Investigating a Physically-Based Signal Power Model for Robust Low Power Wireless Link Simulation

Tal Rusak
Department of Computer Science
Cornell University
Ithaca, New York, USA 14853
tr76@cornell.edu

Philip Levis
Computer Systems Laboratory
Stanford University
Stanford, California, USA 94305
pal@cs.stanford.edu

ABSTRACT

We propose deriving wireless simulation models from experimental traces of radio signal strength. Because experimental traces have holes due to packet losses, we explore two algorithms for filling the gaps in lossy experimental traces. Using completed traces, we apply the closest-fit pattern matching (CPM) algorithm, originally designed for modeling external interference, to model signal strength.

We compare the observed link behavior using our models with that of the experimental packet trace. Our approach results in more accurate packet reception ratios than current simulation methods, reducing the absolute error in PRR by up to about 30%. We also find that using CPM for signal strength improves simulation of packet burstiness, reducing the Kantorovich-Wasserstein (KW) distance of conditional packet delivery functions (CPDFs) by a factor of about 3 for intermediate links.

These improvements give TOSSIM, a standard sensor network simulator, a better capability to capture real-world dynamics and edge conditions that protocol designers typically must wait until deployment to detect.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: Model Development, Model Validation and Analysis

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Wireless link simulation, wireless sensor networks

1. INTRODUCTION

Many wireless sensor network deployments have observed significant differences between behavior in controlled environments such as test labs or simulation and behavior in the field. These differences have in many cases caused applications to be unable to successfully collect the desired data [11]. The reasons for these failures are very difficult to determine due to the highly constrained nature of sensor network hardware, including only a few bits of output information and little memory to save performance logs. The remote nature of many sensor network deployments make the study of the operation of the network and debugging protocols and applications even more difficult.

Increasing simulation fidelity, so simulators can capture edge cases and complexities encountered in real deployments, will reduce the gulf between testing and practice. It will also allow sensor network developers to use the resources of a PC to debug and develop full scale applications. Creating such models will facilitate theoretical studies regarding the nature of such systems. Low-power wireless networks have proven difficult to simulate because of a large number of factors that impact their operation and a limited theoretical understanding. In particular, the precise modeling of the variation of noise and signal power when receiving a packet in WSNs is an open problem. A special challenge in creating these models is that WSNs share the RF environment, and especially the 2.4 GHz frequency range, with 802.11 wireless networks, microwaves, cordless telephones, and many other interference sources.

Wireless simulators have traditionally relied on analytical models for signal strength and noise. These models allow developers to explore a huge space of possible configurations and network designs and also enable a good understanding of packet dynamics. Such simulations have had success in modeling highly-complex environments, such as the cell phone network's coverage of cities. For example, the WiSE tool [6] developed in the mid-1990's and several commercial tools available today [1, 4] model the signal power of wireless networks in complex environments with relatively low errors. Such systems use computer-based modeling tools to express the geometry of the region being simulated.

We take a different approach to modeling sensor networks. Rather than simulate an arbitrary configuration of nodes based on analytical models, we examine how to simulate a specific configuration based on experimental traces. This experimental approach, which we call *physically-based simulation*, has an additional benefit—it allows us to validate our models by comparing to real world phenomena. Unlike purely analytical approaches, which are not grounded in a real network and therefore cannot be validated, using measurements from a deployment allows us to compare the resulting simulated behavior with the observed behavior.

MSWiM'08, October 27–31, 2008, Vancouver, BC, Canada.

Using physical-layer measurements in the form of 4 Hz RSSI traces and 1 kHz noise+interference traces, we explore using probabilistic models to recreate behavior that is representative of what is observed on the real network. Simply replaying traces is insufficient, as it does not allow users to simulate experiments longer than the traces, and can also lead to overfitting time constants [7].

Physically-based simulation has been applied successfully in our previous work to modeling noise and interference [10] for the purpose of simulation. In our prior work, we proposed the Closest-fit Pattern Matching (CPM) algorithm for modeling noise and interference based on traces from deployed networks. In this paper, we propose extending this approach to modeling signal strength variations. CPM uses conditional probability distributions to model trends in the data trace. Based on the past k values for the variable (noise, signal strength, etc.), CPM samples from the probability distribution of what the next value will be. We present the algorithm in greater depth in Section 2.4.

There are two research challenges, however, in applying this technique to signal strength. The first challenge is biased sampling. Unlike noise+interference, which can be sampled at any time, signal strength can only be sampled on successfully received packets. This means that the signal strength trace is only partially observable. Only using received packet signal strengths skews the distribution and may lead to different packet reception ratios than those observed in reality. Therefore, an algorithm needs to generate estimates of missed signal strength measurements.

We propose two solutions to address biased sampling. One algorithm, called Average Value (AV), simply assumes all missed packets have the average observed signal power. The other, called Expected Value PMF (EVP), fills in a probability mass function of expected signal strengths based on the reception probabilities of the signal strengths of observed packets. We find that EVP leads to a lower maximum packet reception ratio (PRR) error bound as compared to AV. EVP bounds the absolute error in PRR to 22% as compared to experiment depending on the link, while AV bounds the same error measure by 30%.

The second challenge is phase and sampling precision. For some physical layers, such as the one we study (802.15.4), there is a very sharp 1.5dB transition between low and high packet reception ratios. The radio hardware, however, can only produce readings at the precision of a single dB. When the radio reports the signal strength of a received packet, this is the sum of the noise+interference and the actual signal strength. The sharpness of the SNR-PRR curve and similarity between the two values means that the relative phase of interference and signal is important.

To address the problem of phase and sampling precision, we explore whether assumptions of in-phase, out-of-phase, or neutral phase additive models lead to more accurate simulation. We find that for each experiment we need to individually evaluate which phase assumption to use, and that choosing the correct assumption can lead to reductions of error in absolute packet reception rate by up to 30%.

Another advantage of this model is that it allows sensor network designers to choose a particular algorithm and phase assumption that best fits their deployment location. We provide an overview and several examples of fitting models to experimental traces of both noise+interference and signal power in Section 5.1.

