# MobiNet: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks

*Priya Mahadevan*
*UC San Diego*
*pmahadevan@cs.ucsd.edu*

*Adolfo Rodriguez*
*IBM, RTP and Duke University*
*razor@cs.duke.edu*

*David Becker*
*Duke University*
*becker@cs.duke.edu*

*Amin Vahdat*
*UC San Diego*
*vahdat@cs.ucsd.edu*

## Abstract

The current state of the art in evaluating applications and communication protocols for ad hoc wireless networks involves either simulation or small-scale live deployment. While larger-scale deployment has been performed, it is typically costly and difficult to run under controlled circumstances. Simulation allows researchers to vary system configurations such as MAC layers and routing protocols. However, it requires the duplication of application, operating system, and network behavior within the simulator. While simulation and live deployment will clearly continue to play important roles in the design and evaluation of mobile systems, we present *MobiNet*, a third point in this space. In MobiNet, the communication of unmodified applications running on stock operating systems is subject to the real-time emulation of a user-specified wireless network environment. MobiNet utilizes a cluster of *emulator* nodes to appropriately delay, drop or deliver packets in a hop by hop fashion based on MAC-layer protocols, ad hoc routing protocols, congestion, queuing, and available bandwidth in the network. MobiNet infrastructure is extensible, facilitating the development and evaluation of new MAC layers, routing protocols, mobility and traffic models. Our evaluations show that MobiNet emulation is scalable and accurate while executing real code, including video playback.

## 1 Introduction

Wireless mobile systems have become increasingly popular in the past few years. Of particular interest has been the proliferation of *ad-hoc* wireless networking where mobile nodes form peer relationships with one another to relay information through the network.

One key challenge in this area is evaluating these protocols and applications in a scalable and accurate manner. It is difficult and costly to deploy development software on a large number of real mobile nodes. Further, live deployment makes it difficult to obtain reproducible results. To overcome these limitations, researchers have developed simulation engines to mimic the behavior of mobile systems by modeling packet loss, queuing delays and MAC-layer behavior. Application code is typically re-written to conform to the simulation environment. This approach requires increased development effort and also leads to loss in accuracy as the behavior of a unmodified application running over a real OS, network stack and hardware is lost. Finally, accurate simulation environments face significant scalability limitations, often topping out at a few tens of mobile wireless hosts.

MobiNet is an emulation environment designed to overcome some of the accuracy and scalability challenges in mobile evaluation. The goal of our work is to allow users to evaluate the behavior of their wireless systems under a range of conditions in a controlled, reproducible environment. System aspects that we would like to allow users to control include MAC layers, routing protocols, mobility patterns and traffic models. MobiNet supports flexible deployment of the above models, thereby allowing researchers to study and improve the performance of wireless applications and protocols.

To support the above types of experiments, we designed MobiNet to emulate a target mobile network on a scalable LAN cluster with gigabit interconnect, enabling researchers to deploy unmodified IP-based software and subject it to faults, varying network conditions, different routing protocols, and MAC layer implementations. Edge nodes running user-specified OS and application software are configured at the IP-layer to route packets through one or more MobiNet *core* nodes that cooperate to subject the traffic to the bandwidth, interference patterns, congestion, and loss profile of the target network topology.

We must address several key challenges to successfully emulate large-scale multi-hop wireless networks. Behavior of MAC layer (*e.g.* various flavors of 802.11) significantly impacts the performance of wireless networks. Node mobility plays an important role in wireless environments. Ad hoc routing protocols are critical for relaying packets. Therefore our emulation supports: i) various MAC layers ii) routing protocols iii) node movement patterns. Each of the above layers must be deployed in a modular manner, allowing users flexibility and control over their experiments. Further, we would like to structure our emulation in a scalable, accurate and extensible manner. Our scalability tests show that a single MobiNet core can forward up to 89,000 packets per second. Using just one MobiNet core and 2 physical edge nodes, we have been able to emulate a 200-node topology, forwarding application packets in real time. Along with scalability, MobiNet also provides good accuracy. We validated our MAC and routing protocols against other simulators and found that our results compared favorably with those obtained from ns2. We also present results from running unmodified binaries (video playback) that demonstrate the power and flexibility of our system.

