Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5523--03-10,054

NRL Mobile Network Emulator

William Chao
Joseph P. Macker
Jeffery W. Weston

Communication Systems Branch
Information Technology Division

February 6, 2003

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 6-2-2003	**2. REPORT TYPE** Formal	**3. DATES COVERED** *(From - To)* June 1, 2002 to December 31, 2002
4. TITLE AND SUBTITLE NRL Mobile Network Emulator		**5a. CONTRACT NUMBER** N0001403WR20018
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER 0602235N
6. AUTHOR(S) William Chao, Joseph P. Macker, and Jeffery W. Weston		**5d. PROJECT NUMBER** 03214
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER 55-5943-G35
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5320		**8. PERFORMING ORGANIZATION REPORT NUMBER** NRL/FR/5523--03-10,054
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 800 North Quincy St. Arlington, VA 22217		**10. SPONSOR / MONITOR'S ACRONYM(S)**
		11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Naval Research Laboratory (NRL) Mobile Network Emulator (MNE) allows experimenters, developers, and researchers to test actual mobile networking technology and scenarios under more controlled laboratory conditions with low cost and repeatability. In this report, we describe the emulation design and various software and hardware components. We also give simple examples of how the mobile emulation system has been applied with a set of ancillary visualization tools, motion generators, and network analysis tools. Finally, we discuss how this emulation environment can supplement more abstract simulation studies and more expensive and time-consuming field trials of mobile network systems and software.

15. SUBJECT TERMS

Emulator Mobile Network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** William Chao
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	18	**19b. TELEPHONE NUMBER** *(include area code)* 202-767-2186

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

CONTENTS

NRL MOBILE NETWORK EMULATOR

INTRODUCTION

There is a flurry of recent scientific activity in the study of future information networking systems, especially in the area of mobile, wireless networking for more autonomous operations. While numerous simulation environments and models have been developed for mobile routing and networking analysis, there is a growing interest in the development of a real-low-cost, flexible wireless mobile internetwork protocol (IP) test environment that provides flexible, dynamic topology control and manipulation. The Naval Research Laboratory (NRL) Mobile Network Emulator (MNE) does this. The test suite also provides direct and indirect support for node motion control and modeling. Along with an overview of our mobile emulator design, we describe developed methods and ancillary tools for mobile network analysis and performance studies. The emulation environment has been implemented with widely available, off-the-shelf commercial hardware within a commercial open source environment; low operating cost and flexibility were primary goals. The tool allows experimenters, developers, and researchers to test actual mobile networking technology and scenarios under more controlled laboratory conditions with low cost and repeatability. In this report, we describe the emulation design and various software and hardware components. We also give simple examples of how the mobile emulation system has been applied with a set of ancillary visualization tools, motion generators, and network analysis tools. Finally, we discuss how this emulation environment can supplement more abstract simulation studies and more expensive and time-consuming field trials of mobile network systems and software.

RATIONALE

The fact that mobile networking is a “hot” topic of present technology research and development hardly needs saying. Recent technical publications are inundated with reports of promising technologies and approaches for a better wireless future. This is understandable as the future applications for wireless networking are expanding rapidly and in many different dimensions. Wireless data networking, once viewed as a limited real estate with only privileged players or amateur operators, is now seen as a more or less limitless public resource. Much of this is due to the proliferation of low power, high bandwidth wireless local area network (WLAN) technologies. Spatial reuse of nonlicensed spectrum is now standard practice across metropolitan areas and is applied within a diverse set of industries. Even nontechnical private users are getting in on the act with the recent explosion of inexpensive wireless home networking turnkey solutions.

Many earlier wireless networking technologies tended to rely exclusively on significant infrastructure planning and hierarchy for general operations (e.g., cellular data systems). Newer technologies are addressing adaptive, self-organizing networking that can robustly interoperate and adapt to changes (minor or severe) within a localized network region or potentially across a larger wide area network (WAN). One emerging technology area, mobile ad hoc networks (MANET) [1], focuses on support for autonomous operation of dynamic, wireless networks within designated localized areas of a given IP-based wireless network. This technology is seen as an important advancement for the future of network-centric military operations and has many commercial and emergency-related applications as well. In addition to performing important research in this technology area, NRL scientists have played a leadership role in technology standardization and in the development and progression of MANET technology within academic forums.

Manuscript approved December 19, 2002.

Work on related MANET technologies is progressing rapidly within both commercial and academic forums and related Internet standards organizations. This technology has seen significant progress in the area of simulation and modeling for supporting protocol development, modeling, and analysis. In the recent past, we have performed extensive work within the various simulation environments to study mobile networking problems and protocol solutions within the Information Technology Division (ITD). NRL has also played a critical role in orchestrating a number of field trials of prototype mobile networking systems for various sponsors, including the Office of Naval Research (ONR) and the Defense Advanced Research Projects Agency (DARPA). Orchestrating a field trial of wireless mobile networking involves significant cost and logistical issues relating to mobile platforms, support personnel, network and experiment automation, antennas, and support equipment. The significant cost and logistics required to execute such a field trial can be limiting in terms of achieving meaningful testing that exercises a practical number of mobile nodes over appropriate test scenarios within the given time. Field trials are an important component of testing, but while important simulation and field trial work continues, we have become increasingly convinced that a flexible emulation environment is needed to bridge the gap between the more abstract simulation and actual field trial work, especially in the arena of mobile, wireless networking.

In conceiving our work, we envisioned a mobile network emulation system that was low-cost, flexible, and controllable. We desired to support the emulate of node motion in several ways including the use of captured motion files from live experiments, self-generation of motion as needed, and the reuse of related motion tools used within common simulation environments. In this way, both live field trials and simulation test results can be more directly compared to the emulator-generated results to better validate models and experiments under similar conditions. The emulation environment also provides a low-cost wireless mobile environment to execute a wide variety of traffic testing and motion scenarios using the same hardware equipment and software environments used in live wireless, mobile networks. In this sense, the emulator serves as a tremendously valuable, cost-saving debugging and analysis tool to supplement simulation and field studies.

