Software Architecture For Scalable Multi-hop Mobility Emulation With Indoor Wireless Grids

Gautam Bhanage and Yanyong Zhang
Wireless Information Network Labarotory (WINLAB)
Electrical And Computer Engineering,
Rutgers University, NJ 08901, USA
(gautamb, yyzhang)@winlab.rutgers.edu

Abstract

Wireless multi-hop mobility emulation on an indoor stationary testbed requires tools that can create arbitrary topologies and adapt network connectivity. We begin with a brief comparison of current options for mobility emulation, and describe the need for a better emulation mechanism. Based on these requirements we implement the bursty PER filtering engine in the linux kernel. We further build on this engine and implement our discrete and streamline mobility models. Through this study we describe our methodology and finally present a case study to demonstrate the effectiveness and the ease of our approach.

Index Terms

Mobility emulation, MANET Testbed architecture.

1. Introduction

Baseline testing of research ideas is usually done through simulations as they allow for a tight design and feedback loop. More detailed performance evaluation, however, is preferably done through testbed emulations or deployments on an actual platform. Mobility emulation within multi-hop networks on a space constrained testbed requires a suitable technique for topology creation and run time topology adaptation. Wireless topologies can be emulated either by wired or wireless testbeds. For example, the EMPOWER emulator [18], is one technique that models a wireless mobile node on a wired testbed. The downside of emulating wireless links using wired networks leaves out the details from the physical and the medium access layer such as the effects of propagation, interference, multi-path, physical layer capture, and interactions of the MAC itself. We will compare a few prominent approaches to topology creation on a wireless testbed and explore their possibilities for mobility emulation.

Though emulation of networks on a wireless testbed seems to be a better alternative, mapping real world multi-hop topologies on space constrained testbeds in an efficient and accurate manner presents a significant challenge. The following techniques have been proposed to achieve controlled connectivity in wireless broadcast domains: (1) Transmit Power control, (2) Physical shielding/Attenuation, (3) Noise generation, and (4) MAC filtering. Transmit power control suffers from poor control with COTS WLAN cards (E.g. Atheros 5212 chipset), where minimum transmit power is $1mW$ and sensitivity is getting better (up to $-92dBm$). Hence even with an indoor path loss exponent of 3, it is difficult to create topologies by just controlling the transmit power. Attenuation provides an excellent approach to create topologies and emulate mobility, with the only problem being that they do not scale well. Clancy et. al [20] show that it is possible to create multi-hop topologies with controlled attenuation through programmable RF matrix switches. The MiNT testbed [9] has a similar approach to topology creation with RF attenuation and power control. Both solutions share the drawback of either incurring high cost due to the use of RF matrix switches and programable attenuation or severely limiting the number and type of emulated topologies with fixed attenuators without human intervention. Increasing the noise floor in certain parts of the network provides another way of topology emulation [14]. An exhaustive search is needed to find the appropriate set of nodes whose PER values between the sender and the receiver are within the specified range. The discussion in [13] shows that there is a limitation on the number and type of topologies that can be created through this approach. [14] shows it is possible to create simple string topologies with 4 to 6 hops, but creation of arbitrary topologies is still very difficult, especially with the limited number of noise sources.

On-off MAC filtering allows the link to be emulated as either up or down. Software approaches proposed in [22] and [16] use this method for mobility emulation but both of them lack the ability to model partial link failure for fine-grained control. Similarly, the APE testbed [12] emulates mobility with humans carrying laptops along particular paths, which does not permit scaling or remote execution. Mobility emulation on static nodes has been attempted in an earlier study by spatially switching the mobile node and

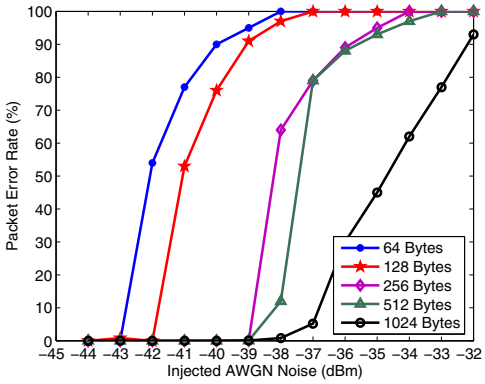

Figure 1: Experimental evaluation to test the performance of the noise injection mechanism for link degradation with varying packet sizes.

