# Using IF-Scale Delays to Emulate the Effects of Site-Specific Multipath in a Digital Wireless Channel Emulator *

Jason Matusiak
Gardetto Engineering
Hanover, MD, USA
jason@gardettoengineering.com

Richard Graham
EurekaSound LLC
Sewickley, PA, USA
rick@eurekasound.com

Keith Taylor
University of Maryland
College Park, MD, USA
taylor23@umd.edu

Eric Anderson
Carnegie Mellon University
Pittsburgh, PA, USA
andersoe@ece.cmu.edu

Brenton Walker
Lab for Telecom. Sciences
College Park, MD, USA
brenton@umd.edu

## ABSTRACT

Laboratory-based wireless testbeds are essential for performing repeatable experiments with experimental protocols and devices. Several many-to-many wireless emulators have been proposed and built, but existing systems still rely on simplified channel models, such as free-space path loss, or general stochastic fading models. We have begun using site-specific channel modeling software to drive experiments. In this paper we describe our prototype digital channel emulator with multiple discrete-time taps, and several experiments using a site-specific, 3D ray-tracing channel model, controlling and highlighting the effects of RF multipath. We describe and evaluate a technique for emulating RF-scale interference with discrete delay taps with a resolution on the timescale of IF.

## Categories and Subject Descriptors

I.6.7 [**SIMULATION AND MODELING**]: Simulation Support Systems —*Environments*

## Keywords

wireless emulator, wireless channel model

## 1. INTRODUCTION

When designing a new wireless device or protocol, it is imperative to test it thoroughly. Whereas wired systems can be set up and tested extensively in a lab, wireless systems are much harder to fully exercise; especially mobile ad-hoc network (MANET) or delay-tolerant network (DTN) devices designed to form self-configuring networks over a large geographic area. Testing such nodes requires creating separation between devices, mobility, and dynamic channel conditions. Live field testing is of course the most realistic environment, but field tests are expensive, difficult to choreograph and monitor, and not very repeatable. Having a multi-node wireless testbed that can realistically and repeatably emulate a wide variety of site-specific conditions is very much needed.

Several laboratory-based or localized wireless testbeds have been proposed and built. Some rely on automated mobile or stationary nodes transmitting over the air with either low power or crippled antennas to limit the radio range, and therefore the geographic scale of the experiment; for example ORBIT [15], MINT [5], or PHAROS [12]. These systems make experiments cheaper and easier to manage than full-scale field tests, but they will still have issues with external interference, reproducibility, and realism. Other laboratory-based testbeds take RF-isolated devices and feed them into a larger many-to-many channel emulator. By dynamically adjusting the channel conditions between devices according to some channel model, they have the ability to emulate a wide variety of different environments. A fully analog example of this is MeshTest [4], and full-matrix test systems for this purpose have since become available [8]. A drawback of this approach is that such systems can only control attenuation, and may suffer from unrealistic artifacts due to reflections, leakage, and undesirable multipath in the system. To more realistically emulate the RF conditions of real environments, some experimenters want control over deeper channel characteristics such as multipath, propagation delays, and fading. The Carnegie Mellon University (CMU) wireless emulator is a digital many-to-many wireless testbed which avoids many of the problems of purely analog systems, and potentially provides users with complete control over multipath, signal phase, and propagation delays [9, 1].

Even with such powerful testbeds, experiments can only be as realistic as the channel models used to drive them. MeshTest has used simple variations of free space path loss or two-ray ground-bounce formulas to compute desired attenuations. The CMU emulator supports a variety of loss and fading models and antenna types, but they are gen-

*This work was funded by the Laboratory for Telecommunications Sciences, US Department of Defense. The opinions expressed in this paper reflect those of the authors, and do not necessarily represent those of the Department of Defense or US Federal Government.


*WiNTECH'13,* September 30, 2013, Miami, Florida, USA.

http://dx.doi.org/10.1145/2505469.2505474.

eral stochastic models, only dependent on inter-node distances. In [20] the authors use the CMU emulator to perform environment-specific channel modeling; they have a set of tunable channel and fading models for different environments, and they choose a model based on the nodes' environment. However this is still not completely site-specific, as the channel model does not take into account the specific arrangement of objects and surfaces surrounding the nodes.

Ray tracing is a method of channel modeling where the individual paths that a signal can take, including reflections and possibly diffractions and transmissions through solid objects, are computed. Given a physical model of an environment, ray tracing produces a list of rays between the transmitter and receiver, along with signal power, propagation delay, phase on arrival, and field orientation. In [19] the author uses ray tracing to model an urban environment and drive an attenuator-based testbed. Because those experiments used a single-path attenuator-based system, the individual rays had to be combined using a mathematical receiver model to give a single path loss prediction. However the fully-realized CMU emulator DSP engine gives us control over multiple paths and propagation delays, so we are able to take advantage of all the information the ray tracing model provides.

Using ray tracing information to drive the CMU emulator still presents several challenges, though. The main problem we address in this paper is the fact that the sample rate, and therefore the time resolution of the delay line, is much coarser than the period of a 2.4GHz WiFi signal. Since the RF signal will cover several complete cycles between each sample, we cannot directly capture realistic RF phase-based interference using only discrete taps. Our approach is to substitute IF-scale phase interference in the down-converted sampled signal for the expected RF interference.

In this work we use prototype components of a next-generation CMU emulator and site-specific ray tracing software with custom 3-D models of the University of Maryland (UMD) campus to perform highly detailed emulated mobile wireless experiments. We highlight the main challenges to running realistic experiments and describe the solutions we developed. We report results for a number of experiments: arbitrary multi-tap experiments verifying the behavior of the system, simple 2-4 ray mobile scenarios, and more complex mobile experiments based on a detailed model of the UMD campus.

## 2. BACKGROUND

In this section we provide some background on ray tracing channel models and the CMU wireless emulator. This will make clear why ray tracing models are ideal for use with the CMU Emulator, and vice versa, and also explain the challenges to implementing this realistically.

