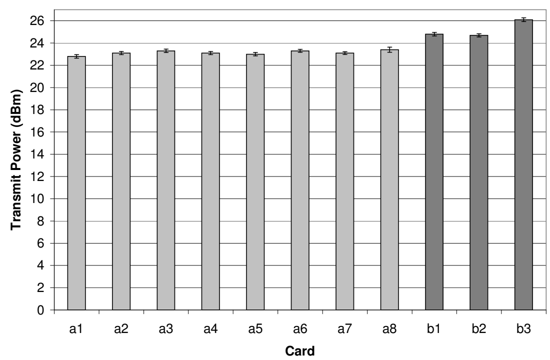

Figure 7.23: Senao Card Power

The cards fell into two distinct sets “a” and “b”. The cards in set a averaged quite close to 23 dBm of transmit power with very little variation. Set b, on the other hand, had significantly more power and variation. While these cards were marketed, sold, and labeled as identical they were purchased at different times, from different vendors, and contain MAC addresses that fall into two distinct ranges corresponding to sets “a” and “b”.

The data support the following conclusions. Even cards that appear outwardly to be identical, may actually be different and have different transmit power. Cards that are actually identical, may have very little transmit power variance, but they may also have significant variance.

As a result, transmit power variation seems to be one likely contributor to link asymmetry.

Transmit modulation quality variation. A large number of factors such as transmitter linearity may cause cards to vary in the fidelity of the transmitted signal that they produce. Similar cards should produce similar fidelity.

Receiver quality variation. Likewise, a receivers can also vary in quality such as linearity and signal acquisition. Again, similar cards should have similar receive fidelity.

Receiver noise floor variation. The noise floor of the receiver is determined largely by the performance of the low noise amplifier (LNA). LNAs are designed to amplify the weak signal received at the antenna into a larger signal that can be processed without introducing much noise into the signal. Anything that touches a signal, however, adds some degree of noise. The figure of merit for LNAs is their "noise figure", measured in dB, which quantifies how much noise they introduce into the signal.

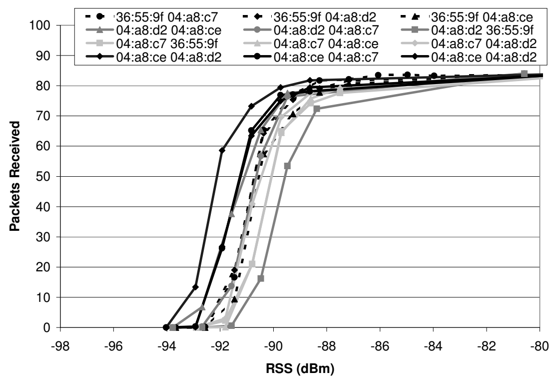

Figure 7.24: Packet Delivery Rate Variation

To quantify how much asymmetry might be from the combined effects of receiver noise floor variation, transmit quality variation and receive quality variation, the pairwise packet delivery rate was measured between all possible pairs of four wireless cards using 2 Mbps broadcast packets. In this case, coaxial cable and a variable attenuator were used to vary the transmit power between these nodes. This test corrected for transmit power variation in order to isolate the desired effects. The received signal strength was varied between -80 and -98 dBm.

Figure 7.24 shows the results. Approximately 3 dB of variation was observed over all of the links that were measured.

Interference variation. A final potential contributor to link asymmetry is interference level variation. Interference variation between two nodes is likely to contribute to asymmetry in a way that is highly site-specific and variable over time. Evaluating its impact requires doing a careful study of interference and its variability in time and space. An important distinction of interference with respect to the previous potential factors is that it is highly unlikely to be constant. Most sources of interference e.g. undesired 802.11 traffic, non-802.11 data traffic, cordless phones, microwaves, etc. are bursty on some time scale.

Several researchers have observed asymmetric links that have a fairly consistent constant bias. Thus, in at least some cases it seems unlikely that bursty interference is the cause of link asymmetry.

Summary. This section has discussed several potential causes of link asymmetry and has shown that several of these are contributing factors. While in some cases one factor such as transmit power asymmetry may be the dominant factor, in many cases, asymmetry may result from the additive effects of several causes. Importantly, this section has shown that link asymmetry can exist even when using homogeneous hardware and when external interference does not play a role. Thus, protocol designers should consider link asymmetry even when hardware is uniform. Symmetric signal propagation can, however, be assumed between two antennas since asymmetry is physically impossible in this case.

7.2 WLAN Performance Analysis

This section leverages the link behavior characterization discussed earlier to analyze WLAN uplink behavior. The aim of this analysis is to gain insight into the issue of hidden and exposed nodes. Why do WLANs seem to work reasonably well despite the fact that almost nobody uses RTS/CTS? Traditional analysis would indicate that network performance could grind to a halt if hidden nodes are present and RTS/CTS is not utilized.

To answer these questions this section analyzes the performance of an operational and heavily utilized wireless network in CMU's Tepper School of Business. This network consists of a 17 access point 802.11b network on channels 1, 6, and 11, and covers a single large campus building. A radio map of this building was constructed by sampling received signal strength throughout the building, and storing the physical location of each sample. For the sake of this analysis, each node was considered to have the same transmit power, which as discussed previously, is likely to be violated in a real network.

The likelihood of hidden terminals and exposed nodes was then analyzed as follows. First, in software, a random distribution of 400 nodes within this building was created taking into account the likelihood of a particular location's occupancy. I.e. nodes were much more likely to be located in lecture halls than offices. The actual observed access point locations and channel assignments were used. Each node picked a random recorded set of access point signal samples at it's location in the radio map. Each node was then associated with the access point having the strongest signal.

This analysis then looked at each node in the network and analyzed its pairwise interaction with all other nodes in the network as depicted in Figure 7.25. Each primary node considered is labeled "A" in Figure 7.25. The nodes with which A's interaction are considered are labeled "B". Considering A's interaction with its access point we have 6 possible cases. If A and B are associated with the same access point then they must be able to communicate with it and: a) they are in carrier sense range of each other or b) they are out of carrier sense range - "hidden" - from each other. If A and B are associated with different access points, then we have four cases: c) A and B are in carrier sense range of each other, but B does not interfere with

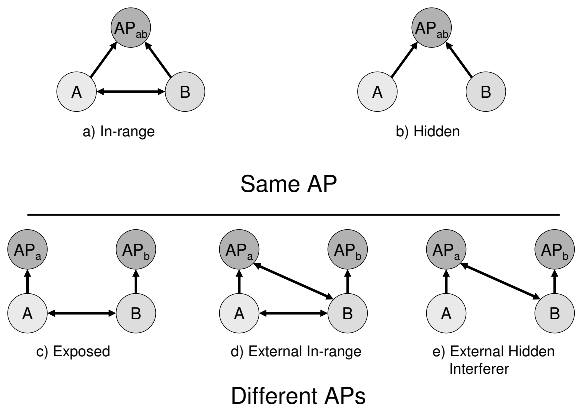

Figure 7.25: Infrastructure Topologies

A's transmissions to its access point. d) A and B are in carrier sense range and B's transmission **can** interfere with A's transmissions to its access point. e) A and B are out of carrier sense range from each other and B's transmissions can interfere with A's transmissions to its access point. Again, only A's interaction with its access point is considered. B's interactions with its access point are considered when in turn when it is "A". f) A and B are out of carrier sense range and B's transmissions do not interfere with A's transmissions to its access point. B does not affect A in this case; hence this case is not depicted in Figure 7.25.

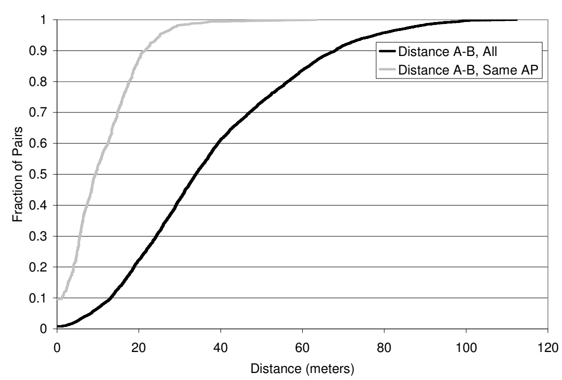

Figure 7.26: Distance CDF

In this analysis, path loss between nodes and access points is computed directly from radio map measurements. Between nodes, however, there are no direct measurements. To model path loss between these nodes a log distance path loss model [49] is used with a d_0 of 1.0 meter pl_{d0} of 40.0 dB and a path loss exponent n of 5.0. Figure 7.26 plots the CDF of pair distance for all pairs and also for pairs associated with the same access point. This CDF is for a single execution of this analysis (runs

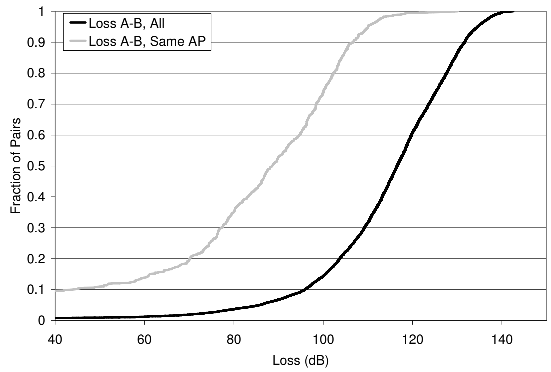

Figure 7.27: Loss CDF

produce very similar results, so the results of other runs are omitted). The corresponding CDF of path loss values using the node-node path loss model is shown in Figure 7.27.

Table 7.2 shows the frequency of the various types interaction found in a single run of this analysis (other runs were quite similar). Clearly, hidden nodes were very uncommon. The reason for this can be found by analyzing Figures 7.26 and 7.27. The wireless network in this building is fairly dense; thus, nodes associated with the same AP tend to be quite close to each other. To be out of range requires a loss of 115 dB which occurred for very few pairs associated with the same access point. Exposed pairs and external interferer pairs, however, were much more common.

Total Pairs	159600
Same AP Pairs	12230
Hidden Pairs	406
Exposed Pairs	11438
External Interferer Pairs	34374

Table 7.1: WLAN Performance Analysis Summary

The impact that hidden nodes might have on performance was analyzed next. Note that just because A and B are hidden does not mean that A's transmissions with B will always fail. For each hidden pair, the data obtained in Section 7.1.4 was used to estimate what the probability that A's transmissions would be received by its access point if it interfered with a transmission from B. The path loss measured in the radio map was used to compute the RSS at A's access point for both A and B and then the capture probability was computed from the table in Section 7.1.4. RSS values that were not in this table were computed via extrapolation.

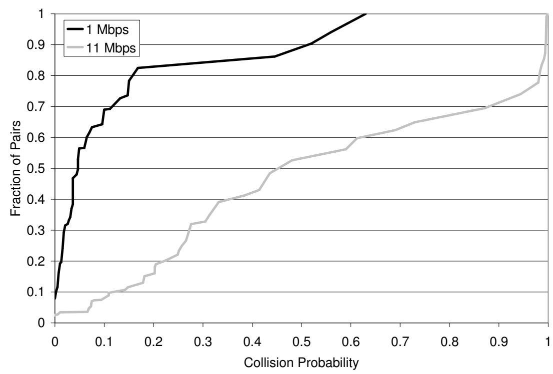

Figure 7.28: Hidden Node Collision Probability, 1 Mbps vs. 11 Mbps

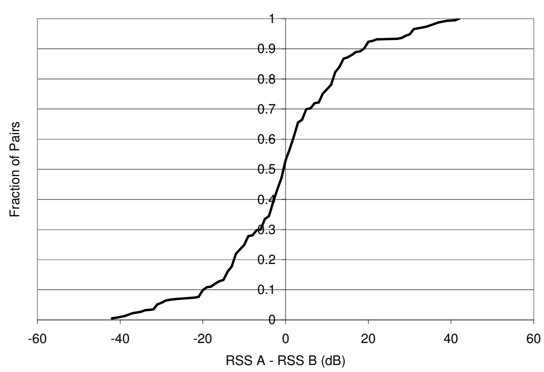

Figure 7.29: Hidden Node RSS Difference CDF

Figure 7.29 shows the result for both 1 Mbps transmissions and for 11 Mbps transmissions. At 1 Mbps, very few of the hidden pairs are likely to have high collision probabilities. For these cases, A will receive more throughput to its access point than B will. At 11 Mbps, however, there is a fair chance for collision. In practice, most nodes in this network would likely communicate at 11 Mbps, so hidden nodes could interfere with each other. Nevertheless, the scarcity of hidden nodes indicates that they are not likely to present much of a problem.