The remainder of this paper is organized as follows: Section 2 describes the details of the MobiNet framework. We briefly describe MobiNet's accuracy and scalability in section 3. Section 4 describes our experiences in deploying real unmodified applications over MobiNet. We discuss related work in section 5 and present our conclusions in section 6.

## 2 The MobiNet Framework

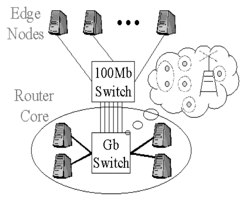

Figure 1: MobiNet Architecture

We borrowed some basic design principles from the publicly available ModelNet [10], an emulation environment for wired and static networks. However, MobiNet required a complete reimplementation of the system given the inherent differences between wired and wireless networks. The MobiNet architecture is composed of *edge nodes* and *core nodes* as shown in Figure 1. Edge nodes in MobiNet can run arbitrary architectures and operating systems and could even be a combination of different devices such as laptops, PDAs, etc. Our current experiments have been performed on edge nodes running linux. They run native IP stacks and function as they would in real environments with the exception that they are configured to route IP traffic through MobiNet cores. MobiNet core nodes run a modified version of FreeBSD to emulate topology-specific and hop-by-hop network characteristics.

Target applications run on edge nodes as they would in a real setting. However, to decrease the number of client

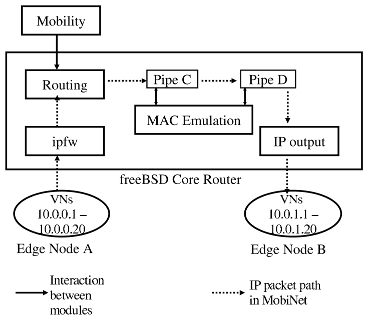

Figure 2: MobiNet Modules

(edge) machines required for large-scale evaluations, our architecture allows for *Virtual Edge Nodes* (VNs). VNs enable the multiplexing of multiple application instances on a single client machine, each with its own unique IP address. Since MobiNet clients use internal IP addresses (10.*), the number of clients that can be multiplexed onto an edge node is not limited by IP address space limitations, but rather by the amount of computational resources ( *e.g.* threads, memory) that the target application uses. All VNs are configured to route their traffic through one of the cores. The MobiNet cores emulate wireless network behavior at multiple layers while eventually routing packets to the edge node hosting the destination VN.

Emulation at the core involves capturing node movement patterns, dynamic routing, and MAC layer effects such as collisions and capture. To this end, we have a mobility module, routing module and MAC layer module in the MobiNet core. The physical layer also plays an important role in wireless networks, hence we have support for free space propagation model and two-ray ground reflection model[1]. By dividing mobile emulation behavior under functional lines, MobiNet's modules are more easily developed and replaced. This allows experiments to use different combinations of modules, leading to a more flexible and powerful emulation framework. Figure 2 depicts the interactions between the different modules in MobiNet. The mobility model is implemented as a user level application that downloads new node movement files into MobiNet core's kernel at user specified time intervals. The routing module uses this information to find new routes when existing routes become stale. Once a packet enters the system, it is handed up by the *ipfw* module in the FreeBSD kernel to the MobiNet module. The routing module within MobiNet is now responsible for finding a path in order to send this packet to its destination. The path is basically a list of nodes through which the packet has to traverse before reaching its destination. Once the path has been obtained, the MAC-layer module emulates the packet according to the specified attributes of each *pipe* in the path. Pipes in MobiNet correspond to the transmission capacity of their associated nodes. The packet traverses through every intermediate node's pipe, thereby being subjected to queuing delays and congestion at every node. Once the packet successfully reaches the last hop in the path, the packet is sent to the virtual node hosting the packet's destination. Thus, transmitting a packet from source A to destination B via nodes C, D, and E will involve sending the packet through pipes A, C, D and E before finally relaying it to destination B. Each pipe maintains a drop-tail queue for storing packets that need to be transmitted from the corresponding node. All attributes of pipes such as bandwidth, queue size and loss rate are user-configurable and can be downloaded into the core's kernel using the *sysctl* function call in FreeBSD.