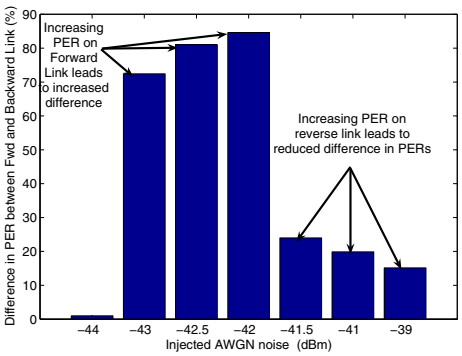

Figure 2: Emulation of asymmetric links by measurement of difference in PER between forward and reverse link. Plot is a function of the noise injected at one antenna near the receiver of the forward link

traversing it through a region highly affected by noise [19]. Apart from the need for node switching at run time, it is not very clear as to how a real world (controlled link PER) mobility scenario may be mapped to the wireless testbed.

To do a sanity check, we performed a simple experiment with arbitrary node *A* sending UDP traffic to node *B* with noise injection deployed at the receiver. Figure 1, shows that we need varying noise levels for varying packet sizes to achieve the same link PER and for most packet sizes the PER changes from 0 to 100% within $5dB$ of injected noise. Figure 2, shows the asymmetricity of links achievable with noise injection at the node *B*. It shows that as the noise is increased at the receiver beyond a threshold, it starts affecting the reception at node *A*. Both these experiments show it is difficult to control link PER with the use of current noise injection setup.

Based on the comparison of the existing topology generation and mobility techniques for wireless grids, we are still in need of an approach that allows: flexible node mapping, is reproducible, works without human intervention, and is affordable. The rest of the paper is organized as follows. Section 2 describes our approach to controlling link connectivity. Section 3 provides details of the testbed architecture and Section 4 highlights preliminary results obtained from the testbed. Section 5 shows an application of the mobility emulation apparatus. Finally, Section 6 provides the concluding remarks.

2. PER Based Packet Filtering

Standard MAC filtering relies on discarding all MAC frames seen from a particular source MAC address based on a black list. This is popular since it does not require external hardware (noise injection) or man power (moving RF shields) nor incurs extra cost for experimentation. Drawback of this approach lies in the fact that the emulated links are binary, either ON (PER of 0%) or OFF (PER of 100%). Hence, we call the conventional approach *on-off* MAC filtering. From a mobility emulation perspective, we will not be able to emulate intermediate PERs with this approach. This puts a significant limitation in terms of the accuracy of the emulation, specially since previous studies [4] have shown that intermediate PERs can and do occur in wireless links. Here we show how our *PER based packet filtering* scheme proposed in [3] extends *on-off* MAC filtering technique to lay the ground work for mobility emulation.

2.1. Basics

The key idea is to emulate partial packet losses on links by dropping MAC frames in direct proportion to the desired link PER. This mechanism is implemented as a kernel module based on [15]. Each interface on a node can now independently administer packet filtering at different filtering rates for each MAC address. The modified kernel patch was implemented and tested on a Linux machine running a debian distribution with the 2.6.12 kernel[1]. To ensure efficient implementation, special care was taken while using critical path instructions. The kernel patch requires *netfilter* support to be enabled and compiles as a kernel module.

2.2. PER Pattern

For realistic emulation, packet drops produced by our system should closely approximate those recommended by channel models. Considerable number of channel error models rely on i.i.d distribution of channel errors. However, further research revealed that channel errors are bursty in

1. A slightly older kernel version is being used for compatibility with other pre-installed software. The kernel patch can be ported with minimal effort to later kernel versions.

nature due to inherent attenuation, inter-symbol interference (ISI), doppler shift and multipath fading [2]. Since initial efforts by Gilbert [11] and Elliot [10], bursty channel errors have been modeled by a simplified two state Markov model [21], where one of the states indicates an error and the other indicates a successful transmission. To avoid substantial packet processing overhead in the kernel, instead of implementing a Markov model we use mean error lengths and error free periods for controlling error burstiness. In fact, a trace based evaluation of channel error behavior in [17] evaluated the mean and standard deviation of packet error probability, error length, and error free lengths. Hence using similar average value estimate we should be able to approximate the channels packet drop patterns.