### 2.1 Ray Tracing

Ray tracing is a method of channel modeling which attempts to explicitly identify the exact propagation paths through a specific environment. This approach is purely physics-based, treating electromagnetic waves as rays that propagate according to the laws of geometric optics. Individual rays are cast from a transmitter, undergoing reflections, transmissions, and diffractions computed based on the Geometric Theory of Diffraction [10] before reaching a receiver. This requires a detailed description of the objects in an environment, including positions, shapes, and material properties. The received signal is calculated by summing the contribution of each individual ray that reaches the receiver. This approach goes beyond calculating path loss, providing more detailed information of each path such as phase, direction of arrival, and time of arrival. These methods are used to evaluate the electric field strength and make predictions of signal characteristics in a site-specific environment, allowing for immense customization of environmental detail. Signal predictions can be done in both real world representations and hypothetical environments. In contrast, empirical path-loss models are ultimately derived from and validated against specific data sets, and are generally applicable only to environments and wavelengths similar to the ones present in that data. Ray tracing can, in principle, be applied to *any* environment, so long as the relevant properties are known.

Ray tracing techniques have long faced criticism for their accuracy-sensitive coverage mapping due to large computation and data requirements [13]. The inclusion of reflected, transmitted, and diffracted ray interactions during propagation in realistic 3D environments often results in complex and slow calculation procedures, especially compared to 2D environments. Much recent work has focused on optimization and preprocessing while maintaining accuracy [7]. Along with advances in ray tracing methods, computation times have been greatly reduced by the progression of high performance hardware including current multi-core processors and GPGPUs.

Although computational requirements have become less of an obstacle, the accuracy of ray tracing is tightly coupled to the amount of detail in the modeled environment. Accurate predictions require a precise description of the buildings, terrain, and other objects including their shape, position, elevation, and material properties (permittivity and conductivity). General building footprints, positions, and elevations can be obtained from aerial photography or LIDAR (Light Detection and Ranging) and can be expensive, but many digital databases are available providing this data. However, since it is uncertain how accurate this data is and the material properties of each surface may not be known, using this data in a ray tracing environment can only provide approximations for RF signal characteristics. Near perfect accuracy would require complete knowledge of an environment on the order of the shortest RF wavelength, detail which is difficult and time-consuming to gather. Total accuracy would include the impossible task of modeling all transient objects.

Despite having large data requirements, many researchers have shown that accurate results can be obtained even with generalized environment details. A series of studies [16, 14, 3] comparing ray-traced estimates with direct measurements have found mean errors between 1 and 7 dB and standard errors between 1 and 8 dB. For this paper we use Remcom's EM wave propagation software suite, Wireless InSite (WI) [18], to model both very simplified idealized models with one and two reflective planes, and a real-world UMD campus model.

### 2.2 CMU Wireless Network Emulator

The CMU wireless emulator is a digital many-to-many channel emulator. The output signal from each node is attenuated, mixed to a lower frequency, digitized, processed through a DSP engine implemented in an FPGA, and regenerated and mixed back up to RF at each output. The

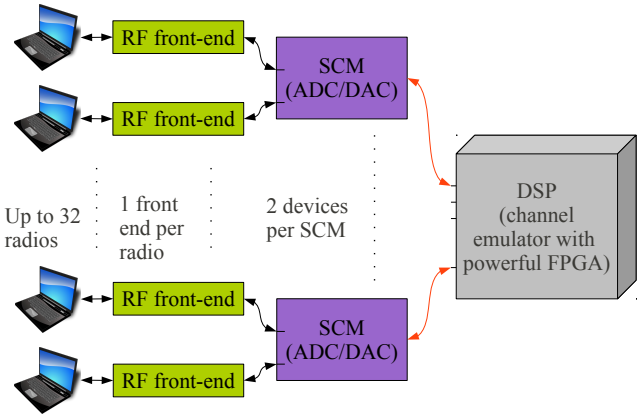


**Figure 1: Block diagram of the main components of the CMU emulator**

CMU emulator has gone through several iterations; the system we describe here is similar to that described in [1] but the specific details apply to a prototype "gen-3" system [21].

The rough block diagram of the emulator is shown in figure 1. The RF front end has a transmit/receive switch that toggles when it detects transmit energy from the radio, and handles mixing of the signal down/up to/from some intermediate frequency (IF) that can be digitized. The Signal Conversion Module (SCM) has A/D and D/A converters to digitize and re-generate the signals, and a high-speed optical interface to communicate these samples with the main FPGA on the DSP board. The prototype DSP engine implements a discrete-time tapped delay line as shown in figure 2. 256 of the most recent signal samples are kept in a rotating buffer, and "taps" corresponding to different signal delays can be placed at arbitrary offsets in the buffer, with real-valued coefficients. For a given input/output pair, up to four taps can be placed, and during each sample period the taps are summed, and the signal is regenerated and mixed back up to RF at the receiving node.

Each SCM has a small FPGA (Xilinx Virtex-6 LX75T), which is sufficient to implement the DSP for 4x4 18-tap channels. We use a single SCM as a self-contained two-node emulator for the experiments described in this paper.

To describe the DSP implementation, let $x_i(t)$ be the sample recorded $t$ time steps ago at input $i$. Let $a^k_{i,j}$ be the coefficient of the $k$-th tap on input $i$ to output $j$, and let $\delta^k_{i,j}$ be the time delay (in units of the sample rate) of that tap. Then the output signal, $y_j$, at output $j$ is given by:

$$y_j = \sum_{i \neq j} \sum_{k=1}^{K_{i,j}} a^k_{i,j} \; x_i(\delta^k_{i,j}) \qquad (1)$$

where $K_{i,j}$ is the number of taps used between input $i$ and output $j$.

This time-domain DSP engine is almost exactly what we need to take advantage of the information produced by a ray-tracing channel model. For each of the $K_{i,j}$ strongest rays between transmitter $i$ and receiver $j$, we can place a tap with time delay $\delta_{i,j}$ and scale coefficient $a_{i,j}$. It was observed in [19] that for moderately complex campus environments, including four or fewer rays in a ray tracing model produced sufficiently accurate predictions. Therefore the four taps we have available in our prototype should be sufficient to emulate real environments realistically.

