Simulating Frame-Level Bursty Links in Wireless Networks

Daniel Lertpratchya
d.lertpratchya@gatech.edu

George F. Riley
riley@ece.gatech.edu

Douglas M. Blough
doug.blough@ece.gatech.edu

School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332

ABSTRACT

In this paper, we propose a stochastic bursty-link model to simulate bursty behavior observed in wireless communications. Our stochastic bursty-link model works at the frame level where the probability of correctly receiving a frame is dependent on the results of previous transmissions. To accomplish this, our model uses a bursty probability adjustment function to adjust the probability of correctly receiving a frame based on the history of the link. When an appropriate trace file is available, our model can directly derive a bursty probability adjustment function from the trace file. When a trace file is not available, our model can simulate different bursty characteristics by selecting a function from a set of ideal bursty probability adjustment functions. We show that our model can simulate different bursty behaviors observed in real wireless links, using both trace file analyses and ideal bursty probability adjustment functions. In addition to studying the ability to simulate bursty behaviors observed in real wireless links, we also study the effect of incorporating bursty behavior into wireless simulations. We show that incorporating bursty behavior into wireless simulation has a significant impact on the performance of a wireless routing protocol.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Verification

Keywords

ns-3, wireless simulation, bursty wireless channel

1. INTRODUCTION

Network simulations have been used extensively to evaluate the performance of wireless networks. With the scale and complexity of the wireless ad-hoc networks and sensor networks, network simulation has become an indispensable tool in studying wireless network performances.

Several network simulators are readily available for use in the research community and the industry. ns-2 [3], ns-3 [4], OPNET [8], and GloMoSim [24] are examples of well known network simulators. Most, if not all, network simulators support wireless network simulation such as IEEE 802.11 to some extent. One of the important components of wireless network simulation is the signal propagation model. Most of the network simulators support theoretical signal propagation models such as Friis free-space model [10] and the log-distance model [9]. A comparative study between different propagation models in ns-3 was done by Stoffers and Riley [21]. Kotz et al. [13] showed that many of the commonly adopted assumptions in the wireless network simulations are too simplistic. Among these simplistic assumptions is the signal propagation model used in the simulators.

One of the characteristics of wireless communication is that the wireless links are bursty. Multiple studies have shown that wireless links are bursty and that the burstiness affects the experimental results [6, 7, 12, 19]. A few studies have been done with the goal of quantifying the wireless link burstiness. At the physical layer, coherence time is the time during which the radio signal is considered to be stable. A Markov chain describing the burstiness of a link at bit level was proposed by Mushkin and Bar-David [18]. One of the first metrics to measure link burstiness was proposed by Srinivasan et al. [20]. The authors defined a burstiness metric, β, by using conditional probability delivery function (CPDF), which can be obtained from packet delivery traces. The burstiness metric measures if a link is closer to an independent link or an ideal bursty link. The authors showed that most wireless links are bursty and β can be used to predict network protocol performance on a bursty link.

Even though link burstiness is a widely recognized characteristic of wireless communications, most of the currently available network simulators do not provide support for modeling wireless link burstiness. A few models have been proposed to model burstiness [11,15,23]. Lee et al. [15] proposed a noise model such that the level of noise depends on the history of the previous noises. Gómez et al. [11] proposed an auto-regression model to replicate a bursty behavior by modeling received powers where the model is obtained from analyzing trace files. A major limitation of modeling sig-

Simutools 2014, March 17-19, Lisbon, Portugal

DOI 10.4108/icst.simutools.2014.254618

nal or noise variation is that obtaining accurate empirical data is very difficult or impractical. Vlavianos et al. [22] showed that getting an accurate measurement of received signal strength indicator (RSSI) and signal-to-interference-plus-noise ratio (SINR) is difficult due to many factors. For example, according to 802.11 specifications, RSSI is only measured during the PLCP preamble and not the whole frame. Moreover, RSSI resolution is dependent on the device chipset [16] and often reported as an integer only. SINR, which must be derived from RSSI, inherits all inaccuracies from RSSI [22].

In this paper, we propose a stochastic bursty-link model to simulate bursty behavior of the link at the *frame* level. By modeling link burstiness at the frame level, we are able to create bursty links and avoid the low-level inaccuracies and complexities. The underlying idea of our stochastic bursty-link model is that the probability of successfully receiving a frame is dependent on the history of the previous receptions. The model adjusts the probability of successfully receiving a frame based on the results of previous frame receptions. Our model can directly simulate a bursty behavior of a real wireless link given that an appropriate trace file is available to the model. Our model can also simulate bursty behavior by using appropriate functions if a trace file is not available. We show, through simulation, that our model can closely simulate real wireless links with different bursty behaviors.

The rest of the paper is organized as follows. In Section 2, we present the underlying idea of our stochastic bursty-link model. In Section 3, we discuss the two important parameters of our model and how to obtain the appropriate parameters for link modeling. We present our detailed implementation in the ns-3 simulator in Section 4. We evaluate our stochastic bursty-link model against other models and present the simulation results in Section 5. Finally, Section 6 concludes the paper.

2. MODELING BURSTY LINK BEHAVIOR

In this section, we present the stochastic bursty-link model for wireless simulation. Our model is motivated by the observation that, in a bursty link, the probability of correctly receiving a frame depends on the frame reception history of the link. Our goal is to mimic the same behavior where the probability of successfully receiving a frame is also dependent on the history of the link.

The high level idea of our stochastic bursty-link model is as follows: the bursty-link model keeps track of transmission results as observed by a receiver on each link in an internal cache. When calculating the probability that the current transmission will be successful, the model looks at the history of the previous transmissions and adjusts the probability according to the history. The model imposes a time-to-live (TTL) limit for a cache entry to prevent very old transmission results from affecting the probability value. If the cache is empty, the model simply uses the probability according to a default metric such as the distance between the sender and the receiver.

The adjustment to the probability values is made by consulting a function called a bursty probability adjustment function ($BPA(n)$) since it changes the probability values if the bursty behavior is taken into account. The parameter n is the size of the burst where $n > 0$ represents the number of consecutive received frames and $n < 0$ represents the number of consecutive missed frames. In other words, $BPA(n)$ is the change in probability of successfully receiving the next frame given that the previous n consecutive frames were received (when $n > 0$) or missed (when $n < 0$). Note that with this definition, the burst size 0 is undefined. However, we can use $BPA(0) = 0$ to represent the case where the cache is empty and no adjustment is made to the probability value.