A similar analysis for external interferer pairs was then conducted; again the capture data measured earlier was used. The results are shown in Figure 7.30 and Figure 7.31. In this case we have three possible outcomes a collision, A's packet is captured - the desired outcome, B's packet is captured - thus causing A's packet to fail. At 1 Mbps the odds of a A's packet not being captured are extremely small. While the odds of A's packet being received at 11 Mbps are somewhat worse, they are still very good. Thus, even though there are a large number of external interferer

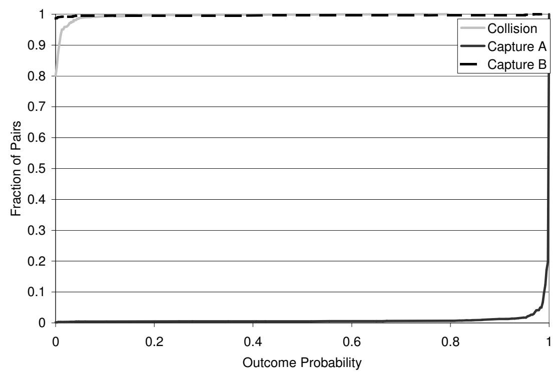

Figure 7.30: External Interferer CDFs, 1 Mbps

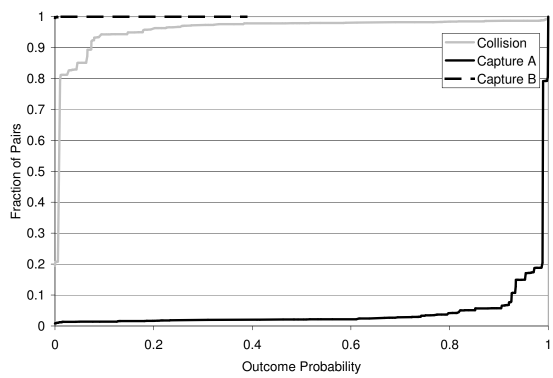

Figure 7.31: External Interferer CDFs, 11 Mbps

pairs, the impact of this phenomenon on performance is minimal.

To understand why external interference has so little impact, consider Figure 7.32 which shows a CDF of difference in received signal strength at A's AP from A and B. For the vast majority of external interferer pairs, A enjoys a significant advantage in signal strength.

This analysis leads to the conclusion that the most serious inefficiency that is likely to plague this network is exposed nodes. Consider Figure 7.32 which shows a CDF of the difference in received signal strength at A's AP from A and B. For the vast majority of pairs, A again enjoys a significant advantage. Hence, A need not defer when B is transmitting.

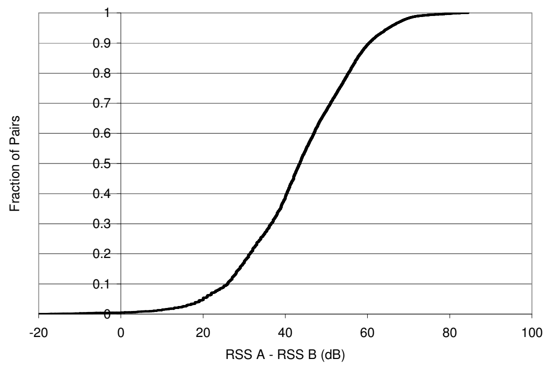

Figure 7.32: External Interferer RSS Difference CDF

7.3 Device Characterization

This section now considers experiments that measure particular aspects of wireless NIC behavior such as perceived received signal strength versus actual received signal strength. Many of these tests were conducted in conjunction with the Roofnet group as described earlier. In addition to providing the control necessary for these tests, the emulator allows these tests to be automated which greatly reduces execution time while eliminating the error associated with manually conducting similar experiments.

NIC Signal Measurement Characterization

Many researchers have proposed techniques that rely on signal strength and/or noise floor measurements provided by the card. Two common examples are signal strength based device location [3] and SNR-based rate selection [24]. The success of these proposed techniques hinges on the accuracy of NIC signal measurement; very little information, however, has been published regarding the accuracy of these measurements in actual hardware.

To investigate the accuracy of signal measurements made by current 802.11b cards, the signal measurement behavior of five wireless cards was tested. Each card was the exact same model: an Engenius NL-2511CD Plus Ext2 card. Using the emulator to connect a single transmitter-receiver pair enabled precise control of the received signal strength (RSS) at each card (the transmitter was constant while alternately measuring each receiver). For each signal strength between -70 dBm and -100 dBm at 2 dB intervals 500 packets of 1500 bytes each were sent at 1 Mbps. The average signal strength (RSSI) and noise measured by each card were then computed (along with 95 % confidence intervals).

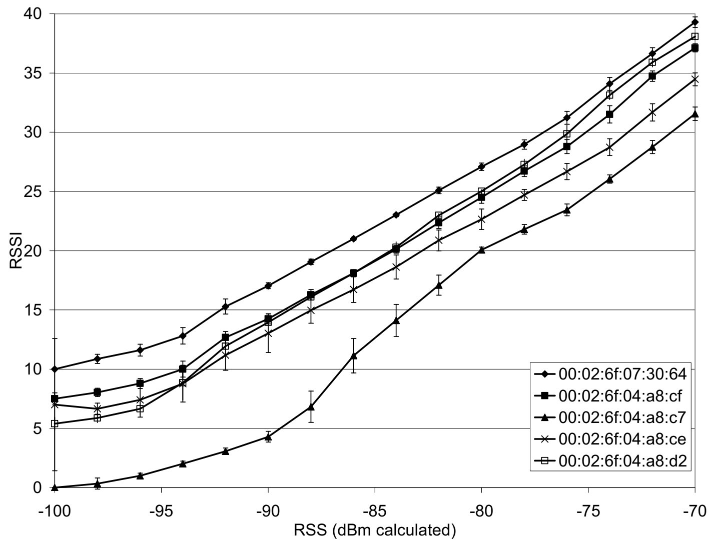

Figure 7.33: Per-card RSSI Variation

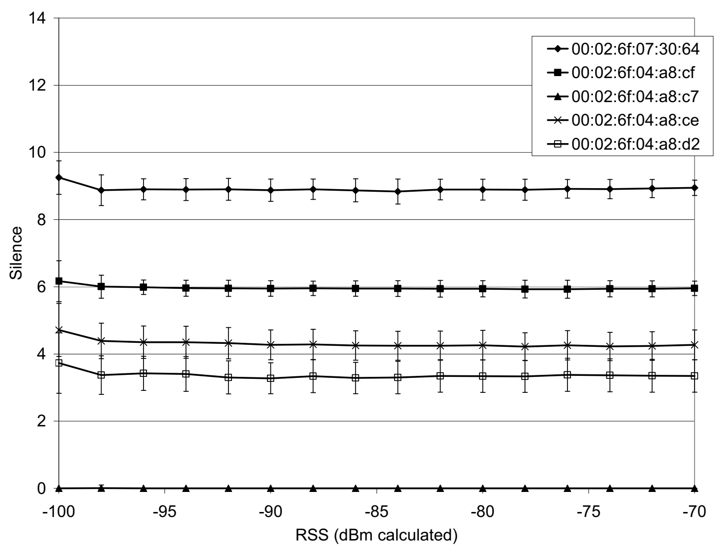

Figure 7.34: Per-card Noise Variation

As shown in Figure 7.33 there is approximately 10 dB of variation in the measurements even for the exact same model of card. This is clearly inadequate for many purposes. For most cards, however, this variation seems to be caused by a constant bias. This implies that each card's measurement behavior, RSSI, for a given RSS can

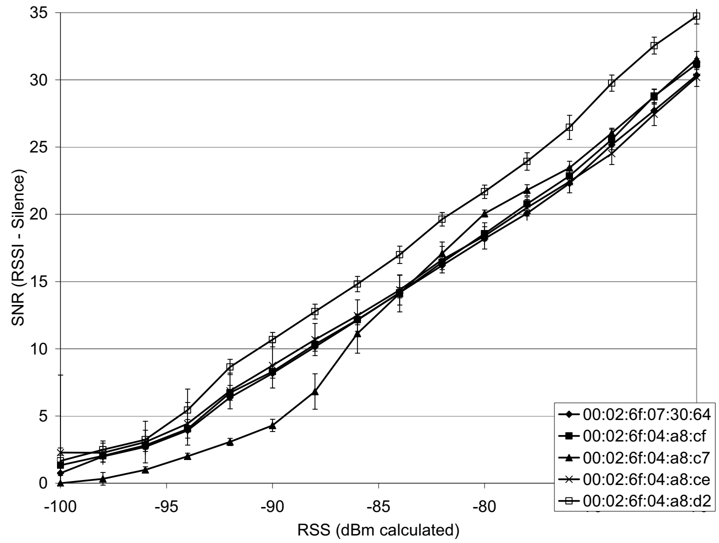

Figure 7.35: Per-card RSS Variation after Correction

be defined as: $RSSI(RSS) = RSS + E_c + E(RSS)$. Where RSSI is the measured signal strength, RSS is the actual signal strength, E_c is a constant (per-card) error term, and E(RSS) is each card's variation of from the base E_c for a particular RSS. Ideally, each card would have a lookup table that would give the E_c as well as E(RSS) for each RSS. Lacking such a table, however, we can leverage the fact that most of the error is contained in E_c to correct RSSI.

One very simple method of obtaining a good estimate of E_c is to min-filter the noise measurements (the filtering eliminates spurious noise measurements). As shown, in Figure 7.34, the noise measurements over the same set of tests shows very similar variation. That is, each card's variation in RSSI closely matches it's variation in measured noise. Figure 7.35 shows the variation in RSS when using this technique. With the exception of one card, this lowers the variation to approximately 4 dB. This is a greatly reduced variation, but may not be low enough for some purposes (e.g. signal strength based location). Complete card characterization of the relationship between RSSI and RSS is possible, but may not be worth the per-card testing required.

Multipath Performance

The potential of applications to estimate the amount of multipath present using information obtained from the NIC's equalizer was then analyzed. On the Engenius NL-2511CD Plus Ext2 cards (and all other cards based on the same chipset), a register - "MPMetric" - is available to estimate the amount of multipath interference present

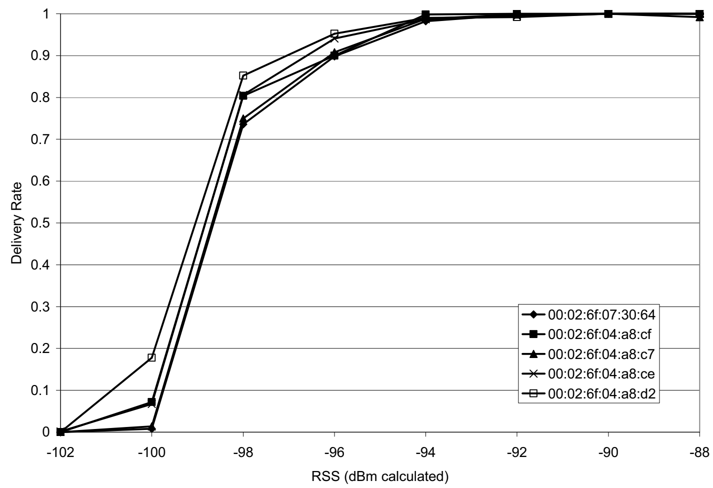

Figure 7.36: Per-card Delivery Rate Variation

during reception.