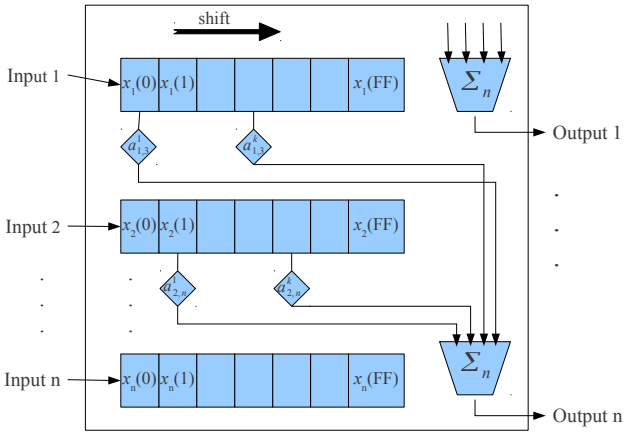


**Figure 2: Schematic of the time-domain channel model implemented in the CMU emulator.**

The time delay corresponding to each sample in the buffer is determined by the sample rate of the A/D converter; in our case 200MHz, so each time step is 5ns. This corresponds to about 1.5m of propagation distance. However the wavelength of a 2.4GHz signal is on the order of 10cm, so this system does not directly give us the ability to model completely arbitrary multipath.

## 3. ADJUSTING DELAY FOR IF PHASE

To achieve realistic multipath emulation, we have to take into account the power, delay, and phase of each ray as it arrives at the receiver. The CMU emulator gives us control over the power and the delay. At each time step of the experiment, the power and delay of the strongest rays in the ray tracing prediction can be used to set a tap in the DSP engine. The phase is entirely a function of the delay, modulo a 180° phase inversion for each reflection. A 180° phase inversion can be affected by changing the sign of the scale coefficient on a tap.

There are two main obstacles to using ray tracing predictions with the emulator the way we have described. First, for short distances, the inherent processing latency of the emulator may be greater than the actual time of flight of the ray, and the range of the delays may cover a time span longer than our sample buffer can hold. Second the time resolution of the samples, and therefore the discrete taps on the sample bins, is several times longer than the period of the RF signal. This is illustrated in figure 3. During the time spanned by one bin, the RF carrier will go through several complete cycles. During the time spanned by multiple bins, the actual RF carrier's phase advances constantly during that real time. This RF phase advance is, in general, different than the phase advance of the digitized IF; an artifact of delay and the RF, IF, and sample rate frequencies.

To address the first issue, we choose here to only deal with relative delays, and note that the time spanned by the 256 sample buffer is over 1200ns, or a path length *difference* of almost a quarter mile. We implement the relative delays by always placing the first tap of a given input/output pair on the first (most recent) sample bin, and arranging all the other later taps relative to that. This ignores the absolute time delay of a path, so this implementation would not be effective for nodes trying to use time-of-flight measurements

to determine their relative locations. It would also be an issue for nodes that try to synchronize their transmissions to focus their combined energies on a receiver [2].

The ideal solution to the second problem is to implement fractional delay filters [11]. A fractional delay filter applies a function to several consecutive buffer samples in order to effectively reconstruct (interpolate) a sample at an arbitrary intermediate time. Fractional delay filters would be ideal, but there are several reasons why they might not be practical. First, they add to the processing latency of the emulator; a 10-stage filter will require 10 FPGA clock cycles to compute, and some emulators may not be able to support the computational load, especially in a many-to-many emulator where the number of cross-connections grows quadratically. Second, the FPGA may not have enough multipliers to support the computation based on several consecutive taps. This was the case in the gen-2 CMU emulator, where running any multi-tap experiment required reducing the system to only three inputs. Finally, fractional delay filters will require an overhaul of our prototype DSP engine and the control protocol implemented in the FPGA and system control software.

The main contribution of this paper is to propose, analyze, and experiment with a technique for emulating phase-realistic multipath interference using only discrete taps. We compute the expected RF phase of each ray, relative to the first ray, and then allow the delay of the tap for that ray to be adjusted slightly so that it taps a bin which is close to the correct phase at IF. That is, we substitute IF-scale interference for RF-scale interference. This technique is not ideal, but we have found that based on the metrics of signal strength and throughput, that it achieves realistic results, and captures multipath effects that would not be possible by manipulating attenuation alone.

## 3.1 Procedural Description

The process is identical, and independent, for all pairs of inputs and outputs, so this discussion will simply refer to one "input" and one "output". The following procedure is implemented in a Java program, `RayParser` that reads in complex impulse response (CIR) predictions from WI and sends configuration commands to the emulator.

Let $x(t)$ be the sample digitized at the input at time $t$. Let $B$ be the sample buffer with $M$ entries, $B[0], \ldots, B[M-1]$. We call the location $B[i]$ the $i$th "bin". At time $t$, $B[i]$ holds the sample $x(t-i)$. Let $R_k = (\delta_k, a_k, \phi_k)$, $k = 1, \ldots, K$, be the $k$th ray predicted between the transmitter and the receiver. Each ray consists of: $\delta_k$ the time of flight, $a_k$ the path loss (on a linear scale, not in logarithmic/dB), and $\phi_k$ the phase on arrival. Let $f_{\text{samp}}$ be the sampling frequency of the system (in our case 200MHz), let $f_{\text{RF}}$ be the carrier frequency of the original signal, and let $f_{\text{IF}}$ be the carrier frequency of the signal after downconversion. Here we assume that $f_{\text{IF}}$ divides $f_{\text{samp}}$, so $D = f_{\text{samp}}/f_{\text{IF}}$ is an integer. This simplifies delay line phase calculations greatly.

Assume the rays are sorted in order of arrival, so that $\delta_k < \delta_{k+1}$. We start by converting the rays to taps. Let $\tau_k = (i_k, a_k)$ be the tap corresponding to the $k$th ray. It consists of a bin index, $i_k$, and the scale coefficient of the ray, $a_k$. The scale coefficient is just the path loss of the $k$th ray. The bin index is computed as $i_k = \text{round}((\delta_k - \delta_0)/f_{\text{samp}})$. This always places the first tap on bin 0, and all successive taps are placed relative to the first tap.