MOBILE NETWORK EMULATION SYSTEM DESIGN

Our MNE design is based upon open source operating system capabilities. Although it can be easily extended to work on other systems, it is presently designed to run under Linux environments with IPTABLES network filtering capability installed. IPTABLES is fairly common network packet filtering software provided on newer operating systems to support firewall functions and filtering. If IPTABLES were not available, we would expect that the MNE approach could easily be adapted to use other dynamic network filtering interfaces. The emulator includes a software process that writes location information (e.g., emulated GPS longitude and latitude data) to shared memory. Other processes within the testing system, such as traffic generators, can read from this shared memory and use the dynamic information provided. The MNE can also be used to generate motion patterns that can be communicated via a back channel using IP multicast to all nodes participating in the emulation. In this way, emulation nodes can listen in on this channel to pick up pertinent location information for other nodes. The emulation modes can then compare this information with their own emulated location. By having this location and other information (e.g., transmit power) available locally, nodes can determine the nodes with which they should and should not be able to establish effective communication links. They can do this using a variety of wireless propagation models, the simplest of which is a basic range model. Using this model, when the calculated distance to a given node is beyond the specified range, all incoming packets from that node on the emulated (wired or wireless) interface are dropped before being delivered to the appropriate application using a MAC address filter. To the network applications, this makes the local link between the two nodes appear unusable, just as it would be in the real world, albeit with a hard limiting metric in this case.

Figure 1 shows a high-level overview of the MNE and how it is typically used in practice. Shown are two separate network interfaces per device. One of the physical network interfaces acts as a control channel while the other is used to artificially control and manipulate a dynamic network topology among the participating nodes. Figure 1 is a bubble diagram of a sample manifested topology.

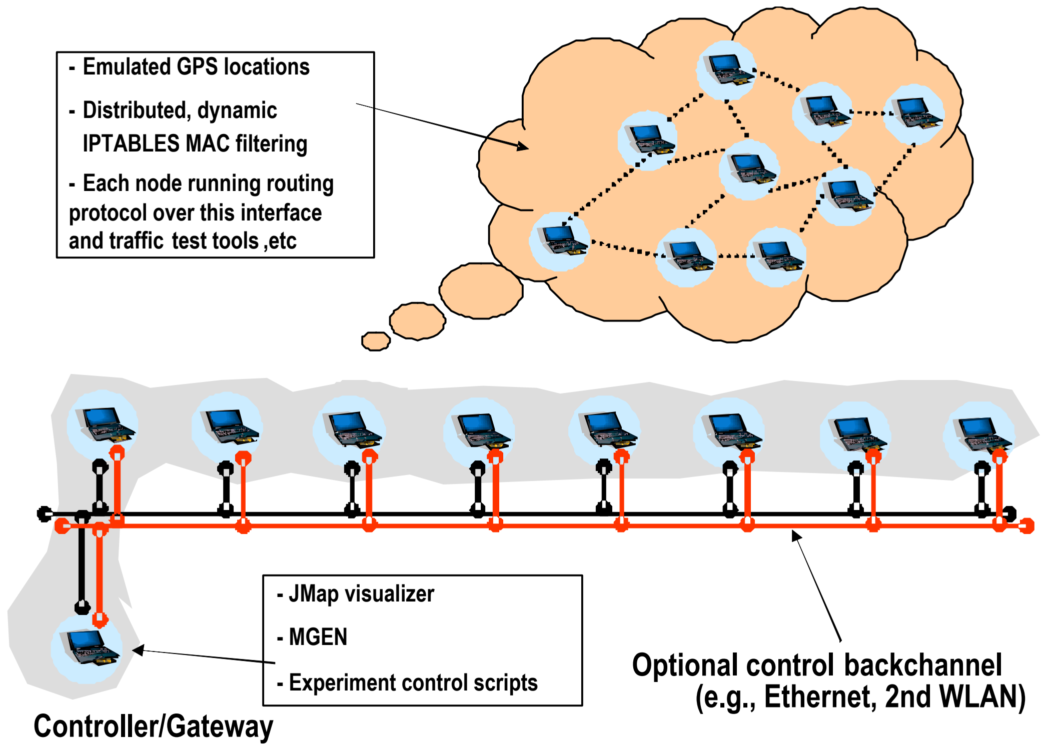
- Emulated GPS locations
- Distributed, dynamic IPTABLES MAC filtering
- Each node running routing protocol over this interface and traffic test tools ,etc
- JMap visualizer
- MGEN
- Experiment control scripts
Optional control backchannel (e.g., Ethernet, 2nd WLAN)
Controller/Gateway

Fig. 1 — A high-level view of the MNE

We reemphasize that the emulator does not presently attempt to emulate all conditions that would be experienced in the real world, including hidden terminal effects and detailed time-varying channel models. By using actual WLAN hardware in an ad hoc mode, some of the MAC layer effects can be realized in an emulator run, but detailed performance and other issues in specific real world environments should be cross-checked with a detailed simulation model or a real environmental experiment.

FUNCTIONAL COMPONENTS

Figure 2 shows the major function components of the MNE and their relationships to other system programs.