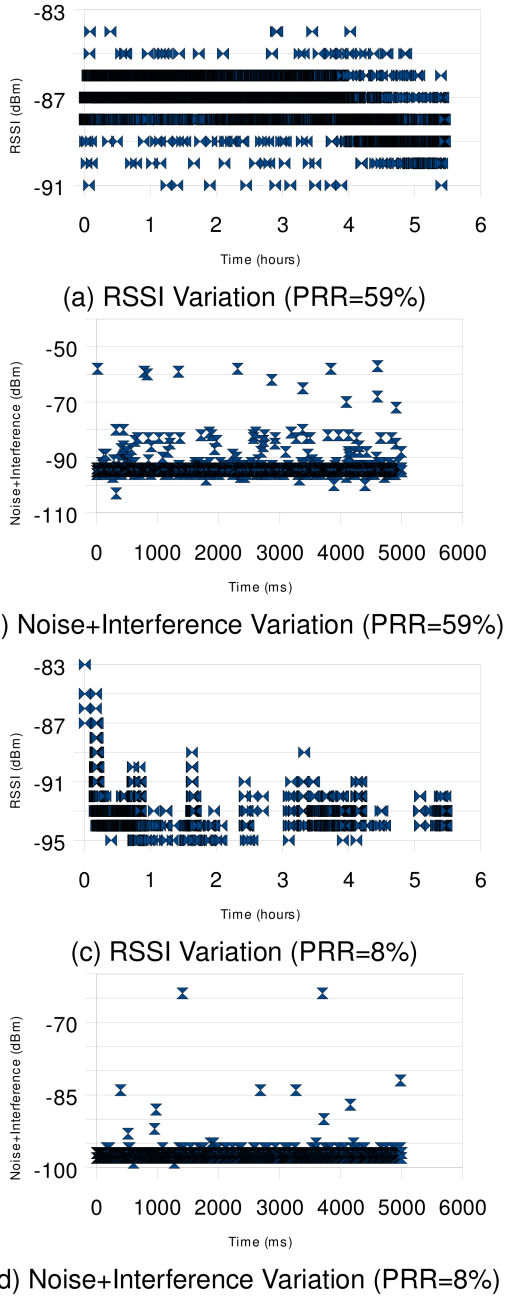

Figure 1: This figure shows experimental variations in RSSI ((a) and (c)) and Noise+Interference values ((b) and (d)) from two representative sensor network deployments. We see that both of these parameters are non-constant and do not vary consistently across different environments.

Finally, we compare fixed-PRR links of the various simulation methods to the Kantorovich-Wasserstein (KW) distance on conditional packet delivery functions (CPDFs). CPDFs describe packet delivery probability given n consecutive successes or failures. As each x value is equally important in a KW distance measure, CPDFs lend more weight to the rare than the common case, and so better represent the complexities of real-world networks than simple measures such as μ of a Gilbert-Elliot channel.

The rest of this paper is organized as follows. In Section 2, we review related work about TOSSIM, a standard simulator for wireless sensor networks, about modeling signal strength, and about physically-based simulation. In Section 3, we present algorithms for the modeling of signal power in sensor network systems. In Section 4, we present our experimental work and provide an overview of the traces used to validate the proposed methods. In Section 5, we compare simulation results to experimental traces by using absolute PRR differences and the Kantorovich-Wasserstein (KW) distance [8] using the concept of CPDFs [10]. Finally, in Section 6 we provide concluding observations.

2. BACKGROUND AND RELATED WORK

2.1 The TOSSIM Simulator

TOSSIM simulates TinyOS-based sensor network applications [12, 13, 14]. TOSSIM replaces several low-level hardware components with software equivalents. Application code runs unmodified in the simulator, enabling developers to test implementations in addition to algorithms. TOSSIM has advanced network simulation features: it simulates packet capture, implements acknowledgments (including false positive acknowledgments), and has a robust noise model [10].

2.2 Signal Power Models for Sensor Networks

Sensor network simulators have taken varying approaches to modeling signal power. Released versions of TOSSIM, for example, assume signal power $|\mathbf{S}|$ to be constant, and allow the user to input a gain (attenuation) value for each link in the simulation. Currently, gain is either manually input for each link or modeled using a tool [19] which simulates overall network structures, but not the temporal variations between individual links in the network. Figure 1 illustrates the noise+interference and RSSI variations over typical, representative experiments. There is a longer term variation in the signal power of received packets, which is approximated in this figure by $RSSI = |\mathbf{S} + \mathbf{N}|$, where $|\mathbf{N}|$ is the noise+interference value. Since $|\mathbf{S}| \gg |\mathbf{N}|$ for received packets, however, it is unlikely that the variations illustrated in this figure, taken from experimental traces, are a result of noise variations alone. We see that TOSSIM's assumption that signal power is constant is a simplification to reality.

Other models of sensor networks have attempted to apply the aforementioned analytical models to sensor network links. One such analytical model that has been investigated in the context of sensor networks is the *log normal shadowing power* model. In this technique [9], power is predicted using the path loss and shadowing analytical models. On the dBm scale, the signal power S is given by

$$S = P_t + K - 10\lambda \log_{10} \frac{d}{d_0} + \Psi(\mu, \sigma) \qquad (1)$$

where S is the desired signal power, P_t is the transmit power, λ is the path loss exponent, $\frac{d}{d_0}$ is a physical parameter proportional to distance, and Ψ is a Gaussian random variable, where μ is the mean and σ is the standard deviation. Bae and Kim [5] conducted an investigation attempting to apply this model for use with the TI/Chipcon CC2420, a radio commonly used in sensor network deployments (including the experiments we performed), and gave a set of parameters that apply to this radio and to the 2.4 GHz frequency range. When we apply this model, we use the parameters given in Table I of that study [5].