To summarize, the main ideas of our model are as follows.

1. The model keeps track of transmission results on each wireless link in an internal cache.
2. The probability of successfully receiving a frame is adjusted based on the previous transmission results.
3. Time-to-live is introduced to limit the time during which a previous transmission result can affect future transmissions.

Two major components of our stochastic bursty-link model are the bursty probability adjustment function and the time-to-live of a cache entry. Since the choice of the two parameters will have significant impact on the behavior of our model, it is important that they are picked carefully when modeling a bursty link. In the next section, we discuss two methods to obtain appropriate values for these parameters.

3. BURSTY PROBABILITY ADJUSTMENT FUNCTION AND CACHE TTL

In this section, we present two methods to obtain appropriate bursty probability adjustment function and time-to-live for a cache entry: from a trace file and from a set of predefined functions. A trace file approach is suitable when the goal is to simulate a specific link and it is possible to get a trace file from that link. We present a set of simple functions that can be used to simulate bursty link behavior in the case where getting a trace file is not possible or the goal of the simulation is simply to simulate bursty links that are not modeled after specific links.

3.1 Trace File-based Values

The first method of obtaining a bursty probability adjustment function and cache TTL is through the analysis of a real trace file. By analyzing a real trace file, we can observe the bursty characteristic of the wireless link and obtain appropriate values for our model.

Since we are interested in analyzing burstiness of a link, we need to be able to keep track of the number of consecutive frames received or missed. One way is to keep track of the sequence number in the 802.11 header. We have to make sure that all frames of interest are sent with the same physical layer parameters; for example, wireless channel, modulation scheme, and data rate. This requirement can be met by looking at the Radiotap header.

Given a trace file, it is possible to calculate the conditional probability delivery function ($CPDF(n)$) from the trace file. $CPDF(n)$ gives the probability of correctly receiving a frame for different burst sizes n. A positive burst size represents a number of consecutive successful frame receptions while a negative burst size represents a number of consecutive frames missed. One example of a $CPDF(n)$ from a real trace file is shown in Figure 1. As seen from Figure 1, the link exhibits a bursty behavior where the probability of successfully receiving a frame depends on the results

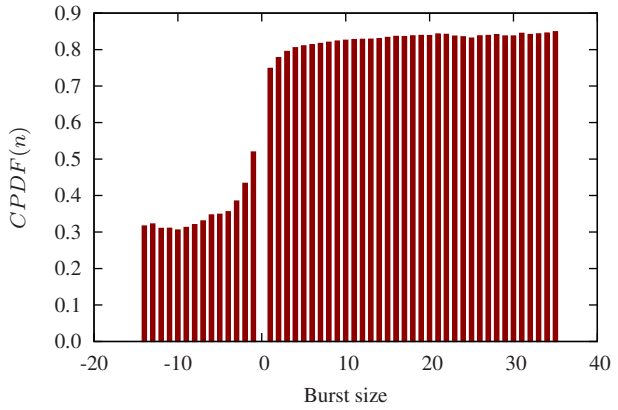

Figure 1: An example of CPDF from a trace file. The link exhibits a bursty behavior since the probability of successfully receiving a frame depends on the results of previous transmissions.

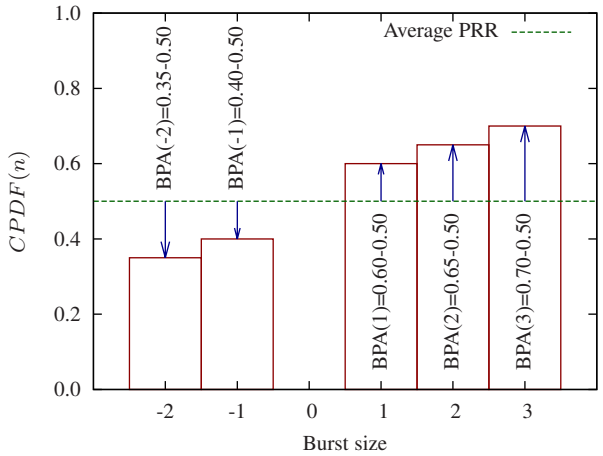

Figure 2: An example of calculating bursty probability adjustment from CPDF.

of previous transmissions. For example, if the source node is sending two frames, the probability of correctly receiving the second frame depends on the result of the first transmission. If the first frame was correctly received, the probability of correctly receiving the second frame is 0.7484. If, however, the first frame was missed, the probability of correctly receiving the second frame is only 0.5192.

A domain of $CPDF(n)$ of a finite trace file is a finite set since the number of consecutive frames received or missed is finite. For example, the domain of $CPDF(n)$ of the trace file in Figure 1 is $\{-14, -13, \ldots, -1, 1, 2, \ldots, 35\}$ since the largest number of consecutive frames missed is 14 and the largest number of consecutive frames received is 35. Notice that 0 is not included in the domain since the burst size 0 is undefined.

To obtain $BPA(n)$ from $CPDF(n)$ starting from a trace file, we first calculate the average packet reception ratio (PRR) from the trace file. The average PRR represents the probability of correctly receiving a frame when bursty characteristic of the link is not taken into account. Let $\mathbb{D}$ be the domain of $CPDF(n)$. Let $\triangle = \max(\mathbb{D})$ (the largest number of consecutive frames received) and $\triangledown = \min(\mathbb{D})$ (the largest number of consecutive frames missed). $BPA(n)$ can be defined as:

$$BPA(n) = \begin{cases} CPDF(n) - \text{PRR}, & n \in \mathbb{D} \\ 0, & n = 0 \\ CPDF(\triangledown) - \text{PRR}, & n < \triangledown \\ CPDF(\triangle) - \text{PRR}, & n > \triangle \end{cases}$$

We include the last two cases where $n < \triangledown$ and $n > \triangle$ to handle a case when simulating a burst size larger than the bursts observed in the trace file.

An example of how to obtain $BPA(n)$ from $CPDF(n)$ is shown in Figure 2.