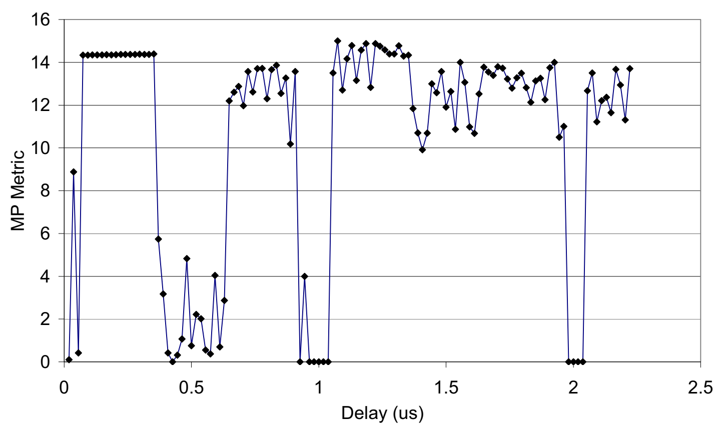

Figure 7.37: Two-ray MP Metric vs. Delay

As the documentation on the Prism 2.5 MPMetric register is scant, the emulator's ability to measure the behavior of this register is critical in understanding its performance. Figure 7.37 shows MPMetric as a function of delay spread for two equal-strength rays. These measurements were obtained from the two-ray test described earlier, and use the five Engenius cards used previously. From this test, it can be inferred that if significant multipath reception is present, MPMetric is likely to be high. MPMetric was the measured in the presence of no multipath as shown in Figure 7.38. From this test it is seen that the MPMetric register may also go high

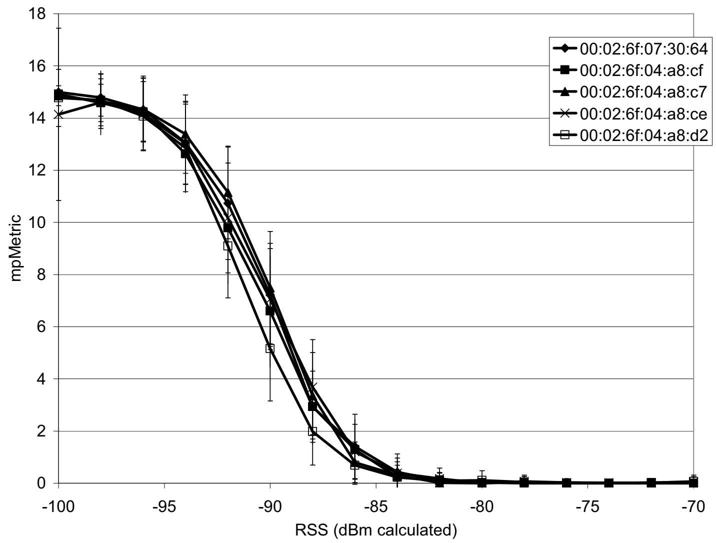

Figure 7.38: One-ray MP Metric vs. RSS

whenever the signal conditions are marginal irrespective of multipath. This suggests that a high MPMetric reading is a likely indicator of multipath when the received signal strength is high, but it is not a useful indicator of multipath when the received signal strength is weak.

7.4 Case Studies

This section presents three case studies that illustrate how users can conduct various types of link-level tests. These tests were chosen to illustrate some of the range of capabilities provided by the emulator. These experiments were conducted using the Prototype Version of the emulator [27].

7.4.1 802.11b Rate Selection

The first case study looks at the problem of 802.11b transmit rate selection. This study illustrates how the emulator can be used to fairly compare different rate selection algorithms.

When selecting a transmit rate, a fundamental tradeoff that wireless protocols must make is throughput vs. range: higher transmit rates increase throughput but at the cost of range and robustness to interference. Rather than selecting a fixed point in this tradeoff, wireless protocols such as 802.11b support multiple transmit rates. This allows wireless NICs to potentially select the best transmit rate in a given environment and at a given moment.

Selecting the best rate, however, is a difficult problem and several schemes have been proposed. The emulator allows a controlled comparison of the performance of these schemes on real hardware. For illustrative purposes three schemes are examined: ARF - auto rate fallback [63, 7], SNR signal-to-noise ratio based scheme (with the same goal as [24], but very different mechanisms), and ERF - Estimated Rate Fallback. Each of these approaches is described below.

The transmission rate selection implementations are based on the HostAP mode Prism driver for Linux. Extensive alterations were made in order to take fine-grained control of rate selection out of the firmware, and put it into the driver. These alterations enable per-packet control over transmit rate, and effectively disable firmware rate control.

ARF Implementation. Auto rate fallback attempts to select the best transmit rate via in-band probing using 802.11's ACK mechanism. ARF assumes that a failed transmission indicates a transmit rate that is too high. A successful transmission is assumed to indicate the current transmit rate is good, and that a higher rate might possibly be useful.

The ARF implementation works as follows. If a given number of consecutive packets are sent, then increment to the next highest transmission rate. If a given consecutive number of packets are dropped then decrement the rate. If no traffic has been sent for a given amount of time, then use the highest possible transmission rate for the next transmission. In this ARF implementation, the increment threshold is set at 6, the decrement threshold at 3, and the timeout value at 10 seconds. (The Prism 2.5 firmware based ARF algorithm uses a decrement threshold of 3 and a timeout of 10 seconds, but is somewhat different than the algorithm used here since retries are implemented entirely in firmware.)

SNR Implementation. SNR-based approaches attempt to eliminate the overhead of probing for the correct transmission rate by selecting the optimal transmission rate for a given SNR. These schemes typically ignore multipath interference, and assume that card RSSI/noise floor measurements are completely characterized on a per-card basis.

SNR-based rate selection algorithms are faced with the fundamental problem that the information they need to make the rate selection decision is measured at the receiver. The SNR-based implementation leverages receiver based reception information, like RBAR [24], but eliminates the per-packet overhead and works with standard 802.11. The key insight that the SNR-based algorithm leverages is the fact that instantaneous path loss between two given points is symmetric in both the sending and receiving directions [1]. Hence, it's possible to estimate SNR at the receiver by observing traffic in the reverse direction. Further details of this scheme are omitted as they are beyond the scope of this work.

Estimated Rate Fallback. While signal based transmission rate selection has the benefit of quickly setting the transmission rate, this technique may be inadequate in some situations. Auto rate fallback, on the other hand, has the advantage of implicitly taking all relevant channel factors into consideration, but may probe more than necessary. A simple hybrid algorithm was developed that uses both SNR and ARF in conjunction the on-card measurements of multipath. This scheme is called Estimated Rate Fallback (ERF).

The basic idea of ERF is to run the ARF and SNR-based schemes in parallel, and then to select the appropriate estimate. This is done by using the SNR-based estimate unless one of the following is true: multipath is detected, or the SNR estimate is near a decision threshold (2 dB in this implementation). This allows ERF avoid the multipath weakness of the SNR-based approach while reducing the need for card characterization.

Rate Selection Algorithm Comparison The performance of the previously discussed transmission rate selection algorithms is now evaluated using three emulated signal propagation environments. In all cases, the same test is used to measure performance.

Under lightly loaded traffic conditions, optimal rate selection is not strictly necessary since a lower transmission rate can simply be used. Rate selection becomes critically important, however, when the wireless network is running at capacity. For two of these tests, this fully loaded condition for a single transmit-receive pair is examined. For the third test, a lightly loaded situation is examined.

To measure performance of a single transmitter under full load, as many unicast UDP 1400-byte packets as possible are sent from the transmitting node to the receiving node under the given signal environment. For the lightly loaded scenario, 100 packets are sent over 10 seconds and the number successfully received is measured.

These tests highlight the emulator's ability to enable controlled comparison of rate selection mechanisms with a high degree of repeatability. For each experiment this section will briefly discuss how the experiment would have fared using an alternate approach.

Fixed RSS. The first test used to evaluate the various rate selection mechanisms was to measure performance when the received strength was constant and the source sent as much traffic as possible as described above. Figure 7.39 shows the results. As

[1] This discussion assumes a single receive and transmit antenna. This approach can be modified to support the general case.

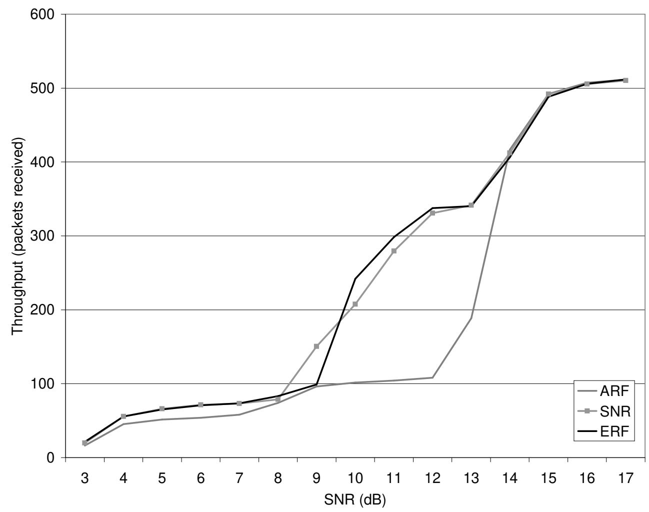

Figure 7.39: Rate Selection for Fixed RSS

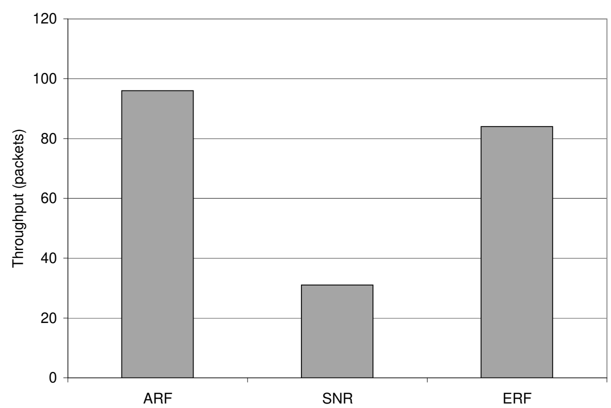

Figure 7.40: Rate Selection for Under Multipath

expected, SNR performs well. ARF, on the other hand, performs poorly at intermediate signal levels where it is periodically probing for a higher bandwidth that will never be useful. ERF, is able to match SNR performance quite closely.

Obtaining this result using real-world experimentation would be possible, but tedious since positioning nodes to obtain a particular fixed RSS is difficult. Simulation might be used, but would only yield useful results if the hardware were modeled accurately.

Multipath. Next, rate selection performance in a multipath environment was measured by commanding the emulator to introduce a delayed copy of the primary signal from the sender to the receiver (ideally this would be both directions) with a fixed delay of 1 symbol period. With the RSS of the primary ray set to -77 dBm, the delayed ray strength is set to -84 dBm. As shown in Figure 7.40, ERF and ARF perform much better than SNR since SNR sends at 11 Mbps. This also masks the fact that SNR uses multiple retries to even attain this throughput. This test demonstrates that multipath can cause the SNR-based scheme to fail, although it is unclear whether this situation is common enough to worry about in many environments. Nevertheless, ERF is able to use hardware information to eliminate even this situation.

Eliciting this result using real-world experimentation would essentially require a highly controlled large-scale RF test range. Using simulation would simply not be feasible.

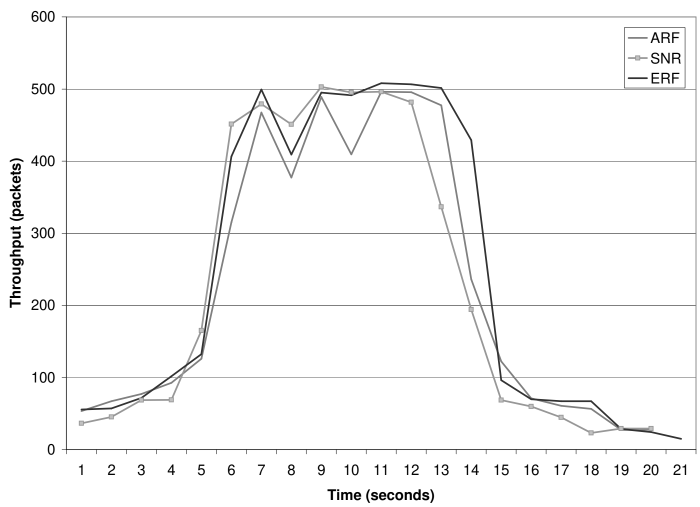

Figure 7.41: Rate Selection for Driveby Emulation

Fast Fading. Performance in a fast fading environment was then tested by measuring throughput during a replay of a "drive by" scenario similar to that shown in Figure 7.41. (In this experiment, the controller is simply emulating the fast fading caused by multipath, and is not actually emulating multiple signal copies. Hence, the multipath differences in the various algorithms are not demonstrated by this experiment.) Figure 7.41 shows that in this scenario, all algorithms perform similarly though ARF and ERF generally outperform SNR when the signal is marginal, while SNR and ERF generally outperform ARF when the signal is strong.

