Towards Large-Scale Mobile Network Emulation Through Spatial Switching on a Wireless Grid

Kishore Ramachandran, Sanjit Kaul, Suhas Mathur, Marco Gruteser, Ivan Seskar
WINLAB / Electrical and Computer Engineering Department
Rutgers, The State University of New Jersey
94 Brett Rd
Piscataway, NJ 08854
{kishore,sanjit,suhas,gruteser,seskar}@winlab.rutgers.edu

ABSTRACT

Experimentation with large mobile networks is notoriously tedious and expensive. We present the architecture and work-in-progress implementation of the m-ORBIT testbed, a mobility emulator using spatial switching, which facilitates mobile system experiments with 802.11a/b/g wireless network interfaces. The emulator does not require any physically moving parts—it emulates mobility by switching over an array of 128 spatially distributed radios. Instead of using hardware antenna switches, we implement spatial switching in software over Gigabit Ethernet links to the radio nodes. Preliminary results support the scaling of this approach to a large number of radios at relatively low cost. Packet error rate measurements also indicate that an experimenter can create multi-hop topologies by injecting additive white Gaussian noise into the environment. We demonstrate through an Ad hoc On Demand Distance Vector routing case study how this emulator enables mobile systems experiments and plan to make the emulator available for remote access by the research community.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design–wireless communication

General Terms

Design, Experimentation

Keywords

Mobility emulation, testbed, spatial switching, noise injection

1. INTRODUCTION

Several factors make experimental research with mobile systems particularly tedious. Mobile systems inherently require node mobility, which is difficult to achieve without organizing a team of human assistants [1]. Radio interfaces based on the IEEE 802.11 standard, often the radio of choice for a mobile network, have a range of up to several hundred feet. Therefore, experiments must be conducted in an area large enough to allow nodes to move beyond each others radio range. Moreover, changing radio channel characteristics over time lead to poor reproducibility of results. Even in a controlled radio environment, the exact movements of different nodes may be hard to repeat.

Understandably, the community heavily relies on a simulation-based research methodology, even though Allman and Falk [2] have persuasively argued for the importance of carefully choosing between different methodologies. While they referred to wired networks, the need for validating and improving simulation models has also been recognized by colleagues in the mobile networking field [3, 4]. These efforts could intensify if researchers had easier access to experimental capabilities.

Mobile network emulators facilitate access to experimentation. They can provide a middle ground between pure simulation and laborious full-scale experiments [5, 1] with increased realism but still acceptable reproducibility of results. Perhaps, the most widely used emulation strategy—probably due to its ease of deployment—is based on software MAC filtering (e.g., [6]), which controls the topology of a network without moving nodes. It provides an emulation environment at the network layer and above but carries the same disadvantages as network simulators at the link and physical layer. The "Testbed on a Desktop" [7] exemplifies an approach that uses a more realistic physical and link layer. It requires, however, a carefully planned hardware setup involving antenna shielding and attenuation. Especially, if topologies should change during an experiment, variable attenuators must be strategically placed into the setup, as illustrated by RAMON [8]. EWANT [9] eliminates variable attenuators through a hardware switch to connect a radio to different antennas over time. This number of antennas is limited by the antenna switch and does not scale to larger networks.

This paper presents the design and implementation of M-ORBIT, a mobility emulator that uses spatial switching and noise generation. To our knowledge, this is the first mobility emulator implementation that provides—without a team of human assistants—repeatable large scale experimentation and a network stack with most of the physical and link layer intricacies. We have built the emulator on the stationary ORBIT indoor wireless network research testbed, which is described elsewhere [10]. The emulator is fully software controlled, allowing us to make it remotely accessible for the research community.

Key contributions include an implementation of spatial switch-

SIGCOMM'05 Workshops, August 22–26, 2005, Philadelphia, PA, USA.

ing in software and noise generation to create larger topologies. Spatial switching relies on a large number of static radio interfaces, whose antennas are arranged in a grid topology. To emulate moving nodes, the system will select sending and receiving radios from the grid that best represent the positions of the moving nodes as given by a mobility scenario. To emulate large-scale networks the system can raise the noise floor in the environment, which emulates a greater distance between transmitter and receiver.