To obtain the bursty probability adjustment function, we first calculate the average PRR of the link. In Figure 2, the average PRR of the link is 0.5. The next step is to calculate $CPDF(n)$ for different burst sizes n. The differences between the average PRR and the $CPDF(n)$ are $BPA(n)$. By analyzing the example $CPDF(n)$ in Figure 2, we obtain the following bursty probability adjustment function.

$$BPA(n) = \begin{cases} CPDF(n) - 0.5, & n \in \{-2, -1, 1, 2, 3\} \\ 0, & n = 0 \\ -0.15, & n < -2 \\ 0.20, & n > 3 \end{cases}$$

To calculate the appropriate time-to-live for a cache entry, we use the burstiness metric (β) proposed by Srinivasan et al. [20]. The burstiness metric is a scalar value between -1 and 1 used to measure burstiness of a given link. A value close to 0 means the link is more independent (the probability does not change much when burst occurs) while a value close to -1 or 1 means that the link is very bursty (the probability changes rapidly when burst occurs). Links with negative correlation have negative βs.

The burstiness metric of the link reduces as the duration between packets increases [20]. By decreasing the sampling rate when calculating β, the value of β decreases. To calculate the appropriate cache TTL, we decrease the β sampling rate until β is sufficiently close to 0, which means that the transmissions are almost independent of each other. The time between samples is now representing the duration where the results of previous transmissions have only a slight effect on the outcome of the next transmission. Thus, the time-to-live is equal to the time between samples.

The advantage of getting $BPA(n)$ and TTL from a real trace file is that the bursty characteristics of the simulated link will be almost identical to the real wireless link. The drawback of the trace-file based approach is that appropriate trace files must be available. However, getting trace files may not be feasible in all scenarios, for example, when the number of nodes is large. If the network consists of N nodes, we need to obtain $O(N^2)$ trace files, which may not be feasible. We would like to be able to incorporate bursty characteristics when simulating arbitrary links.

3.2 Synthetic-link Values

As discussed earlier, using a real trace file is the most accurate method of simulating a link. However, obtaining a trace file may not be feasible in all cases. The other method of obtaining a bursty probability adjustment function is to select one from a set of predefined functions. This method

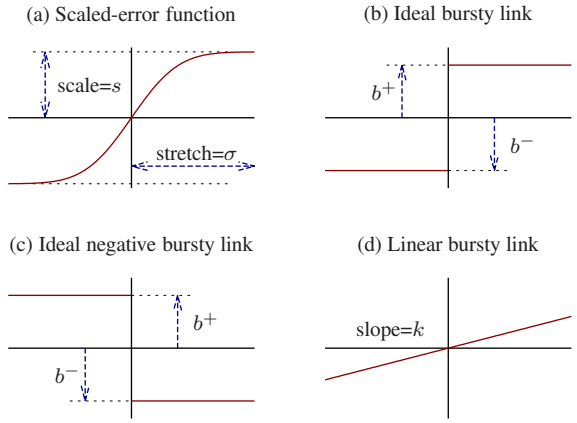

Figure 3: Examples of possible synthetic bursty probability adjustment functions. The x-axis is the burst size and the y-axis is the $BPA(n)$.

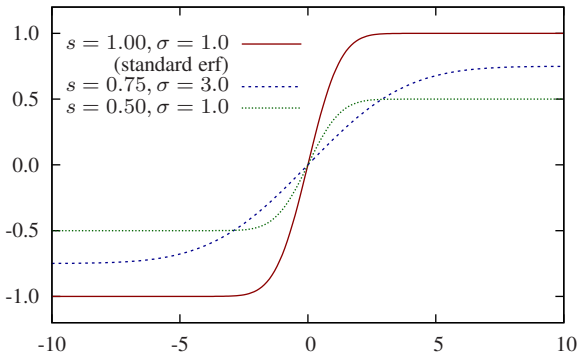

Figure 4: Examples of different scaled-erf functions with different scaling (s) and stretching (σ).

is suitable for a simulation where obtaining trace files is not possible or not feasible. The method is also suitable for a simulation where the goal is not to model links after specific real world links, but to incorporate bursty links into the simulation. We propose a set of functions in Figure 3.

Figure 3 (a) represents a BPA function that resembles the error function (erf). Figure 3 (b) and (c) represent the ideal bursty links where the bursty probability changes rapidly when a burst occurs. A BPA function in Figure 3 (c) shows a link with negative correlation, which has been observed in real wireless link [20]. Figure 3 (d) represents a linear bursty link where burstiness gradually increases.

The parameters b^+, b^- of the ideal bursty functions and the slope k of the linear bursty function, can be selected depending on the burstiness level desired. We note that there is no limitation on the actual values of the bursty probability. The only real limitation is the validity of the values and that the values returned by the model will be within a reasonable range. It is possible to use any function or a set of discrete values as a bursty probability adjustment. As mentioned previously, analyzing a trace file will result in a set of discrete values. A closed-form function, if desired, can be obtained by applying an appropriate statistical method.

The scaled-erf bursty function in Figure 3 (a) deserves special attention. We use an error function that has been *scaled* and *stretched* to obtain the bursty probability adjustment function. Our scaled-erf function takes two arguments: a scale (s), and a stretch (σ). In the standard error function, the function range is from -1 to 1. We scale the range of erf function by using s so that the range is not limited to between -1 and 1. The scaling s represents the limit to where the bursty probability adjustment function converges. We include the stretching factor, σ, to stretch the error function along the x-axis. The stretching factor changes the step sizes between bursts. The differences between two burst sizes in a function with large σ will be smaller than a function with small σ. Figure 4 shows examples of different scaled-erf functions with different parameters.

As seen in Figure 4, the effect of the scale, s, is to change the limit of the erf function. In other words, s represents the maximum probability change due to bursty behavior. The effect of the stretching factor, σ, is to stretch out the erf function along the x-axis. A large σ means that the probability differences between two burst sizes is small. For example, the probability difference between burst size 2 and burst size 1 of the standard error function is 0.2718 while the difference is 0.2338 when $\sigma = 3$.