This experiment demonstrates the benefits of being able to replay the exact same signal trace. Comparing these rate selection algorithms in a real drive-by experiment would be difficult since even slight variations in mobility would cause channel inconsistency across experiments. Hence, it would be difficult to separate the effects on performance due to the different algorithms from the effects due to RF channel variation.

In practice, experiments that include mobility are also very cumbersome to execute in the real-world especially as the number of mobile nodes increases. A simulated test would result in a much coarser grained use of the signal fading trace and fail to simulate the effects of rapid fading due to vehicle mobility. Hence, confidence in the accuracy of such a simulated test would be greatly reduced.

7.4.2 Bluetooth Interference

One well known problem that can afflict wireless networks in a license free band is interference from competing sources. To illustrate the emulator's ability to investigate interference from arbitrary sources a simple experiment was conducted involving two 802.11b nodes communicating in the face of interference from a Bluetooth source. As shown in Figure 7.42, each node was positioned 50 meters from the other two nodes.

This experiment was conducted using one or more of three RF Nodes connected to the emulator prototype: "Orchid", "Hermes", and an interferer ("Nice" or a Bluetooth source). For experiments conducted in an emulated physical environment (i.e. where manual control of channel parameters is not required), a log-based path loss model derived from the local environment was used. For each of the experiments discussed, obtaining realistic results using traditional methods would be difficult or inaccurate.

Figure 7.43 shows the results of communication between Hermes and Orchid for four scenarios (each value is an average of 25 trials with 95% confidence intervals shown), two of which - the "Yagi" cases - will be discussed in the next section.

In the "Isotropic, No Interference" test, Hermes and Orchid communicate with omnidirectional antennas with no interference (using a TCP benchmark with traffic from Orchid to Hermes). Communication is only around 1.25 Mbps due to the distance between the nodes.

In the "Isotropic, Interference" test, Hermes and Orchid communicate as before, but the Bluetooth source is configured to broadcast a constant 15 dBm signal with Bluetooth modulation. TCP communication between Orchid and Hermes is not possible in this case.

7.4.3 Flexible Antenna and Multi-element Air Interface Support

Complete control over signal propagation also allows the emulator to emulate arbitrary types of antennas. To illustrate this, the ability of directional antennas to

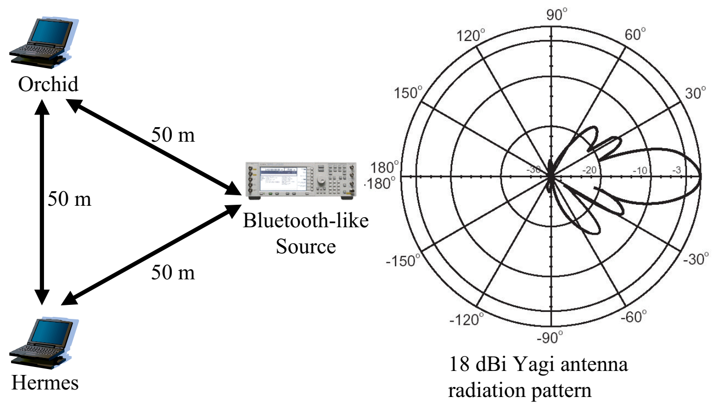

Figure 7.42: Directional Antenna Topology

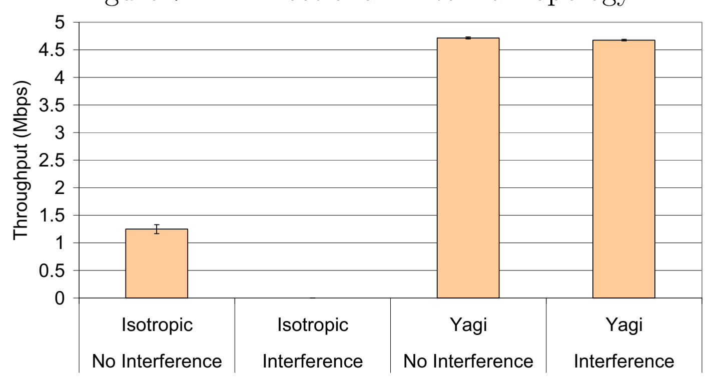

Figure 7.43: Directional Antenna Results

improve range and spatial reuse by minimizing the effects of interfering Bluetooth traffic (discussed in 7.4.2) was analyzed. Orinoco cards were used for these tests.

The "Yagi" tests repeat the "Isotropic" tests discussed previously, but with 18 dBi Yagi antennas [55] attached to Orchid and Hermes. These antennas are aimed directly at each other. Figure 7.42 shows the radiation pattern for these antennas. Note that for Orchid and Hermes, the Bluetooth source lies along a side lobe with approximately 22 dB and 18 dB respectively less gain than the primary lobe. As shown in Figure 7.43 these directional antennas successfully increase the communication rate and also mitigate the effects of external interference.

7.5 Summary

A clear understanding of wireless device performance is critical in understanding how wireless networks function and how they might be improved. Despite this need, little data exists for modern wireless networks on important performance issues such as packet capture, collision, off-channel reception and interference and how these interplay with issues such as hidden and exposed nodes.

This section has presented a large controlled study of 802.11 link and device behavior aimed at replacing convention and assumption with measured device behavior. The link-level performance data in conjunction with a wireless LAN radio map has shown that hidden nodes are uncommon in dense wireless networks, and that true collisions are unlikely for low transmission rates. Moreover, this section has shown that interference from nodes that are hidden but external to a given wireless cell have little impact on wireless network performance due to packet capture. These findings hint that a capture-aware MAC could reduce the number of exposed nodes by allowing simultaneous transmissions in cases where it poses no harm.

This section has also measured off-channel interference and reception behavior. These tests of revealed that off-channel interference rejection can perform very well, and confirmed the usefulness of leveraging this feature to allow more than three channels to be used in the 802.11b networks. These results show, however, that off-channel reception behavior is quite poor and this feature should be used with great caution.

In addition, fine-grained measurements of device behavior have enabled an accurate understanding of metrics such as RSSI. In particular, imperfections in RSSI measurements were discussed, and methods of mitigating these imperfections were illustrated. The data presented in this section should enable a better understanding of wireless device performance, and act as a catalyst for improving wireless simulator fidelity.

This chapter has demonstrated that physical layer wireless network emulation can provide detailed link and device level insights that can be difficult to obtain using other techniques.

Chapter 8

Network Experiments

Chapter 7 illustrated how physical layer wireless network emulation is a powerful technique that can provide important insights into link-level wireless network behavior. This chapter discusses how physical layer wireless network emulation provides similar benefits at the wireless network layer.

As an illustrative example, this section considers the important problem of wireless LAN access point selection. This problem is analyzed in an enterprise setting for two reasons. First, enterprise wireless LANs are an important market segment, and handoff-sensitive applications such as voice over IP (VoIP) are of increasing importance within this segment. Secondly, enterprise wireless LANs have unique features such as trust between access points and a wired backhaul infrastructure that simplify this discussion and provide an opportunity to leverage the wired infrastructure to improve wireless performance.

The use of physical layer emulation can be applied to a wide range of problems; this problem was chosen to demonstrate several of the key benefits achievable via this technique such as: a controlled experimental environment, fair comparison across experiments enabled by signal environment repeatability, improved "exploratory" experimentation, and a shortened development cycle.

8.1 Access Point Selection

Access point selection is a fundamental issue that has a large impact on wireless LAN performance. The 802.11 standard places the burden of solving this problem on the clients. Access points provide mechanisms for client association and disassociation; clients are left to discover access points and implement policies that provide good performance.

Achieving good performance is challenging for several reasons:

- The limited capabilities of wireless LAN hardware. 802.11 clients are half-duplex and operate on a single channel. Thus, data communication and looking for new access points are largely mutually exclusive operations.
- Development of wireless network software is cumbersome. The development cycle of wireless software requires programmers to move devices in physical space to explore the effects of parameters and algorithms.
- Testing of wireless network software is difficult. Likewise, the testing of and debugging of software require physically moving devices. As a result, exercising all code paths of wireless software can be difficult to achieve.

The development cycle of wireless drivers and firmware is complicated by the need to run wireless experiments to test, debug, and optimize the algorithms and software under development. These experiments are typically conducted in a deployed testbed which can even be co-located with a production network. The resulting lack of experimental control and repeatability can be serious drawbacks that hinder the ability to test software, discover problems, and determine the causes of the problems observed.

Physical layer wireless emulation provides researchers and developers with several important benefits as discussed in previous chapters.

8.2 802.11 Background

Before discussing how the emulator aids the development of improved access point selection, this section reviews the access point selection mechanisms provided by 802.11.

Access Point Discovery. The first task that a disconnected client must perform is access point discovery. 802.11 provides two mechanisms for doing this: beacons and probes.

802.11 access points periodically send beacons announcing their presence and capabilities. Clients can leverage these beacons to passively discover access points by simply listening a certain amount of time for incoming beacon frames. The primary disadvantage of this approach is the latency involved. Beacons are typically sent out at 100 ms intervals, thus a station must listen to each channel for at least 100 ms to ensure that all beacons on that channel have been received (in practice a little more time is needed since beacon frame transmission can be delayed from the nominal transmission time.) Thus scanning eleven 802.11b/g channels requires over one second, scanning eight 802.11a channels requires nearly one second, and scanning 19 802.11a/b/g channels requires approximately two seconds.

To reduce the latency required to discover stations, 802.11 provides an active access point discovery mechanism using probe frames. To actively discover access points on a given channel, a client sends out a probe frame and then waits for a certain amount of time for responses. Access points that hear the probe frame send probe response frames containing the same capability information that is contained in beacon frames. While significantly faster than passive scanning, this process can still take several hundred milliseconds; moreover, recent trends in access point implementation point to a possible increase in this latency as more access point functionality is moved out of firmware and into drivers and centralized access point controllers.

Selection Policy. Access point selection policy is not specified in the 802.11 standard. Given the paucity of information provided by access points, however, the only realistic interoperable policy is to select the access point with the strongest signal. (Proprietary extensions from some vendors enable the use of other load-sensitive policies.)

Access Point Authentication Association. After selecting a suitable access point, a client joins the access point by first authenticating and then associating. Association is accomplished by the client sending an association request to an access point which sends back an association response indicating the results of the request. Similarly, a client leaves an access point by first sending a disassociation request frame.

In an enterprise setting, initial authentication will incur a high latency due to the need to contact a central authentication server. Authentication with subsequent access points can then be performed very quickly using a variety of fast authentication approaches [10]. For simplicity, this discussion will consider the case of an open network where authentication is not required. Networks with fast authentication should see similar performance.

Roaming. 802.11 roaming between access points is supported using the mechanisms presented earlier for access point discovery and association. When a client desires to search for new access points it uses passive or active scanning to discover new access points. It then disassociates with the old access point and associates with the new access point.

8.3 Measuring Access Point Selection Performance

Controlled, accurate, and detailed measurement of access point selection performance is fundamental to understanding the behavior of access point selection and developing improved policies, protocols, and implementations. Physical layer emulation provides an ideal environment in which to conduct this measurement, and provides insights into behavior that can be extremely difficult to obtain using other techniques.

This section discusses a series of experiments that quantify access point selection performance. For each test, this section will discuss how it is implemented in the emulator. Results of running these tests then illustrate the insight that controlled experiments provide.

8.3.1 Two Access Point Tests.

This discussion first considers tests that examine a single roam operation between two access points. This is an ideal test from which to begin access point selection analysis since direct control over the signal propagation environment allows the behavior of a "perfect" client to be fixed a priori. The operation of an actual wireless device's access point selection behavior can then be compared with the theoretical "optimal" behavior. Optimality can be debated since the client lacks a true understanding of the signal environment that it resides in. "Optimal" behavior - as discussed here - means optimal given a complete understanding.

This discussion considers two different two access point tests: "abrupt" and "gradual". Both tests begin with a client associated with a first access point A and ideally end with the client associated with a second access point B; in some cases the client may fail to associate with the second access point. Thus, these tests measure performance for a fixed scenario namely a roam from access point A to B.