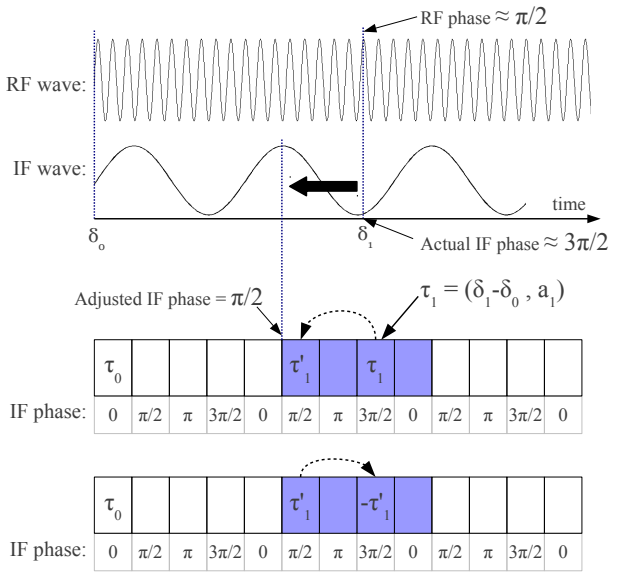


**Figure 3: Adjusting the delay of tap $\tau_1$ to obtain the desired phase at IF, relative to the first tap, $\tau_0$.**

We then adjust all taps $\tau_k$ for $k > 0$ to select for IF phase, relative to $\tau_0$. Recall that $\phi_k$ is the phase at RF of the $k$th arriving ray. Each bin $B[i]$ has associated with it an IF phase $\theta_i = 2i\pi/D \pmod{2\pi}$ relative to $B[0]$. Note that there are $D$ distinct IF phase values available. For each $\tau_k = (i_k, a_k)$ we select the closest $\tau'_k = (i'_k, a_k)$ that minimizes $|\theta_{i'_k} - \phi_k|$. More specifically:

- Choose $\theta = \underset{\theta \in \{\theta_0, \ldots, \theta_{D-1}\}}{\text{argmin}} |\theta - \phi_k|$
- Set $i'_k = \underset{i \in [0,M] \,:\, \theta_i \in \{\theta, \theta \pm \pi\}}{\text{argmin}} |i - i_k|$
- Set $\tau'_k = (i'_k, a_k)$ if $\theta_{i'_k} = \theta$,
  or $-\tau'_k = (i'_k, -a_k)$ if $\theta_{i'_k} = \theta \pm \pi$

Note that we are using the fact that by inverting the scale coefficient, $a_k$, we are effectively rotating the phase of the tap by $\pi$. It is possible that there is an $i$ with $\theta_i = \theta \pm \pi \pmod{2\pi}$ closer to $i_k$ than any $i$ with $\theta_i = \theta$. This can reduce the absolute delay time offset error of the selected tap.

For example, suppose $f_{\text{RF}} = 2462$MHz, $f_{\text{samp}} = 200$MHz, $f_{\text{IF}} = 25$MHz, and two rays: $R_1$ arriving in 10ns, and $R_2$ arriving in 69ns (a delay of 59ns and phase of 92.88° relative to $R_1$). $R_1$ would be placed in the first tap with a delay of zero. $R_2$ would be tentatively placed in tap 2 with a delay of 12 clocks (a 60ns offset from $R_1$), but would be adjusted to a delay of 10 clocks so that it would be 90° out of phase (at IF) with $R_1$.

One drawback of this simplified technique is that the $f_{\text{IF}}$ must be chosen based on $f_{\text{RF}}$ and $f_{\text{samp}}$. This means the prototype testbed only supports one RF carrier at a time. In our case, changing the WiFi channel requires re-tuning the $f_{\text{IF}}$. Another drawback of this technique, compared to using fractional delay filters, is that it is not completely realistic. That is, the effects on the received power show the expected phase-based interference, but the actual RF seen at the receiver is not exactly the same as combining individual paths with the actual phases and delays.

## 3.2 Analytical Description

The act of combining time-domain taps can be written analytically as follows. We explain it for two taps, but it easily generalizes to more. Let $A_0$ be the amplitude of the original signal, let $m(t)$ be the modulated message at baseband, and let $\omega_c$ and $\omega_{lo}$ be the RF carrier and local oscillator (LO) frequencies (in Hz, so $\omega_* = 2\pi f_*$) respectively. Then the analytic representation of the transmitted signal is

$$x_0(t) = A_0 e^{i\omega_c t} m(t) \tag{2}$$

Then the RF signal seen at the receiver after a time delay of $\tau_1$ is

$$y_1(t) = A_1 e^{i\omega_c(t-\tau_1)} m(t-\tau_1) \tag{3}$$

where $A_1$ is the received amplitude of that path. When two distinct rays reach the receiver with delays $\tau_1$ and $\tau_2$ the resulting RF signal at the receiver is

$$\begin{aligned} &y_1(t) + y_2(t) \\ &\quad = A_1 e^{i\omega_c(t-\tau_1)} m(t-\tau_1) + A_2 e^{i\omega_c(t-\tau_2)} m(t-\tau_2) \end{aligned} \tag{4}$$

In the testbed we start by mixing the original signal to IF. This corresponds to multiplying by $e^{-i\omega_{lo} t}$

$$\begin{aligned} \widetilde{x_0}(t) &= A_0 e^{-i\omega_{lo} t + i\omega_c t} m(t) \\ &= A_0 e^{i(\omega_c - \omega_{lo})t} m(t) \end{aligned} \tag{5}$$

Then a delayed and attenuated copy of the down-converted IF signal looks like

$$A_1 e^{i(\omega_c - \omega_{lo})(t-\tau_1)} m(t-\tau_1) \tag{6}$$

This is different from a down-converted copy of the delayed RF signal. It requires a phase correction by a constant $\phi_1 = e^{-i\omega_{lo}\tau_1}$ to give:

$$A_1 e^{i(\omega_c - \omega_{lo})t - i\omega_c\tau_1} m(t-\tau_1) \tag{7}$$

If we add together two phase-corrected down-converted paths we get

$$e^{i(\omega_c - \omega_{lo})t} \left( A_1 e^{-i\omega_c\tau_1} m(t-\tau_1) + A_2 e^{-i\omega_c\tau_2} m(t-\tau_2) \right) \tag{8}$$

and multiplying by $e^{i\omega_{lo} t}$ mixes back up to the carrier frequency

$$\begin{aligned} &A_1 e^{i\omega_c(t-\tau_1)} m(t-\tau_1) + A_2 e^{i\omega_c(t-\tau_2)} m(t-\tau_2) \\ &\qquad = y_1(t) + y_2(t) \end{aligned} \tag{9}$$

With arbitrary control of the delays, $\tau_i$, and the ability to make arbitrary phase corrections, then delays applied to the IF signal could be made to produce the correct combined signal at RF. Unfortunately, with our prototype, we only have control of the *delays* on the scale of IF.