The remainder of this paper is organized as follows. Section 2 reviews the goals that influenced our system design and lists key design choices. Section 3 describes our implementation of spatial switching using virtual MAC addresses. The implementation is based on the Click Modular Router [11] framework. The evaluation in section 4 concentrates on a case study to test an AODV routing protocol implementation on M-ORBIT.

2. DESIGN CONSIDERATIONS

M-Orbit's design is driven by the following key design goals:

Large-Scale Multi-Hop Experimentation. The testbed should support tens of mobile nodes. In addition, it should be able to create topologies where not all nodes are in direct communication range, so that experimentation with multi-hop routing protocols is possible.

Include link layer properties. The results should reflect realistic link layer interactions and mobile nodes should be able to run arbitrary network layer protocols and PC-level applications to generate network traffic.

Repeatability. In a radio-based testbed slight changes in the radio environment can significantly affect reproducibility of results. The emulator should allow full automation to enable experimenters to conduct a large number of trials to obtain statistically significant and reproducible results.

The mobility emulator design and implementation is based on a stationary indoor testbed that is part of the Open Access Research Testbed for Next-Generation Wireless Networks (ORBIT) infrastructure. The indoor testbed currently comprises 128 IEEE 802.11a/b/g radio interfaces attached to 64 static nodes arranged on an 8 by 8 grid, as shown by Fig. 1. The antennas are mounted on the sides of crates, at 45 and 225 degree positions when looking at the topside of a node. The antennas are connected through shielded cables to the Atheros-based wireless cards. Every node is a small form factor PC with 1GHz Via C3 CPU, 512 MB RAM, and 20 GB hard disk. The nodes also have two 1000BaseT Ethernet ports, which are connected to each other and to external servers through a 2-layer hierarchy of switches. Raychaudhuri and colleagues describe the stationary testbed in more detail [10, 12].

2.1 Spatial Switching

We believe that, at reasonable expense, a software spatial switching approach best meets these goals. It emulates mobility by switching a moving node to different radio and antenna positions as time progresses. Thus, the emulated path of a moving node comprises a number of discrete steps that approximate the actual path a moving node would take. Figure 1 illustrates this concept. Note that at any position packets are transmitted over real radio interfaces, thus this emulator can be used to evaluate the effect of interference or other physical and link layer effects on higher layer protocols. We implement spatial switching in software using the Gigabit Ethernet connections available on the ORBIT testbed, because it allows us

Figure 1: Spatial Switching. The path shows an actual path of a mobile node, which the system emulates by choosing the radio node that best approximates the current position. The testbed comprises currently 64 nodes with two 802.11 a/b/g interfaces each.

to scale to a large number of nodes at much lower cost than using hardware antenna switches.

This approach is more flexible than modulating the signal-to-noise ratio (SNR) by attenuating the signal at the transmitter or receiver. Continuously changing the level of attenuation could also simulate the effect of mobility without moving any nodes. However, emulating multiple moving nodes would require extensive synchronization between transmitter and attenuation controller to control attenuation on a per packet basis. For example, SNR must be modulated separately at the receivers to accurately emulate broadcast frames. Each receiver's SNR controller must then be aware of frame start, frame duration, and frame source to apply the correct SNR modulation. We avoid these complexities by instead influencing the path loss through real changes in position.