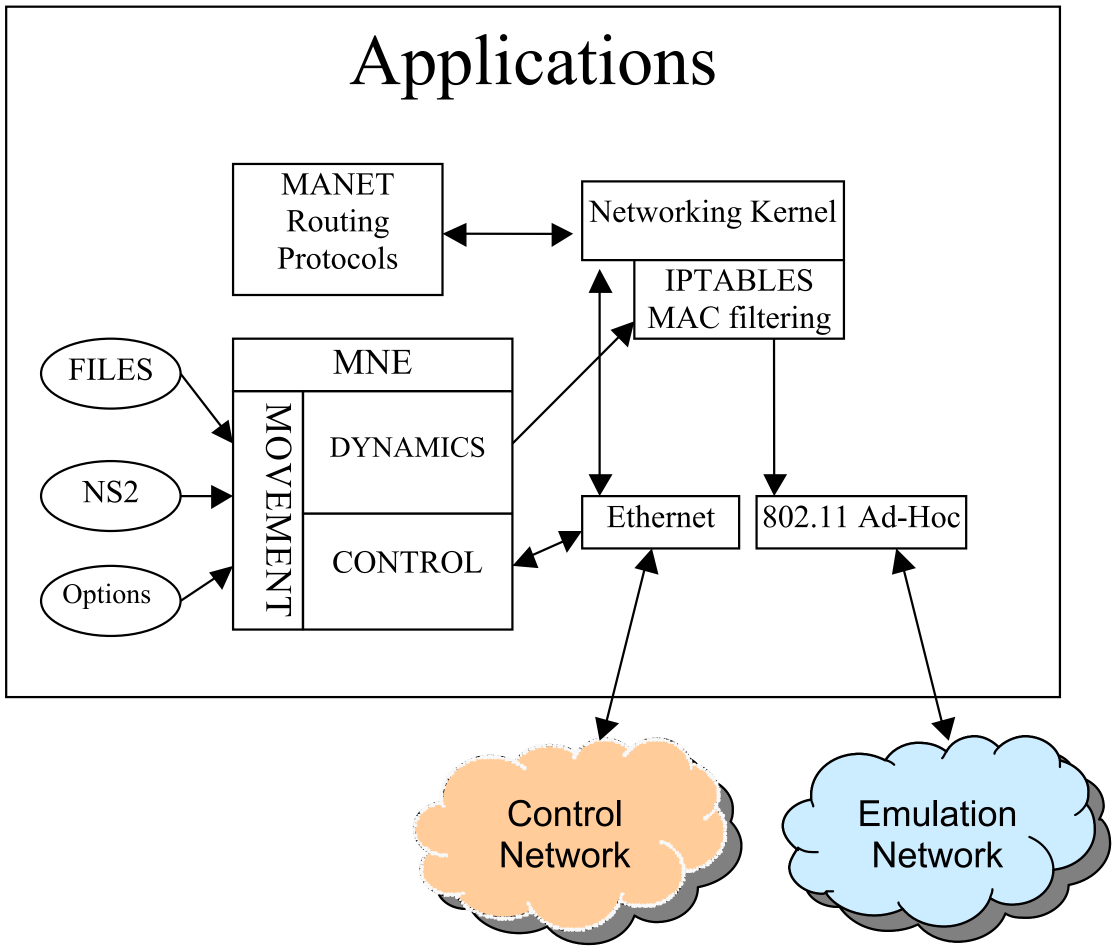

Fig. 2 — MNE components

Hardware Components

Control traffic to run a dynamic emulation is designed to operate independently from the user traffic sent over the dynamic network topology created by the emulation. The location information can be sent via a wireless or wired interface regardless of which interface is being used for emulating dynamic routing and user traffic. For example, we can make a testbed with a single hard-wired interface, a single wireless interface, or both a wired and a wireless interface. In the single-wired interface environment, the routing protocols can be exercised fully, but actual results may suffer under congested tests since control traffic to run the experiment is sharing the same channel as potentially congested user traffic. This possibility is still supported in function by the design and it may be cost- or complexity-advantageous to use a single interface (e.g., Ethernet) for basic testing purposes. Continuous transport of control traffic is supported even under dynamic motion and virtual network fragmentation because the control channel uses a well-known multicast address and port number that is not filtered by IPTABLES and is received continuously by all nodes. Instead of using a wired Ethernet, the use of a wireless interface (such as a WLAN) may better reflect some of the MAC layer behavior and interference issues unique to wireless operation. In both of these cases, when a single interface emulation environment is used and the network gets too congested, the location information sent out by the MNE can interfere with the normal test data, resulting in a lower throughput and increased latency. Similarly, the normal test traffic can interfere with the location information, resulting in nodes receiving stale position updates for other nodes. If nodes have out-of-date position information, blocking and unblocking may not be done at the correct times. Some priority packet processing of the control traffic can be done to provide probabilistic improvements to this condition, but interference effects cannot be completely avoided in this case. Therefore, we strongly recommend that two network interfaces are used for general operation of the MNE, especially when congested user data testing is desired. Since TCP traffic will naturally operate near or at network congestion levels, this is a fairly typical assumption to make for most realistic testing scenarios. In addition to two interfaces, the most recommended configuration is to use both a wireless (e.g., 802.11 in the ad hoc mode) and a hard-wired Ethernet interface. The normal test data, including the routing protocol and user data, can be sent out via the wireless interface, just as it would be in a real-world scenario. The hard-wired interface is used as the supplemental back channel for the emulation control information and status. Once again, a second wireless interface on a separate channel could also be used for the backchannel instead of a wired interface, but this is likely of no benefit for a tabletop emulation and may create additional wireless channel interference. We therefore recommend a typical wired Ethernet interface for control channel purposes.

Software Components

IPTABLES Filtering

In the initial design, a simple range model was chosen to emulate a nominal wireless link range, and a method for blocking packets from given nodes was needed. The input filtering function of IPTABLES is perfect for this. IPTABLES (IP6TABLES) is a standard tool that comes with the Linux operating system to filter the packets on a given network interface. Before a data packet is passed on for further network kernel processing, it can be examined by applying a set of rules to determine if the packet should be accepted or dropped from the “INPUT” chain. A similar set of rules applies to the “OUTPUT” chain. It is important to note that inserting and deleting an entry in the input chain of the IPTABLES can emulate the wireless connectivity or topological dynamics.

When two nodes are out of range from each other, blocking the output of wireless interface does not emulate the wireless neighbor link outage properly, because in broadcast packet radio systems, other nodes may be in range while a particular node is out of range. Only by selectively blocking input from transmitting source nodes can we more realistically emulate an out-of-range situation in a broadcast network. The blocking and unblocking is done by inserting or deleting the MAC address of the tested interface into the input filter of IPTABLES. Filtering by IP address was not chosen to perform the blocking and unblocking function because the nature of MANET is to forward one node’s IP traffic via other nodes. Blocking a particular IP address is therefore incorrect behavior and would disrupt the emulation of the IP forwarding capability. In addition, nodes cannot be blocked via IP if they do not have an IP, as when a node is performing broadcast requests or discovery services prior to obtaining a routable source address. While the uniform range model used in the initial design of link activation/deactivation is simple, it does not fit a more realistic model of probabilistic packet loss over dynamic network links. In recent updates to the link range mode, we have introduced the notion of a fuzzy range to emulate asymmetric wireless links and to add imperfection to the range model characteristics. In this case, the MNE randomly adds or subtracts a fuzzy range value from the uniform range to coarsely emulate the time-varying nature of many typical wireless channels. More sophisticated propagation channel models can also be added to the MNE design; there is even a *mangle* function available in IPTABLES that could emulate channel errors in a more sophisticated way. We do not support this functionality in the present emulator, but future versions may include such features.

Another recent design enhancement we have developed is a simplified emulation of wireless blockage within the emulated area that can be used to partially emulate real-world wireless obstacles such as buildings, towers, and hills. At present, we are considering such enhancements to link modeling, but we also wish to preserve the low complexity, low overhead nature of the present MNE design.

The Use of a Multicast Back Channel and Reliability

The nodes in the MNE environment need to know the location of every other node in the test, as they use such information to determine link propagation and dynamic connectivity effects in real time. Since the updating of this information needed to be sent to all participating nodes, multicasting the data is the logical design choice. Since basic IP multicast transport is an efficient but unreliable group delivery mechanism, we needed some multicast reliability to ensure proper operation of the control channel. A reliable multicast program called *mdpchat* [2] was used as the framework for the MNE multicast control, and the rest of the MNE – namely the multicast of movement information and MAC filtering – were built on top of *mdpchat*. Using *mdpchat* as our base design, we have the ability to reliably and efficiently send multicast location information to the other nodes over the backchannel with a reasonably high degree of confidence.