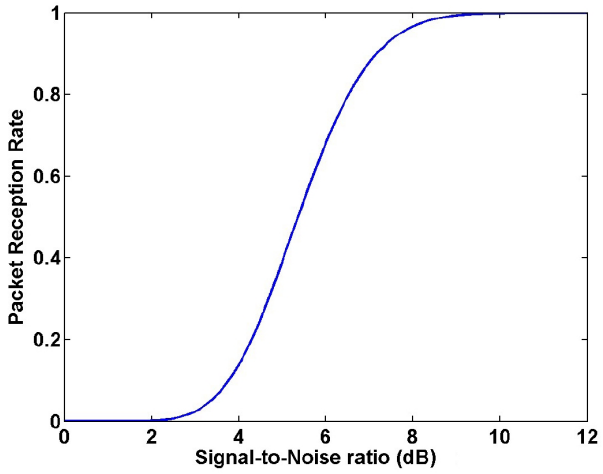

Figure 2: TI/Chipcon CC2420 SNR/PRR Curve [10].

2.3 Physically-Based Simulation

This paper extends the physically-based radio link simulation algorithm introduced in our previous work [10]. The equation that underlies this model is

$$SNR = \frac{|\mathbf{S}|}{|\mathbf{N}|} \qquad (2)$$

where $|\mathbf{S}|$ is the magnitude of the signal power of a received packet, $|\mathbf{N}|$ is the resultant magnitude of any environmental noise or disruption not caused by the network being implemented, and SNR is the signal-to-noise ratio [10]. Note that in a logarithmic scale, (2) may be expressed as

$$SNR_{\text{dB}} = |\mathbf{S}|_{\text{dBm}} - |\mathbf{N}|_{\text{dBm}} . \qquad (3)$$

This expression for SNR can be mapped to a packet reception rate using the function given in Figure 2. The TOSSIM simulator implements this model as follows: it models $|\mathbf{N}|$ using the CPM algorithm [10], reviewed in the next section. We see from Figure 1 that modeling noise is desirable since there are substantial short term variations in noise values. As mentioned above, TOSSIM currently assumes $|\mathbf{S}|$ to be constant.

2.4 The Closest-Fit Pattern Matching (CPM) Algorithm

The CPM algorithm uses an experimental trace to create a conditional model of observed values [10]. First, an experimental trace is collected using mote hardware in the environment to be simulated at frequency x. In its model generation phase, CPM scans the trace and computes a prob-

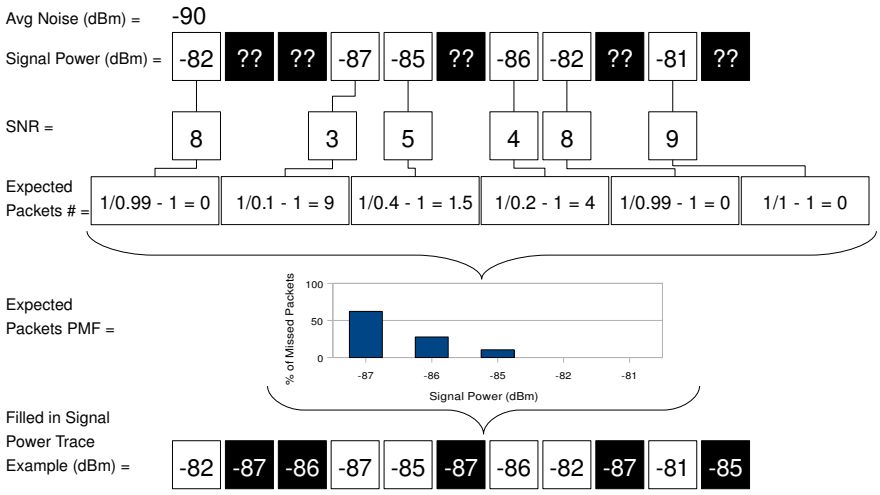

Figure 3: An example of the EXPECTED VALUE PMF algorithm on an input, assuming an average noise of -90 dBm. This trace shows 6 of 11 packets were received, with RSSI values of -82 dBm, -87 dBm, -85 dBm, -86 dBm, -82 dBm, and -81 dBm. The RSSI values of the five missed packets are not known, and this is indicated by "??"s in the figure. Extrapolating from expected PRRs of the RSSI values of the received packets, there should be 14.5, not 5 lost packets: for example, only 1 of 10 packets at -87 dBm should be received. Note that for simplicity, we use PRR values that are approximates to those given in Figure 2.

ability distribution of the expected value v given k prior values. To run CPM, a simulation replays the first k values from the trace; then the algorithm uses the probability distribution constructed during model generation to sample the next value. If the prior k values do not match a pattern observed in the real network, then CPM samples from the most common pattern.

If k is the length of the trace, then CPM will simply replay the trace. If $k = 0$, then CPM takes independent samples. Lee et al. [10] found that $k = 20$ leads to the best results when simulating external interference.

3. ALGORITHMS FOR MODELING SIGNAL POWER

To improve the TOSSIM simulator, we propose to collect signal power traces over long periods of time and then to use the CPM algorithm to predict signal power. There is a substantial research challenge in producing traces of signal power.

3.1 Collecting Signal Power Traces

Collecting signal power traces is more complex than collecting noise traces for several reasons. First, any signal power value refers to a link between a pair of nodes; thus, any trace needs to involve both a sender and a receiver. Noise, on the other hand, is local to an individual node in the network.

Furthermore, signal power is not known to the sensor network mote directly. The best estimate is the $RSSI = |\mathbf{S}+\mathbf{N}|$ upon packet reception. Due to the steepness of the SNR-PRR curve (Figure 2), $RSSI$ must be corrected for this noise error. This is not just a matter of subtracting the noise value since the phases of waves must be considered.

In addition, while noise can be sampled discretely in any environment, and there will always be a sample when one is requested, this is not the case for signal power. Recall that signal power requires the communication between *two* nodes; if the link fails and a packet is not delivered, then there will be no $RSSI$ or signal power value available for this time. Thus, the signal power trace needs to be post-processed and filled-in for completeness and to avoid missing samples.