2.2 Split-stack Architecture

The software spatial switching system uses a split-stack architecture, as illustrated in Fig. 2. Throughout an experiment the application and network layer of a mobile node reside on the same machine, denoted as a virtual mobile node. As time progresses, it uses the link and physical layers of different grid nodes by reconfiguring the tunnel. The virtual mobile node can be either a dedicated grid node or a server that is on the same local area network as the grid nodes. The network stacks of the virtual mobile nodes and the grid nodes are tied together by spatial switching components. On the virtual mobile nodes, they provide a virtual network interface, *fake0*, which is associated with a grid node radio interface. This means that most applications can be integrated with this system by changing the routing table to point to this virtual interface. We are also planning to implement cross-layer interfaces to access radio properties on the grid node such as signal-to-noise ratio, and a dynamic mobility interface to control the (virtual) movement of a node during run-time. To date, we have completed a spatial switching implementation that relies on a static predefined path.

Consider the scenario where one mobile node sends a packet to a second mobile node. The emulator would represent the mobiles through two virtual mobile nodes and two grid nodes, let us refer to them as VMN1 and VMN2, and GN1 and GN2, respectively. The packet will be created by an application running on VMN1 and processed by its regular network stack. However, instead of sending the packet over a wireless interface, VMN1 will tunnel the packet over the wired gigabit Ethernet interface to GN1. GN1 will

Packet Size (byte)	Gb Crossover RT Latency (μs)	Gb Switch RT Latency (μs)	802.11 11Mbps Latency (μs)	Relative Overhead w/ 2 Switches
60	48	5	609	9.5%
1000	129	20	2035	8.3%

Table 1: Latency overhead through tunneling, measured at low load. In the ORBIT testbed setup with two layers of switches, tunneling packets to and from the virtual mobile nodes increase packet latencies by less than 10% for 11Mbps radios.

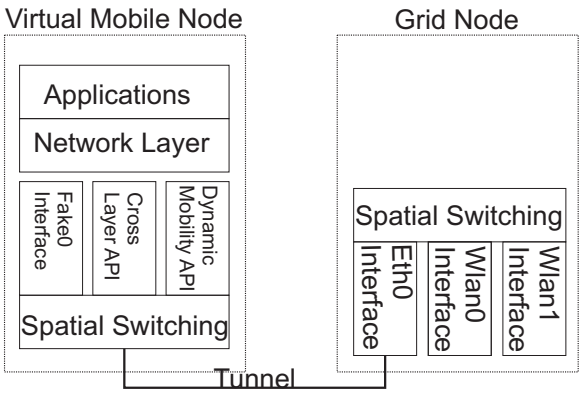

Figure 2: Split-stack architecture. The network stack of a single mobile node is split between a virtual mobile node and a grid node. For every additional mobile node in the experiment, the emulator requires an additional virtual mobile node and grid node.

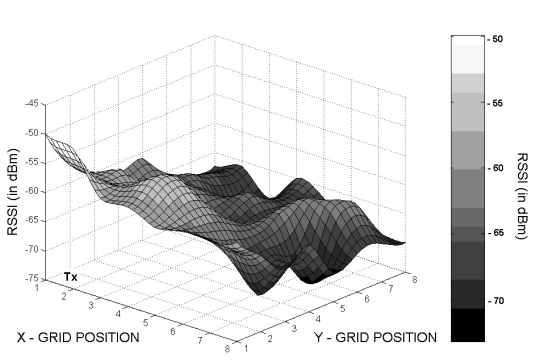

Figure 3: Signal power measured at each grid node's position with a signal analyzer. Even at 1mW transmit power the dynamic range in the ORBIT environment is not large enough–all radios are withing communication range.

then transmit the packet over the wireless interface to GN2, who will tunnel the packet to VMN2. At VMN2 it is again handed to the network layer and, if destined for this host, to an application. When one of the nodes moves, only the associated grid node will change.

This approach guarantees minimal latency overhead when switching among nodes but every packet transmitted over the wireless network must first traverse two tunnels. On an uncongested network, this approach therefore adds a constant latency overhead to every packet. This overhead amounts to less than 10% in our setup. This number includes operating system delays, packet transmission durations, and store-and-forward switch latencies. To obtain these values, we have measured the total duration of reflecting a packet 10000 times (round-trip) between the grid nodes and then calculating the mean packet latency. This was implemented using the same Click modular router framework [11] that we also use in the spatial switching implementation.