Link PER is specified as a fraction $\frac{N_{dropped}}{N_{dropped}+N_{passed}}$, where $N_{dropped}$ is the number of packets that will dropped in a sequence, and N_{passed} is the number of packets that are passed. For example, with $N_{dropped} = 10$ and $N_{passed} = 30$, the node will drop 10 packets in a row, and then pass the next 30 packets. While maintaining the same PER, the user can control the size of the burst by varying the values of $N_{dropped}$ and N_{passed}. For example, a PER of 10% can be achieved by dropping 1 packet out of every 10 packets or by dropping 10 packets out of every 100, but the implications of these two options are completely different when ad hoc routing is used. If the routing protocol loses only one packet in ten, it could still populate routing tables, but loss of 10 consecutive control messages from a node could significantly degrade the performance. Filtering settings for all the interfaces and source addresses can be checked by the *procfs* entries under */proc/net/mackill*.

2.3. Applicability

PER based packet filtering focuses on creating large-scale complex network topologies, and some low level details may be lost for this reason. Specifically, we would like to emphasize that this scheme is more suitable when wireless protocols above MAC layer are studied, and care must be taken with both PHY and MAC layers.

First let us look at the problems with the PHY layer. Consider a network consisting of four nodes: *A*, *B*, *C* and *D* with connectivity as: $A \longleftrightarrow B \longleftrightarrow C \longleftrightarrow D$, where A and D cannot hear from each other. However, mapping this topology on a testbed may result in all four nodes falling in a single collision domain. It is noted that even though MAC filtering enforces the topology by dropping frames, the nodes are still within each other's interference range, leading to a decreased aggregate throughput and a noticeable increase in delay. MAC filtering techniques will also lose some unicast MAC-layer interactions. Since ACKs and re-transmissions for unicast transmissions happen in the hardware, it is impossible to model the effect of these in software.

3. Mobility Architecture

Our PER filtering based scheme supports two approaches to mobility:

- Snapshot mobility
- Streamline mobility

Snapshot mobility model is one of the most popularly supported approaches where experimentation and evaluation is performed with discrete network states. E.g. An experiment where a node moves from an arbitrary point *A* to a point *B* might be performed by sampling system performance at intermediate points *C* and *D* along the trajectory between *A* and *B*. However, if we want to measure the system performance as the node moves from *A* to *B*, over time *t* secs, snapshot mobility does not help. Our unique streamlined mobility approach permits the experimenter to evaluate system performance in a timed mobility experiment. Such an approach is typically useful since it allows the experimenter to measure the transient response of the system (E.g. how do certain routing protocols react to a particular rate of change of topology).

Further in this section we will briefly discuss, the implementation of each of the mobility approaches and the system architecture for allowing automatic, repeatable mobility experiments.

3.1. Snapshot Mobility

Connectivity of a topology with *N* nodes is represented by a $N \times N$ PER-link state matrix L_{per}. An entry $l_{i,j}$ in the matrix shows the *PER* between nodes *i* and *j*. Calculating the L_{per} is a three step process. We start by evaluating the next discrete position of the node by applying standard mobility models like the random waypoint, reference point group mobility, or any other model [6]. Using new node coordinates we then evaluate the Euclidean distances. Finally, we can either use propagation models or real measurements to map calculated distances to link PER. This process should be performed iteratively to evaluate the network state at all snapshot points.