MobiNet emulation is a three step operation: topology creation, assignment of VNs and pipes to hosts and cores respectively, and application execution. A user creates a desired topology, MobiNet distributes pipes associated with each node in the topology across the cores to distribute emulation load, assigns VNs in our emulated topology to edge nodes, and configures and executes the applications in the MobiNet emulation framework. We now describe each of MobiNet's modules in more detail.

### 2.1 Mobility

The mobility module is a user-level application that generates various node positions and neighbor lists consisting of nodes within a node's transmission range. This information is downloaded in real time into the kernel of the MobiNet cores at regular user-specified intervals. Alternatively, we could calculate these positions and neighbor lists in real time within the core's kernel. Doing so, however, would cause significant overhead since floating point operations would be required in the kernel.

One interesting parameter in MobiNet's emulation is that of the interval used to refresh node positions within the core's kernel. If the interval is too high, valuable kernel processing is wasted in reading new node coordinates for values that have changed little. If it is too low, it leads to inaccurate results. MobiNet attempts to bridge the gap between kernel performance and accuracy by choosing an interval value that provides good performance and accurate results under a wide variety of emulations. We found that setting the node position refresh rates to 0.5 seconds provides good results for our test scenarios, with node velocities up to 20 m/s. We stress that the refresh interval is user configurable and node coordinates can be downloaded into the kernel at a much lower granularity.

Our current mobility application supports the random waypoint mobility model described in [1], though MobiNet can use arbitrary movement models. In our applications, users specify the topology size, the duration of the experiment, the maximum speed of nodes, the movement *pause time*, and the interval of the desired output. The mobility application creates time-indexed movement files that include the current positions of each node and the neighbor lists for each node. These movements files can be read by the MobiNet core during the execution of the experiment.

### 2.2 MAC Layer Emulation

Our modular emulation approach is amenable to a wide range of models for the MAC layer. We implemented our MAC layer based on IEEE's 802.11 standard specification for RTS-CTS-Data-ACK in MobiNet. The details of our implementation are described in [5]. The physical layer plays an important role in the performance and energy consumption of mobile and wireless systems. The free space model and the two-ray model predict the received power as a deterministic function of distance [1]. Our physical link model supports free space propagation and two-ray ground reflection model [1]. Power level at which packets are received determines if one or both packets are dropped due to noise or if one packet is captured by the other.

### 2.3 Dynamic Routing

As with all other MobiNet modules, the routing layer is implemented as a pluggable module in the FreeBSD kernel. The MobiNet core makes a call to this routing module to retrieve paths for the packets that it receives. We have implemented the Dynamic Source Route (DSR) [3] protocol in the MobiNet core. While we chose DSR in our current implementation, DSR can be replaced with any other ad-hoc routing protocol such as AODV [8], DSDV [9], or TORA [7]. Our generic design and the fact that each component in MobiNet is pluggable and not dependent on other components enable us to implement a broad range of routing modules in the kernel with relative ease. Detailed implementation is described in [5].

## 3 Evaluation

In this section, we briefly describe our experiences using MobiNet for evaluating ad hoc wireless applications. Our evaluation focused on testing MobiNet for scalability as well as accuracy.

We have written and tested a simple application in native TCP/IP and in the ns2 network simulator to enable comparisons between MobiNet emulation and ns2 simulation. The application establishes simple constant bit rate (CBR) streams between senders and receivers using UDP. Each sender sends data to exactly one receiver. Our CBR communications consists of 64-byte packets sent from each node (sender) at the rate of 4 packets per second. While it is impossible to guarantee that both versions function identically, the simplicity of our test application leads to it exhibiting very similar behavior in both environments. Using this application, we have executed a number of experiments to evaluate the performance, scalability, and accuracy of the different modules in MobiNet. The goal of our accuracy and routing overhead tests were to reproduce the experiments described in [1].

In all of our experiments, MobiNet edge nodes consisted of Pentium 4 2.0 GHz PCs with 512 MB memory running linux version 2.4.2. We use a single Pentium 3 dual processor with 2 GB memory supporting FreeBSD version 4.5 as our MobiNet core. Our experiments on ns2 were conducted on a machine similar to our edge nodes. MobiNet provides various packet statistics that enable us to determine the number of packets sent, packets dropped due to MAC collision, and other useful metrics. Likewise, we make use of ns2 trace files to extract these metrics.