We can express the delay-selection process analytically as follows. For clarity let $\omega_{if} = \omega_c - \omega_{lo}$ be the IF. Also, without loss of generality and to simplify the equations, we will treat the first copy of the signal as having delay 0 (and therefore phase 0), and will only have to deal with one delay value, $\tau$. For frequency $\omega_c$ and delay $\tau$, let $\phi_c(\tau)$ be the phase of a signal with frequency $\omega_c$ after a delay of $\tau$. That means that $e^{i\omega_c\tau} = e^{i\omega_c\phi_c(\tau)}$ Similarly, let $\phi_{if}(\tau)$ be the phase of a signal with frequency $\omega_{if}$ after a delay of $\tau$.

We start by considering both signals as pure analytic signal tones; that is, a complex exponential with no modulated data. Then the signal arriving with the first ray has the form $A_0 e^{i\omega_c t}$, and the delayed attenuated signal looks like $A_1 e^{i\omega_c(t-\tau)}$. Summed at the receiver they appear as

$$\begin{aligned} A_0 e^{i\omega_c t} + A_1 e^{i\omega_c(t-\tau)} &= e^{i\omega_c t}\left(A_0 + A_1 e^{-i\omega_c\tau}\right) \\ &= e^{i\omega_c t}\left(A_0 + A_1 e^{-i\omega_c\phi_c(\tau)}\right) \end{aligned} \tag{10}$$

Note that the factor $(A_0 + A_1 e^{-i\omega_c\tau})$ is now just a complex constant multiplying the tone $e^{i\omega_c t}$. Then the two down-converted rays look like $A_0 e^{i\omega_{if} t}$ and $A_1 e^{i\omega_{if}(t-\tau)}$ and summed together we have

$$\begin{aligned} A_0 e^{i\omega_{if} t} + A_1 e^{i\omega_{if}(t-\tau)} &= e^{i\omega_{if} t}\left(A_0 + A_1 e^{-i\omega_{if}\tau}\right) \\ &= e^{i\omega_{if} t}\left(A_0 + A_1 e^{-i\omega_{if}\phi_{if}(\tau)}\right) \end{aligned} \tag{11}$$

Since we cannot control $\phi_c(\tau)$ on the scale of $\omega_c$, in order to emulate the desired signal (eqn 10) we need to choose an alternate delay $\tau'$ such that

$$A_0 + A_1 e^{-i\omega_c\phi_c(\tau)} \approx A_0 + A_1 e^{-i\omega_{if}\phi_{if}(\tau')} \tag{12}$$

or essentially

$$\omega_c\phi_c(\tau) \approx \omega_{if}\phi_{if}(\tau') \tag{13}$$

This corresponds to selecting a delay whose phase at IF matches the desired phase at RF of the actual delay. Because the wave is periodic there are many such $\tau'$ to choose from. The constraint on *which* $\tau'$ comes from the need to *not* decorrelate the signals modulated on the RF carrier.

With a modulated signal, $m(t)$, the sum of the two rays at the carrier frequency looks like

$$\begin{aligned} &A_0 e^{i\omega_c t} m(t) + A_1 e^{i\omega_c(t-\tau)} m(t-\tau) \\ &\quad = e^{i\omega_c t}\left(A_0 m(t) + A_1 e^{-i\omega_c\tau} m(t-\tau)\right) \\ &\quad = e^{i\omega_c t}\left(A_0 m(t) + A_1 e^{-i\omega_c\phi_c(\tau)} m(t-\tau)\right) \end{aligned} \tag{14}$$

and the sum of the down-converted rays at IF looks like

$$\begin{aligned} &A_0 e^{i\omega_{if} t} m(t) + A_1 e^{i\omega_{if}(t-\tau)} m(t-\tau) \\ &\quad = e^{i\omega_{if} t}\left(A_0 m(t) + A_1 e^{-i\omega_{if}\tau} m(t-\tau)\right) \\ &\quad = e^{i\omega_{if} t}\left(A_0 m(t) + A_1 e^{-i\omega_{if}\phi_{if}(\tau)} m(t-\tau)\right) \end{aligned} \tag{15}$$

Again, we wish to find a $\tau'$ such that for all $t$:

$$\begin{aligned} &A_0 m(t) + A_1 e^{-i\omega_c\phi_c(\tau)} m(t-\tau) \\ &\quad \approx A_0 m(t) + A_1 e^{-i\omega_{if}\phi_{if}(\tau')} m(t-\tau') \end{aligned} \tag{16}$$

Or at least so that the effect on the matched filter at the receiver is the same. In general we have reasoned that the best approximation will be choosing $\tau'$ as close as possible to the original $\tau$, so that the effect on the matched filter in the receiver will be similar to that of the desired signal (eqn 14). If we choose the closest tap that matches the desired phase at IF and take advantage of the 180° phase inversion achieved by inverting a tap's scale parameter, we will never have to move a tap more than 1/4 of an IF period relative to the first tap. On the other hand, if there are more than two taps, two non-first taps may move up to 1/2 IF period relative to each other.