By using s and σ to modify the shape of the error function, we can use the scaled-erf function to approximate the remaining three functions by selecting the appropriate parameters. For instance, to approximate an ideal bursty function with $b^+ = b^-$, we set s to b^+ and select a small σ such that the burst size 1 and -1 have values sufficiently close to b's (i.e. $BPA(1) \rightarrow b^+$ and $BPA(-1) \rightarrow b^-$). We can use a negative scale to get a function like the one in Figure 3 (c). For a linear bursty function, we can select a very large σ to stretch out the error function.

To get an appropriate time-to-live for a cache entry in a synthetic bursty link, we have to use a value within a range of suggested values, since we do not have a trace file to analyze like we did in Section 3.1. In [20], the authors suggested that the duration of correlation is usually around 500 ms. We present our own observation regarding the appropriate TTL from the trace files we generated for simulation in Section 5.

The advantage of using a predefined function is that it does not require a trace file. The predefined functions can be used in a simulation with a large number of nodes where obtaining trace files is not feasible. The drawback of using a function is that the characteristics of the simulated link may not match exactly to real wireless links.

3.3 Algorithm Complexity

We end this section with some discussion regarding the complexity of the stochastic bursty link model.

Since the probability of correctly receiving a frame in our model is dependent on the results of previous transmissions, each node in the simulation has to keep track of transmission results from all other nodes in the network. In other words, if there are N nodes in the network, each node has to keep track of all previous transmission results from the remaining $N - 1$ nodes.

To reduce the memory requirement, the following two methods may be used. First, the model may ignore links with average PRR below a certain threshold. The model simply returns the probability of correctly receiving a frame without considering burstiness, which means that the model does not need to keep track of the history of that link.

For example, the model may choose to ignore all links that have average PRR less than 0.1. In practice, applying this method means that the model will exclude all links where the two nodes are too far apart.

The second method to reduce memory requirement is to purge the history whenever it is possible. For instance, all cache entries older than the cache TTL can be removed. Moreover, the history can be purged when the transmission results switched between success or failed since a burst is defined as consecutive successful or failed frame receptions.

4. ns-3 IMPLEMENTATION

Next, we present our implementation of the stochastic bursty link model in the ns-3 simulator. We start by presenting a quick overview of the standard WiFi module in ns-3 then proceed to explain our modification to incorporate the stochastic bursty link model.

The current WiFi model in ns-3 consists of multiple components working together. The two components that are directly related to our work are `WifiChannel` and `WifiPhy` [14]. Every WiFi device has its own `WifiPhy` while a single `WifiChannel` object serves as a channel that glues all `WifiPhy`s operating in the same wireless channel together.

The main responsibility of `WifiChannel` is to pass the signal between `WifiPhy`s when transmission occurs. Given the signal, each receiving `WifiPhy` can calculate the signal-to-interference-plus-noise ratio (SINR) of the receiving packet. Packet error rate (PER) is then calculated based on the SINR and other physical layer parameters. Finally, PER is used to determine if the transmission is successful or not.

To summarize, the current steps to determine a transmission result in ns-3 WiFi module are as follows.

1. Calculate SINR of the packet
2. Calculate PER from the SINR
3. Determine if the transmission is successful

To incorporate the stochastic bursty link model in ns-3, we made changes to the default WiFi module in ns-3. We introduce a new object called `BurstyHelper` to the WiFi module. `BurstyHelper` serves as the brain of the model with two important functions. First, `BurstyHelper` acts as a memory module by keeping track of history of transmission results between all `WifiPhy`s. Second, `BurstyHelper` is responsible for readjusting the PER when burst is detected. Every `WifiPhy` holds a pointer to the `BurstyHelper` object.

At the end of the reception, `WifiPhy` first calculates the PER based on the SINR. `WifiPhy` then consults `BurstyHelper` by calling `ReadjustPer`. `BurstyHelper` looks at the cache of previous transmissions from the sending `WifiPhy` to the receiving `WifiPhy` to see if there was a burst prior to the current packet. If there was a burst, `BurstyHelper` adjusts the PER by consulting the bursty probability adjustment function and returns the new PER to `WifiPhy`. `WifiPhy` then determines the result of the reception using the adjusted PER value. Finally, `WifiPhy` reports the result of the reception to `BurstyHelper` so that `BurstyHelper` can cache the result for future use.

Overall, the following changes were made to WiFi module.

1. A new object called `BurstyHelper` is included in the WiFi module
2. `BurstyHelper` keeps track of transmission history between all pair of `WifiPhy`s
3. `WifiPhy` stores a pointer to the `BurstyHelper` object
4. `WifiPhy::EndReceive` now takes a pointer to the sending `WifiPhy`
5. `WifiPhy` calls `BurstyHelper` to check if any adjustment to the default PER is required
6. `WifiPhy` determines the outcome of the reception and reports the result to `BurstyHelper`

The new steps to determine a transmission result in the modified WiFi module are as follows.

1. Calculate SINR of the packet
2. Calculate the default PER from the SINR
3. Look at the history of the transmissions
4. Adjust the PER if necessary
5. Determine if the transmission is successful
6. Save the result of the transmission

5. EVALUATION

We evaluate our stochastic bursty-link model by using the implementation in ns-3. We evaluate our stochastic bursty-link model in three different aspects. First and most importantly, we evaluate how well our model is able to replicate the bursty behavior observed in real wireless links. Second, we study how well our stochastic bursty-link model can simulate real wireless links with different bursty characteristics. Our goal is to study the difference between using a discrete BPA function and using a synthetic function to simulate wireless links. Finally, we study how our stochastic bursty-link model affects the routing protocol performance.

5.1 Trace Files Generation

As stated earlier, the most important goal of our model is to replicate bursty characteristics observed in real wireless links. To evaluate our model against real wireless links, we obtained trace files from an indoor office environment. We used three laptops and one desktop to gather traces files. BackTrack 5 R1 was installed on all devices. One laptop equipped with a wireless adapter based on the Atheros chipset [5] was selected as a packet sender. An application that continuously broadcasts UDP packets at the rate of 100 packets per second using 802.11g was installed on the packet sender. Transmission power and data rate were kept constant throughout the trace files collection. The remaining machines were used to capture the packets. One capturing laptop was equipped with Atheros-based wireless adapter while the other laptop was equipped with a RaLink-based wireless adapter [2]. The capturing desktop was equipped with a RaLink-based wireless adapter.