With our mobility application, we simulated random waypoint mobility using various seeds and pausetime values, resulting in different movement patterns. For most of our experiments, we specified a neighbor-refresh interval of 0.5 seconds. We found that our interval of 0.5 seconds gives us comparable results with lower intervals such as 0.2 seconds and also with the continuous movement pattern that ns2 supports.

### 3.1 Core Performance

One of the experiments we have executed, tested the ability of the MobiNet core to process packets. The goal was to find the number of packets per second the MobiNet core router could emulate without saturation.

The setup comprised of 200 VNs that were distributed across 2 edge nodes (100 virtual IP addresses mapped to each edge node). We disabled the mobility module to decrease the overhead due to DSR. Thus DSR is invoked only once for a source-destination combination. Once a route to a particular destination has been found by the routing module, the route does not change. A sender application was associated with every VN on one edge machine, while a listener was associated with every VN on the other edge machine. Each sender application sent 64-byte UDP packets at a constant bit rate to a specific listener, thereby accounting for 100 flows from the sender edge machine to the lister edge machine. Each source sent packets in exactly 1 hop to exactly one destination which was also the node's sole neighbor. Thus, there were no packet collisions. We also set the DIFS and SIFS values in the 802.11 specifications to zero as the goal was to gauge the maximum number of packets that could be sent through a single MobiNet core. MobiNet core runs with a clock resolution of 10Khz, meaning that we are able to accurately emulate each packet hop to within 0.1 ms accuracy. Even for end-to-end path lengths of 10 hops, packet transmission delays are accurate to within 1 ms, sufficient for our target wireless scenarios, especially when considering end-to-end transmission, propagation, and queuing delays. This accuracy holds up to and including the peak emulation rate because MobiNet's emulation runs at the kernel's highest priority level. We measured throughput in terms of packets per second and CPU utilization at the core for different packet sending rates. We ran similar tests but with different topologies, so that each packet from the sender must traverse 3 hops and 5 hops respectively before reaching the destination. Again, we ensured that there were no collisions and nodes just had their communication partners as their neighbors. As the number of hops increased, we found that the total number of packets that the core could forward per second decreased as it now had to perform more work per packet. We summarize our results in Table 1.

| CPU utilization at core | Pkts/sec forwarded for 1 hop | Pkts/sec forwarded for 3 hops | Pkts/sec forwarded for 5 hops |
|---|---|---|---|
| 50% | 43.5K | 25K | 16K |
| 70% | 63.5K | 38K | 23K |
| 90% | 78K | 47K | 30K |
| 100% | 89K | 50K | 35K |

Table 1: Forwarding capacity at the Core

### 3.2 MAC layer and routing accuracy

Validating the behavior of our MAC layer implementation is difficult as no known emulation or simulation technique can accurately predict the bit error rates or radio interference under arbitrary deployment scenarios. To gain some baseline confidence in the accuracy of our 802.11 MAC model, we conduct micro-benchmarks to compare MobiNet's MAC layer performance with that of ns2 for a variety of topologies and packet transmission rates. Since the packet transmission rate is dependent upon the timing and rate of collisions, we hypothesize that if MobiNet and ns2 deliver the same packet throughput under a range of conditions, the packet collision and backoff behavior is likely to be similar. We experimented with several topologies and packet sending rates. For each of our topologies, we found that MobiNet and ns2 had similar packet delivery ratio. Our detailed results for different topologies are described in [5].

The next step was to validate routing accuracy in our emulator. We achieved this by comparing experimental results obtained from MobiNet to that from ns2 for our simple CBR communication. We used the 802.11 MAC protocol and DSR implementations available in ns2. Using our mobility model, we generated movement files that were used by ns2 and MobiNet. We varied the maximum speed and pause time in our experiments and for each of the above, we found that MobiNet's packet delivery ratio matches that of ns2. We also compared the number of control packets transmitted by our DSR implementation with that of ns2 in the above experiments and again found that MobiNet compared favorably with ns2. Again the experiments and results are described in [5]. All the above tests helped validate the MAC and routing accuracy of our emulator.