2.3 Radio Range Scaling Through Noise Generation

To contain facilities cost, the radio nodes are deployed with three feet spacing—thus, the current grid occupies a 24ft by 24ft area. The path loss in this area is to small to create a multi-hop network, the nodes are fully connected even at 1mW transmit power setting.

Figure 3 shows the received signal power at each of the 63 nodes positions, while the node at the corner position 1-1 was emitting beacons. These measurements were conducted with the Yellowjacket handheld 802.11b/g protocol analyzer, whose signal measurements are, according to the specifications, accurate within 1 dB [13]. The graph shows that the dynamic range of our environment is approximately 25 db. At the edges on the far side from the transmitter the signal power varies between -65 and -75 dBm, wheres the receive threshold of most 802.11 cards lies at approximately -90 dBm. The graph also shows that radio propagation in our indoor environment exhibits multi-path effects. As such it differs substantially from the free space propagation models commonly used in ns-2 simulations and provides an interesting alternative evaluation environment.

The system can emulate different node spacings by injecting additive white gaussian noise (AWGN) into the environment. In the absence of interference, the bit error rate of a wireless receiver is a function of the signal-to-noise ratio (SNR). According to wireless communication theory, greater distance between transmitter and receiver will lead to lower SNR (and increased bit errors) through increased path loss, when transmit power and noise floor remain constant. The same SNR can be achieved by generating an increased level of AWGN, which will reduce the SNR by raising the noise floor at the receiver. The same concept is commonly used in testing wireless systems such as cell phones in a laboratory environment.

We placed four antennas about 3ft inward from the corners of the grid. The antennas are attached, through a 4-way splitter, to a AWGN waveform generator. Four antennas more evenly distribute the noise level over the grid. Using only one antenna to inject noise would create highly asymmetric links.[1]

[1]Consider the SNR at different positions between two radios A and B at opposite corners of the grid, with a noise source next to A. Signal S and noise N power are inversely proportional to the square of the distance d from the transmitter. Therefore, when A transmits a packet the $SNR = \frac{\frac{S}{d^2}}{\frac{N}{d^2}} = \frac{S}{N}$ remains constant across all receivers, in this idealized model. When B replies to A (or just acknowledges the frame), however, the noise source is colocated with the receiver. Then, $SNR = \frac{\frac{S}{d^2}}{\frac{N}{(x-d)^2}}$, where x is the distance between the transmitter and receiver. As d approaches x the SNR falls of sharply, therefore the packet cannot be received even at much lower noise levels than in the previous case. By generating noise near both receivers, SNR levels will be symmetric on this link

2.4 Experimenter's Interface

An experimenter has to perform two main tasks to configure a mobility experiment: Configuration of the virtual mobile nodes and defining the paths of mobile nodes.

First, the experimenter designates one ORBIT node as virtual mobile node for each mobile node that should be emulated.[2] On these nodes, the experimenter installs applications and perhaps network layer software as appropriate for the experiment. The only constraint, that we are aware of, is that the routing layer must be instructed to use the virtual fake0 interface for communications with mobile nodes—we have added a default routing table entry for this interface.

In the second step the experimenter defines a path for each virtual mobile node. This requires creating a configuration file for each mobile node that contains (time, node name) tuples. The first tuple should contain emulation time 0 and define the grid node that belongs to the virtual mobile at the beginning of the experiment. Nodes are named according to their coordinates in the grid. Each consecutive tuple marks a change in the grid node that is associated with the virtual mobile. By coordinating the time intervals with the AWGN level, emulation of different speeds is possible. The configuration tool translates these configuration files into Click configuration files for each node that is part of the experiment.