We created trace files using `tcpdump`. All wireless interfaces were switched to monitor mode with `airmon-ng` [1]. The trace files contain information about the received signal strength of each frame provided by the Radiotap header. All captures were done within the Klaus Advance Computing Building on the Georgia Institute of Technology campus with a duration of one hour per trace file. We obtained about 100 trace files for the simulations.

5.2 Evaluation Against Real Wireless Links

We evaluate two variations of our stochastic bursty link model: using discrete BPA functions directly analyzed from trace files and using a scaled-erf BPA function. We used `MatrixPropagationLossModel` as a base propagation loss model for both variations. We set the loss such that the average PRR of the link is equal to the PRR of each trace file. For the discrete BPA function, we followed the steps described in Section 3.1 to obtain the BPA function. First, we calculated cache TTL for each trace file by decreasing β sampling period until $|\beta| < 0.1$ and then used the TTL as a parameter to our model. Finally, the discrete BPA function of each trace file was obtained by analyzing the CPDF of the trace file.

For the scaled-erf BPA function, we manually tuned the two parameters of the scaled-erf function (s and σ) to match with the shape of the CPDF from the trace file. To obtain the TTL, we analyzed the 100 trace files. We varied the β sampling period of the 100 trace files and looked at the distributions of β's. At the sampling period of 10 ms (counting every packet), all 100 trace files had $|\beta| > 0.1$. When the sampling period decreased to 100 ms (counting every 10 packets), the number of trace files with $|\beta| > 0.1$ dropped to 28. At the sampling period of 500 ms (counting every 50 packets), only 4 trace files still had $|\beta| > 0.1$. Thus, we used the TTL of 500 ms for the scaled-erf BPA function. The value of 500 ms is also in agreement with the previously suggested value [20]. Note that this TTL is used in *all* scaled-erf BPA functions. In the case of discrete BPA function, the TTL is different for different trace files.

We compared the results of our model with the trace file and two other models: the default ns-3 model and BEAR [11]. For the default model, we used the `MatrixPropagationLossModel`. We set the loss such that the average received signal strength is equal to the average received signal strength of each trace file. We added a fading effect by adding a `RandomPropagationLossModel` with a `NormalRandomVariable` with mean 0 and variance corresponding to the trace file. For BEAR, the parameters were tuned using the received power traces from the trace files with the order of 3.

We presented the simulation results by using the β values. We selected a representative subset of the trace files with varying β values. All results from simulated links were averaged from 1000 simulations. The simulation results are reported in Figure 5.

As seen from Figure 5, our model can simulate burstiness of the trace file from very bursty links ($\beta \to 1$) to almost independent links ($\beta \to 0$). The β values of both the discrete BPA function variations and the scaled-erf BPA variations function differ from the trace files' only slightly. The β values of BEAR show that it can simulate a bursty link to a certain degree. However, the BEAR model is slower to change when compare to our model since it uses an auto-regression model to simulate links. The default LogDistance with fading model in ns-3 does not exhibit any bursty behavior as the β values are close to 0; that highlights its memory-less behavior. We show the detailed results from one of the trace files in Figure 6, Figure 7, and Figure 8. Figure 6 shows the PRR of the trace file and different simulated links. Figure 7 shows the CPDF of the trace file and the simulated links. Figure 8 shows the distribution of burst sizes of the trace file and the simulated links.

In this set, the average packet reception ratio of the trace file is about 0.67. As seen from Figure 6, the PRR of the default model remains relatively stable when compared to the trace files and other models. Figure 7 confirms that the CPDFs of both variations of our model closely resemble the CPDF from the trace file while the CPDF of the default model is almost flat. In other words, the randomness of the default model does not capture the bursty effect observed in a real network. Figure 8 shows the distributions of burst sizes of the trace file and the simulated links. The distribution of the burst size of our model is almost identical to the trace file while the distributions of burst sizes of the default model and BEAR are noticeably differ from the trace file. Note that the number of instances shown in Figure 8 are cumulative (e.g. burst size 3 is also counted in burst size 4).

5.3 Simulating Links with Different Bursty Characteristics

In this section, we compare between the two variations of our model. Specifically, we compare between using a discrete BPA function from a trace file and using a scaled-erf function. We would like to see how well the scaled-erf function simulates different CPDFs from trace files when compared to directly using discrete BPA functions from trace files.

To compare between the two methods, we selected four trace files with different CPDF shapes and show the results from simulations with discrete BPA functions and scaled-erf functions. The four sets of CPDFs are reported in Figure 9.

In Figure 9 (a), the trace file shows a slightly bursty behavior with small burstiness metric. Figure 9 (b) shows an example of a very bursty link where the probability shifts quickly depending on the frame reception history. In Figure 9 (c), the link also shows a slightly bursty behavior but with different shape from the link in Figure 9 (a).

As shown in Figure 9, our model with discrete BPA functions can simulate links with different bursty behaviors. The burstiness metrics of the discrete BPA functions are almost identical to the trace files. The scaled-erf model can simulate the links in Figure 9 (a), (b), and (c) well but cannot simulate the link in Figure 9 (d) due to its strange CPDF shape. The shape of the CPDF in Figure 9 (d) does not resemble any scaled-erf function. In this case, the scaled-erf function performs poorly while the discrete BPA function is still able to mimic the strange CPDF shape.

As noted earlier, one drawback of using a scaled-erf function instead of a discrete BPA is that, the function may not be able to simulate some CPDF shapes. The shapes of the CPDF simulated from the scaled-erf function will be more smooth than the real trace files as the function is not discrete. Thus, a discrete BPA function may be needed to simulate links with strange CPDF shapes.

5.4 Effect of the Simulated Bursty Link on Routing Protocols

In this section, we investigated the effect of our stochastic bursty-link model on routing protocols. Our goal was to study the effect on routing protocols when the bursty behavior is incorporated into the wireless simulation. We studied two scenarios with routing protocols: a diamond topology and a random network. First, we studied the diamond topology since the topology is simple and the effect of incorporating the stochastic bursty-link model can be easily observed. Next, we proceeded to study the random topology.