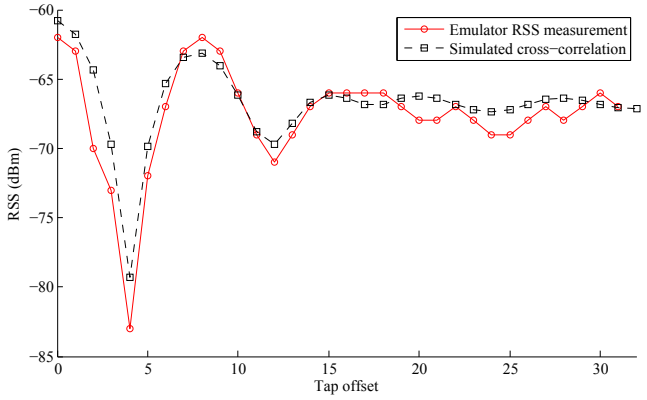


**Figure 4: Simulated cross-correlation of the 802.11 DBPSK DSSS PLCP Preamble 128-bit SYNC and experimental signal strength measurements of a two-tap emulator experiment with different tap offsets.**

In reality the effect of selecting delay based on phase will depend intricately on the signal, $m(t)$. In our case, 802.11b beacons are sent with a DBPSK signal spread by an 11MHz DSSS code [6]. Figure 4 shows the simulated cross-correlation of the code at different discrete offsets corresponding to the emulator settings used in our experiments. In our case a 1/4 period of IF corresponds to two bins, so we never have to adjust any tap to more than two bins away. However, the plot shows that for rays closer than six bins apart, a shift of as little as two bins can have a large effect on inter-symbol interference. At longer delay differences the symbols are decorrelated, and adjusting taps has little effect.

# 4. EXPERIMENTS AND RESULTS

Most of our results are from received signal strength (RSS) measurements from commodity WiFi cards, however we also performed simpler experiments using just a pure sine wave at IF as a source. All of our WiFi experiments were performed using Atheros AR5424 802.11a/b/g WiFi cards on Dell Latitude R6500 laptops running Ubuntu 10.04 with ath5k drivers. The kernel drivers were modified slightly as described in [19] to eliminate some measurement irregularities.

RSSI measurements are based on the power received in WiFi beacons that are transmitted at approximately 51.255ms intervals, resulting in approximately 19.51 measurements per second. All experiments were done on 802.11b WiFi channel 11, corresponding to a $f_{\mathrm{RF}}$ center frequency of 2462 MHz, which was mixed down to a $f_{\mathrm{IF}}$ of 25 MHz by using a $f_{\mathrm{LO}}$ of 2437 MHz.

A difficulty with comparing the WI model RSS predictions to the emulator-based experiments is that both experiments and predictions exhibit rapid and dramatic fluctuations in RSS at successive points in space. We would like a metric that reveals that some experiments capture these fluctuations better than others, even though the average absolute signal strength is about the same. We achieve this by computing root mean squared error (RMSE) over the time series averaged over increasingly larger rolling windows. Figure 5 shows the RMS error when both the reference (model) and experiment results are smoothed by averaging over increasingly larger spatial windows.

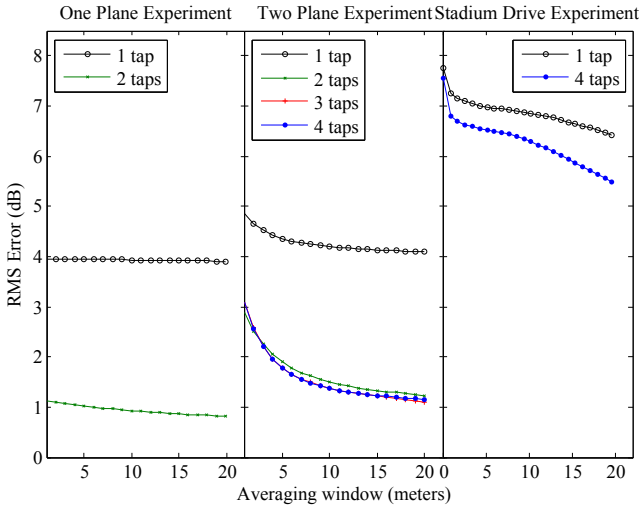


**Figure 5: Emulated channel gain error (RMSE) as a function of spatial resolution.**

## 4.1 Simple Two-Tap Tests

In our first two-tap test we injected an 25MHz unmodulated sine wave directly into the A/D of the SCM. We routed the digitized signal to the first tap of the digital delay line and then sequentially stepped a second tap while monitoring the summed output on a spectrum analyzer. Since our sample frequency (200 MHz) is exactly eight times the input frequency (25 MHz), there were exactly eight possible phase orientations. These eight phases allowed us to completely null the output signal when the 2nd tap had a delay relative to the first tap of 4, 12, 20, 28, etc., since these all represent a 180° shift of the input sine wave. Selecting tap 2 delays of 0, 8, 16, 24, etc., added the input sine wave in precise phase alignment and we observed a 6 dB increase in output power (doubling the output amplitude quadruples the output power). Other tap 2 delays generated the expected sums $\mathrm{Re}(e^{i\frac{\pi}{4}k} + e^{i(\frac{\pi}{4}k - \frac{\pi}{4}\delta)})$, where $k$ represents the increasing index of the sine wave samples marching through the delay line and $\delta$ represents the tap 2 setting.

The next two-tap experiment used a down-converted and attenuated WiFi access point transmitting IEEE 802.11b beacons as described above. The digitized samples were routed to the SCM delay line and two taps with equal scale coefficients were used. The first tap was set at the first bin (delay of zero) and the second tap was varied from a delay of zero samples to 31 samples (155 ns). The taps were digitally summed, converted back to analog, and then up-converted to the original RF frequency to provide an input to another WiFi enabled laptop computer so that RSS measurements could be recorded. At each step, we let the WiFi cards settle and we measured the receiver's reported RSSI value.