We use the ORBIT NodeHandler infrastructure to start an experiment. It provides a scripting language to execute shell commands in parallel on a group of ORBIT nodes, and allows remote booting and shutdown of nodes. Starting the experiment involves executing the applications and installing the click configurations on the nodes. Changing the associated grid nodes is controlled through a timer on each node. This timer starts as soon as the click scripts are installed. NodeHandler uses multicast to send the commands, thus it also achieves a high degree of synchronization among the nodes. Ott and colleagues [12] provide more information on the NodeHandler scripting as well as a measurement collection infrastructure.

3. IMPLEMENTATION

We have implemented the spatial switching concept on the ORBIT Testbed through MAC address translation using the Click modular router framework [11]. Click runs on a Linux 2.4.26 kernel and is integrated with the madwifi stripped driver[3] for wireless cards with Atheros chipsets. Every grid node and virtual mobile node requires a custom Click configuration, which includes the mappings between the virtual mobiles' MAC addresses and the MAC addresses of the corresponding grid node. A configuration tool automatically generates these scripts from path descriptions of the mobile nodes.

The VMN scripts contain most of the switching logic, while the grid nodes just forward packets from their VMN to the wireless interface and vice versa. Conceptually, a VMN script maintains an set of Ethernet headers, which change over time. These headers define the source and destination grid node over which packets are switched, depending on their destination. The VMN scripts install the virtual network interface *fake0* and intercept all packet sent to this interface. On every packet, the VMN script changes the source and destination addresses in the packets' Ethernet header to the wireless interfaces of the grid nodes that are associated with the source and destination VMNs. We will refer to this header as the *wireless header*. It then double encapsulates the scripts the packet with a second Ethernet header that contains the source and destination addresses to send the packet to the VMN's associated grid node. We will refer to this second header as the *wired header*. To account for this double encapsulation the maximum transfer unit on *fake0* must be reduced to 1486 octets). Finally, click sends the packet directly to the *eth0* interface.

```
FromHost(fake0, 192.168.100.2/24)

// Split into streams based on dst MAC
-> whichVM_cl::Classifier(
    0/000FEA4D9510,0/000FEA4A8AF0,
    0/000FEA4CB750,0/FFFFFFFFFFFF);

// add double Ethernet header
whichVM_cl[0] // 1. stream
  -> Strip(14) // remove header
  -> ee_VM73_VM43 // add wireless header
  -> ee_Own // add wired ethernet header
  -> Q; // send to output queue

// Similarly, handle packets for
// other destination and broadcasts.
whichVM_cl[1]
  -> Strip(14)
  -> ee_VM73_VM53
  -> ee_Own
  -> Q;

whichVM_cl[3]
  -> Strip(14)
  -> ee_VM73_ALLVM
  -> ee_Own
  -> Q;

// Send outgoing packets to eth0
Q::Queue -> ToDevice(eth0);

// Forward received packets to host
fd :: FromDevice(eth0)
-> SetPacketType(HOST)
-> ToHost(fake0);

// Change the encapsulation headers
// when emulated node moves
PokeHandlers(
wait 12,
write ee_Own.dst 00:0F:EA:4C:A6:BE,
write ee_VM73_VM43.src 00:60:B3:25:BF:E9,
write ee_VM73_VM53.src 00:60:B3:25:BF:E9,
write ee_VM73_ALLVM.src 00:60:B3:25:BF:E9,
wait 3,
write ee_VM73_VM83.dst 00:60:B3:25:C0:11
)
```

Figure 4: Fragment of the click configuration script for a virtual mobile node.

Listing 3 shows the key parts of a VMN configuration file. For brevity, this configuration describes a small scenario with three nodes, only one of whom moves a step. Also, we have omitted the handling of ARP request that the local host sends to the *fake0* interface. The first part of the script defines a graph of packet processing elements that a packet traverses. The encapsulation head-

and SNR falls of with distance.

[2]Our current implementation uses ORBIT nodes as virtual mobile nodes. In future versions, we expect to make external servers available for this purpose.