Abrupt. In the two access point abrupt roam test, the emulator signal environment is constructed as shown in Figure 8.1. The client initially has a superb connection with access point A, and no connectivity with the second access point B. At a certain instant in time, t_{move}, the connections are reversed so that there is no connectivity with A and excellent connectivity with B. This experiment measures how well the client copes with this instantaneous change. Note that signal fading is not utilized in this test.

This experiment - and the rest of the experiments in this section - use the emulator's programmatic control interface to define the experiment. This allows tight control and coordination of the movement of the clients in the emulated signal environment with the measurement of access point selection behavior.

Complete measurement of access point behavior requires instrumentation of the client driver and possibly firmware. This test adopts a less detailed approach in the interest of portability between drivers. At 250 ms intervals, iwconfig is executed on the client to determine which access point it is associated with. Using this approach, the following metrics are extracted:

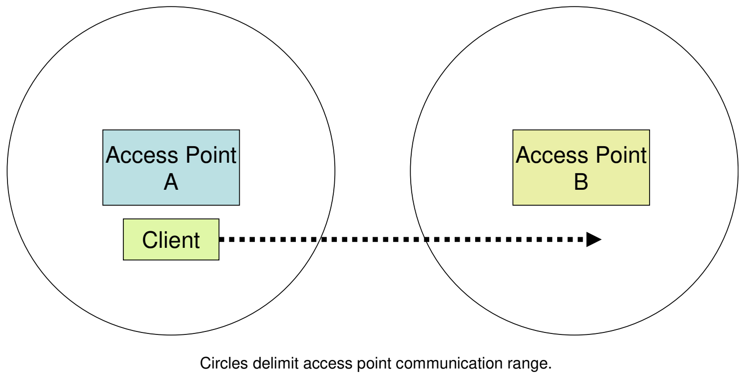

Figure 8.1: Abrupt Roam Test Configuration

- $t_{reaction}$ the amount of time after t_{move} required until the client either associates with access point B, or disassociates with A. Intuitively, this is the amount of time until the client reacts to the new signal environment.

- $t_{completion}$ the amount of time after t_{move} required until the client is associated with access point B and no further associations or disassociations are observed.

- $t_{disassociation}$ the amount of time in the roam process during which the client was disassociated from all access points.

- $n_{associations}$ the number of access point associations. E.g. a client that disassociated with A, associated with B, disassociated with B, then associated with A would yield $n_{associations} = 2$.

The abrupt test was used to analyze the roaming performance of a driver that was very popular at the time of this work: the Madwifi-NG driver. The version used was 0.9.1 which was current when this test was conducted (this driver will be referred to as Madwifi 0.9.1 hereafter.) The wireless cards used were Senao NL-5354MP ARIES2 cards which are based on the Atheros 5212.

Table 8.1 shows the two access point abrupt roaming performance of the Madwifi 0.9.1 driver. The results shown are drawn from a series of 20 repetitions of the abrupt roam test. This test clearly reveals two serious shortcomings of this driver's access point selection behavior:

	$t_{reaction}$	$t_{completion}$	$t_{disassociation}$	$n_{associations}$
Mean	7.32 s	15.24 s	7.92 s	1.0
Median	6.36 s	16.13 s	7.21 s	1.0
Min.	0.75 s	7.88 s	6.19 s	1.0
Max.	21.28 s	38.52 s	17.24 s	1.0

Table 8.1: Abrupt roam performance for Madwifi 0.9.1

- The driver is extremely slow to react to the new signal environment. It took over 7 seconds on average to begin searching for a new access point.
- Once the driver has recognized the need to find a new access point, it is quite slow in doing so. It took an average of nearly 8 seconds to find and associate with the new access point.

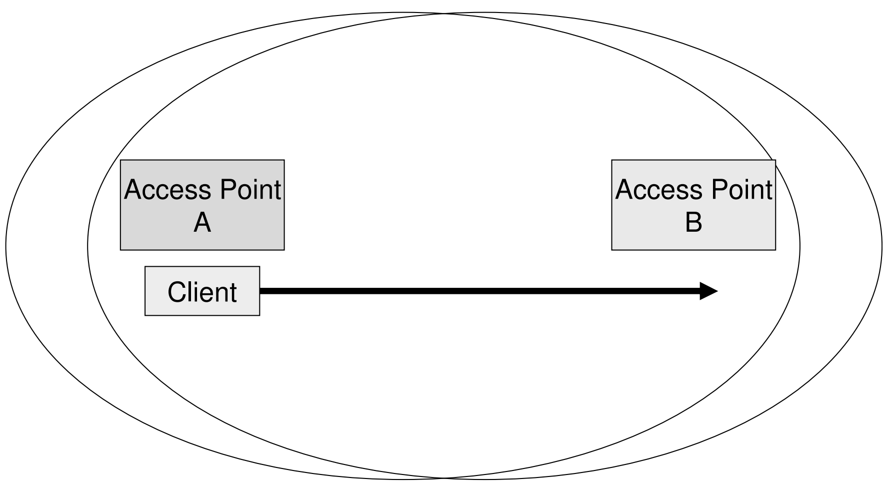

Figure 8.2: Gradual Roam Test Configuration

Gradual. The abrupt roam test just discussed is quite useful in that the "optimal" client behavior is known: once the signal environment has changed, the client should change access points as soon as possible. Real signal environments, however, are not discrete in nature. Thus, while the abrupt test yields insight into driver performance, it is not representative of a typical roam from one access point to another.

To observe behavior under a more typical roam scenario - but one in which the "optimal" outcome is still known - a two access point gradual roam test is used as depicted in Figure 8.2. This test is similar to the abrupt test, except that the change in signal conditions is gradual, and the client is never completely out of range of either access point as shown by the ovals which depict access point range in the figure.

Specifically at a given instant in time, the client begins moving from access point A to access point B over an interval of 20 seconds. It then is stationary at B for 20 more seconds before the test is repeated in the opposite direction. This process was repeated 20 times. Again, fading was not used in this test.

Under this test, the Madwifi 0.9.1 driver failed to roam despite very poor connectivity when near access point B. For the entire test Madwifi 0.9.1 remained associated with access point A.

This test clearly shows that Madwifi 0.9.1's slow reaction to a changing signal environment performs very poorly in a mobile scenario.

8.4 Improving and Tuning Madwifi 0.9.1

The exceptionally poor performance of Madwifi 0.9.1 access point selection is partly caused by the rudimentary support provided by 802.11 for access point selection. The high overhead of invoking the active or passive scanning mechanisms makes it undesirable to do so. The difficulty of implementing, testing, and tuning a high-performance access point selection algorithm led the authors of the Madwifi driver to implement a simplistic placeholder access point policy until a more sophisticated algorithm could be developed. The algorithm that they implemented simply selects the "best" access point and sticks with it. Roaming only occurs when the initial connection is broken. Moreover, the implemented scanning algorithm is very slow, and uses poorly tuned parameters.

While poor performance will occur in real networks, the causes of poor performance can be difficult to isolate. When analyzing access point selection in the emulator, in contrast, many of the causes become obvious.

After observing the poor performance of Madwifi 0.9.1, an investigation was initiated into the causes of this poor performance. The emulator's interactive environment provided an ideal environment in which to conduct this investigation. In this case, the signal environment from the abrupt roam case was used, but the mobility script was turned off. After turning on the Madwifi scan and roam debug logging, roams were initiated interactively by simply moving the client under inspection from near one access point to the other in the GUI. The time at which this occurred was recorded, as was the time at which disassociation with the old access point occurred, and the time that reassociation with the new access point completed.

Conducting this procedure several times quickly revealed a variety of mechanisms that were causing very poor roaming. For instance:

- Scanning was failure driven; that is, the connection with the current access point had to break before scanning was initiated. Proactive scanning code was disabled.

- Low-pass filtering of signal results caused very slow updates. Thus, several scans were required before an accurate view of the signal environment was obtained.

- Caching of scan entries caused invalid results to be used as if they were fresh.

- Multiple attempts were made to associate with access points even if the only information about that access point was not fresh.

Thus, Madwifi 0.9.1 was functional for a scenario with little to no mobility; even very modest amounts of mobility would break network connectivity.

An effort was then undertaken to improve Madwifi's performance. The above problems were all found to be easily amenable to improvement through tuning and small additions to the Madwifi 0.9.1 code as discussed further below. The emulator provided two important benefits during this effort. First, it reduced development cycle time. Traditional development would have required testing by running around with laptops in a real wireless network which would slow development time considerably. The emulator enabled testing simply by moving icons in the interactive GUI. Secondly, the emulator's complete control over the signal propagation environment allowed quick tuning of parameters. The effects of various values such as proactive scan interval and scan cache behavior were easily observable in the emulator without any degree of uncertainty in what the signal propagation environment looked like when tests were conducted.

The abrupt roam and gradual roam tests were then repeated for the enhanced version of Madwifi 0.9.1 - called Madwifi 0.9.1-E. Madwifi 0.9.1-E contains a small number of modifications to address the major shortcomings discussed in the list above. The results are shown in Table 8.2 and Table 8.3. Again, these results are from 20 executions of each test.

In the abrupt roam case, reaction time and completion time have both dropped significantly.

In the gradual roam case slightly different definitions for $t_{completion}$ are utilized - labeled $t_{completionA}$ and $t_{completionB}$ - since the client may switch between access points multiple times during the test. They are defined as follows:

- $t_{completionA}$ the amount of time after t_{move} required until the client is first associated with access point B.

	$t_{reaction}$	$t_{completion}$	$t_{disassociation}$	$n_{associations}$
Mean	0.98 s	4.04 s	3.15 s	1.2
Median	0.94 s	4.00 s	3.08 s	1.0
Min.	0.39 s	3.59 s	2.59 s	1.0
Max.	1.55 s	4.53 s	4.28 s	2.0

Table 8.2: Abrupt roam performance for Madwifi 0.9.1-E

	$t_{reaction}$	$t_{completionA}$	$t_{completionB}$	$t_{disassociation}$	$n_{associations}$
Mean	9.96 s	11.33 s	16.14 s	0.95 s	2.1
Median	10.31 s	11.28 s	15.44 s	0.52 s	2.0
Min.	3.22 s	8.54 s	8.54 s	0.00 s	1.0
Max.	17.56 s	17.56 s	34.52 s	3.47 s	5.0

Table 8.3: Gradual roam performance for Madwifi 0.9.1-E

- $t_{completionB}$ the amount of time after t_{move} required until the client is associated with access point B and no further associations or disassociations are observed.

In contrast to the original driver, the Madwifi 0.9.1-E now roams in the gradual roam case. In fact the movement time was reduced to 14 seconds in this case, and roaming clearly happens well under that. Ideally in this case, roaming would occur halfway between the access points or at at 7 seconds. The enhanced version performs within a couple seconds of this optimal time, but there is still room for improvement. The following section introduces a novel way in which access point selection can be improved by speeding up the scanning process.

8.5 Improving Access Point Selection Using Fast Scanning

Observation of Madwifi 0.9.1 scan performance in the emulator revealed that a scan operation would typically take around three seconds to complete. Thus even Madwifi 0.9.1-E would still take a very long time to locate new access points. This observation inspired a new approach to scanning to reduce this time.

8.5.1 Previous Work

Previous efforts have attempted to reduce scan completion time by reducing the number of channels scanned [10, 54, 41, 40]. The idea is that each access point maintains a list of neighbors. When a station roams away from that access point, it only scans channels of neighbors thereby reducing scanning time.

8.5.2 Fast Scanning Introduction

Consider Figure 8.3 which shows the operation of a traditional, non-optimized 802.11b/g active scan operation. When searching for access points, a scanning client scans all eleven channels by changing to the desired frequency, sending a probe request, and then waiting a period of time for probe responses. While a station is scanning, it cannot communicate with its access point. The time to complete active scanning is dominated by the need to wait for probe responses from access points. As a result, even scanning two channels can be very time consuming since if no access point is heard on a given channel, the maximum channel dwell time must be used. For Madwifi 0.9.1 this is 200 ms. Reducing this dwell time is one option, but can only help to a limited degree and could cause missed probe responses.