Node Motion Emulation

The motion emulation portion of the MNE has three distinct components integrated into one program. This emulated mobility can be used in ad hoc routing protocol tests and other tests that require location movement. The first part of the emulator is the **movement module** that provides the location information — represented in terms of longitude and latitude coordinates — and movement generation. The second component is the **dynamics module** that determines the range and potential wireless blockage and provides a dynamic filtering function for neighboring links. The **control module**, the third piece of the MNE motion emulation, provides a communications backchannel for the dynamic location information and run-time control commands between the nodes that are involved in testing. The Linux command line options of the motion emulator are shown in the following. The details of each option are explained in the next section.

```
Usage: mne -i <interface> -A <addr/port> -m <mode>
 -N <ipv4/ipv6 address> -M <mac address>
 [-t <max heading seconds>][-r <range>]
 [-F <fuzzy range>] [-o <obstacles file>]
 [-f <file_name>] [-s <speed>] [-p <pause>]
 [-h <height>] [-w <width>] [-u <user input flag>]
 [-R <rotation time>] [-S <Start Wait Point>]
 [-l <left/center longitude>] [-L <left/center Latitude>]
 [-g <start longitude>] [-G <start Latitude>]
 [-d <diameter>] [-D <cw|ccw>]
 mode: master|slave|file|random|static|line|circle|waypoints|ns2
```

Movement Module

The **movement module** of the motion emulation is responsible for generating node movement by putting the location information, represented in standardized global positioning system (GPS) format, into shared memory. We use the GPS format because it is compatible with other software tools that use the same shared memory technique for real-world mobile network experiments. In this case, the information is only used to track and report node locations. The movement module supports several modes of operations by using the **–m** command line parameter. The supported option modes are “file,” “random,” “master/slave,” “waypoints,” “ns2,” “static,” “line,” and “circle.” In each of these modes, the movement module is woken up on a periodic polling interval (e.g., once every second) by the control module to post a new location into the shared memory space. The new location is calculated by adding its current location with its current speed on its current heading.

In **file** mode, the movement module plays back a prerecorded motion file for a set of mobile nodes that were logged by the NRL gpsLogger program [3] during a live mobility field test. In **random** mode, the movement module applies a variant of a random vector model for motion [4]. The node picks a random heading and walks for a random number of seconds between 0 and the maximum walk time specified by the "-t" option. It then pauses for the number of seconds specified by the "-p" option before picking a new random heading and repeating the process. The random mode also has height and width options, specified by the "-h" and "-w" parameters, respectively, to restrict the limits of the random vectors. Moving into the virtual wall formed by the boundary restrictions will cause the node to bounce back, rather than continuing on a course that would take it out of the specified bounding box. The speed is controlled by the "-s" option. The longitude and latitude of the starting location are specified by the "-l" and "-L" options, respectively. The longitude and latitude of the upper-left corner of the area bound by the height and width restrictions are specified using the "-g" and "-G" options, respectively. Width and height are measured in meters, while speed is measured in meters per second.

In most modes, each node keeps track of its own location, and sends this location out via the back channel. Thus, the motion emulation is completely distributed. In another mode, the **master/slave** mode, a designated master node sends out the locations on behalf of all slave nodes. The slaves listen to these messages not only to see where other nodes are, but also to get updates on their own current location. The slaves broadcast sign-up messages periodically so that the master and the slaves can start and stop asynchronously. The master/slave mode uses the same command line options as the random mode and can perform the same random vector model variation described above.

We have also developed a **predefined waypoints** mode that is designed to support motion testing where nodes travel to predetermined waypoints, optionally stopping for a time at given *waypoints* if they reach one before their traveling time has expired. However, if time has expired before the node has reached a waypoint, the node moves on to the next waypoint. In this mode, the movement module waypoint rotation time is specified by the "-R" option and the starting location is specified by the "-S" option. The "tracks" are specified by the "-f" command line option, followed by the waypoint's filename. This file contains the longitude, latitude, flag, and single-letter name assigned to each waypoint on the track. The flag indicates whether the waypoint is a pass-through point, or if nodes should stop there during a waypoint rotation.

In the **ns2** motion mode, the "-f" option specifies the name of an NS2 motion file. The NS2 file is a particular movement description file used by the NS2 network simulation environment [5]. Various motion generators, such as the Bonn University Java-based mobility generator [6] and Carnegie Mellon University Monarch mobility generator [4] can generate these NS2 movement files. These tools can produce a variety of different motion models, including random vectors, grid motion, and clustered motion. The lines of the NS2 files are assumed to have times in terms of whole seconds, distances in meters, and speeds in meters per second. The "-N" option specifies which node number from the NS2 file is be used for the node that is running the MNE. The "-l" and "-L" options are used to specify the longitude and latitude of the starting location (0,0) of the mobility generators, respectively.

The **static** motion mode is pretty much self-explanatory – the node starts in the location specified by the "-l" and "-L" options and never moves from that position. In **line** mode, the node travels back and forth between the location specified by "-l" and "-L" options and the location designated by the "-g" and "-G" options. In **circle** mode, the node moves in a circle around a central point specified by the "-l" and "-L" options, starting at the location specified by the "-g" and "-G" options. Instead of calculating a point on the circle to start at, the "-d" option may be used to specify the diameter of the circle. In this case, the point at the top of the circle will be used as the starting location. By default, the node will move clockwise around the circle unless the "-D ccw" option is given.

Link Dynamics Module

The **link dynamics module** of the MNE is responsible for determining the range between nodes and if any obstacles are between nodes and carrying out the blocking action using IPTABLES or IP6TABLES. When the node name given by the "-N" option contains a colon (":"), it is assumed to be an IPv6 address, and the dynamics module will block the incoming packet using IP6TABLES when blocking is appropriate. When there is no colon in the node name, it is assumed to be an IPv4 address, and thus blocking is done with IPTABLES when the node goes out of range. The "-M" option, which specifies the MAC address, must be given so that the blocking action can be activated using IPTABLES' MAC address filtering function. The distance between each node is presently calculated periodically (e.g., once per second). The blocking or unblocking takes place accordingly. This is similar to APE (Ad-hoc Protocol Evaluation) and MobiEmu [7], except that we use the "input" chain to block the packet, whereas they use the "prerouting" chain.