[3]Available from `http://www.pdos.lcs.mit.edu/~jbicket/madwifi.stripped/`. The madwifi stripped driver is a fork from the regular madwifi driver that provides integration with Click. It also provides a pseudo-ibss mode that we used for our experiments because the ad-hoc mode implementation in the standard madwifi driver is not stable.

ers that the script generates are dependent on the destination VMN. Therefore, a classifier elements divides the incoming packets into three streams (two for the other VMNs, one for broadcast) based on their destination MAC address. In each stream Ethernet encapsulation elements replace the headers to send the packet via two grid nodes to the destination VMN. All streams converge on the *eth0* interface.

When a node moves, the VMN needs to update its Ethernet headers with new MAC addresses. The PokeHandlers command executes such updates at predefined times (this relies on time synchronization between nodes). The *wait* times are relative to each other, and zero represents the start of the experiment. At time 12, in the example, the VMN switches to a different grid node. Therefore, it must update the wired header with the new destination address, and all wireless headers with the new source address. Simultaneously, all other VMNs must change the destination address in one of their wireless headers. This is illustrated at absolute time 15 in the script. The destination address in the header for VMN 8-3 is updated, because this VMN has moved to a new grid node.

Double encapsulating in the VMN scripts means that a GN node only has to keep track of its associated VMN. When it receives a double encapsulated packet on the wired interface, it removes the wired header and transmits the remaining packet over the wireless interface. Whenever a broadcast or unicast packet is received on the wireless interface, however, it has to add another Ethernet header to forward the packet to its current VMN. Similar to the VMN script, a PokeHandler command changes this header when a VMN moves away or a new VMN arrives at the grid node. The madwifi stripped driver only accepts packets with 802.11 headers as opposed to other network drivers that expect 802.3 headers. Therefore, we integrated the GN script with the pseudo-ibss script supplied with the madwifi stripped driver. This script handles the header conversion and implements a pseudo ad-hoc mode.

4. CASE STUDY: AD-HOC ON DEMAND DISTANCE VECTOR ROUTING

The following case study illustrates how the emulator can be used for mobile systems experiments and how noise generation can create multi-hop topologies on the testbed. To this end we measure goodput on a multi-hop network running the Ad-hoc On Demand Distance Vector (AODV) protocol.

The case study required installing an AODV implementation and a goodput measurement application on the nodes, and generating a random-walk mobility scenario. The first step was straightforward: We chose the Ad-hoc On-demand Distance Vector routing implementation from Uppsala University (AODV-UU).[4] After recompiling our kernel with netlink support and invoking the aodv daemon with a command line parameter AODV was up and running. The crucial command line parameter instructs it to use our virtual interface *fake0*. Similarly, we could use a variety of different throughput measurement applications. This shows how our spatial switching implementation at the network device driver level allows the use of unmodified routing protocol implementations.

To create mobility scenario, we implemented a perl script that generates a 2D random-walk pattern. To allow easier interpretation of the results, we defined a 5min experiment with only one mobile node conducting a random walk over the full grid and two stationary VMN assigned to grid nodes 1-3 and 8-3. We have raised the noise level so that communications between the two stationary nodes must be routed over the mobile node, because the stationary

[4]Available at `http://user.it.uu.se/~henrikl/aodv/`

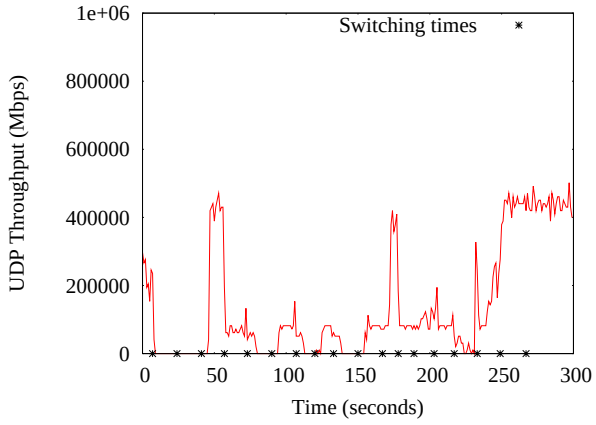

Figure 5: UDP goodput over a 2-hop AODV route. The two endpoint nodes are stationary. Goodput changes over time as the intermediary node moves between areas with different packet error rates. The markers show the times at which the intermediary node moves to a different grid node. The changes in throughput appear well correlated.

nodes are out of each other's communication range. The mobile node switches its position on average every 20 seconds.

