Repeatable and Realistic Wireless Experimentation through Physical Emulation

Glenn Judd and Peter Steenkiste
Carnegie Mellon University
Pittsburgh, PA, USA
glennj@cs.cmu.edu prs@cs.cmu.edu

Abstract

In wireless networking research, there has long existed a fundamental tension between experimental realism on one hand, and control and repeatability on the other hand. Hardware-based experimentation provides realism, but is tightly coupled to the physical environment and circumstances under which experiments are carried out. To overcome this, researchers have understandably embraced simulation as a means of evaluation. Unfortunately, wireless simulation is plagued with inherent inaccuracies. To overcome the stark tradeoff between the realism of hardware-based experimentation and the repeatability of simulation-based experimentation, we are developing a wireless emulator that enables both realistic and repeatable experimentation. Unlike previous emulators, our approach simultaneously achieves both a high degree of realism and fine-grained repeatability by leveraging physical layer emulation.

1 Introduction

As wireless network deployment and use become ubiquitous, it is increasingly important to make efficient use of the finite bandwidth provided. Unfortunately, research aimed at evaluating and improving wireless network protocols is hindered by the inability to perform repeatable and realistic experiments.

Techniques that have proven successful for analyzing wired networks are inadequate for analyzing wireless networks since they are fundamentally different. While the physical layer can frequently be ignored in wired networks, in wireless networks the 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. As a result, traditional methods of experimentation have difficulty satisfying the needs of wireless researchers.

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, support for a large number of nodes, isolation from production networks, and integration with wired networks and testbeds. We now discuss how alternative methods of experimentation fare with respect to this list of desirable properties.

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 issues since the behavior of the physical layer is tightly coupled to the physical environment and precise conditions under which an experiment is conducted. The mobility of uncontrolled radio sources, physical objects, and people makes these conditions nearly impossible to reproduce. It is also difficult to avoid affecting colocated production networks. Moreover, configurability and management of even a small number of mobile nodes distributed in three dimensions is cumbersome; this lack of manageability also makes integration with external networks problematic.

For these reasons, many researchers have understandably embraced simulation as a means of evaluation. This approach solves the problem of repeatability, configurability, manageability, modifiability, and (potentially) integration with external networks, but faces formidable obstacles in terms of realism. Wireless simulators are confronted with the difficult task of recreating the operation of a system at all layers of the network protocol stack as well as the interaction of the system in the physical environment. To make the problem tractable, simplifications are typically made throughout the implementation of the simulator. Even fundamental tasks such as deciding what a received frame looks like [1] diverge greatly from the operation of real hardware. 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 5.

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 5, with the exception of a few very small-scale emulators, and emulators hardwired to a particular physical location, these emulators have typically 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.

We are developing a wireless emulator that enables both realistic and repeatable wireless experimentation by accurately emulating wireless signal propagation in a physical space. Unlike previous approaches, this emulator provides a real MAC layer and a realistic physical layer while avoiding adopting an uncontrollable or locale-specific architecture. The key technique we use to accomplish this is digital emulation of

signal propagation using an FPGA.

This emulator provides an attractive middle ground between pure simulation and wireless testbeds. To a large degree, this emulator should be able to maintain the repeatability, configurability, isolation from production networks, and manageability of simulation while obtaining much of the realism of hardware testbeds, and in many cases, this emulator should provide a superior platform for wireless experimentation. This emulator 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 (for example, changing the MAC hardware currently can only be simulated, while introducing directional antennas can be emulated). Real world evaluation is still useful for verifying the operation of the emulator in real settings or for settings with physical environments that are not currently reproduced in the emulator.

Our discussion proceeds as follows: Section 2 describes the architecture of the emulator we are developing; Section 3 discusses our current emulator prototype; Section 4 presents several experiments that illustrate several aspects of our prototype's functionality; Section 5 discusses related work; and Section 6 concludes our discussion.