The dynamics module also supports several fine-tuning operations. The "-r" option can set the nominal communication range of link operation. When the distance between a node and its peer exceeds the specified range, packets from that node are dropped by entering its MAC address into the IPTABLES filter. When an out-of-range node comes back in the range boundary, the entry is removed from the MAC address filter. Future versions on the NRL MNE may provide source transmit power and other parameters over the control channel to allow for more heterogeneous link dynamic modeling. The "-F" option makes the range "fuzzy" to mimic a more dynamic, real-world condition where the achievable network link may go in and out when a node is near the edge of acceptable communications range. This option presently provides crude time-varying link emulation but does provide some variability even when nodes are static. Future work (e.g., the transmit power information mentioned above) may allow us to emulate more complex time-varying channel models. As mentioned, the emulator also allows for obstacles. This can be used to emulate buildings or other obstructions that block direct line-of-sight wireless communications between a given pair of nodes. The "-o" option specifies an obstacle file. Each line of the file defines a pair of points (latitudes and longitudes) that form a line that causes a wireless obstacle. A direct wireless link between a given pair of nodes is blocked by this obstacle line if any part of the line falls directly between the two nodes.

Control Module

The control module is a modified version of *mdpchat*, the reliable multicast tool we mentioned and referenced earlier. The emulator uses *mdpchat* to send GPS location information between the nodes via a multicast channel. Modification of *mdpchat* was done to have it wake up the **control module** once per second to send its GPS location to other nodes via MDP. When the option "-u" is set to 1, the control module also supports the following run-time commands from user input so that emulation parameters can be altered or changed in real time during the course of an experiment.

The **"help"** command gives a brief list of commands that the MNE supports. The "**range** <meter>" command changes the communication range of all participating emulators by sending an "in flow" command to itself and all other nodes. The "**block**" command toggles the blocking behavior of all participating emulators by sending another "in flow" command to all nodes, including itself. When the blocking is turned off, all the "iptables" and "ip6tables" commands are not passed on to the kernel, and thus no further blocking or unblocking is done. The command **"gps"** toggles the ability to display the locations of other nodes when the multicast packets are received. The "**tx"** command toggles the ability to display the transmitted information of its own location.

OTHER RELATED TOOLS

During the course of its design, the MNE was integrated and tested with several other mobile network testing support tools. Those tools are not the focus of this document, but they are mentioned here to clarify and demonstrate their use in an MNE experiment. The Multi-GENerator (MGEN) is used to generate traffic and to receive the user test traffic for performing analyses of mobile network performance in support of the dynamic routing of user data [3]. JavaMap (JMap) [8] is a Java-based mobile network visualization tool developed at NRL and used to support a wide variety of testing and experimental applications. JMap dynamically displays the positions and general status of network nodes on a map using their GPS locations. For MANET routing experiments, JMap has also been extended to display the dynamic routing links between nodes in an experiment. Recent versions of JMap work for both IPv4 and IPv6 addressing systems. As mentioned, MGEN is often used with MNE to provide network test traffic and GPS location data in a real mobile experiment for display in the local node JMap tool. This approach is exactly the same approach taken in a live experiment with actual GPS data. When used with MNE, MGEN reads from the shared memory created by the MNE tool, so it is important that the MNE software is started first during a test.

Another tool of interest that is often used in testing is the TRace Plot Real-time (TRPR) NRL software tool developed as a traffic analysis tool capable of analyzing data in real-time and creating output for plotting by gnuplot, a standard multiple-operating-system-supported plotting tool. Plots such as data throughput rate, latency, jitter, and loss can be produced in real time, or after a test has been completed. TRPR also serves as a statistical post-analysis tool and can aggregate a large set of data for quick summary viewing or produce multiple plots for detailed viewing. Examples of TRPR output from an actual MNE experiment can be seen in the Results section of this report.

Overall, the MNE design is unobtrusive to the types of IP and network protocols used during testing. At present, NRL has used the tool in testing a number of MANET protocols including variants of the Optimized Link State Routing (OLSR) [9] protocol, a version of OLSR for IPv6, and variants of the Ad-hoc On-demand Distance Vector (AODV) [10] protocol.

A SAMPLE TEST SCENARIO

The following command lines illustrate a simple test scenario that uses the NRL MNE and a set of NRL network traffic tools to provide user data.

```
mne -i eth0 -m ns2 –f manhattan_movement.txt -A 224.100.1.1/5000 -r 300 -N
  $MYIP -M $MYMAC -l $long -L $lat –F 30 –o obstacle_file.txt
mgen -i eth1 mgenscripts/ManyTo1.mgen
drec -f -i eth1 -p 6000 data/drec.log
```

In this example, MNE uses a backchannel, *eth0*, and the multicast address/port is 224.100.1.1/5000. An ns2 movement file, *manhattan_movement.txt,* was produced by the Java-based BonnMotion tool and emulates the motion style of vehicles moving within a Manhattan-like city grid system. A fuzzy propagation range is set to 30 m. The valid maximum link range is set to 300 m. While optional, in this scenario a set of link interference obstacles is defined in the file *obstacle_file.txt*. In this example, the MGEN toolkit is used to establish user test traffic on interface *eth1*, which is the actual emulated, dynamic interface. The test pattern used for user traffic is ManyTo1.mgen, in the *mgenscripts* directory. Local received data are logged into the data file at *data/drec.log*.

RESULTS

To compare the MNE test results to real-world test results, a sample live test was conducted at NRL using a prototype MANET routing protocol. In the live test, 10 nodes were used. One remained in a central location (xcom), while the other nine nodes moved every 4 min from one waypoint to the next. A test pattern consisting of three phases of data volume was used. The first phase involved the nine mobile nodes each sending 10 kbps to the stationary node (xcom) for 10 min and xcom sending the same amount of data to itself. During the second phase, the rate for each node was increased to 50 kbps for 10 min. The rate was again increased to 100 kbps per node in the third phase. The motion waypoint rotation period for each phase remained 4 min.

OLSR Tests

The MNE test results are compared with those of the live test in the following figures. The data rate performance comparison is shown in Figs. 3 and 4. The throughput rate in Fig. 3 is the real-world throughput and the throughput rate in Fig. 4 is the emulated movement. The link range for the MNE was set to 300 m. The GPS location that was logged during the live test is used as input to the emulator's file playback. Figure 5 shows a JMap picture of the nodes and OLSR links. The links depict the routes between the peer nodes from xcom's perspective. Xcom is designated as "1:100" in the figure.