Figure 5 shows the UDP goodput between the two stationary nodes as measured with the Rude & Crude tools.[5] Five repetitions within one hour showed low variance. In the graph we have also marked the times at which the position of the mobile node changes. All 802.11 radios were configured to a bitrate of 1Mbps, therefore the theoretical maximum goodput over a two-hop route protocol overhead is about 400Kbps. The graph shows how the throughput varies between zero and the maximum at different positions of the mobile node.

The high throughput variance of the AODV experiment can be explained with the different link qualities that the mobile node experiences at different positions. To this end, we measured packet error rates between different sender and receiver pairs to characterize the network topology created by noise generation. For this experiment we configured one radio to transmit access point beacons and one radio on each of the other 63 nodes as receivers that log the RSSI and sequence number for each received beacon.[6] This was implemented in Click with the madwifi stripped driver.

Figure 6 shows for each receiver the percentage of beacons received at the noise setting used for the AODV experiment. Each experiment duration was one minute, during which the node marked with *tx* transmitted 600 beacons. Many nodes have high PERs, but have not completely lost communication. This means some nodes will sporadically receive packets from a transmitter even though the PER is too high for most applications. This does not match well the network topologies commonly used in simulator evaluations. However, it is consistent with experimental observations by other researchers [5].

The transmitter positions correspond to the endpoints of the AODV measurement. Referring back to figure 5 we can interpret the variations in goodput. The mobile VMN is initially assigned to grid node 7-5. From the left packet reception graph, we can see that

[5]Available from `http://rude.sourceforge.net/`

[6]`iwpriv ath0 setrxhdr 1` configures the madwifi stripped driver to prepend prism headers to each received packet, which contain RSSI.

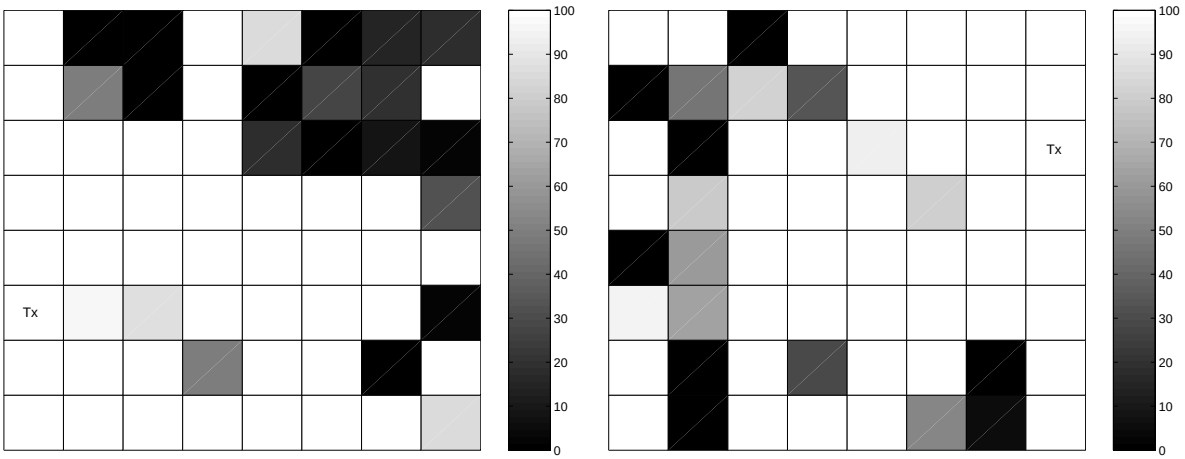

Figure 6: Packet reception rate with raised noise floor. The transmitter is at positions 1-3 (left) and 8-6 (right). Some nodes receive no packets, which allows the creation of multi-hop topologies.