3.2. Streamlined Mobility

Goal of the streamline mobility model is to allow experimenters to set experiment duration and measure overall system performance. This also allows node mobility emulation at independent velocities. We can change the PER patterns with time to emulate packet drops that are caused by the mobile node's relative speed. To emulate mobility for node *i* we vary all entries $l_{i,j}$ and $l_{j,i}$ in L_{per} for every node *j* connected to node *i* as a function of time and other channel parameters such as modeled path loss, fading and interference. Granularity of such discrete mobility is

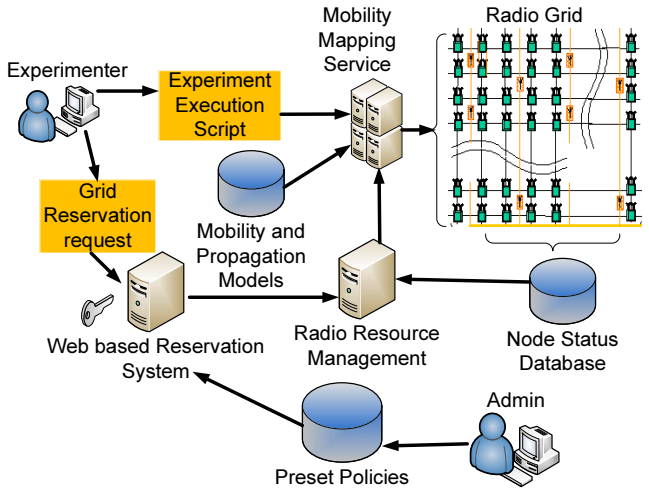

Figure 3: Proposed software architecture for topology creation and mobility mapping for automatic, repeatable experimentation on the ORBIT grid.

dependent on the sampling frequency of the matrix L_{per}. Rate of change of $l_{i,j}$ and $l_{j,i}$ is directly proportional to the relative velocities of nodes i and j.

3.3. Network Architecture

We have implemented our topology creation and mobility emulation setup on the ORBIT radio grid [8]. The setup consists of a grid of 400 802.11 radio nodes with reproducible wireless channel models. Every node is a small form factor PC with 1GHz Via C3 CPU, 512 MB RAM, 20 GB hard disk and three ethernet ports, one of which is used for node configuration and control. Majority of the ORBIT nodes are fitted with Atheros 5212 based IEEE 802.11 a/b/g cards while remaining have Intel cards. We used Atheros cards for our experimentation.

Figure 3 describes the broad network setup of our proposed experimentation architecture. Components of our software architecture are implemented with C++/perl/shell scripts. The experimenter provides the experiment parameters through a control script after making a web reservation of the grid. The system estimates the grid usage, maps the topology on the nodes, configures wireless interfaces, starts routing and data flows while also maintaining network topologies and dynamically controlling link PERs. Logs are collected and parsed by the execute script. We are able to store and reproduce all of our topologies.

4. Results

We will provide some representative results for showing system performance with snapshot and streamline mobility.

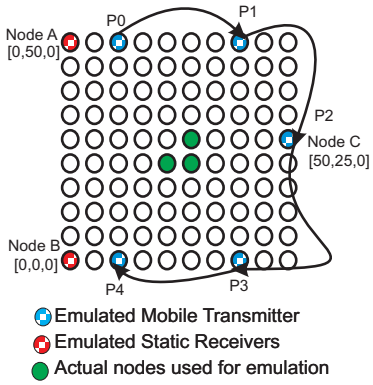

Figure 4: Nodes and setup for mobility emulation. Trajectory of the emulation and virtual positions are shown by nodes *A*, *B* and *C*.

4.1. Snapshot Emulation

In this experiment we test link performance at discrete points in time or space. Assume a real world scenario as shown in Figure 4 that needs to be mapped to static nodes, in which *A* and *B* are the static nodes, while *C* denotes the mobile node traveling from virtual position *P0* through *P4*. As mentioned earlier PER at each of the positions may be modeled based on:

1) Actual Measurements: Measurement traces from the field could serve as guidelines for modeling.
2) Propagation Models: In the absence of absolute measurements, standard propagation models may be used to evaluate power (and hence PER) at the receiver. Effects of other properties such as channel fading and interference can also be modeled in terms of PER.

We use the latter approach in this study. For the sake of simplicity, we adopted a model that assumes a linear relationship between PER and distance. Note that this relation may be easily changed when evaluating realistic settings.