### 3.3 Scalability

Given the accuracy of our emulation experiments, we next consider the scalability of our emulation environment. One of the main benefits of MobiNet over using a simulator such as ns2 is experiments can be run in realtime. Simulators that do not run in realtime have an advantage that however complex the experiment, it eventually completes. On the other hand emulators that run in realtime find the load too great at some stage. However, for typical experiments, MobiNet is capable of forwarding up to 89,000 packets per second and thus has a distinct advantage over ns2 with respect to time taken to complete experiments upto this capacity.

To quantify this benefit, we compare the time required to run experiments in ns2 and MobiNet as a function of numbers of CBRs. We used a 200 node topology with nodes distributed randomly in a 3000 meter by 600 meter rectangle (resulting in the same node density as our previous experiments). For MobiNet, the 200 nodes were distributed across 2 MobiNet edge nodes. The ns2 experiments were run on a single machine with the same configuration as the MobiNet edge node. We disabled node mobility in this case to reduce the overhead due to finding routes with DSR. Here, DSR only needs to find routes to destinations once (at the start of the experiment). We varied the number of CBR sources from 10 to 40, with each sender once again transmitting 64-byte packets at the rate of 4 packets per second. Each node sent a total of 1200 packets. Figure 3 shows the computation time necessary to execute the experiment for MobiNet emulation and ns2 simulation. This is the time it takes for the experiment to complete multiplied by the number of machines used in the experiment. In real time, this experiment takes 5 minutes, as it takes each CBR source 300 seconds to transmit its share of packets. As a result, MobiNet using 3 machines (2 edges and 1 core) emulates the experiment in 15 minutes. In contrast, ns2 simulation time of the experiment increases linearly with the number of CBR nodes. In the case of 40 nodes transmitting, the ns2 simulation lasted 134.5 minutes, compared to MobiNet's 15-minute emulation.

## 4 Deploying Real Applications

In this section we demonstrate the utility and generality of our infrastructure by deploying and evaluating real unmodified code, a video player over MobiNet. We used XAnim as our sample application. XAnim is a program that plays a wide variety of animation, audio and video formats on Unix X11 machines. Running the same application on ns2 would be difficult to impossible. Our goal was to study the performance of the the video player in a ad hoc wireless network as a function of node movement.

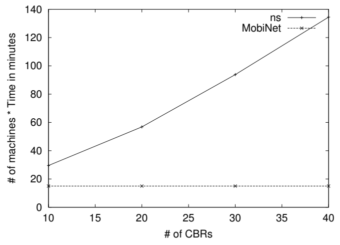

Figure 3: Scalability in MobiNet vs. ns2 as a function of time

We started with a wireless topology consisting of 50 nodes moving according to the random waypoint movement model, where the maximum random speed was set to 1 m/s. The nodes in our topology were hosted on two edge machines, thus each edge node was responsible for 25 VNs. We randomly chose two VNs from our topology. XAnim was deployed over one of the VN, while the display was set to the other VN. Communication between these two nodes ran over the x11 protocol. The VN executing XAnim would send its packets to the MobiNet core, which would use DSR to find a route to the VN hosting the display. The packets were emulated according to our 802.11 implementation in the MobiNet core and then sent to the destination VN which would display the movie. Due to node movement, if existing routes went stale, DSR was used to find fresh routes to the destination VN. The video clip was replayed in a continuous fashion for 2 minutes. For lower node mobility scenarios, packet drops due to broken routes was low and we observed that the video played in an almost continuous manner. In a highly mobile environment, we found that the video clip would stall for a while during packet drops. Once routes were found, the clip would start playing again.Unlike CBR communication, in the x11 communication that takes place between the XAnim nodes, loss of vital packets due to node movement leads to the application stalling for a while. We recorded the total number of XAnim packets exchanged between the two VNs for different values of pause time and for different values of maximum speed. We averaged the results over

| Pause time (s) | 1 m/s | 5 m/s | 20 m/s |
|---|---|---|---|
| 0 | 14500 | 13708 | 5490 |
| 30 | 15596 | 13728 | 13031 |
| 60 | 14927 | 14565 | 13207 |
| 300 | 16200 | 16100 | 16086 |

Table 2: x11 packets exchanged between 2 VNs for various maximum speeds

several runs of the experiment and present them in Table 2.