We suggest two separate steps to account for these challenges: (1) filling in missing signal power values into the experimental trace, and (2) correcting for the phase differences between noise and signal traces. Each of these algorithms assumes that a trace of RSSI values has been collected by measuring the received signal power from a network of two motes. This is accomplished by a TinyOS application that we wrote that sends packets at an interval of 4 Hz on the sending side, and records the RSSI value at reception. The algorithms also expect that the environmental noise in the environment that we aim to simulate has been characterized, such that the average noise N (in dBm) over the period of the RSSI-collection experiment is known or can be approximated well. A TinyOS application similar to `RSSISample` [3] can be used to measure average noise. Furthermore, we make one assumption of three about phase differences between noise and signal in each of these algorithms, quantified by p, where

$$p = \begin{cases} -1, & \text{Noise and signal are assumed in phase} \\ 0, & \text{No correction for phase difference} \\ 1, & \text{Noise and signal are assumed out of phase} \end{cases}. \quad (4)$$

We test all three assumptions in this paper by setting the phase correction factor p when constructing the signal power trace.

3.2 The Expected Value PMF (EVP) Algorithm

First, we consider the EXPECTED VALUE PMF (EVP) Algorithm. Pseudocode is given in Algorithm 1.

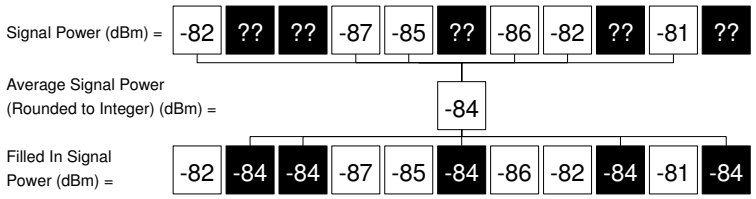

Figure 4: An example of the Average Signal Power Value **algorithm. This trace shows 6 of 11 packets were received, with RSSI values of −82 dBm, −87 dBm, −85 dBm, −86 dBm, −82 dBm, and −81 dBm. The RSSI values of the five missed packets are not known, and this is indicated by "??"s in the figure. In this algorithm, these missing values are filled in with the average signal power value, rounded to an integer.**

Algorithm	Expected Value PMF	Average Signal Power Value
Running Time (seconds)	11.529	11.244

Table 1: Running time of the preprocessing code using the suggested algorithms on a laptop with a 2.6 GHz processor. This run generated signal power traces and TOSSIM scripts for both directions of the link under the three phase assumptions studied. The RSSI traces had over 80,000 packets. Times were measured using Java's `System.currentTimeMillis()` function. The output can be used for many TOSSIM simulations.

Algorithm 1: Expected Value PMF

Data: Average noise N, list of successful reception times $\mathcal{T} \subseteq [0, l]$, RSSI trace $\mathcal{R} : \mathcal{T} \rightarrow$ dBm value collected from experiment of length l at 4 Hz, and phase assumption p

Result: A filled in signal power trace $\mathcal{S} : [0, l] \rightarrow$ dBm value mapping for time duration l at 4 Hz

Initialize PMF $\mathcal{P}$;
foreach $t \in \mathcal{T}$ **do**
  $r = \mathcal{R}(t)$;
  Find $s = 10 \log_{10} \left(10^{\frac{r}{10}} + p \times 10^{\frac{N}{10}}\right)$;
  Add mapping $(t, \text{round}(s))$ to $\mathcal{S}$;
  Add s to $\mathcal{P}$ with frequency $\frac{1}{prr(s-N)} - 1$;
foreach $(t \in [0, l]) \notin \mathcal{T}$ **do**
  Add mapping $(t,\ sample(\mathcal{P}))$ to $\mathcal{S}$;
return *S;*

The algorithm accepts as input the average noise in the environment to be simulated N (in dBm), a set of successful reception times $\mathcal{T}$ from an experiment conducted at 4 Hz, an RSSI trace $\mathcal{R}$ which maps time values where the RSSI is known to the corresponding RSSI values measured in dBm, and a phase assumption p. Then, for each of the known RSSI values, a signal power value is generated by correcting the RSSI value for the noise error using the expression

$$s \text{ (in dBm)} = 10 \log_{10} \left(10^{\frac{\mathcal{R}(t)}{10}} + p \times 10^{\frac{N}{10}}\right). \quad (5)$$

The intuition is that if a signal and noise are in phase, then the actual signal power is lower than the RSSI value detected, so $p = -1$. If the signal and noise are out of phase, then the actual signal power is higher than the RSSI value detected, so $p = +1$. Finally, if the phase differences cancel each other out, then $p = 0$ and $s = \mathcal{R}(t)$. After it is computed, s is added to the signal power trace $\mathcal{S}$ at time t.

The algorithm also adds each signal power s computed from the experimental trace to PMF $\mathcal{P}$ at a frequency corresponding to the number of packets that are expected to be lost, quantified by

$$\text{Expected lost} = \frac{1}{prr(s-N)} - 1 \quad (6)$$

where prr maps a signal to noise ratio (SNR $= s - N$ on the dB scale) to the corresponding PRR value, following the curve illustrated in Figure 2. This expression effectively extrapolates the number of packets that *should have* been received at this signal power value, based on the probability of receiving this single packet. One is subtracted to account for the packet that has just been received. Note that this number is stored as a `float` value, so it is possible to have fractional amounts of expected missing packets.

Finally, for every missing signal power value in $\mathcal{S}$, i.e. for every time that a packet was lost in experiment, the PMF of expected missing values $\mathcal{P}$ is sampled, and these times are mapped to signal power values corresponding to the proportion of signal power values that are found to be missing.

The number of expected lost packets is greater than the actual number of lost packets in the example shown in Figure 3. This is also the case in the real traces studied. We conjecture that this may be because of the 1 dBm granularity of the RSSI values collected by the CC2420 radio in our experiments. However, the full reason for this difference and its causes and implications is a topic of future work that we are interested in pursuing. As expected, this observation results in lower power values being common in traces filled in using the EVP algorithm since these values are most likely to be added to the trace.