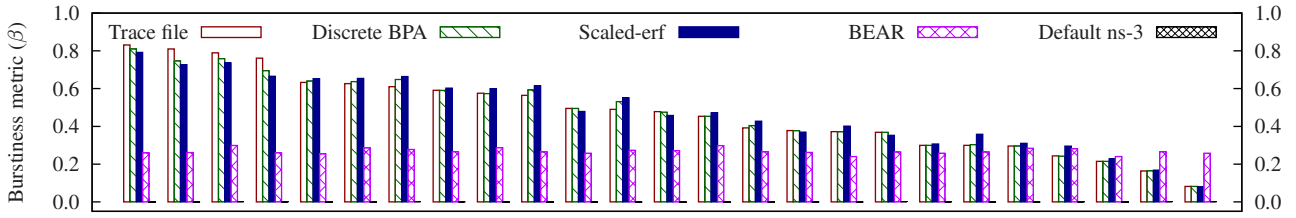

Figure 5: A comparison between simulated links and trace files.

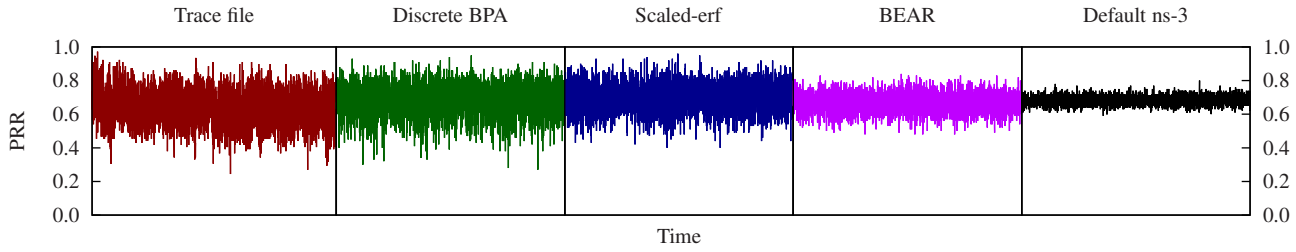

Figure 6: Packet reception ratio at different time for the trace file, our bursty-link model, BEAR, and the default ns-3.

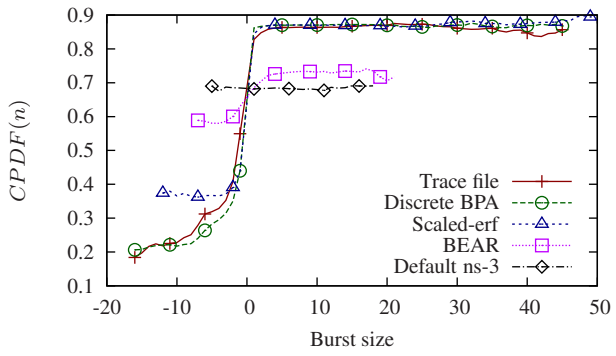

Figure 7: CPDF from one trace file along with its simulated counterparts from different models.

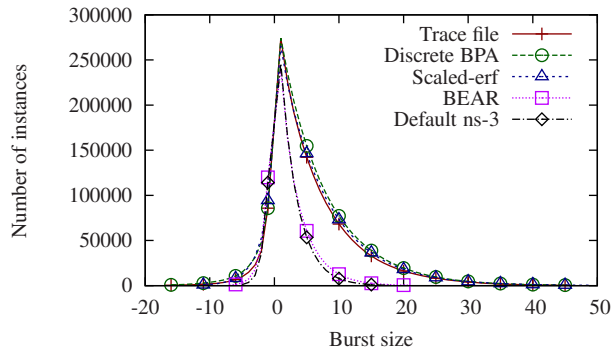

Figure 8: Distribution of burst sizes from the trace file and different models.

5.4.1 *Diamond topology*

We began the study with a network with four nodes placed in a diamond topology as shown in Figure 10. There is one source node s and one destination node t. Node u and v serve as forwarding nodes for routes between s and t. We manually set the propagation loss in the network such that there are only four possible links in the topology: $s \leftrightarrow u$, $s \leftrightarrow v$, $u \leftrightarrow t$, and $v \leftrightarrow t$. The loss on the four links, ϕ, can be set to different values to get different frame success rates. Other links were configured with infinite loss. To accomplish this, we used `MatrixPropagationLossModel` in ns-3 as the base propagation loss model. The topology was selected to ensure that s must reach t by using a routing protocol.

We compared the results between three models: the default ns-3, BEAR, and our stochastic bursty link model. The fading effect was added to the default ns-3 model using the `RandomPropagationLossModel`. For our model, we evaluate three different variations:

1. "Ideal bursty" – ideal bursty link ($b^+ = b^- = 0.2$),
2. "Positive" – positive burst only ($b^+ = 0.2, b^- = 0$), and
3. "Negative" – negative burst only ($b^+ = 0, b^- = 0.2$).

The positive burst only model is a model where the probability of correctly receiving a frame increases when a positive burst occur and the probability resets to the default value when a negative burst occur. In other words, the quality of the link in the positive burst only model is always at least as good as the default link. For the negative burst only model, the link is always at most as good as the default link. The positive burst only model and the negative burst only model are not representatives of real wireless link, but are included for comparison purpose.

To observe the behavior of different models under different network conditions, we varied ϕ such that the frame success rate of the links is between 0.1 (very bad links) and 0.9 (very good links). The UDP application on the source node generates packets at the rate of 20 packets per second. We reported simulation results using packet reception ratio. All results reported are averaged from 1000 simulations. The simulation results are reported in Figure 11. For better readability, the confidence intervals are not shown since the intervals are very small.