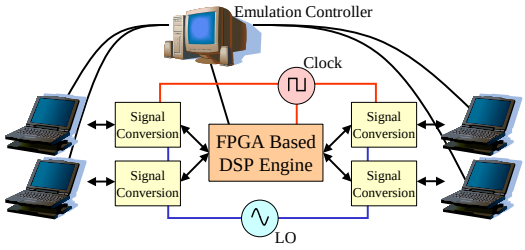


Figure 1. Emulator Architecture

2 Emulator Architecture

The architecture of the proposed emulator is shown in Figure 1. A number of "RF nodes" (e.g. laptops, access points, 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). For each RF node, the RF signal transmitted by its line card is mixed with the local oscillator (LO) signal down to baseband and is then digitized. The digitized signals are fed into a DSP engine that is built around one or more FPGAs. 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. Finally, for each RF node, the DSP combines the appropriately processed input signals from all the other RF nodes. This signal is then sent out to the wireless line card through the antenna port. Given the current state of technology, a DSP engine based on a single FPGA might support as many as 50 RF nodes. Using multiple FPGAs, even larger systems can be built.

The operation of the emulator is managed by the Emulation Controller, which coordinates the movement of RF nodes (and possibly physical objects) in the emulated physical space. The Emulation Controller uses location information (and other factors as dictated by the signal propagation model in use) to control the emulation of signal propagation within this emulated environment. In addition, the Emulation Controller coordinates node (and object) movement in physical space with the sending of data and the operation of applications on the RF nodes. Each RF node runs a small daemon that allows the Emulation Controller to control its operation via a wired network. RF nodes that are not capable of running code may require a proxy to run the daemon on their behalf.

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 [2] or Emulab [3].

In the next section we discuss a proof-of-concept implementation of the emulator architecture just presented. We then present several experiments that demonstrate how this prototype can support a wide range of experiments.

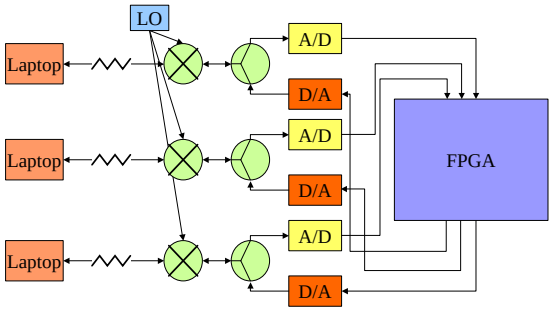


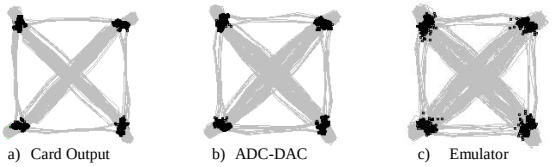
Figure 2. Prototype Hardware Architecture

3 Proof-of-Concept Prototype

3.1 Description

Hardware. To demonstrate the feasibility of the wireless emulator, we have constructed a small prototype designed to validate the emulator's primary functionality by emulating signal propagation between three laptops on a single 802.11b "non-overlapping channel". This prototype was constructed using discrete components for approximately $1,000 excluding the cost of RF nodes and external support equipment; we expect future versions - with integrated components - to cost on the order of several hundred dollars per RF node supported. The results obtained so far show that the approach that we advocate is capable of providing powerful wireless emulation capabilities. We first discuss our prototype's hardware implementation followed by a discussion of the control software.

Figure 2 shows the hardware architecture of the prototype. Each laptop operates on 802.11b Channel 11 which is centered at 2.462 GHz and contains its main spectral elements from 2.451 GHz to 2.473 GHz. The outgoing signal from each laptop is first attenuated and then mixed with a 2.449 GHz LO signal. The resulting output from the mixers (ignoring the signal image) is a signal ranging from 2 to 24 MHz. This signal is then fed into an A/D board for sampling. Each A/D board



a) Card Output (mixed down) EVM: 4.8%

b) ADC-DAC EVM: 6%

c) Emulator Prototype EVM: 7.7%

Figure 3. Physical Layer Fidelity

generates 12-bit digital samples of the incoming signal at 52 Msps, and sends them to the FPGA for processing.

FPGA-Based DSP. Inside the FPGA, the signals are processed to emulate a physical environment. Each outgoing signal is then sent to a D/A board for reconstruction. It is then mixed up and attenuated before arriving at the destination wireless card's antenna port.