3.3 The Average Signal Power Value (AV) Algorithm

We also consider the Average Signal Power Value (AV) Algorithm. Pseudocode for this method is given in Algorithm 2. This algorithm accepts as input the average noise N (in dBm), a set of successful reception times $\mathcal{T}$ from an

experiment collected at 4 Hz, an RSSI trace $\mathcal{R}$ which maps time values where the RSSI is known to RSSI values measured in dBm, and a phase assumption p. Then, the signal power trace is computed for these known times using the expression (5) and all known values are added to the signal power trace $\mathcal{S}$. The average signal power of the received packets, P, is computed (in terms of dBm) and rounded to an integer, to conform to the output of the CC2420 radio for real packet values. Note that this average signal power has a sampling bias, as it only considers received packets. Finally, the algorithm fills in the signal power trace by inserting the average signal power value for all time values that are missing from $\mathcal{S}$. Figure 4 shows an example of the execution of the AV algorithm.

Algorithm 2: AVERAGE SIGNAL POWER VALUE

Data: Average noise N, list of successful reception times $\mathcal{T} \subseteq [0, l]$, RSSI trace $\mathcal{R} : \mathcal{T} \rightarrow$ dBm value collected from experiment of length l at 4 Hz, and phase assumption p

Result: A filled in signal power trace $\mathcal{S} : [0, l] \rightarrow$ dBm value mapping for time duration l at 4 Hz

foreach $t \in \mathcal{T}$ **do**
  $r = \mathcal{R}(t)$;
  Find $s = 10 \log_{10} \left(10^{\frac{r}{10}} + p \times 10^{\frac{N}{10}}\right)$;
  Add mapping (t, s) to $\mathcal{S}$;
Let P be the average value of power values in $\mathcal{S}$, rounded to an integer;
foreach $(t \in [0, l]) \notin \mathcal{T}$ **do**
  Add mapping $(t,\ P)$ to $\mathcal{S}$;
return *S;*

3.4 Implementation and Performance of Signal Power Correction Algorithms

Both of these algorithms have been implemented using a Java preprocessor that accepts text files of traces that are collected from a TinyOS application developed for this purpose. The application outputs the signal power traces $\mathcal{S}$ to a text file that is used as input to a modified version of the TOSSIM simulator, which uses CPM (see Section 2.4) to predict signal power when a simulation is performed. Our preprocessor also automatically generates Python scripts for use with TOSSIM for all three phase assumptions discussed above. The preprocessing step needs to be run only once for an arbitrary number of simulations of a certain wireless link. Table 1 gives timing results collected for these preprocessing steps.

To decide on the effectiveness of these simulation techniques, we collected experimental noise and signal power traces discussed in the next section.

4. EXPERIMENTAL WORK

In conducting this study, we noticed a lack of experimental traces for signal power variations. Thus, we collected our own experimental traces in order to audit these simulations. Each experiment we conduct has one sender mote and one receiver mote, placed at specific locations on the Cornell University campus shown in Figure 5.

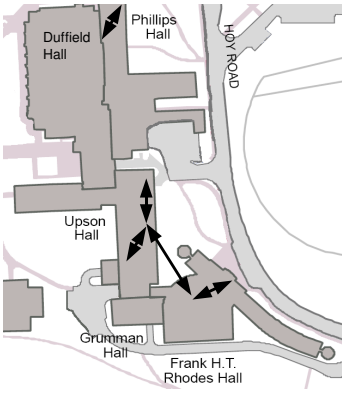

Figure 5: Map of experimental collection locations for this investigation; each pair is represented by an arrow. The link in Phillips hall had two motes separated by one floor, and in all other experiments the motes were located on the same floor. Base map from `http://www.parking.cornell.edu/pdf/Stu_combo_2006.pdf`.

We developed a TinyOS application that sends packets at a rate of 4 Hz from the sender to the receiver. We chose this frequency as a baseline to investigate long RSSI traces and we plan to investigate other collection frequencies as future work. The sender mote uses this application to send packets at this rate; the receiver simply listens for packets and records the sequence number and the RSSI of each received packet. For each experiment, we tested the link in both directions, i.e. first one mote is the sender and the other mote is the receiver, and then the sender and the receiver are switched for the purposes of collecting another trace. We collected an RSSI trace of about 12 hours (or more) for each pair of nodes.

The nodes used were Telosb motes [16] with TI/Chipcon CC2420 radios [2], the radio that is modeled by TOSSIM. The experiments were conducted in Rhodes, Upson, and Phillips Halls on the Cornell University campus. These buildings are high traffic, high use academic facilities around the clock and they are both connected with 802.11abg wireless networks. There are also cordless phones, microwaves, and personal wireless access points in use in both buildings. Thus, there are many factors that can impact the quality of the wireless connection between the nodes.

5. EVALUATION

We evaluate the proposed model on two levels. First, we show that using the proposed algorithms, along with an appropriate choice of assumptions about phase differences between the signal and noise, we can achieve predictive packet reception rates simulation compared directly to experiment. Then, we evaluate the correlation among packets and show that our algorithms provide KW distances lower by a factor of about 3 compared to the best alternative simulation methods.

5.1 Comparing Simulation and Experiment PRRs

In this section, we compare a first order parameter, packet reception rate (PRR), between simulation and the environ-

ment that we are trying to simulate. The different algorithms proposed above have a different level of correspondence to experiment with respect to PRR.

PRR is a very basic simulation parameter. It is possible to get a perfect PRR simulation by simply accepting packets at a rate equal to the PRR that was derived from the experiment. Unfortunately, such a simulation will not take into account temporal variation of signal power or packet reception correlation that is known to be found in sensor network links [10]. Furthermore, in complex networks it may be impossible to dictate the reception rate, since such a simulation does not consider the interactions between different pairs of motes.

PRR is a very difficult parameter for general simulators to get right. For example, we ran simulations of our experiments with the analytical model suggested in Section 2.2 and got PRRs that were very different from those that we observed experimentally. In most cases, these results were so far off that they do not even provide a basis for comparison.

At the same time, correctly modeling packet reception rate is extremely important in wireless sensor networks, especially for those links in the intermediate range. It is vital for protocol designers to have an idea of the proportion of packets that are received as compared to those that are lost in order to correctly account for these losses, either in information, time, or both, when programming the network. Physically-based simulation introduced in TOSSIM 2.0.1 and 2.0.2 has greatly increased the ability of the simulator to correctly capture packet reception rates. For example, Metcalf [15] shows that TOSSIM 2.0.2 produces largely correct results in terms of PRR for good and bad links. In that work, TOSSIM's gain value is set to the average RSSI of the link, which approximates signal power. However, examining the figures in Chapter 4 of [15], we note that PRR differences between experiment and simulation values occur mostly in intermediate links. Although there are relatively few intermediate links in that study, their PRR is not predicted precisely in many cases.