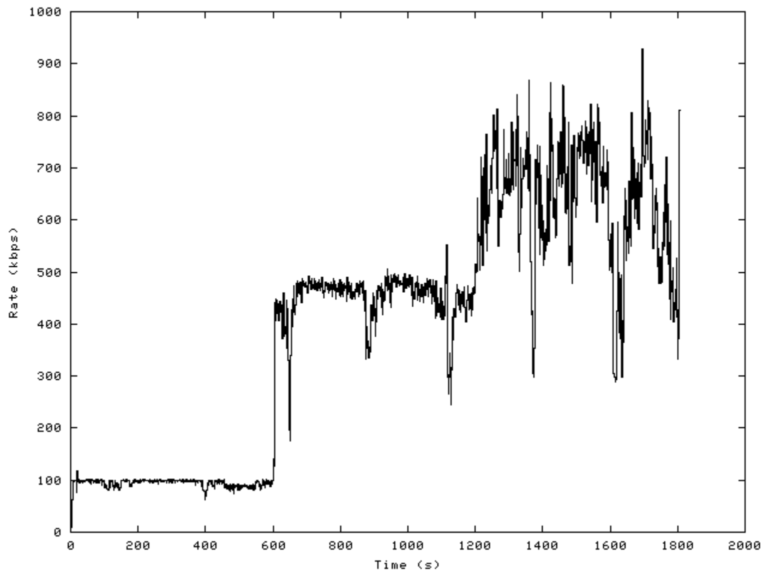

Fig. 3 — Sample live test data rate throughput

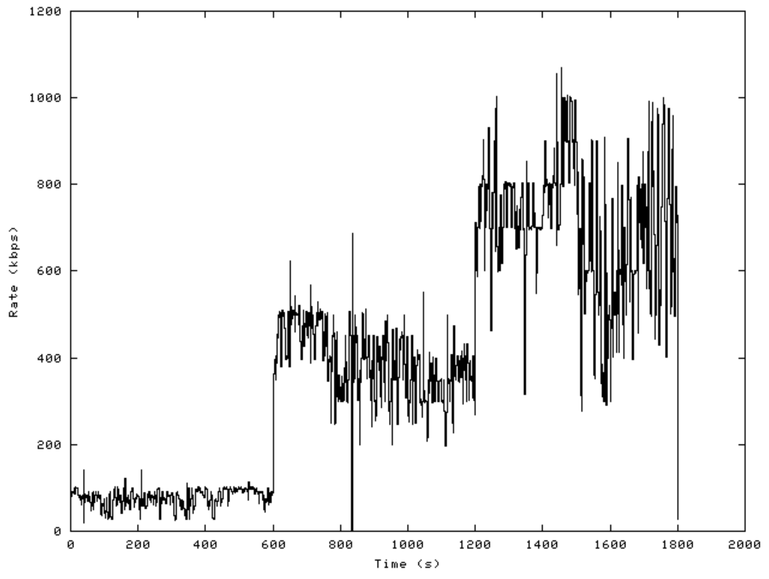

Fig. 4 — Sample emulation test data rate throughput

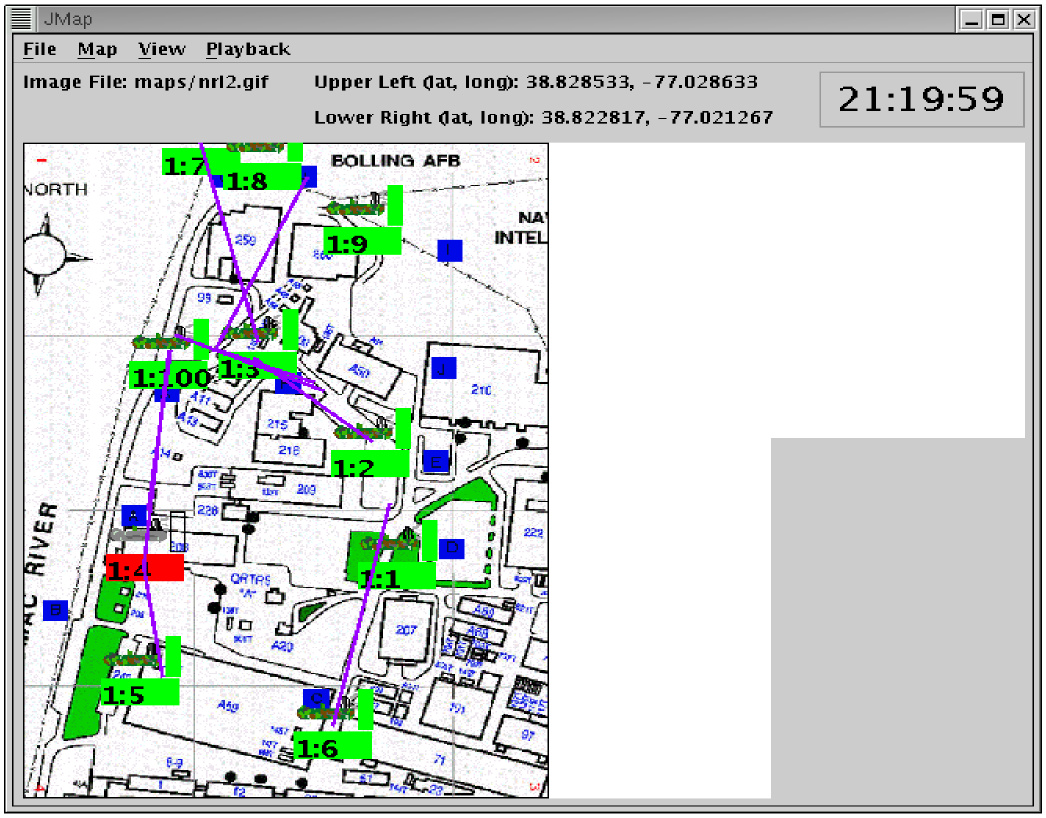
JMap
File Map View Playback
Image File: maps/nrl2.gif
Upper Left (lat, long): 38.828533, -77.028633
Lower Right (lat, long): 38.822817, -77.021267
21:19:59
BOLLING AFB
NORTH
RIVER
1:7
1:8
1:9
1:100
1:3
1:2
1:4
1:1
1:5
1:6

Fig. 5 — JMap visualization snapshot from sample test

Next Generation IP (IPv6) Testing

As another testing example, shown in Figs. 6 and 7, we analyzed a newer version of the OLSR routing code running IPv6 vs the original code designed for IPv4. IPv6 is expected to provide additional control and user data overhead during operation, and we hypothesized that significant performance differences might result from the newer design. We ran two back-to-back tests, first IPv4 OLSR and then the newer IPv6 OLSR, and then compared the MNE test results. Figure 6 shows the IPv4 data throughput rate, while Fig. 7 shows the throughput rate for IPv6. In both tests, the link communications range for nodes within the emulator was set to 300 m. Both tests used identical movement files to provide the GPS locations.