As we stepped through the delays, the beacon signals exhibited both constructive and destructive effects due to the time and phase differences between the digitally delayed signal (tap 2) and the non-digitally delayed signal (tap 1). Referring to figure 4, a large combined power null occurs when the two taps are separated by four clocks (20 ns) since that corresponds to a 180° phase shift of the IF carrier and the approximately 90.9 ns DSSS chips still have significant overlap. With 200 MHz sample rate, each IEEE 802.11b DSSS chip duration is about 18.2 samples. When tap 2 is de-

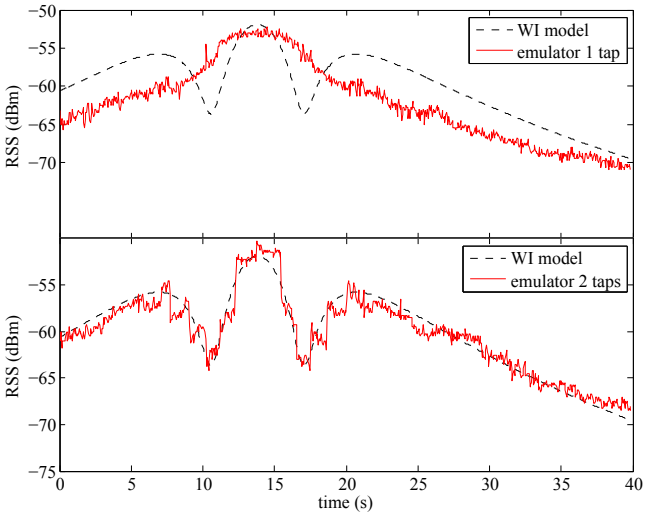


**Figure 6: RSS measurement for the simplified scenario with a single reflective plane in the emulator with 1 and 2 taps, vs the model.**

layed zero or eight samples, the IF carriers are completely in phase and increased RSSI power is observed. Starting at 12 sample delays, the time delay is long enough that the DSSS chip overlap diminishes and the two delayed signals begin to decorrelate. After a long enough delay, the receiver synchronizes to just one of the two summed signals and the beacon's RSSI power levels off.

For comparison, we used Sage [17] to simulate the randomized IEEE 802.11 DSSS PHY [6] 128-bit length sync at our 25 MHz IF and our $f_{\text{samp}}$ and correlated the reference against a two-tap experiment where the spacing between the taps was stepped one buffer delay step at a time. This is also plotted with the experimental results in figure 4. Our emulator results match the predicted correlation well. We note that the magnitude of the nulls and peaks in the emulator data do not exactly match the simulation data. These small differences will be investigated further.

## 4.2 1 and 2 Plane Ray Tracing Experiments

In these tests we used WI ray tracing predictions from a custom mobile scenario. The scenario had a TX antenna 2.44m high, transmitting at 2462MHz, and a series of receivers at the same height along a straight path; all antennas were omni-directional. The ground had "wet earth" characteristics and there were no other obstructions. The array of receivers were spaced 1m apart with a total length of 775m (776 points). The first receiver was roughly 280m from the transmitter, the last was roughly 512m, and the closest receiver was roughly 76m. With this setup, there were only two rays: the direct line of sight (LOS) ray, and a single ground-bounce ray. We ran one experiment using both rays, and a second experiment with only the strongest (LOS) ray. Our control program, `RayParser`, reads WI CIR output files, computes SCM tap parameters, and transmits these to the SCM at a rate of one update every 205ms. At this update interval we expect to see four beacons for each update. We record the received power of the beacons on a receiver laptop and take the average value of every four RSSI beacons received. Figure 6 shows the results of this experiment for both one and two taps compared to the WI-predicted receiver power level. The plot shows that using just one tap roughly approximates predicted power. This

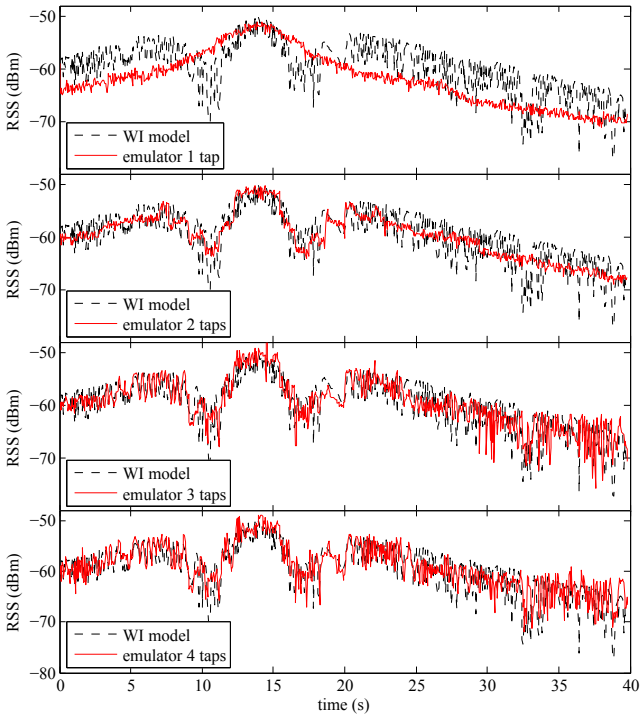


**Figure 7: RSS measurement for the simplified scenario with two reflective planes in the emulator with 1, 2, 3, and 4 taps, vs the model.**

pattern is expected since power of the LOS ray is roughly predicted by free space path loss. Including the second tap captures the prominent interference pattern caused by the ground-bounce ray.

In a second test we added a large, 2km high, 2km long metal wall to the WI model. The wall was positioned perpendicular to the ground, but at an angle to the receiver array. With this second plane there are four possible rays between the transmitter and each receiver: LOS, ground-bounce, wall-bounce, and a ground/wall bounce combination. Figure 7 shows the predicted power compared to our experimental results for 1, 2, 3, and 4 taps. We observe that although two taps match the expected RSS much better than one tap, not much is gained by adding in the third and fourth tap. This seems in agreement with the results in [19], where the author observes that in their real-world urban experiments, WI predictions with relatively few rays gave the best agreement with field test results. These few rays were the highest power chip-time correlated rays.