The model that we propose improves upon the prediction of PRR for the following reasons. First, it considers the variations in signal power which may account for some PRR variations. There are two algorithms proposed for filling-in experimentally determined signal power traces, and due to varying environmental conditions one or the other may be more appropriate. In addition, our model corrects for the phase differences between signal and noise waves for each of the links when converting RSSI to signal power. This correction needs to be tuned to the environment, and it can cause major differences in the overall PRR of a given link.

Note that in any physically-based simulation model, it is almost always possible to get an exact PRR by modifying the signal power with a constant coefficient or a constant additive value. This is, essentially, a brute-force method of searching for the appropriate phase correction. The proposed method shows that using a signal power trace with just three assumptions about phase, it is usually possible to achieve nearly the same result.

In Figure 6, we compare the absolute differences between experimental and simulation PRRs of the various algorithms that we propose in this work under different assumptions about phase. We also perform the same comparison to the corresponding experiments as simulated by TOSSIM 2.0.2. The TOSSIM simulator assumes that signal power is constant, and we input the average RSSI value of the corresponding experimental trace into TOSSIM. This is a common assumption; for example, Metcalf used this approximation as the gain parameter in TOSSIM in the aforementioned study [15]. In each case, noise+interference is modeled by CPM using a noise trace collected in the experimental environment without sending packets.

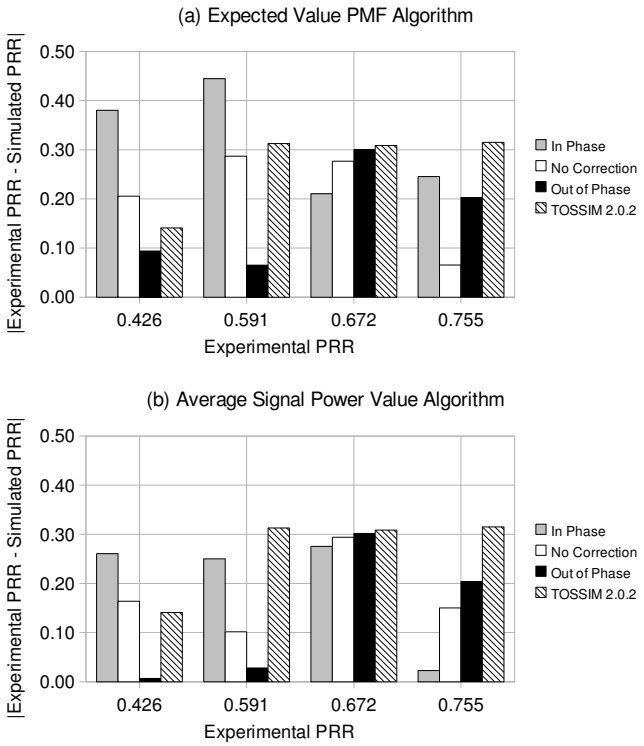

Figure 6: Absolute differences between the PRR of experiments and simulations for various assumptions about phase using (a) the expected value PMF algorithm and (b) the average value algorithm for filling in signal power traces. In both plots, the TOSSIM 2.0.2 value takes gain (signal power) to be the average RSSI, without rounding.

In the present experimental study, we noted similar results: the bad links and the good links perform sufficiently well in TOSSIM 2.0.2. As we can see in Figure 6, however, TOSSIM does not give satisfying results for intermediate links and gives somewhat arbitrary PRR results given a consistently calculated gain value.

By correctly tuning the choice of assumptions it is possible to perform an effective simulation using the suggested technique. Given the appropriate phase assumption, the EVP algorithm approximates PRR to within an absolute difference of 22% in the worse case (less than 10% in all but one case), while the AV algorithm approximates the PRR value to within 30% (less than 10% in all but one case).

Conducting an analysis similar to Figure 6 allows sensor network designers to fit experimental data from their deployment location to one of the algorithms proposed. Thus, it is possible to use this method to choose the best simulation technique for a given wireless link.

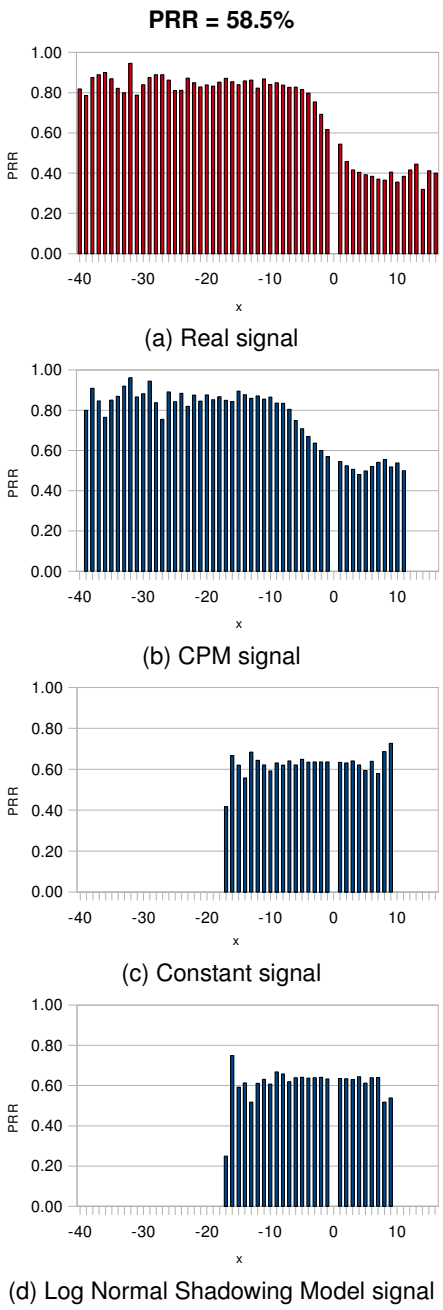

Figure 7: CPDFs for PRR=58.5% link. The negative x-axis plots consecutive packet successes, while the positive x-axis plots consecutive packet failure. We note that the experimental plot (a) is generally correlated in x with a sharp decreasing trend as x increases, and that the trace simulated with CPM (b) also follows such a correlation. The constant signal CPDF (c) and the Log Normal Shadowing simulation CPDF (d) show no apparent correlation. Values for x with no bar indicate that no data was collected from this value, not a PRR of 0%.