Between the three laptops in our prototype, there exist a total of six point-to-point signal channels. For each of these channels, the Emulation Controller is capable of dynamically adjusting the attenuation from the source to the destination by dynamically setting the scaling factors mentioned previously. Hence, for each of the six signal paths, the emulator can recreate effects such as large-scale path loss and small-scale fading (we currently only employ large scale path loss). Our emulator's noise floor is either generated by replacing an RF node with a noise generator, or by using the noise floor naturally present in our emulator.

The use of an FPGA-based DSP gives our prototype the ability to support more complex signal modeling than we have yet implemented. Moreover, the FPGA's "program" can be customized for each individual experiment. For instance, while our current FPGA configuration does not support geometric models of signal propagation, these could be added by configuring the FPGA with multiple signal paths between each source and destination, and adding a programmable delay to each path. Also, one ore more noise generators could be programmed into the FPGA to provide precise control over the noise floor.

Emulation Controller. The Emulation Controller is driven by scripts that specify each node's movement and communication. As the RF nodes move about in the emulated physical space, the Emulation Controller continuously computes attenuation of each signal path and the scaling factors required to emulate this attenuation (a simple path loss model based on measurements in CMU's business school is used). After computation, these scaling factors are sent to the FPGA-based DSP. The Emulation Controller also generates a visual display of node location in the emulated physical environment.

Under direction of the Emulation Controller, we can generate network traffic between any pair of nodes. The Emulation Controller is able to synchronize this traffic with node movement. If desired, however, nodes can be made to generate traffic manually. This is most useful for scenarios in which nodes are stationary.

3.2 Validation

We now present a simple validation of the operation of our emulator by examining two key properties: fidelity and isolation. We then briefly discuss how this validation will be applied to future versions of our emulator.

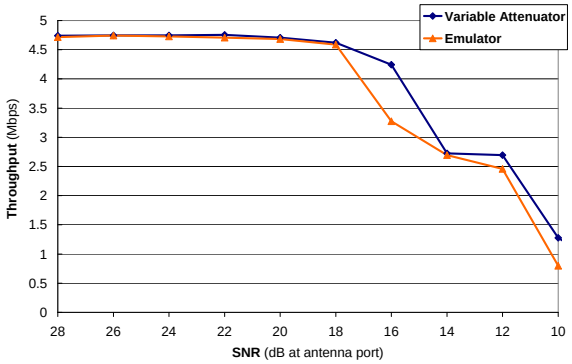


Figure 4. Transport Layer Fidelity

While our current prototype is simple, signal propagation fidelity is reasonably accurate. This can be quantified by measuring error vector magnitude (EVM) which is the relative difference between ideal signal constellation points and observed constellation points. Figure 3 compares the modulation fidelity of an 802.11 signal passing through our emulator (c) with the direct output of an 802.11 card (a). For each case, this figure shows the measured constellation, and EVM. Qualitatively this figure shows that our emulator adds a small amount of error to the modulation. This is quantified by an EVM difference of about 2.9%. Figure 3(b) shows the same measurement for a digitized signal that bypasses our FPGA-based DSP engine. The degradation seen in (c) relative to (b) implies that a good deal of the undesired noise in our emulator is being introduced in digital transmission to or from the DSP engine. By eliminating this noise, we expect to match the performance shown in (b) (future enhancements should actually allow us to surpass this performance).

Figure 4 shows that our prototype's physical layer fidelity translates into transport level fidelity; this figure compares TCP throughput for two laptops connected via a variable attenuator versus two laptops connected via our emulator prototype. Each data point is an average of 20 trials. Confidence intervals are omitted since they are very tight. Clearly high end performance is quite close with performance diverging at low SNR values. The refinements discussed previously should greatly narrow this gap.

An important benefit of our prototype is the ability to conduct experiments in isolation from external sources of interference. While our prototype is not yet specifically shielded against external interference, we performed simple measurements to determine the degree of isolation that our prototype obtained. Using a laptop external to our emulator, we generated a broadcast ping flood at 2 Mbps. With this external node placed at various distances from our emulator, we then measured the TCP throughput between two nodes internal to our emulator. When TCP throughput was unaffected by the ping flood, we inferred that the internal emulator nodes were no longer able to sense the carrier of the external interfering node. Our tests indicate that our emulator is isolated from external nodes that are approximately 18 meters or further away in our environment. We expect to significantly reduce