## 4.3 Real-world Environment Ray Tracing Tests

Our final experiment was based on a real-world urban model. 3D models of a portion of the UMD campus were imported into WI and a scenario was set up with a stationary transmitter, and a receiver that walked past it (see the "Stadium Drive" experiments in [19]). The receiver walked 60 meters in 75 seconds and went from a non-LOS situation, to having LOS (30m apart), to not having LOS again. The building models were realistic and to scale, based on actual CAD drawings of campus. They were modeled as brick surfaces, since that was the dominant material, and the ground was modeled as "wet earth." Running the emulator similarly to above, we logged received WiFi beacons and plotted them against the WI predicted values for one tap and three taps. As can be seen in figure 8, having only the strongest

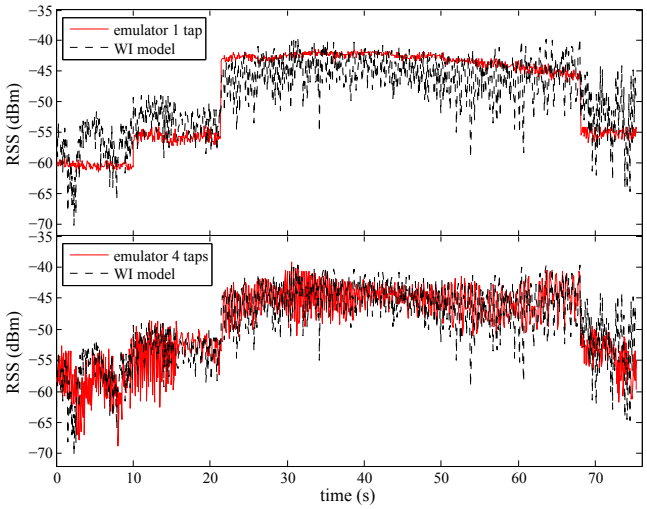


**Figure 8: RSS measurement for the Stadium Drive scenario in the emulator with 1 and 3 taps, vs the model.**

ray produced a signal that mimics the general behavior of the model prediction, but with minimal dynamics. When three taps are included, the signal starts to exhibit some of the multipath fading that was predicted in the real world model. We did not produce the magnitude of fluctuations that the WI model predicted. This will be further studied, but it is most likely due to WI's multi-ray power aggregation algorithms not being specifically tailored to the WiFi signal correlation properties.

## 5. CONCLUSION AND FUTURE WORK

Using the CMU wireless emulator in conjunction with site specific ray tracing software, we were able to implement a four tap multipath wireless emulator system for more realistic fading results. Due to time-resolution limitations in our system, we introduced a method for substituting IF phase interference for RF interference, and showed analytically and experimentally that the system produces quite reasonable results, and that they are much more accurate than using the LOS ray alone. In the future we will investigate fractional delay filters, produce and refine more real-world models and compare the site-specific emulator results to real-world field tests.

## 6. REFERENCES


[1] K. C. Borries, G. Judd, D. D. Stancil, and P. Steenkiste. FPGA-based channel simulator for a wireless network emulator. In *VTC Spring*, 2009.

[2] Y. J. Chang, M. Ingram, and R. Frazier. Cluster transmission time synchronization for cooperative transmission using software-defined radio. In *IEEE ICC*, 2010.

[3] Z. Chen, A. Delis, and H. L. Bertoni. Radio-wave propagation prediction using ray-tracing techniques on a network of workstations. *J. Parallel Distrib. Comput.*, 64(10):1127–1156, Oct. 2004.

[4] T. C. Clancy and B. D. Walker. Meshtest: Laboratory-based wireless testbed for large topologies. In *IEEE TridentCom 2007*, pages 1–6, 2007.

[5] P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: A miniaturized network testbed for mobile wireless research. In *INFOCOM*, 2005.

[6] IEEE Standard 802.11-2012 (Revision of IEEE Std 802.11-2007), 2012.

[7] M. F. Iskander and Z. Yun. Propagation prediction models for wireless communication systems. *Microwave Theory and Techniques, IEEE Transactions on*, 50(3):662–673, 2002.

[8] JFW Industries. 50pma-055 manual. Technical manual, `http://www.jfwindustries.com/documents/50PMA-055_manual.pdf`.

[9] G. Judd and P. Steenkiste. Repeatable and realistic wireless experimentation through physical emulation. *SIGCOMM Comput. Commun. Rev.*, 34(1):63–68, 2004.

[10] J. Keller. Geometric theory of diffraction. *Journal of the Optical Society of America*, 62:1448–1461, 1974.

[11] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine. Splitting the unit delay: Tools for fractional delay filter design. *Signal Processing Magazine, IEEE*, 13(1):30–60, 1996.

[12] A. Petz, C.-L. Fok, and C. Julien. Experiences using a miniature vehicular network testbed. In *VANET '12*, VANET '12, 2012.

[13] C. Phillips, D. Sicker, and D. Grunwald. Bounding the Practical Error of Path Loss Models. *International Journal of Antennas and Propagation*, 2012:1–21, 2012.

[14] L. Piazzi and H. L. Bertoni. Achievable accuracy of site-specific path-loss predictions in residential environments. *Vehicular Technology, IEEE Transactions on*, 48(3):922–930, 1999.

[15] 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 *IEEE Wireless Communications and Networking Conference, 2005*, volume 3, pages 1664–1669. IEEE, 2005.

[16] K. Rizk, J.-F. Wagen, and F. Gardiol. Two dimensional ray tracing modeling for propagation prediction in microcellular environments: Ieee transactions on vehicular technology. *IEEE Transactions on Vehicular Technology*, 46:508–518, 1997.

[17] W. Stein et al. *Sage Mathematics Software (Version 5.10)*. The Sage Development Team, 2013. `http://www.sagemath.org`.

[18] Remcom. *Wireless InSite( Version 2.6.2)*. `http://www.remcom.com/wireless-insite`.

[19] K. Taylor. Using commercial ray tracing software to drive an attenuator-based mobile wireless testbed. Master's thesis, University of Maryland-College Park, 2012.

[20] X. Wang, E. Anderson, P. Steenkiste, and F. Bai. Improving the accuracy of environment-specific vehicular channel modeling. In *WiNTECH '12*, 2012.

[21] X. Wang, K. Borries, E. Anderson, and P. Steenkiste. Network-scale emulation of general wireless channels. In *IEEE VTC2011-Fall*, 2011.