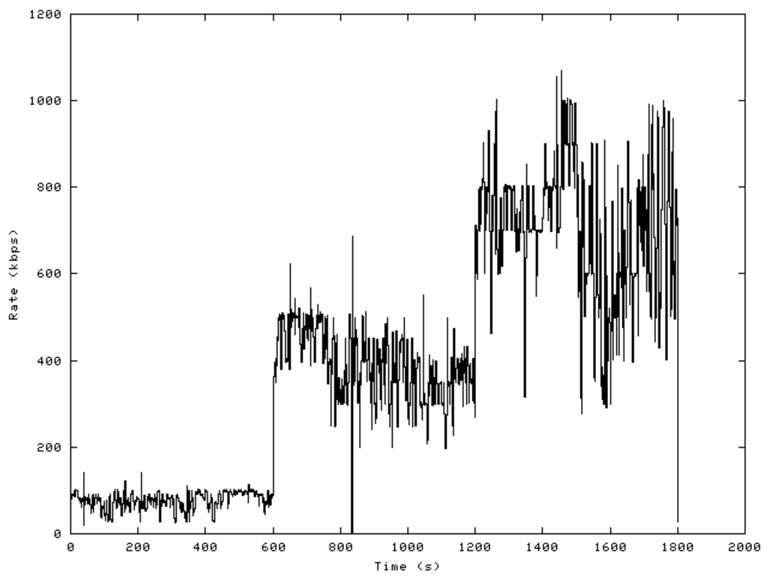

Fig. 6 — Throughput of OLSR IPv4 tests

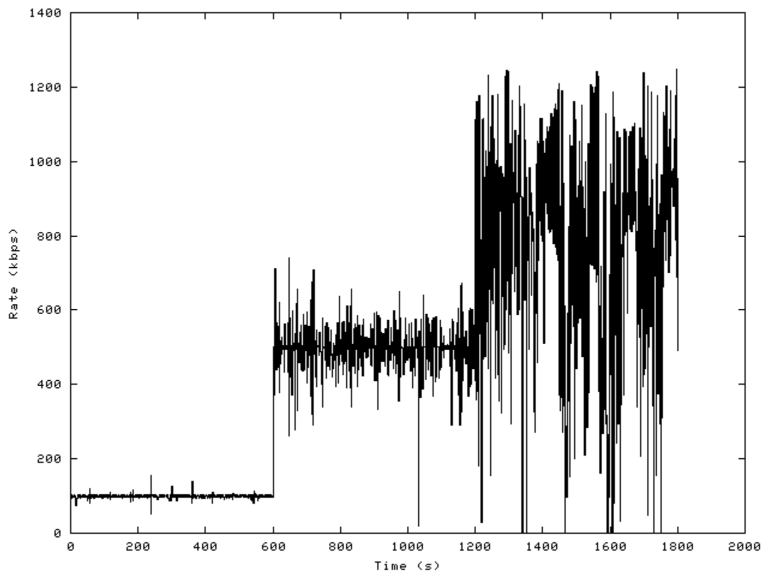

Fig. 7 — Throughput of OLSR IPv6 tests

NS2 Motion Test

The MNE can use simulator-based NS2 motion scripts as input files. The data rate plot shown in Fig. 8 demonstrates the NS2 options of the MNE. An NS2 Manhattan grid movement file was generated with the BonnMotion mobility generator [6] and used as input to the MNE.

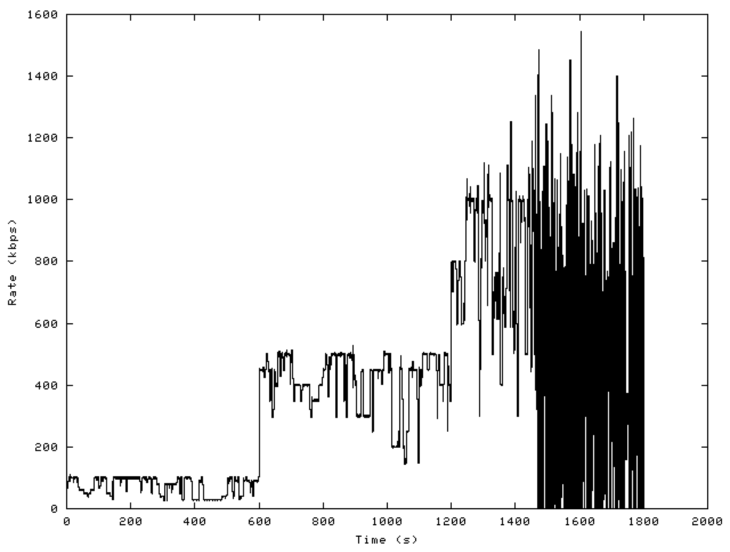

Fig. 8 — Throughput plot for Manhattan grid movement scenario

AODV Test

A test similar to the one done with the prototype OLSR protocol was done with another common routing protocol, AODV. Figure 9 shows the throughput rate graph. The test parameters are 10 nodes, waypoints mode, and three phases of data volume (10, 50, and 100 kbps), just like in the OLSR test. The communications range is again set to 300 m. A noticeable drop in the achieved test throughput can be seen around the 1500-s time frame. We believe this is caused by an implementation bug in the AODV implementation we were using that would have been difficult to find and trace during actual live experiments and may never show up under simulation abstraction. This demonstrates another valuable reason to use the MNE during protocol development and stress testing. In a live wireless, mobile experimental test, it is very difficult to rerun a test in order to reproduce the results due to environmental, cost, and test schedule constraints. However, in an emulated laboratory environment, we can repeat a stress test or other scenarios under controlled conditions in order to discover performance issues and further analyze details and phenomena.

POSSIBLE FUTURE WORK

At present, the MNE is a simple, low-cost mobile emulation toolkit that does not require significant amounts of processing or sophisticated propagation models to produce useful experimental results. As has been mentioned, the link dynamics models are presently coarse. More sophisticated wireless interference and channel models may be useful to add in the future, but such work would need to be traded off with the additional complexity and processing such functionality would introduce to the tool. At present, the tool is reasonable flexible in supporting a wide variety of motion models and routing protocols.