Figure 5. Hidden Terminal Topology

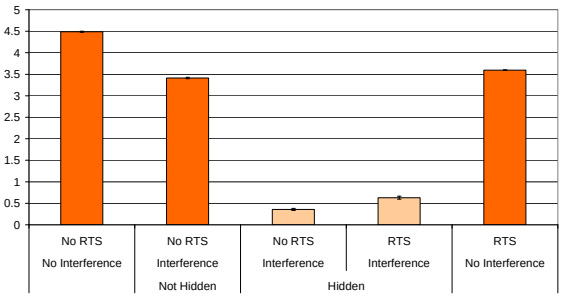


Figure 6. Hidden Terminal Results

this distance when we explicitly address this issue. Nevertheless, compared to a wireless testbed, our current prototype has greatly reduced the possible locations of interfering sources to a relatively small geometric space.

By altering this isolation test to measure internal isolation, we were able to verify that nodes attached to the emulator are effectively isolated against undesired transmission to each other despite their close proximity. In addition, we verified that the signal propagation between laptops in our emulator is unaffected by the movement of nearby people or objects.

We plan to validate the fidelity of future versions of our emulator via a variety of tests: EVM tests of multiple modulations, multi-tone intermodulation tests to demonstrate robustness against intermodulation products, noise floor measurements, packet error rate vs. SNR, and precise measurements of isolation. Where appropriate, we will compare results from these tests with coaxial cable-based tests or open air tests.

4 Sample Experiments

We now present a series of experiments that provide a small sampling of our wireless emulator's capabilities. For each of the experiments presented, obtaining repeatable and realistic results using traditional methods would be difficult. In each experiment, three RF nodes were connected to our prototype: "Orchid", "Hermes", and an interferer ("Nice" or a Bluetooth source).

4.1 Hidden Terminal

We first use our prototype to analyze how the "hidden terminal" problem affects 802.11 networks. Evaluating the hidden terminal problem in a real world environment is troublesome since it is difficult to determine if nodes are in carrier sensing range of each other. Moreover, carrier sensing range constantly fluctuates in the real world. This experiment highlights our prototype's ability to overcome these difficulties by providing precise, independent control over the signal paths between all nodes. This allows us to evaluate the hidden terminal problem by simply commanding the emulator to "disconnect" the desired nodes while leaving the communication between other nodes unaffected.

As illustrated in Figure 5 we arranged our three nodes in a line with all nodes in range of each other. (For simplicity we will speak of spatial relationships in our virtual physical environment as if they were based in a real physical environment). We then measured TCP throughput from Hermes to Orchid while Nice was used to generate interfering traffic using a unicast ping flood directed at Orchid.

As shown in the Figure 6 "No RTS, No Interference" test, throughput between Orchid and Hermes is excellent when there is no interference (each value is an average of 25 trials with 95% confidence intervals shown). In the "No RTS, Interference, Not Hidden" test, we see that when Nice begins interfering, throughput is still quite good (ping packets are much smaller than the TCP packets).

We then created a hidden terminal situation by artificially "severing" the link between Hermes and Nice while leaving the other communication paths unaffected. (The ability to create a hidden terminal situation without "moving" the nodes allows us to directly compare results between the hidden and non-hidden tests.) The "No RTS, Interference, Hidden" test shows that throughput between Orchid and Hermes drops dramatically in this case.

We next analyzed the efficacy of 802.11's RTS/CTS mechanism at overcoming the hidden terminal problem by repeating the previous tests with Hermes set to always use RTS/CTS for frames over 200 bytes. The "RTS, Interference, Hidden" test shows that RTS/CTS is able to double throughput; nevertheless throughput is still much lower than when the interferer was not hidden. Comparing the final "RTS, No Interference" test with the "No RTS, No Interference" case shows that the overhead of RTS/CTS alone is roughly 1 Mbps. Further investigation (and coaxial-based verification) revealed that the cause of this underwhelming improvement was the failure of RTS/CTS to prevent rate fallback. The ability to analyze this type of subtle behavior in a controlled environment is a key advantage of our emulator.