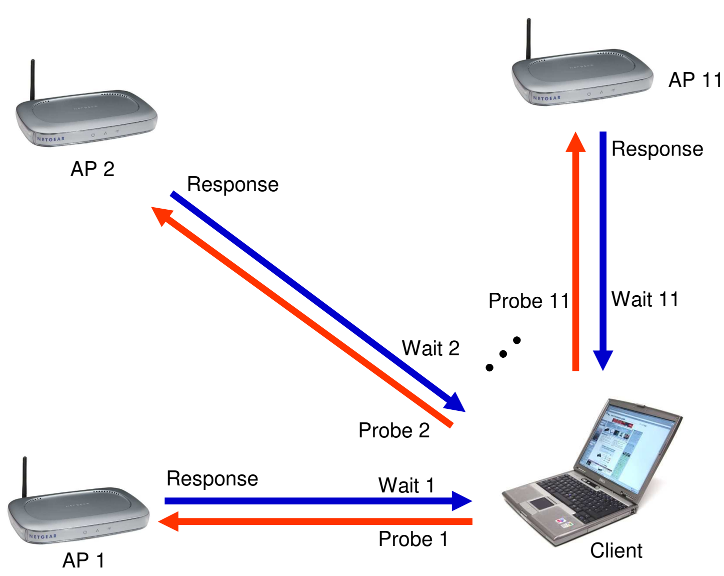

Figure 8.3: 802.11 Scanning

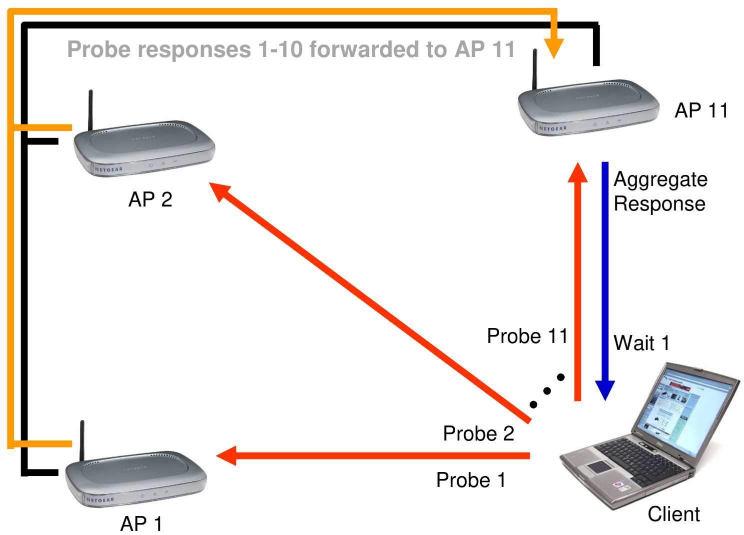

Figure 8.4: Fast Scanning

This section introduces a new approach to reducing scanning time called Fast Scanning, and depicted in Figure 8.4. The idea is to eliminate the time required to listen for probe responses. The key insight leveraged by the "Fast Scanning" algorithm developed here is that scan completion time can be dramatically reduced by changing the scanning protocol so that only a station's current access point (or others on the same channel) respond with a probe response. Access points on other channels forward scan information to the scanning node's access point where it is sent to the scanning access point.

A fast scanning client scans all channels by changing to each channel, sending a probe request, and then changing to the next channel without waiting for a probe response. The last channel scanned is the channel of the currently associated access point which sends an aggregate probe response containing scan information from all associated access points. This approach greatly reduces the amount of time required to complete a scan. Thus all channels can often be scanned in less time than two channels can be scanned without fast scanning. Moreover, the only wait for probe reponses occurs when on the channel of the currently associated access point. As a result, unlike traditional scanning, communication can still take place while waiting for the probe response.

When initially joining a network, fast scanning must be modified since there is no

	$t_{reaction}$	$t_{completionA}$	$t_{completionB}$	$t_{disassociation}$	$n_{associations}$
Mean	6.71 s	6.71 s	9.37 s	0.14 s	1.7
Median	6.61 s	6.61 s	7.85 s	0.00 s	1.0
Min.	3.53 s	3.53 s	4.95 s	0.00 s	1.0
Max.	10.66 s	3.53 s	4.95 s	1.33 s	4.0

Table 8.4: Gradual roam performance for Madwifi 0.9.1-EFS

current access point for the scanning client. In this case, the client may use the first candidate access point to forward probe responses. This case is left as future work.

8.6 Using Emulation to Aid Development

To demonstrate the usefulness of fast scanning, this approach was implemented in the Madwifi 0.9.1 driver. This driver which includes both the earlier enhancements and fast scanning is called Madwifi 0.9.1-EFS.

Again, the emulator proved essential in allowing this work to proceed quickly. Using the emulator's interactive GUI enabled manual testing of implemented features. Bugs in the development process were quickly revealed since complete control over the signal environment eliminated uncertainly and allowed precise testing. No physical movement of devices was required.

8.7 Measuring Fast Scanning Performance

8.7.1 Two Access Point Tests

The performance of the Madwifi 0.9.1-EFS was then measured using the two access point gradual roam test. The abrupt roam test was measured as a sanity check; results are not shown since performance was essentially the same as Madwifi 0.9.1-E since fast scanning was not implemented for the not associated case.

Gradual Roam Scenario.

Table 8.4 shows the two access point gradual roam performance of Madwifi 0.9.1-EFS (again, the figures shown are from 20 repetitions.) The reduced scanning time enabled by fast scanning brings the scan completion time much closer to the optimal 7.0 seconds compared to Madwifi 0.9.1-E. Thus fast scanning appears to be a promising and viable approach.

8.8 Multiple Access Point Performance.

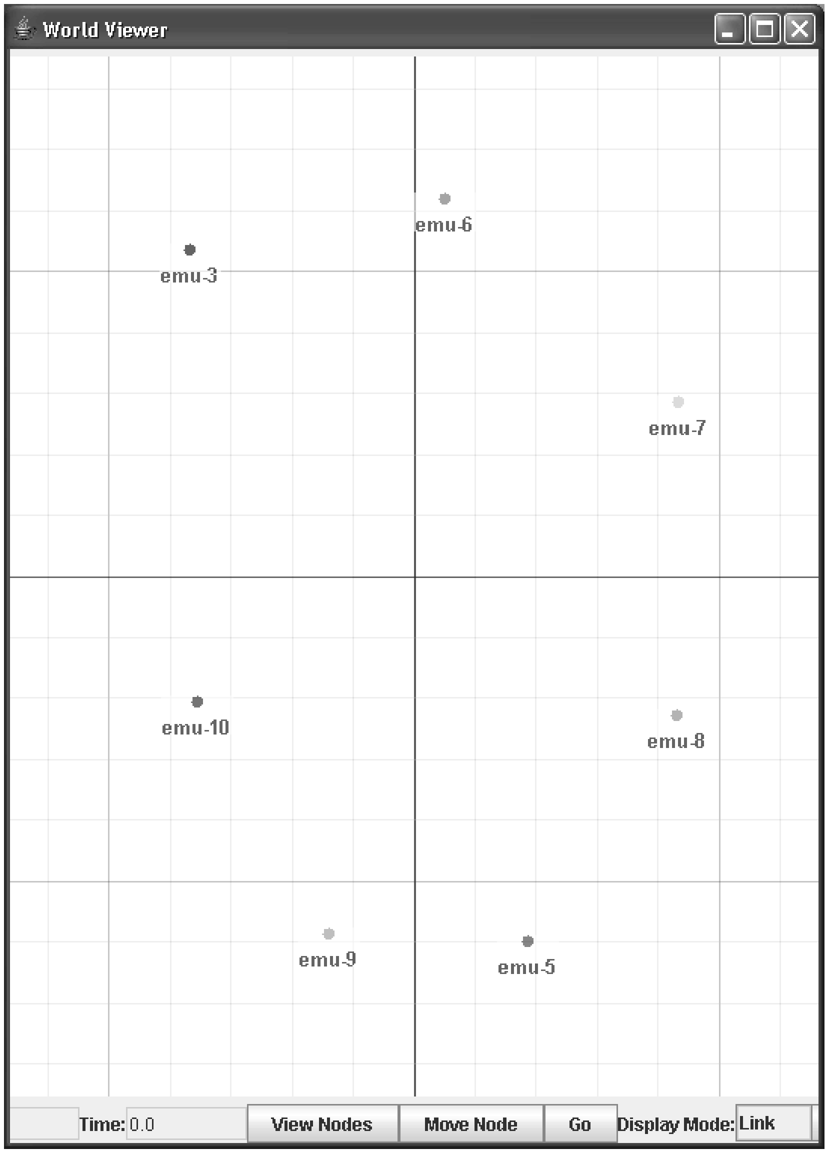

Figure 8.5: Multi-AP Test Topology

The performance of the three Madwifi variations discussed above was then measured for in a multiple-access point high-mobility scenario. The topology for this test is shown in Figure 8.5. Six access points - emu-3, emu-6, emu-7, emu-8, emu-9, and emu-10 - are distributed in the test area. They are set to channels 8, 10, 6, 4, 2, and 10 respectively. A log distance large scale path loss model is used with a d_0 of 1 m, a

	Mean Fraction Received	Median Fraction Received	Standard Deviation
Madwifi 0.9.1	0.41	0.41	0.16
Madwifi 0.9.1-E	0.70	0.69	0.05
Madwifi 0.9.1-EFS	0.77	0.76	0.04

Table 8.5: Multi-access point roam performance comparison

pl_{d0} of 40 dB, and a path loss exponent n of 4.1. Ricean fading with $k = 3$ was used.

The client - emu-5 - navigated a route near all of the access points beginning near access point emu-8 and moving in a counter-clockwise fashion. While the client navigated this route, UDP-unicast traffic was sent from a wired host, through the client's current access point, to the client. Each packet was stamped with an ID, and the client recorded which packets were received successfully. This test executed 20 times. The test was always commenced as the client approached emu-8 at the start of route navigation. The test terminated when completion of a single circuit neared completion. Thus, for each circuit, a particular packet number is sent to the client when it is nearly in the same physical location as it is on other circuits. The traffic sent used a VoIP-like data profile of 240 byte packets sent at 33 packets per second. 4620 were sent during each circuit.

Table 8.5 shows the result. As expected, Madwifi 0.9.1 fares very poorly. Madwifi 0.9.1-E performs reasonably well, and is a substantial improvement over the original driver. Madwifi 0.9.1-EFS is able to increase performance slightly in this case. (The full benefits of fast scanning are somewhat hidden in this test since the VoIP-like data does not push the limits of throughput connectivity and quality differences between Madwifi 0.9.1-E and Madwifi 0.9.1-EFS due to transmission rate are not observable.)

8.9 Summary

This chapter has demonstrated the emulator's benefits enabling network layer tests and experiments. In particular, this chapter has looked at the problem of 802.11 access point selection. The performance of a popular driver was measured and found to be very poor under any sort of mobility. The emulator enabled an investigation into the source of the poor performance which quickly revealed several contributing factors.

An improved version of Madwifi 0.9.1 was developed using the insight gained, and the emulator allowed accurate measurement of the effects of various tuning parameters which resulted in a better driver.

Finally, the above experiments inspired a new approach to access point selection - Fast Scanning - that greatly reduces the time required to look for access points in a wireless LAN. The emulator aided development of this driver by reducing development cycle latency and enabling precise tests to be conducted.

The performance of access point selection in a highly mobile multiple access point scenario was then measured to demonstrate the power of the emulator at conducting a network-wide test. This test revealed that Madwifi 0.9.1-E and Madwifi 0.9.1-EFS greatly improved roaming behavior in this highly mobile environment.

Using traditional techniques to conduct the work discussed in this chapter have been extremely labor intensive, and yielded less accurate insight into driver behavior. For example, the closest analog to the abrupt roam test in a wireless testbed involves running between two distributed access points. Simulation would not have allowed experimentation with the actual driver. Conducting the multi-access point test would have required a large amount of setup. Even then, the mobility would not have been precisely controlled and coordinated with packet transmission. Moreover, the signal environment would have been uncontrolled and differences between tests might have been caused by factors outside of experimental control. In contrast, physical layer emulation enabled realistic, yet precisely controlled experiments to be conducted. This high level of control and repeatability was essential in allowing useful summary statistics to be computed for the tests described earlier.