Figure 5 shows the observed PERs at the two receivers when the sender's position moves. The resulting PER values are as anticipated from such a configuration. For example, initially (at *P0*) when the transmitter is closer to *A* than B, *A* experiences a lesser PER than *B*. As the relative position of the transmitter changes we see difference in PERs at the receivers.

4.2. Streamlined Mobility

To demonstrate streamline mobility we consider a emulated mobile transmitter (*A*) and a static receiver *B* where the mobile transmitter moves away from *B* in a straight line and eventually after traveling a constant pre-determined distance returns to its original position via the same path. In this experiment we measure performance of the link between *A* and *B* as a function of time. The rate of change of link PER

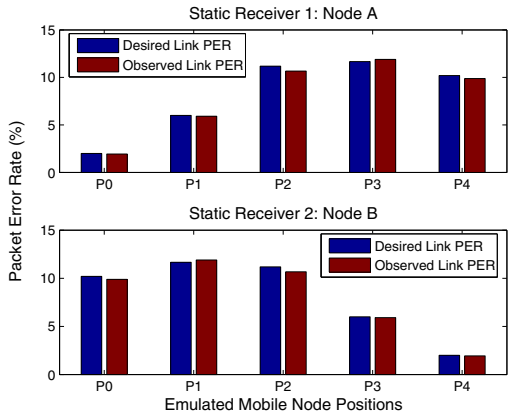

Figure 5: Expected and observed PER at Receivers 1 and 2 with the mobility setup. Receiver 1 is closer to the initial position *P0* and sees a lower PER. As node *C* travels to *P4* PER for node *A* deteriorates. PER seen at Receiver 2 is the opposite of that seen at node *A*

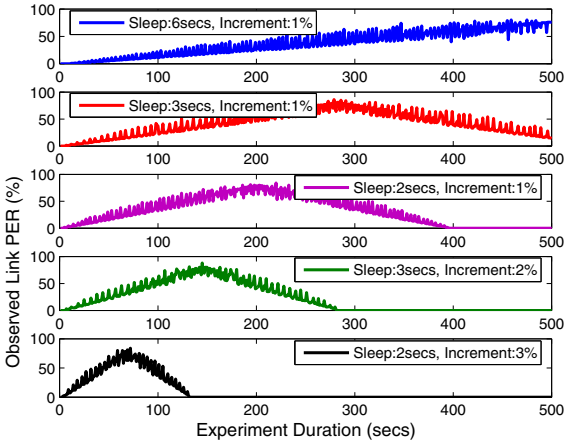

Figure 6: Performance with varying relative velocity of transmitter with receiver. All reception is assumed to be line of sight.

between the nodes will be a function of the velocity of *A*. For the sake of emulation, we assume that the maximum link PER (at the farthest point that *A* travels) is 80%.

Figure 6 shows the measured link PER at node *B* for different velocities of *A*. The topmost subplot shows the slowest movement which is modeled by increasing link PER by 1% every 6 seconds. Similarly, the lowest subplot shows the fastest movement modeled by a 3% increase in PER every 2 seconds. Link rates are varied automatically at run time by scripts on the receivers. Again, it is important to note that these results are indicative and full benefit of this approach could be achieved by evaluating transient response of the system like that of adhoc routing schemes with different group mobility models.

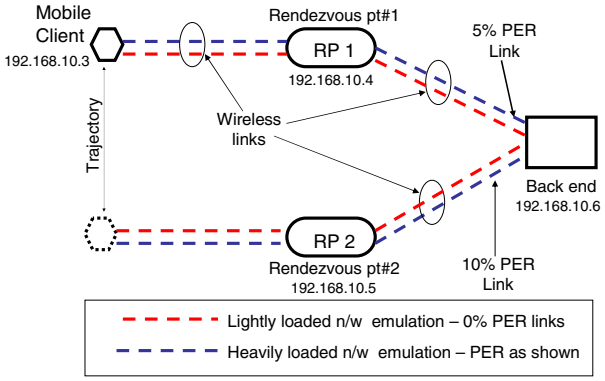

Figure 7: OLSR performance evaluation with the emulated mobility scenario. Mobile node moves between the two rendezvous points which are a part of a mesh connecting to the back end. The network performance of the mesh itself is considered under two conditions as heavily and lightly loaded.