this grid node has good packet reception rate on the uplink as well as the downlink, which explains the corresponding goodput value around 200 Kbps. About 7 seconds later, the VMN switches to grid node 6-8 and the goodput drops to zero. This drop can be explained by the 100% packet loss that grid node 6-8 experiences. In another 17 seconds, the mobile VMN switches to grid node 7-6 and there is no change in the goodput because this grid node also experiences 100% packet loss. About 41 seconds into the experiment, the VMN switches to grid node 8-4 and we see a rise in the goodput value shortly afterwards. This increase in goodput correlates with the packet reception rate for grid node 8-4.

These measurements substantiate that raising the noise floor in the emulator environment can be used to place some nodes out of the communication range of a transmitter. As such, this mechanism allows creating multi-hop topologies in ad-hoc networks.

4.1 Future Work

A continuously moving radio experiences physical effects such as multipath fading, shadowing and, the doppler effect, which affect the received SNR and packet error rate. The constructive or deconstructive summation of several reflections of a signal can cause the signal strength to rise or fall sharply at different points in space and time. In addition, moving into the "shadow" of an obstruction lets the overall signal envelope drop and at high velocity the doppler effect causes frequency shifts.

Thus, the mobility emulator provides the most accurate results in discrete mobility scenarios, where nodes occasionally change position but do not move at high speed while they are communicating. This corresponds well with typical laptop or PDA usage in office and conference environments. More research is needed, however, to accurately emulate the fading and frame error patterns that continuously moving nodes experience. One approach would drop packets in the network stack to emulate frame error rate patterns that are dependent on a mobile node's speed. A second approach, would use more fine-grained variable noise controls on each receiver in conjunction with spatial switching to more accurately emulate movement.

5. CONCLUSIONS

We have presented the design and current implementation status of a mobility emulator based on software spatial switching. This emulator allows experimentation with a sizable number of nodes while using real physical and link layers. Through an ad hoc routing case study we have shown that this emulator can support experiments with mobile nodes that traverse zones of varying connectivity. These first results also indicate that noise generation can emulate larger sized areas on a small testbed, which is necessary for creating single-channel multi-hop topologies.

6. REFERENCES

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

[2] M. Allman and A. Falk. On the effective evaluation of TCP. *ACM SIGCOMM Computer Communication Review*, 29(5):59–70, Oct 1999.

[3] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In *ACM MobiCom '03*, pages 217–229. ACM Press, 2003.

[4] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *ACM/IEEE MSWiM*, pages 78–82, October 2004.

[5] D. Maltz, J. Broch, and D. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, Carnegie Mellon University, Mar 1999.

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

[7] J. T. Kaba and D. R. Raichle. Testbed on a desktop: strategies and techniques to support multi-hop manet routing protocol development. In *ACM MobiHoc*, pages 164–172. ACM Press, 2001.

[8] E. Hernandez and A. Helal. RAMON: Rapid-mobility network emulator. In *IEEE LCN*, Nov 2002.

[9] S. Sanghani, T.X. Brown, S. Bhandare, and S. Doshi. EWANT: The emulated wireless ad hoc network testbed. In *IEEE WCNC*, volume 3, pages 1844–1849, Mar 2003.

[10] 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 WCNC*, 2005.

[11] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug 2000.

[12] M. Ott, I. Seskar, R. Siracusa, and M. Singh. Orbit testbed software architecture: Supporting experiments as a service. In *IEEE Tridentcom*, Feb 2005.

[13] Berkeley Varitronics Systems. Yellowjacket 802.11b/g data sheet. `http://www.bvsystems.com/Products/WLAN/YJ802.11bg/YJ802.11bg.htm`, Dec 2004.