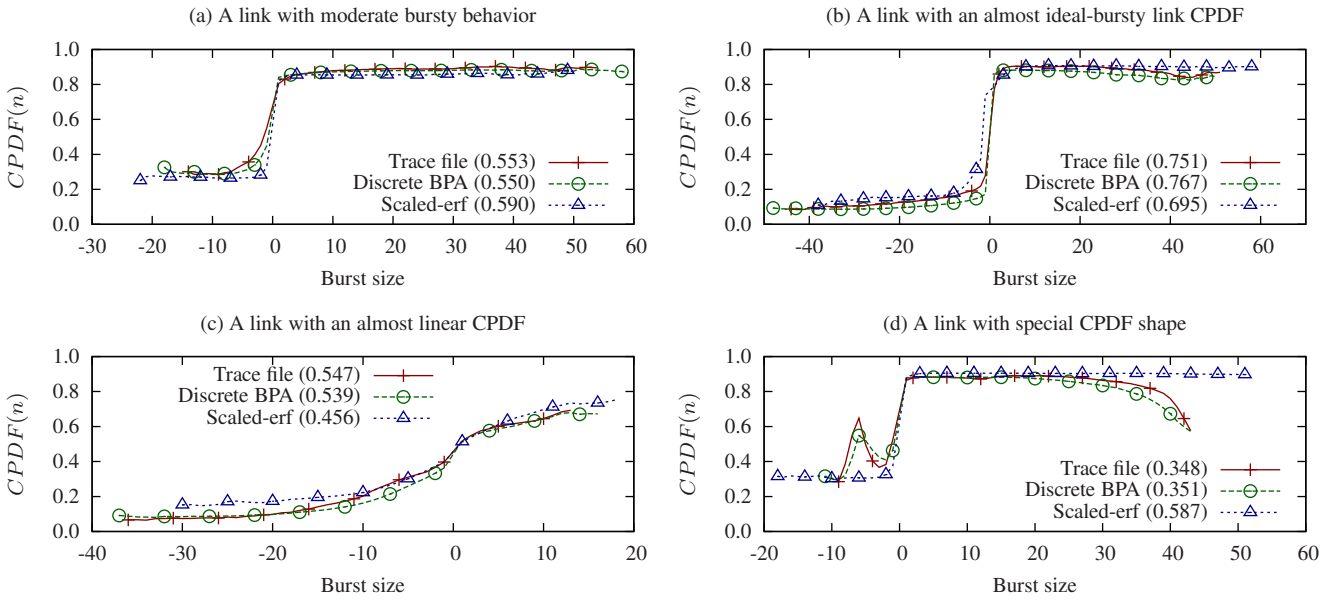

Figure 9: Using a scaled-erf function to simulate links with different CPDF shapes. The number inside the brackets are the burstiness metric (β).

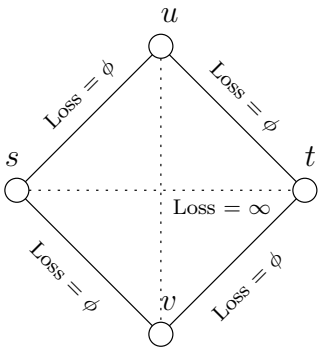

Figure 10: A simple diamond topology used to study performances of different models.

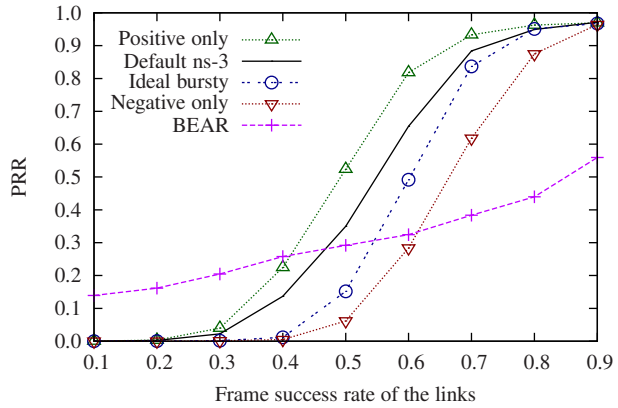

Figure 11: Packet reception ratios of different link models under varying link quality.

As seen from Figure 11, packet reception ratios of all models increase as the link quality improves, which is expected. The positive burst only model has higher PRR than the default ns-3 since the quality of the link increases when positive bursts occur but the link quality never drops below the default values. On the other hand, the link quality of the negative burst only model is always lower than the default ns-3, resulting in lower PRR than the default model. However, the effect of positive burst only and negative burst only are *not* equivalent. The results show that negative bursts have more impact on the PRR than positive bursts. This behavior is expected since DSR takes time to find a new route to the destination when the current route is broken. From DSR's perspective, there is not much benefit from a positive burst other than shorter delivery delay (no retransmission is required at the MAC layer). However, when a negative burst occurs, a DSR route may break and must be re-established. This behavior can be observed in the ideal bursty link model ($b^+ = b^-$) where the effects of negative bursts outweigh the effects of positive bursts.

5.4.2 *Random topology*

Finally, we turn our attention to a random network. In this study, we randomly place 50 static nodes in the deployment area of 500 m by 500 m. Two nodes are selected as a source-destination pair where the distance between the two nodes is at least 300 m to ensure that the two nodes use routing to reach each other. The application on the source node generates packets at the rate of 20 packets per second for 480 seconds. We ran two sets of application: one with UDP and one with TCP.

Again, we compare the results between three models: the default ns-3 model, BEAR, and the three variations of our bursty link model. We use the `LogDistancePropagationLossModel` provided by ns-3 with a path-loss exponent of 3 as the base propagation loss model. We report the simulation results in two aspects: the packet reception ratio and the number of route breaks. All simulation results are averaged from 1000 simulations and reported with 95% confidence interval. UDP simulation results are reported in Figure 12 and TCP simulation results are reported in Figure 13.

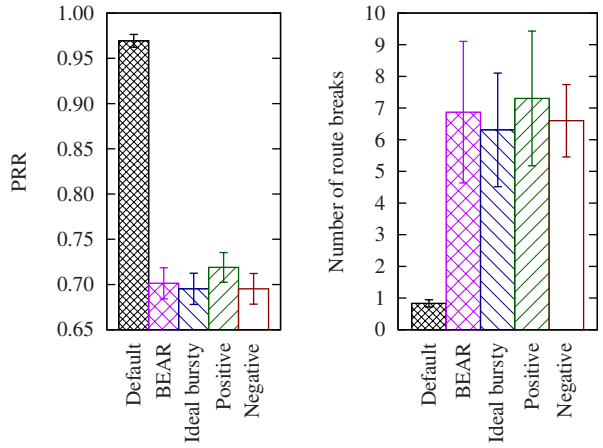

Figure 12: PRR and the number of route breaks of DSR simulation under different link models with UDP application.