5. Case Study: Routing Protocol Evaluation

Considerable research efforts have been diverted towards evaluating the performance of MANET routing protocols [5], [7]. Comparison of these routing protocols are done across a wide range of criterion ranging from stability to scalability. Through this case study we will show the application of our mobility emulation mechanism for routing protocol evaluation.

5.1. Setup

The emulated scenario is as shown in the Figure 7. The setup consists of a back end which connects to the infrastructure. There are two rendezvous points (RPs) which have a wireless link to the back end. The low power mobile node connects to either of the RPs depending on its proximity to them. We measure the performance of the OLSR [1] routing protocol by varying the rate at which the mobile node moves between the two RPs thereby allowing us to see how quickly the route stabilizes. The performance is also dependent on the load at the RPs, thereby deciding the link quality to the back end. For emulating this, we will consider the performance when the system is lightly and heavily loaded. All non-zero link PERs are as shown in the Figure 7. Note that it is possible to do a comprehensive evaluation and comparison across protocols by enhancements such as emulating time varying links even from the RPs and measuring average performance. We do not include these for the sake of brevity.

5.2. OLSR Performance

Topology control and mobility emulation is achieved using mechanisms described here on the ORBIT radio grid.

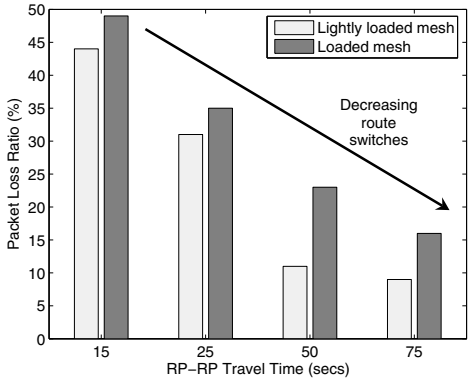

Figure 8: Measured packet drop rate between the mobile node and the back end. Lower travel times indicate a higher velocity of the mobile node between the rendezvous points and correspondingly more route switches.

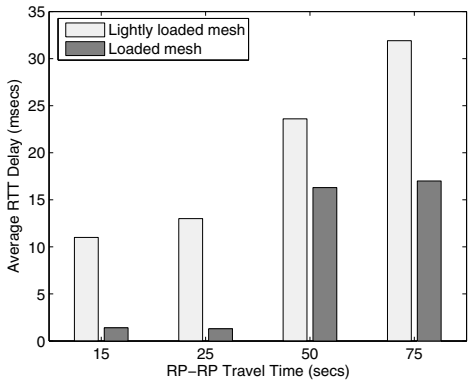

Figure 9: Measured RTT delay between the mobile node and the back end. The mobile node is moving between the two rendezvous points with varying velocities given by the travel times.

We measure the performance of the routing protocol by testing the end to end link conditions in terms of packet loss ratio (PLR) and mean delay. Network conditions are emulated to test performance below saturation. The mobile transmitter while sending packets to the back end moves between the two RPs causing route changes. $RP-RP$ travel time is the time taken by the mobile node to travel from one RP to another. Hence lesser travel times indicate higher node velocities.

Figure 8 shows the observed link packet loss rate (PLR) with different travel times of the mobile node. The PLR results follow intuition that when the node is moving fastest, we have more route changes and hence a higher PLR. The lightly loaded system has a lower PLR as compared to the heavily loaded system under similar mobile node movements. Another important observation is that for higher mobile node velocities, PLR performance is dominated by route changes (which are same for both lightly and heavily loaded systems) resulting in similar performance. However, as the number of route changes decrease with velocity, difference in performance is increased and dominated by packet drops between the RPs and the back end.

Figure 9 shows the average round trip delay between the mobile transmitter and the back end. Measurements show that for a lower PLR, we measure more packets (including ones which are backlogged) with higher delays which would have been usually dropped. Hence as the node velocity decreases with increasing travel times between RPs, PLR decreases and delay increases. Similar trend is observed between the lightly and heavily loaded links between the RPs and the back end.