4.2 Directional Antennas

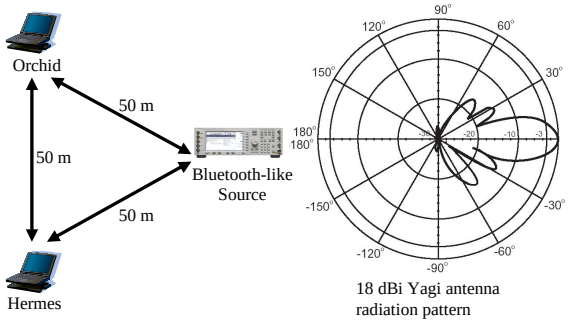


Figure 7. Directional Antenna Topology

Complete control over signal propagation also allows our prototype to emulate arbitrary types of antennas. In this experiment, we analyzed the ability of directional antennas to improve range and spatial reuse by minimizing the effects of interfering Bluetooth-like traffic. As shown in Figure 7, each node was positioned 50 meters from the other two nodes.

Figure 8 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). In the

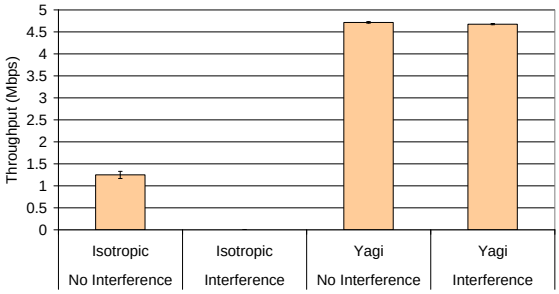


Figure 8. Directional Antenna Results

"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 throughput between Orchid and Hermes is not possible in this case.

The "Yagi" tests then repeat the "Isotropic" tests, but with 18 dBi Yagi antennas [4] attached to Orchid and Hermes. These antennas are aimed directly at each other. Figure 7 shows the radiation pattern for this antenna. 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 main lobe. The results of these tests show that the directional antennas successfully increase communication range and also mitigate the effects of interference.

4.3 Mobility

The mobility experiment illustrates our prototype's ability to easily emulate movement in a physical space, and to coordinate this movement with the transmission of data in a controlled and repeatable manner. Figure 9 contains two screen captures gathered during the operation of this scenario (with some additional explanatory text and illustrations superimposed on the screen captures). The "World Viewer" window on the left is a two-dimensional projection of the emulated physical environment. The window on the right is the Orinoco Link Test utility which performs a link layer ping and reports signal and noise measurements.

As illustrated in the World Viewer window, Orchid and Nice were stationary during this experiment while Hermes moved in a counterclockwise circuit around them. Figure 10 shows the signal strength from Hermes measured by Orchid's wireless card at one second intervals over four circuits, and illustrates our prototype's ability to emulate the effects of mobility at the physical layer in a repeatable fashion. This test also demonstrated our prototype's ability to create an "out of range" situation as demonstrated by the gaps in the measurements recorded by the Orinoco Link Test program when Hermes was most distant from Orchid (Region C).

At two points in this circuit, Orchid measured TCP performance by sending a large amount of data to Hermes. The emulation controller ensured that these measurements were synchronized with node movement and occurred at the same