Chapter 9

Conclusion

In concluding this thesis, this chapter first revisits the motivation behind this work, and the role that this work plays in assisting wireless researchers and developers. Key contributions are then discussed, after which important lessons learned about wireless network behavior - discovered in the course of this work - will be presented. This work then concludes by discussing future directions that wireless technology is likely to take and the role that physical layer network emulation can play.

9.1 Motivation Revisited

The Need for Accurate, Controlled Wireless Experimentation. Wireless networks are rapidly proliferating, providing Internet connectivity in campus environments, homes, and public places. New applications - perhaps sensor networks, vehicular networks, and others currently unforeseen - will further increase the prevalence of wireless networks. Given the massive deployment of wireless networks, understanding and improving wireless network reliability and performance is an increasingly important and timely topic. To accomplish this, researchers need the capability to run realistic and repeatable tests and experiments.

Drawbacks of Previous Approaches. The most obvious way to gain experimental realism is to run experiments on a real wireless network. Unfortunately, conducting wireless experiments on a deployed testbed is challenging due to the distributed topology of wireless networks, the ephemeral nature of wireless signal propagation, and the susceptibility to external interference. Obtaining, repeatable experimentation with real wireless nodes running real applications operating in a physical environment is often not feasible. For this reason, most wireless research to-date has relied on evaluation via simulation.

Wireless simulators do not, however, completely duplicate real hardware in an operational environment, and the correctness of wireless simulation is poor at worst and difficult to validate at best.

Thus researchers have faced a stark alternative: conduct simple yet unrealistic experiments via simulation, or conduct realistic yet cumbersome and unrepeatable experiments via testbeds.

How Physical Layer Wireless Network Emulation Helps. This work has addressed these obstacles by introducing a new experimental approach that strikes a balance between the realism of testbeds and the repeatability of simulation: physical layer wireless emulation. This technique supports real applications running on real wireless devices, and virtualizes only the minimum experimental component - the signal propagation environment - necessary to obtain experimental repeatability.

This work has shown that physical layer wireless emulation achieves fine grained control over RF propagation, enables the analysis of higher layer performance, and facilitates the development of enhanced wireless protocols.

9.2 Contributions

New Experimental Methodology. This most significant contribution of this thesis is the introduction of a new research methodology: physical layer wireless network emulation. This technique has been shown to possess unique advantages over previous techniques. Like wireless testbeds, physical layer emulation supports experimentation with real wireless devices running real applications. Unlike testbeds, however, the signal propagation environment is fully flexible and controllable. Thus, arbitrary signal propagation environments can be utilized in a completely controlled and repeatable fashion. This is similar to the control and repeatability found in wireless simulators, but without sacrificing support for real devices, network software, and user applications like wireless simulators do. Moreover, the complete control over the signal propagation environment enables experiments to be conducted that could not be conducted in a wireless testbed or a wireless simulator.

Solves Problems Necessary for Hardware Realization. This work has solved the problems necessary to implement an operational wireless network emulator. An architecture was developed for converting RF signals to and from digital form, and for processing these signals to construct arbitrary wireless signal propagation environments. In addition, as this work includes functional implementations for the components in this architecture that overcome key challenges such as the synchronization of distributed data conversion components.

Software Architecture for Physical Layer Emulation. At the software level, this thesis has presented an architecture for controlling the hardware-based emulation in real time, as well as an implementation of that architecture.

Trace Capture and Playback. This work developed a simple method for recording, processing, and playing back coarse-grained channel traces using commodity wireless hardware. Techniques were developed for overcoming the major inaccuracies inherent in these traces, and this technique was shown to obtain good results in low a delay-spread environment.

Functional Emulator. Moreover, this work has resulted in fully functional emulator that will continue to be used for wireless research. This emulator will also serve as a template for the development of emulators for use by other researchers.

Device and Link-level Analysis. This thesis has utilized physical layer emulation to analyze 802.11 link-level behavior. This analysis supplants convention with actual measured behavior, and should provide the basis for more accurate understanding of device operation and the development of improved wireless network simulators.

Access Point Selection. In addition, the emulation platform was used to analyze 802.11 access point selection and roaming, and to develop 802.11 protocol extensions that improve access point selection and roaming performance.

Utilizing the techniques developed in this work, researchers can reap similar benefits by conducting realistic tests in a controlled environment, and gaining insight into wireless network and application behavior that is not possible using other techniques.

9.3 Lessons Learned

As touched on in the contributions above, the network, device, and link-level analyses in this thesis have shed light on a number of important elements of wireless network performance. While general aspects of many of these elements can be found in other sources, the specific results are largely unique to this work. This section recaps several of the most important behavioral elements that networking researchers can benefit from.

Capture. Several tests in Chapter 7 illustrated the key role that capture plays in wireless data reception. While the basic effects of capture can be computed analytically, these tests have precisely quantified the performance of an actual implementation.

As a receiver begins to acquire a packet it must first lock onto the incoming signal. The first part of the preamble - before lock is acquired - is the most vulnerable to interference. After lock is acquired, capture effect enables the receiver to reject many interfering signals. For low transmission rates, even interfering signals that are somewhat stronger than the packet being received may be rejected.

If an interfering signal is strong enough, however, the receiver will drop the initial packet in favor of the new packet. If the two signals are nearly equal, then both will be lost and a collision will occur. For low transmission rates, the collision regime is narrow and signals must be nearly equal. For higher transmission rates, the regime is large.

An important result of this capture behavior is that 802.11 control and management frames are very robust to interference, and are unlikely to collide.

Off-channel behavior. 802.11b defines three "non-overlapping channels" in the North American regulatory domain. Several groups have proposed "cheating" by using four slightly overlapping channels, but no fine-grained controlled studies have been conducted to evaluate the efficacy of this proposal. This work has shown that a well-designed receiver can reject off-channel interference sufficiently to make four channels a viable approach.

In addition, some have proposed leveraging the fact that devices can communicate when they are on different channels for various purposes. The results in Chapter 7, however, show that this technique is of very limited use.

Hidden and exposed nodes. The link level measurements in Chapter 7 were used to analyze wireless LAN uplink behavior to gain insight into the prevalence of hidden and exposed nodes. This analysis shows that in a typical office WLAN, the hidden node problem is unlikely to afflict the vast majority of nodes due to the fact that very few nodes associated with the same access point are out of carrier sense range of each other. Moreover, co-channel interference is unlikely to be a problem in 802.11b network uplinks due to capture effect. As a result, 802.11b uplinks suffer most from exposed nodes - the needless deferring of a transmission to avoid collisions. Future MAC designs could achieve higher throughput by being capture aware.

RSSI. Much research has been published using RSSI for various purposes, yet scant attention is often paid to what exactly RSSI is reporting. This work has quantified RSSI, and found that RSSI linearity and accuracy varies across vendors and in some cases across cards from the same vendor. A common feature of RSSI measurements is that near the lowest signal levels, RSSI measurements "flatten out", and several actual RSS values will return the same RSSI. A key lesson is that researchers should measure RSSI on their devices (or a similar device) - using a spectrum analyzer for instance - before reporting values.

Multipath. This work has quantified the multipath reception behavior of a popular 802.11b card. The results show that this device does a good job of rejecting multipath indoors, but does a poor job outdoors. This is a result of the equalizer on the card being designed to mitigate indoor multipath delay spreads only. Hence, using wireless LAN hardware outdoors may result in unstable links due to multipath.

Naive use of channel information. Chapter 8 analyzes the roaming behavior of a popular wireless platform. While the driver analyzed was only a development version, it contained reasonably sophisticated code that attempted to apply many heuristics to improve performance. For instance, channel measurements were low-pass filtered using an exponentially weighted moving average filter, and scan results were aggressively cached. The undesired effect of these "enhancements", however, was to make access point selection decisions based on very stale information.

A key lesson is that naively applying standard performance enhancing heuristics without considering the behavior of wireless channels may backfire.

9.4 Future Work

This thesis has introduced physical layer network emulation as an attractive alternative to traditional methods of wireless network experimentation. While this work has developed solutions necessary to make this technique viable, there are many open avenues work to pursue. This section discusses promising directions in which this work can be extended in the short term. The following section will discuss trends in wireless technology, and how physical layer emulation can play a role further in the future.

9.4.1 Scale

The clearest avenue of future work to pursue is that of increasing emulator scale. Several alternative architectures to the one used in this work were discussed in Chapter 2. Further development and implementation of these alternative architectures should provide improved emulator scale and fidelity; research in this direction is a high priority for future work.

Moreover, the channel modeling techniques utilized in the emulator's current incarnation assume that two unique and independent channels exist between every pair of RF Nodes attached to the emulator. This results in the need to support an order n^2 number of channels. Reducing fidelity of channel modeling for nodes that are more distant could reduce the need for independent channels and also increase emulator scale.

9.4.2 Improving Wireless LANs

Chapter 8 has analyzed one important aspect of wireless LAN operation: access point selection and roaming. The work in that chapter, however, is only a beginning. Fast-scanning can be improved to handle cases where nodes are not yet associated with an access point, and to handle cases where encryption is in use.

In addition, many other important areas of wireless networking operation such as rate selection and load balancing are excellent candidates for analysis using the emulator developed in this thesis. Previous work in these areas has suffered greatly from a lack of a controlled, realistic experimental environment. Emulation overcomes this obstacle, and should enable insights that are extremely difficult to achieve using other techniques.

9.5 Trends in Wireless Technology

A common theme in future wireless networks is the increasing prevalence of data. Data traffic is bursty and difficult to predict; as a result, data networks benefit greatly from statistical multiplexing of resources. While previous voice-centric wireless networks - especially TDMA and FDMA networks - had fewer network level effects to contend with, statistical multiplexing will cause network-level effects to play prominent roles in the data-centric networks of the future.

Physical layer wireless network emulation provides powerful insight into network level behavior by striking a balance between control and realism that is achieved by no other technique. While experiments in real networks will always be necessary, emulation will enable future wireless networks to be developed with fewer real-network experiments. This will enable faster, more reliable development of wireless networks, and will provide a greater understanding of network operation than can be attained in other ways. Simulation will continue to play a role as a technique of last resort that provides intuition when the scale or functionality of an experiment cannot be investigated using emulation or real networks.

This section now elaborates on emerging trends in consumer wireless technology, and what role physical layer emulation can play.

9.5.1 Cellular

The mass adoption of wireless technology - led by cellular - is arguably the most fundamental technological development in the last decade. In the coming years, this will continue as cellular adoption increases penetration in existing markets, and enters new markets in the developing world.

Contemporary cellular devices are, however, no longer merely mobile telephones, but are increasingly data-capable and data-centric. Future cellular devices will continue this trend, and will be full-blown computing platforms with ever-increasing requirements for both sending and receiving data. As a result, cellular networks are already becoming more data oriented. Future cellular networks will be IP-based and support data as a first-class service.

9.5.2 Metropolitan Area Networks

To-date, metropolitan area wireless networks have occupied a relatively small niche of the wireless market. With the advent of WiMAX (IEEE 802.16) networks, however, the WMAN market is undergoing a dramatic transformation into a commodity-driven market providing consumer Internet connectivity. Initial users of 802.16 networks will be traditional data devices such as laptops (laptops are - in fact - already beginning to ship with integrated WiMAX support.) WiMAX technology will, however, form the basis of some fourth generation cellular networks. As a result, in the coming years, there will be no distinction between cellular networks and WMANs; WMANs will simply be part of standard cellular service.

9.5.3 Wireless Local Area Networks

WLAN technology has flourished over the last decade, and continues to grow at a rapid rate. While originally targeted towards enterprise LAN replacement, several innovative usages of WLAN technology have emerged such as “hotspots” and “wireless mesh” networks. In the long term, however, cellular/WMAN networks will likely greatly reduce the need for both hotspots and mesh networks. Enterprise WLAN networks, however, will persist due to both performance and security issues that make running a LAN over the WMAN/Internet unattractive. Similarly, home WLAN networks will also persist. In addition, small device-to-device ad hoc usage of WLAN technology - such as that found in portable video game devices - is also likely to persist.

9.5.4 Short-range Wireless

Short-range wireless networks are primarily targeted at “cord replacement” i.e. allowing electronic devices to communicate without requiring wires. Bluetooth devices currently provide low-bandwidth wireless connectivity for headsets, keyboards, etc. UWB devices will be available reasonably soon that will eliminate the need for wires between everything from computer monitors to video recorders.