Thus, this study provides a good baseline and can be easily extended to perform large scale comparison of routing protocol performance for MANETs.

6. Conclusion and Future Work

We proposed and implemented the Bursty PER filtering scheme which is capable of recreating topologies and mobility scenarios from real life. We show that our snapshot and streamlined approach to mobility emulation provide a realistic solution while permitting remote execution, repeatability, and ease of use. Our architecture is currently being evaluated and used as a basis for large scale performance evaluation of ad-hoc routing protocols for tactical MANET.

References

[1] OLSR routing protocol. $http://www.olsr.org$.

[2] H. Bai, H. Aerospace, and M. Atiquizzaman. Error modeling scheme for fading channels in wireless communications. IEEE Com. Surveys, vol. 5, No.2, Forth quarter, 2003.

[3] G. D. Bhanage, Y. Zhang, and I. Seskar. On topology creation for an indoor wireless grid. In *ACM WINTECH*, pages 81–88, 2008.

[4] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 31–42, 2005.

[5] E. Borgia. Experimental evaluation of ad hoc routing protocols. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 232–236, 2005.

[6] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[7] I. Chatzigiannakis, S. E. Nikoletseas, and P. G. Spirakis. Analysis and experimental evaluation of an innovative and efficient routing protocol for ad-hoc mobile networks. In *Algorithm Engineering*, pages 99–110, 2000.

[8] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *IEEE Infocom* , March 2005.

[9] P. De, A. Raniwala, S. Sharma, and T. Chiueh. MiNT: a miniaturized network testbed for mobile wireless research. *Proceedings of IEEE (INFOCOM)*, pages 2731–2742, Mar. 2005.

[10] E. Elliott. Estimates of Error Rates for Codes on Burstynoise Channels. *Bell system technical journal*, 42:1977–97, Dec. 1963.

[11] E. Gillbert. Capacity of a Bursty-Noise Channel. *Bell system technical journal*, pages 1253–65, 1960.

[12] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordstrom, and C. Tschudin. A large-scale testbed for reproducible ad hoc protocol evaluation. In *Proceedings of IEEE WCNC*, March 2002.

[13] J. Lei, R. Yates, L. Greenstein, and H. Liu. Wireless link SNR mapping onto an indoor testbed. In *Proceedings of IEEE Tridentcom*, Feb 2005.

[14] S. Kaul, M. Gruteser, and I. Seskar. Creating Wireless Multi-hop Topologies on Space-Constrained Indoor Testbeds Through Noise Injection. *Tridentcom*, 4:2731–2742, Mar. 2006.

[15] H. Lundgren. Mackill: Topology Creation Utility. Open source software from Upsalla university, 2003.

[16] J. Macker, W. Chao, and J. Weston. A low-cost, IP-based mobile network emulator (MNE). *in Proceedings of Military Communications Conference*, pages 481–486, Oct. 2003.

[17] G. Nguyen, R. Katz, B. Noble, and M. Satyanarayanan. A Trace-Based Approach for Modeling Wireless Channel Behavior. *Proc. Winter Simulation Conf.,*, pages 597–604, Dec. 1996.

[18] L. Ni and P. Zheng. Empower: A network emulator for wireline and wireless networks. *IEEE InfoCom*, 2003.

[19] K. Ramachandran, S. Kaul, S. Mathur, M. Gruteser, and I. Seskar. Towards Mobility Emulation Through Spatial Switching on a Wireless Grid. . *ACM E-WIND Workshop, Philadelphia, PA*, 2005.

[20] T. Clancy and B. Walker. MeshTest: Laboratory-Based Wireless Testbed for Large Topologies. In *IEEE TridentCom*, May 2007.

[21] J. Yee and E. Weldon. Evaluation of the Performance of Error Correcting Codes on a Gilbert Channel. *IEEE Transactions on Communication*, 43:2316–23, Dec. 1995.

[22] Y. Zhang and W. Li. An integrated environment for testing mobile ad-hoc networks. *In* Proceedings of ACM MobiHoc, June, 2002.