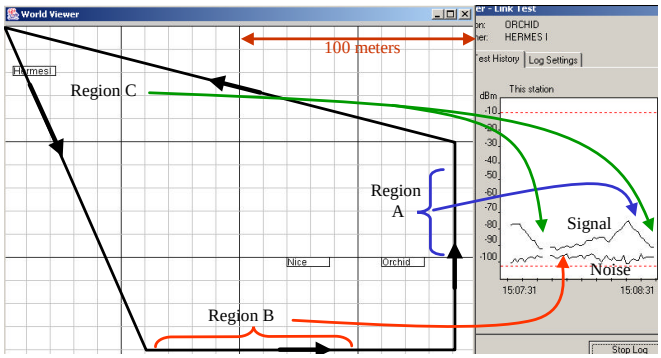


Figure 9. Mobility Scenario Overview

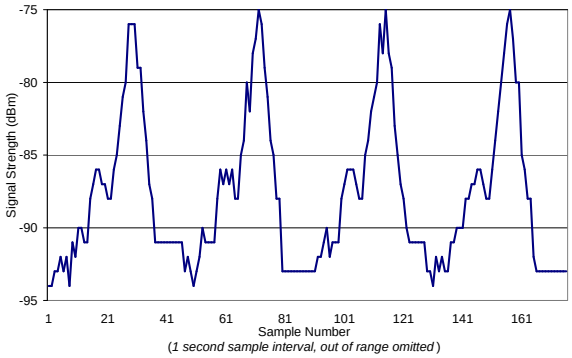


Figure 10. Signal Modeling Repeatability

location each time. The first transmission was initiated when Hermes was near Orchid and the signal strength between the two nodes was strong (Region A). The second test was initiated far away where the signal was weak (Region B). Table 1 shows the results of these tests averaged over 20 circuits. The small standard deviation of the Region A throughput demonstrates that our prototype's control over the physical layer can translate into repeatable TCP performance despite the inherent variability from sources such as the operating system. As the signal strength becomes weaker, however, these higher level factors can produce some variation in results as evidenced by the higher standard deviation seen in Region B. Importantly, the statistical properties of the results are constant.

Location	Average	Std. Dev.
Region A	4.64	0.0343
Region B	1.80	0.225

Table 1. Regional TCP Throughput (Mbps)

5 Related Work

Wireless Simulators. For several years now, ns-2 [5] 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 contains an incorrect value for 802.11 ACK timeout value. In addition, ns-2's physical layer is particularly simple [1]. As a result, some researchers are opting to use commercial simulators such as QualNet [6] and OpNet [7] since they claim better

support for current standards. Despite these claims, however, it is unclear how well these simulators reflect actual hardware.

Wireless Emulators. Emulation has proven to be a useful technique in wired networking research [3, 8, 9], and it has an even larger potential in the wireless domain.

A common approach that has been taken for wireless emulation [10, 11, 12] 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.

More recently efforts have been made to develop RF emulators that accurately emulate down to the physical layer. RAMON [13] 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 and the cost of programmable attenuators severely limits this approach.

[14] proposes extending Emulab into the wireless domain using either fixed wireless nodes scattered in a building or passive carriers to carry wireless nodes. Clearly this approach can achieve a high degree of realism, but has serious obstacles in terms of control and repeatability (since this is essentially a wireless testbed). As there can be significant uncontrolled interference, this approach can only hope to achieve what the authors term "coarse repeatability". In contrast, our approach allows for far greater control and repeatability.

Channel Emulators / Fading Simulators. The most functionally similar approach to the wireless emulator that we are developing is provided by commercial channel emulators [15, 16]. The goal of these emulators, however, is quite different. Rather than supporting emulation of all channels in a wireless network, commercial channel emulators 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). This makes these emulators 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.

6 Conclusion

Understanding and improving wireless network performance is increasingly important. Unfortunately, repeatable experimentation with real wireless nodes operating in a real environment is not feasible. For this reason, most wireless research 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 difficult to validate.

We propose overcoming these obstacles by developing an emulator capable of providing real-time modeling of the wireless communication channel between all pairs of nodes attached to this emulator. This approach maintains much of the realism of wireless testbeds without sacrificing the repeatability of simulation. We have developed a proof-of-concept prototype, and have used this prototype to show that physical layer wireless emulation is a powerful means of enabling realistic and repeatable wireless experimentation.

References

[1] M. Takai and J. Martin. Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks. *Proc. of MobiHoc 2001*, Oct. 2001.

[2] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. *Proc. of HotNets-I*, Oct. 2002.

[3] 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. *Proc. of OSDI 2002*, Dec. 2002.

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

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

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

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

[8] K. Fall. Network emulation in the vint/ns simulator. *Proc. of The Fourth IEEE Symposium on Computers and Communications*, Jul. 1999.

[9] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. "scalability and accuracy in a large-scale network emulator". *Proc. of OSDI 2002*, Dec. 2002.

[10] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz. Trace-based mobile network emulation. *Proc. of SIGCOMM 1997*, Sep. 1997.

[11] P. Mahadevan, K. Yocum, and A. Vahdat. Emulating large-scale wireless networks using modelnet. *Poster and Abstract Mobicom 2002*, Sep. 2002.

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

[13] E. Hernandez and S. Helal. "ramon: Rapid mobility network emulator". *Proc. of the 27th IEEE Conference on Local Computer Networks (LCN'02)*, Nov. 2002.

[14] B. White, J. Lepreau, and S. Guruprasad. Lowering the barrier to wireless and mobile experimentation. *Proc. of HotNets-I*, Oct. 2002.

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

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