## 5 Related Work

Zhang and Li [12] have built an infrastructure for testing mobile ad hoc networks. However, their work does not support any routing protocol. Furthermore, their scheme does not restrict application bandwidth, making experimental results inaccurate for a range of important application characteristics. Noble and Satyanarayanan [6] use trace-based network emulation to play back measured mobile network characteristics to real applications. Our approach generalizes this technique, allowing users to generate their own mobility scenarios. Netbed [11] is a network testbed comprising real mobile nodes using real mobile hardware and software. In contrast to our work, Netbed is a real testing environment, not an emulation or simulation infrastructure. Emwin [13] and JEmu [2] are network emulators similar to MobiNet. However, they both do not have the level of scalability that we have achieved with MobiNet. There is also no support for plugging in ad hoc routing protocols. Judd and Steenkiste [4] describe an approach for wireless experimentation using a real MAC layer. While using a real MAC layer has advantages, scalability is limited as discussed above. Comparison between different MAC layers also becomes more difficult to perform.

## 6 Conclusions and Future Work

The overall goal of our work is to support controlled experimentation of a variety of communication patterns, routing protocols, and MAC layers for emerging ad hoc and wireless scenarios, including laptops, and PDAs. Current approaches to such experimentation include simulation and live deployment. While each clearly has its relative benefits and will continue to play an important role in mobile system design and evaluation, this paper argues for the power of modular, real-time emulation as another important point in this space.

To this end, this paper presents the design and evaluation of MobiNet, a scalable and accurate emulator for mobile, wireless and ad-hoc networks. MobiNet provides accurate mobile and wireless emulation, comparing favorably with existing network simulators while offering improved scalability. It allows researchers to rapidly experiment with a variety of MAC, routing, and communication (layers 2-4) protocols that may not be easily available in live deployments. MobiNet also supports the deployment of different mobility and traffic models. We further show the power of our emulation environment by running an unmodified video playback application communicating across an emulated large-scale multi-hop 802.11 network using DSR on stock hardware/software.

In most of our experiments, we validated MobiNet against ns2 to increase our confidence in the accuracy of our results. We felt that this was an appropriate choice because ns2, with mobile/wireless extensions, has undergone significant development and validation and remains one of the most popular simulators available. We leave comparisons against real wireless networks for future work. A detailed study of application performance under different traffic traces is another ongoing effort.

## References

[1] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, October 1998.

[2] J. Flynn, H. Tiwari, and D. O'Mahony. A Real-Time Emulation System for Ad Hoc Networks. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, January 2002.

[3] D. Johnson and D. Maltz. Dynamic Source Routing in ad hoc wireless networks, Mobile Computing, edited by Tomasz Imielinski and Hank Korth. pages 153–181, 1996.

[4] G. Judd and P. Steenkiste. Repeatable and Realistic Wireless Experimentation through Physical Emulation. In *Proceedings of HotNets-II*, November 2003.

[5] P. Mahadevan, A. Rodriguez, D. Becker, and A. Vahdat. MobiNet: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks . In *Technical Report CS2004-0792*, July 2004.

[6] B. Noble, M. Satyananarayanan, G. Nguyen, and R. Katz. Trace-based Mobile Network Emulation. In *Proceedings of SIGCOMM*, September 1997.

[7] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, April 1997.

[8] C. Perkins. Ad Hoc On Demand Distance Vector(AODV) routing, Internet-Draft, draft-ietf-manet-aodv-spec-00.txt. November 1997.

[9] C. Perkins and P. Bhagwat. Highly dynamic Destination Sequenced Distance-Vector(DSDV) for mobile computers. In *Proceedings of SIGCOMM 94 Conference on Communications Architecture, Protocols and Applications*, August 1994.

[10] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[11] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, Mi Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.

[12] Y. Zhang and W. Li. An integrated environment for testing Mobile Ad-Hoc Networks. In *Proceedings of MobiHoc*, June 2002.

[13] P. Zheng and L. Ni. EMWIN: Emulating a Mobile Wireless Network using a Wired Network. In *Proceedings of WOWMOM*, September 2002.