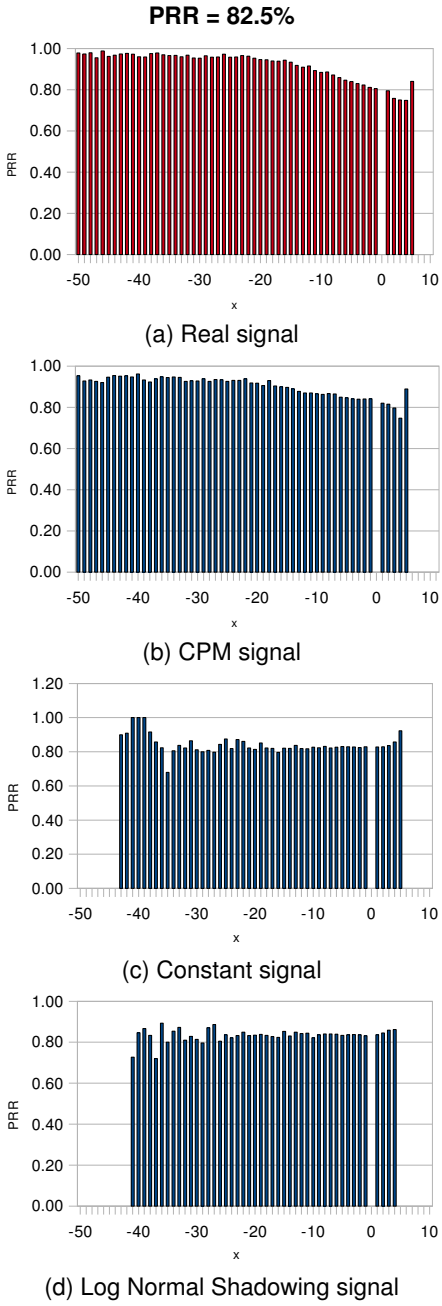

Figure 8: CPDFs for PRR=82.5% link. The negative x-axis plots consecutive packet successes, while the positive x-axis plots consecutive packet failure. We note that the experimental plot (a) is correlated in x with a gently decreasing trend as x increases, and that the trace simulated with CPM (b) also follows such a correlation. The constant signal CPDF (c) shows no apparent correlation, while the Log Normal Shadowing simulation CPDF (d) appears to have a slightly increasing PRR value as x increases. Values for x with no bar indicate that no data was collected from this value, not a PRR of 0%.

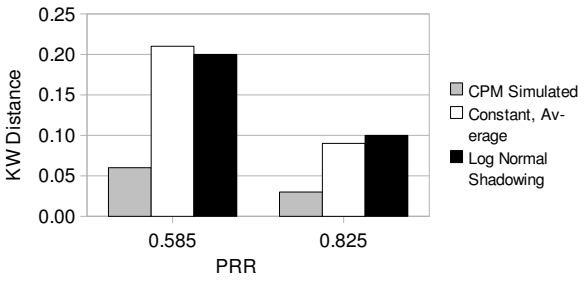

Figure 9: The KW distance comparing various signal power modeling algorithms to the real power algorithm.

5.2 KW Distances of Fixed-PRR Simulations

We note that not only is our approach effective at reproducing experimental PRRs in simulation, it can also model packet correlation effectively. Such correlations were shown to have an effect on higher-order protocols [10]. Clearly, the overall PRR of a long-term experiment is not the only factor that has to be considered for correct simulation.

To investigate the impact of applying our model for signal power on packet reception correlation, we apply a constant noise so that signal power is the only factor varied. Then, we change the noise value for the various simulation models such that the packet reception rates are essentially the same. We consider the following several signal power modeling algorithms in this fashion. *Real signal power* traces are from actual experiments and are corrected using the algorithms given above. *CPM signal power* traces are generated by inputting the real power traces into the CPM algorithm and keeping a history of $k = 20$. *Constant signal power* traces assume that signal power is constant and pre-determined. *Log Normal Shadowing* signal traces use the analytical expression suggested in Section 2.2 to model signal power.

For each of these simulations, we build a conditional probability plot of packet reception based on the condition

$$\mathcal{X}(x) = \begin{cases} |x| \text{ successful sequential receptions}, & x < 0 \\ x \text{ sequential packet losses}, & x > 0 \end{cases}, \tag{7}$$

following an idea first introduced in our previous work [10]. Such a distribution has been called a CPDF. Note that CPDFs investigate trends in packet reception burstiness [18] on the time scale of one packet only.

In Figures 7 and 8, we present CPDFs of two intermediate links that we investigated. The experiment from which this link was collected received about 59% of the packets, and varying the constant noise value led to the two different PRRs for the links in the two figures. For the purpose of this analysis, we used the EVP algorithm, which provides the tightest maximum error bound in PRR over many experiments. In addition, we use the out of phase assumption, which provides a PRR close to the experimentally discovered PRR using the experimental noise characterization. Note that the CPDF corresponding to the CPM algorithm shows a more similar correlation to the Real signal CPDF than any other simulation method. In Figure 7(a), we note that there is an overall sharp decrease in PRR as x (as defined in (7)) increases. The CPDF corresponding to the CPM model (Figure 7(b)) is the only other that appears to show any such decrease. On the other hand, the CPDFs of constant and log normal shadowing signal simulations (Figures 7(c) and 7(d)) appear to present little correlation. Similarly, in Figure 8(a), we see a more gentle decrease in PRR as x increases. Again, the CPM signal CPDF (Figure 8(b)) is the only simulation model that captures this decrease effectively. The constant signal CPDF (Figure 8(c)) shows no apparent correlation, and the CPDF generated from the log normal shadowing model (Figure 8(d)) actually shows a slightly increasing PRR as x increases. Recall that CPDFs are generated from a probabilistic simulation, so some outliers in the overall trends are to be expected.