9.5.5 RFID.

Embedded RF-based identification tags will eventually replace bar-codes on products. While primarily designed for retail purposes, widespread RFID will enable devices to discover rich information about objects in their surrounding environment.

9.5.6 Intervehicular Networks

Automobile manufacturers are currently developing intervehicular wireless networks. These are primarily targeted towards safety applications. The devices used in these networks, however, will likely provide WLAN access as well that will be used to communicate with roadside services and infrastructure (e.g. to pay a toll without requiring a custom transponder.)

9.5.7 Sensor Networks

Sensor networks are currently a very popular research topic, though commercially they have yet to make significant inroads. In the next several years, they will likely continue to be an active area of research. Commercial prospects, however, are unclear.

9.5.8 The Future Role of Emulation

The contention-based MACs used in WLAN, sensor, and intervehicular networks make physical layer network emulation particularly useful for these networks. Nevertheless, the increasing data present in cellular/WMAN networks makes physical layer network emulation attractive even for these networks. Short-range networks may also benefit as the number of devices interacting may grow to be quite large, though technical challenges must be overcome to digitally emulate UWB signals. In the near term, RFID networks are not likely to require physical layer emulation, but should challenging multi-reader environments become common, network emulation will be useful to understand the network level effects present.

A common element in both future WLANs and WMANs will be the use of multiple air interface devices such as MIMO technology. The current emulator architecture supports these in a straightforward fashion with each antenna element supported as an independent RF Node. This simple approach limits emulator scale and models extraneous channels (such as those between antenna elements on a single device.) An important area of future research will be scalable support for multiple air interface devices as discussed in Chapter 2.

In closing, the increasing deployment of data-centric wireless networks make physical layer network wireless emulation an attractive technique for conducting future wireless experiments. The precise design presented in this thesis will need to adapt, however, to accommodate future technology. Nevertheless, the design challenges should be resolvable, and physical layer wireless network emulation will be able to provide powerful insight for years to come.

Bibliography

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements from an 802.11b Mesh Network. In *Proceedings of the ACM SIGCOMM Conference on Network Architectures and Protocols*, Portland, August 2004.

[2] G. Arredondo and W. Chriss. A multipath fading simulator for mobile radio. *IEEE Transactions on Communications*, 21(11):1325–1328, November 1973.

[3] P. Bahl and V. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *Proceedings of PIMRC 2000. London, UK*, September 2000.

[4] J.R. Ball. A real time fading simulator for mobile radio. *The Radio and Electronic Engineer*, 52(10), 1982.

[5] V. Bose. Design and implementation of software radios using general purpose processor, June 1999.

[6] E. Boutillon, J. Danger, and A. Gazel. Design of high speed awgn communication channel emulator. *Analog Integrated Circuits and Signal Processing*, 34(2):133–142, February 2003.

[7] B. Braswell. Modelling data rate agility in the 802.11a wireless local area networking protocol, March 2001.

[8] E. Casas and C. Leung. A simple digitial fading simulator for mobile radio. *IEEE Transactions on Vehicular Technology*, 39(3):205–212, August 1990.

[9] Rooftop Communications. The rooftop c++ protocol toolkit. `http://www.rooftopcommunications.com`.

[10] Cisco Corporation. Cisco Compatible Extensions. `http://www.cisco.com/web/partners/pr46/pr147/partners_pgm_concept_home.html`.

[11] P. De, R. Krishnan, A. Raniwala, K. Tatavarthi, N. Syed, J. Modi, and T. Chiueh. MiNT-m: An Autonomous Mobile Wireless Experimentation Platform. In *Proc. of Mobisys 2006*, Uppsala, Sweden, June 2006.

[12] P. De, A. Raniwala, S. Sharma, and T. Chiueh. MiNT: A Miniaturized Network Testbed for Mobile Wireless Research. In *Proc. of Infocom 2005*, Miami, Florida, March 2005.

[13] Analog Devices. Ad9430. `http://www.analog.com/en/prod/0,,AD9430,00.html`.

[14] Analog Devices. Ad9736. `http://www.analog.com/en/prod/0,,AD9736,00.html`.

[15] Elektrobit. Propsim c8 wideband multichannel simulator. `http://www.propsim.net/`.

[16] Elektrobit. Propsound cs. `http://www.propsim.com/index.php?1983`.

[17] K. Fall. Network emulation in the vint/ns simulator. In *Proc. of The Fourth IEEE Symposium on Computers and Communications*, Red Sea, Egypt, July 1999. IEEE.

[18] M. Fink. Time-reversed acoustics. *Scientific American*, 281(5):91–97, 1999.

[19] G. J. Foschini and M. J. Gans. On limits of wireless communications in a fading environment when using multiple antennas. *Wireless Personal Communications*, 6(3):311 – 335, March 1998.

[20] C. Fullmer and J. Garcia-Luna-Aceves. Solutions to Hidden Terminal Problems in Wireless Networks. In *Proceedings of Sigcomm 1997*, Cannes, France, 1997.

[21] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. In *UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013*. UCLA Computer Science Department, 2002.

[22] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: A software environment for developing and deploying wireless sensor networks. In *Proceedings of the 2004 USENIX Technical Conference*.

[23] E. Hernandez and S. Helal. Ramon: Rapid mobility network emulator. In *Proc. of the 27th IEEE Conference on Local Computer Networks (LCN'02)*, Tampa, FL, November 2002. IEEE.

[24] G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Multi-hop Wireless Networks. In *Proceedings of MobiCom2001. Rome, Italy*, September 2001.

[25] W. Honcharenko and H. Bertoni. Prediction of wideband RF propagation characteristics in buildings using 2D ray tracing. In *Proceedings of VTC July 1995*, 2005.

[26] W.C. Jakes. *Microwave Mobile Communications*. Wiley, New York, NY, 1974.

[27] G. Judd and P. Steenkiste. Repeatable and realistic wireless experimentation through physical emulation. In *HotNetsII*, Boston, MA, November 2003. ACM.

[28] G. Judd and P. Steenkiste. A simple mechanism for capturing and replaying wireless channels. In *E-Wind 2005*, Philadelphia, PA, August 2005. ACM.

[29] G. Judd and P. Steenkiste. Using Emulation to Understand and Improve Wireless Networks and Applications. In *Proceedings of NSDI 2005*, Boston, MA, May 2005.

[30] J. Kaba and D. Raichle. Testbed on a Desktop: Strategies and Techniques to Support Multi-hop MANET Routing Protocol Development. In *Proc. of Mobihoc 2001*, Long Beach, CA, October 2001.

[31] Walt Kester. MT-007: Aperture Time, Aperture Jitter, Aperture Delay Time Removing the Confusion. `http://www.analog.com/en/content/0,2886,760%255F788%255F91284,00.html`.

[32] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of MSWiM 2004*, Venice, Italy, October 2004.

[33] S. Kurkowski, T. Camp, and Michael Colagrosso. Manet simulation studies: The incredibles. *Mobile Computing and Communications Review*, pages 50–61, October 2005.

[34] R. Lackey and D. Upmal. Speakeasy: The military software radio. *IEEE Communications*, 33(5):56–61, 1995.

[35] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems*, 2003.

[36] T. Lin, S. Midkiff, and J. Park. A dynamic topology switch for the emulation of wireless mobile ad hoc networks. In *Proc. of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, Tampa, FL, November 2002. IEEE.

[37] P. Mahadevan, K. Yocum, and A. Vahdat. Emulating large-scale wireless networks using modelnet. In *Poster and Abstract Mobicom 2002*, Atlanta, GA, September 2002. ACM.

[38] S. McCanne and S. Floyd. UCB/LBNL/VINT Network Simulator - ns (version 2). `http://www.isi.edu/nsnam/ns/`, April 1999.

[39] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh. Exploiting Partially Overlapping Channels in Wireless Networks: Turning a Peril into an Advantage . In *Proceedings of IMC 2005*, Berkeley, CA, 2005.

[40] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the ieee 802.11 mac layer handoff process. *ACM SIGCOMM Computer Communication Review*, 33(2), 2003.

[41] Arunesh Mishra, Minho Shin, and William Arbaugh. Context caching using neighbor graphs for fast handoffs in a wireless network. In *Proceedings of Infocom 2004*, Hong Kong, 2004.

[42] B.. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz. Trace-based mobile network emulation. In *Proc. of SIGCOMM 1997*, Cannes, France, September 1997. ACM.

[43] OPNET Tech. Opnet. `http://www.opnet.com`.

[44] K. Papagiannaki, M. Yarvis, and W. Conner. Experimental characterization of home wireless networks and design implications. In *Proceedings of Infocom 2006*, Barcelona, Spain, April 2006.

[45] K. Pawlikowski, H. Jeong, and J. Lee. On credibility of simulation studies of telecommunications research. *IEEE Communications*, 40(1):132–139, January 2002.

[46] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *Proc. of HotNets-I*, Princeton, NJ, October 2002. ACM.

[47] R. Punnoose, P. Nikitin, and D. Stancil. Efficient simulation of ricean fading within a packet simulator. In *Proc. of the IEEE Conference on Vehicular Technology*, Boston, MA, September 2000. IEEE.

[48] A. Rajkumar, B. Naylor, and L. Rogers. Predicting rf coverage in large environments using ray-beam tracing and partitioning tree represented geometry. *Wireless Networks*, 2(2):143–154, 1996.

[49] T.S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice-Hall, Englewood Cliffs, NJ.

[50] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In *Proc. of WCNC 2005.*, New Orleans, LA, March 2005.

[51] L. Roberts. Aloha packet system with and without slots. *ARPA Network Information Center, TR ASS Note 8*, 1972.

[52] A. Saha, K. To, S. PalChaudhuri, S. Du, and D. Johnson. Physical implementation and evaluation of ad hoc network routing protocols using unmodified simulation models. In *Proceedings of the ACM SIGCOMM Asia Workshop*, 2005.

[53] Scalable Network Tech. Qualnet. `http://www.scalable-networks.com/`.

[54] Minho Shin, Arunesh Mishra, and William Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. In *Proceedings of Mobisys 2004*, Boston, MA, 2004.

[55] SmartAnt Telecomm Ltd. Yagi antennas. `http://www.smartant.com/Products/ISM/2.4G/FYW24-01518BFL.pdf`.

[56] Spirent Communications. Tas4500 flex5 rf channel emulator. `http://www.spirent-communications.com/`.

[57] Azimuth Systems. Azimuth w-series wi-fi test solutions. `http://www.azimuthsystems.com/index.asp?p=213`.

[58] C. Tai. Complementary reciprocity theorems in electromagnetic theory. *IEEE Trans. on Antennas and Propagation*, 40(6):675–681, 1992.

[59] M. Takai and J. Martin. Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks, October 2001.

[60] F. Tobagi and L. Kleinrock. Packet switching in radio channels: Part ii–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *IEEE Trans. on Communications*, 23(12):1417–1433, 1975.

[61] UCLA. Whynet Project. `http://chenyen.cs.ucla.edu/projects/whynet`.

[62] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of OSDI 2002*, Boston, MA, December 2002. USENIX Association.

[63] V. van der Vegt. Auto Rate Fallback. `http://www.phys.uu.nl/~vdvegt/docs/gron/node24.html`.

[64] B. White, J. Lepreau, and S. Guruprasad. Lowering the barrier to wireless and mobile experimentation. In *Proc. of HotNets-I*, Princeton, NJ, October 2002. ACM.

[65] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of OSDI 2002*, Boston, MA, December 2002. USENIX Association.

[66] Xilinx. Ise foundation. `http://www.xilinx.com/ise/logic_design_prod/foundation.htm`.

[67] Xilinx. Ml401 documentation. http://www.xilinx.com/products/boards/ml401/docs.htm.

[68] Xilinx. Platform studio and the edk. http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm.

[69] Xilinx. Virtex-4 family overview. http://direct.xilinx.com/bvdocs/publications/ds112.pdf.

[70] Xilinx. Virtex-ii pro and virtex-ii prox x platform fpgas: Complete data sheet. http://direct.xilinx.com/bvdocs/publications/ds083.pdf.

[71] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proc. of MobiSys 2004*, Boston, MA, June 2004. ACM.