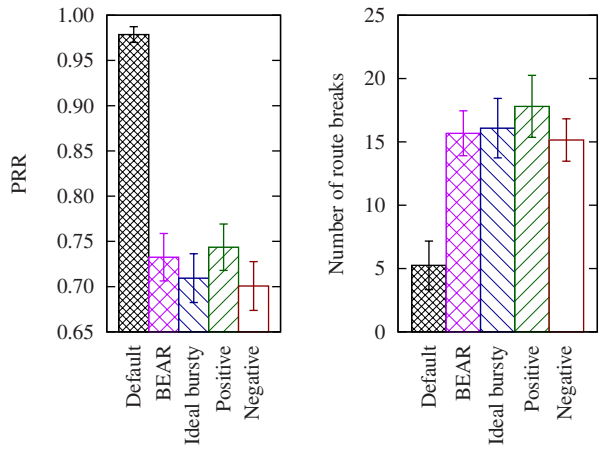

Figure 13: PRR and the number of route breaks of DSR simulation under different link models with TCP application.

As seen in Figure 12 and Figure 13, the default model has the highest packet reception ratio with almost 100% delivery. All models with memory effect have substantially lower packet reception ratios. The lower packet reception ratios of other models result from more frequent link breakage during the simulations. The simulation results show that the performance of DSR is significantly different when bursty behavior is incorporated into the simulation. All models with memory experienced about 4 to 7 link breaks with UDP application and about 14 to 15 link breaks with TCP application. The default ns-3 model experienced only about 1 link break with UDP application and about 4 link breaks with TCP application. The more frequent route breaks mean that DSR has to re-initiate the route discovery process more often. The smaller number of route breaks of the default ns-3 model results in higher PRR but the results are not realistic. This is in general agreement with published results from DSR in a real wireless ad hoc network testbed, which showed that route breaks occurred much more frequently even in a well-planned network as small as two hops [17].

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new stochastic frame-level bursty-link model for wireless network simulation. Our model simulates bursty behavior in wireless links by changing the probability of correctly receiving a frame based on the history of the wireless link. Our model can directly simulate a real wireless link by modeling the bursty characteristics from the trace file or simulate a bursty link by using predefined functions. We have implemented the model in ns-3 simulator and showed that our model is able to replicate bursty behavior observed in real wireless links. We also comprehensively studied the effect of using our stochastic bursty-link model on the routing protocol performance and showed that the routing protocol performance was significantly affected by the bursty behavior of the wireless links.

Even though we have shown that our stochastic bursty-link model is able to simulate variety of bursty characteristics observed in real wireless links, the research in this area is still far from complete. Further refinements to our model are possible. We are currently investigating the possibility of using different bursty probability adjustment functions based on the time when the last burst occurred.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant CNS-0958015.

7. REFERENCES

[1] AirCrack-ng. `http://www.aircrack-ng.org`.

[2] MediaTek. `http://www.mediatek.com`.

[3] The ns-2 network simulator. `http://nsnam.isi.edu/nsnam`.

[4] The ns-3 network simulator. `http://www.nsnam.org`.

[5] Qualcomm Atheros. `http://www.atheros.com`.

[6] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 121–132, New York, NY, USA, 2004. ACM.

[7] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '05, pages 414–425, New York, NY, USA, 2005. ACM.

[8] X. Chang. Network simulations with opnet. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 307–314 vol.1, 1999.

[9] V. Erceg, L. Greenstein, S. Tjandra, S. Parkoff, A. Gupta, B. Kulic, A. Julius, and R. Bianchi. An empirically based path loss model for wireless channels in suburban environments. *IEEE Journal on Selected Areas in Communications*, 17(7):1205–1211, 1999.

[10] H. Friis. A note on a simple transmission formula. *Proceedings of the IRE*, 34(5):254–256, 1946.

[11] D. Gómez, R. Agüero, M. García-Arranz, and L. Muñoz. Replication of the bursty behavior of indoor wlan channels. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, pages 219–226, 2013.

[12] C. Jiao, L. Schwiebert, and B. Xu. On modeling the packet error statistics in bursty channels. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, pages 534–541, 2002.

[13] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 78–82, New York, NY, USA, 2004. ACM.

[14] M. Lacage and T. R. Henderson. Yet another network simulator. In *Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator*, 2006.

[15] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *6th International Symposium on Information Processing in Sensor Networks. IPSN 2007.* , pages 21–30, 2007.

[16] G. Lui, T. Gallagher, B. Li, A. Dempster, and C. Rizos. Differences in RSSI readings made by different wi-fi chipsets: A limitation of wlan localization. In *2011 International Conference on Localization and GNSS*, pages 53–57, 2011.

[17] D. Maltz, J. Broch, and D. Johnson. Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed. In *2000 IEEE Wireless Communications and Networking Confernce. WCNC.* , volume 3, pages 992–997 vol.3, 2000.

[18] M. Mushkin and I. Bar-David. Capacity and coding for the gilbert-elliott channels. *IEEE Transactions on Information Theory*, 35(6):1277–1290, 1989.

[19] D. Niculescu. Interference map for 802.11 networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 339–350, New York, NY, USA, 2007. ACM.

[20] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: measuring wireless link burstiness. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 29–42, New York, NY, USA, 2008. ACM.

[21] M. Stoffers and G. Riley. Comparing the ns-3 propagation models. In *2012 IEEE 20th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 61–67, 2012.

[22] A. Vlavianos, L. Law, I. Broustis, S. Krishnamurthy, and M. Faloutsos. Assessing link quality in IEEE 802.11 wireless networks: Which is the right metric? In *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC 2008.* , pages 1–6, 2008.

[23] K. Wolter, P. Reinecke, T. Krauss, D. Happ, and F. Eitel. Ph-distributed fault models for mobile communication. In *Proceedings of the Winter Simulation Conference*, WSC '12, pages 429:1–429:12. Winter Simulation Conference, 2012.

[24] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *12th Workshop on Parallel and Distributed Simulation. PADS 98. Proceedings.*, pages 154–161, 1998.