To quantify these observations, we measure the Kantorovich-Wasserstein (KW) distance [8] between CPDFs of *real signal power* and the CPDFs of the simulation techniques discussed above. The KW distance computes the distance between probability distributions in a manner that places more emphasis on the rare rather than the common situation. In Figure 9, we show the results of this calculation. The code used to compute the KW distance uses an equivalent metric known as Earth Mover's distance [17].

The CPM algorithm provides the best KW distance as compared to real signal power. In particular, the CPM algorithm improves the KW distance of the intermediate links studied by a factor of about 3. This shows that applying CPM to the signal power traces output from the algorithms suggested in Section 3 is effective at modeling the temporal correlation of packets. At the same time, the overall behavior of the link is similar to the behavior (i.e. the PRR) of the corresponding experimental link. Thus, applying our model leads to a radio link simulation that is quite similar to the behavior of the real link being modeled, both in terms of packet reception correlation and in terms of packet reception rates.

6. CONCLUSIONS

This paper presents an improvement to the TOSSIM simulator by suggesting a way to model reception power of wireless links. In particular, we suggest two algorithms to fill in for signal traces that are inherently missing from experiment. We also make three assumptions about phase. By examining the PRR of links in simulation and experiment, we note that choosing correct parameters in the suggested model can lead to very close PRR simulations as compared to the experimental trace. Using the KW distance, we show that our simulation technique preserves the packet reception correlation as compared to a real signal power simulation.

The algorithms presented in this paper, along with the CPM algorithm that we suggested in previous work, can be used to create a high-fidelity simulation of wireless links in TinyOS sensor networks using the TOSSIM simulation framework. Such accurate models and simulations can help protocol developers learn about the effectiveness of their implementation by collecting traces of signal power and noise from the intended deployment environment.

7. ACKNOWLEDGMENTS

We would like to thank Kannan Srinivasan for useful discussions on this topic. Portions of this work were conducted at the Wireless Networks Laboratory, Cornell University, under the direction of Zygmunt Haas. Cisco Systems, Inc.

provided generous funding that enabled us to conduct the experimental portion of this project.

8. REFERENCES

[1] Wireless Valley Software, `http://www.connect802.com/w-valley.htm`, 2003-2005.

[2] CC2420 Datasheet, Texas Instruments Inc., `http://focus.ti.com/docs/prod/folders/print/cc2420.html`, 2007.

[3] TinyOS repository (`http://www.tinyos.net/`), `/tinyos-2.x-contrib/stanford-sing/apps/RssiSample`, 2007.

[4] Mentum Inc., `http://www.mentum.com/`, 2008.

[5] S. Bae and K. Kim. Analysis of Wireless Link for Mobile Sensor Network. In *Proceedings of ICEE'06*, 2006.

[6] S. Fortune, D. Gay, B. Kernighan, O. Landron, R. Valenzuela, and M. Wright. WiSE design of indoor wireless systems: practical computation and optimization. *Computational Science and Engineering, IEEE*, 2(1):58–68, 1995.

[7] N. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[8] C. Givens and R. Shortt. A class of Wasserstein metrics for probability distributions. *Michigan Math. J*, 31(2):231–240, 1984.

[9] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[10] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 21–30, 2007.

[11] P. Levis. TinyOS: An open platform for wireless sensor networks, invited tutorial. In *MDM'06: IEEE 7th international conference on mobile data management*, 2006.

[12] P. Levis, D. Gay, V. Handziski, J. Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, C. Sharp, R. Szewczyk, et al. T2: A second generation OS for embedded sensor networks. Technical report, TKN-05-007, Telecommunication Networks Group, Technische Universitat Berlin, 2005.

[13] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SENSYS'03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, 2003.

[14] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. TinyOS: An Operating System for Sensor Networks. *Ambient Intelligence*, 2005.

[15] C. Metcalf. TOSSIM Live: Towards a Testbed in a Thread. *Master's Thesis*, 2007.

Experimental PRR = 0.59

	Real Noise	CPM Noise	Constant Noise
Real Signal	0.65	0.58	0.59
CPM Signal	0.56	0.56	0.58
Constant Signal	0.45	0.64	0.45

Table 2: Comparison of simulation strategies on an intermediate link experiment with an experimental PRR of about 59%. Note that introducing the CPM algorithm for signal power modeling, as suggested in this work, improves the accuracy of PRR as compared to the constant signal power assumption currently used in TOSSIM.

[16] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *ISPN'05: Proceedings of the 4th international conference on Information processing in sensor networks*, pages 364–369, 2005.

[17] Y. Rubner, C. Tomasi, and L. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 59–66, 1998.

[18] K. Srinivasan, M. Kazandjieva, S. Agarwal, and P. Levis. The beta-factor: Measuring Wireless Link Burstiness. In *SENSYS'08: Proceedings of the 6th international conference on embedded networked sensor systems*, 2008.

[19] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *SECON'04: Proceedings of the Conference on sensor and ad hoc communications and networks*, pages 517–526, 2004.

APPENDIX

A. COMPARING PHYSICALLY BASED SIMULATION PRRS

In Table 2, we take one experiment and compare various possible simulation techniques using the physically-based simulation model and TOSSIM. Results from several different algorithms are shown. First, we consider concurrent pattern matching (*CPM*), with history size $k = 20$, applied to either noise or power. Second, we assume that the noise or signal power is *constant*. In this case, we take the average value of the RSSI of the packets received in the experimental trace. Finally, we consider *real* signal power or noise, which is the experimentally collected noise or signal power. In the case of signal power, we use the out-of-phase EVP model to correct and fill in the experimental trace.

Note that in TOSSIM 2.0.2, the signal power is modeled as constant, while the environmental noise is modeled with CPM. From this data, we can see the CPM algorithm can greatly improve the PRR correctness when applied to the signal power, as suggested by this work.