The authors are interested in enhancing the MNE in the future to better support the testing and automation of experimentation involving not just mobile routing, but also distributed autoaddressing and configuration of nodes. MNE can also be enhanced to support more dynamic user traffic scenarios as mobile nodes change roles throughout an experiment. The MNE tools would be useful to support such experiments under controlled or probabilistically defined conditions.

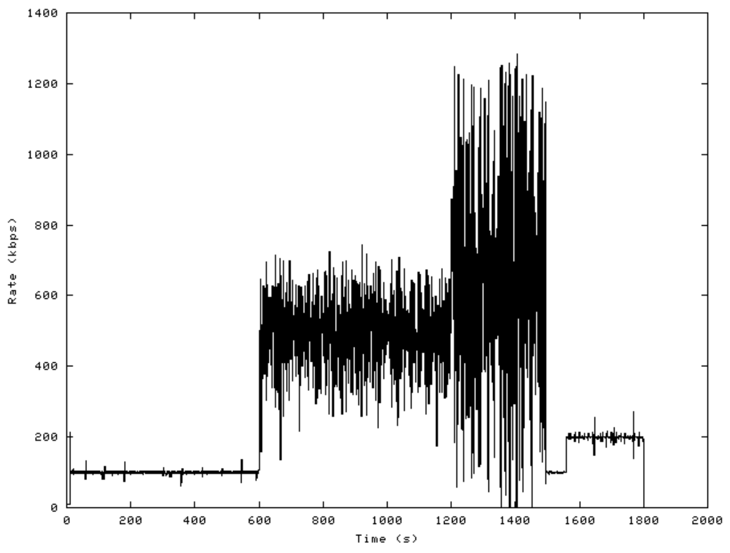

Fig. 9 — Throughput of AODV testing

SUMMARY AND CONCLUSIONS

We have described the design and the features of an NRL-developed, low-cost Mobile Network Emulator (MNE). The MNE is a valuable advancement as a mobile wireless testing toolkit for several reasons. First, it provides flexible emulation of motion and supports multiple modes of motion generation including various internal motion models, live experiment motion traces, and motion input from other tools (e.g., simulation environments). Second, the MNE is simple to deploy in a laboratory environment by building on standard, low-cost operating system kernel functions for packet filtering. Third, while remaining low cost and easily integrated, the MNE provides a sophisticated backchannel control mechanism to eliminate performance issues in congested and distributed network testing. Fourth, the MNE realizes the ability to run repeated, automated mobile networking experiments under a wide variety of controlled test scenarios within a small laboratory space. Thus, the MNE helps supplement the heavy cost and logistics burdens associated with live field trials.

In addition to a rich set of supported node movement models, the MNE has optional features to better emulate wireless communications, including a fuzzy range feature and blockage models. The emulator is not limited to one kind of movement and it can use the mobility files generated by other applications or live traces from actual mobile field experiments. The use of IPTABLES MAC address filtering makes the MNE easy to install within existing operating systems and makes its integration and overhead processing fairly seamless.

In addition to describing the MNE design, we presented a number of examples of emulator use and test trials using several types of movement and different MANET routing protocols. We also briefly described NRL–developed data analysis and visualization software tools typically used with the MNE to perform experiments. We showed in Fig. 3 that the MNE could use an actual live testing mobility trace file and this helped provide some validation of the emulation model correctness. Also in Fig. 3, we observed that the MNE produced results comparable to those of a much more expensive and time-consuming real-world experiment. In conclusion, the MNE is a good, inexpensive mobile network emulation tool that supplements simulation and actual real world experimental work. The MNE easily runs on typical laptop computers with a variety of standard interface cards providing a low-cost, flexible benchtop mobile network laboratory. It can also be easily configured to produce automated tests under a wide variety of mobility and traffic scenarios, reducing the need for personnel monitoring and interaction during testing periods.

ACKNOWLEDGMENTS

During development of the MNE, many people provided a great amount of help, ranging from ideas and coding to integration and testing. Julie Feys and Chris Morse helped with integrating the early test environments and providing graphic files and maps. Aaron Woo wrote the code for NS2 motion script file processing and integrated the IPv6 test bed. Ron Lee provided expertise in IPv6 routing and compiled and debugged the IPv6 routing code that was tested. Alex Powell coded an initial version of the MAC address filter interface, which he researched and adopted from the APE Testbed MAC killer. Brian Adamson kindly provided his mdpchat program and technical know-how regarding numerous protocols and test tools that are used in conjunction with MNE. Kenn Flynn provided JMap and made numerous modifications to support IPv6 address display and topology display capabilities. The GFC (Geodesy Foundation Classes) software [11] was used in calculating distances and headings between two sets of longitudes and latitudes.

REFERENCES

1. S. Corson and J. P. Macker, “Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Consideration,” IETF Request for Comments: 2501, January 1999.

2. J.P. Macker and R.B. Adamson, “The Multicast Dissemination Protocol Toolkit,” Proceedings IEEE MILCOM 99, Atlantic City, NJ, Oct. 31-Nov. 3, 1999, paper 21.1.

3. Protean Research Group Software Tools, Naval Research Laboratory, http://pf.itd.nrl.navy.mil.

4. D.A. Maltz, J. Broch, and D.B. Johnson, “Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed,” Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon Univ. (http://reports-archive.adm.cs.cmu.edu/anon/1999/abstracts/99-16.html), March 1999.

5. L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, “Advances in Network Simulation,” *IEEE Computer* **33**(5), 59-67, May 2000.

6. Bonn University Mobility Generator, 2002, http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/.

7. H. Lundgren, E. Nordström, and D. Lundberg, “The APE Network Testbed,” Proceedings IEEE Wireless Communications and Networking Conference (WCNC) 2002, Orlando, FL, March 17-21, 2002.

8. JAVA-based Mobile Visualizer and Mapping Application, Naval Research Laboratory, http://pf.itd.nrl.navy.mil.

9. T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, “Optimized Link State Routing Protocol,” Proceedings IEEE National Multi-Topic Conference (INMIC) 2001, Lahore, Pakistan, December 28-31, 2001.

10. H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin, “A Large-scale Testbed for Reproducible Ad-hoc Protocol Evaluations,” Proceedings IEEE Wireless Communications and Networking Conference (WCNC) 2002, Orlando, FL, March 17-21, 2002.

11. S. Blackburn, Geodesy Foundation Classes, http://samblackburn.com/gfc/index.html.