MASSIVE: An Emulation Environment for Mobile Ad-hoc Networks

Michael Matthes, Holger Biehl, Michael Lauer, and Oswald Drobnik
Department of Computer Science, J.W. Goethe-University
D-60054 Frankfurt/Main, Germany
Email: {matthes, biehl, lauer, drobnik}@tm.cs.uni-frankfurt.de

Abstract

Developing and evaluating protocols and applications for mobile ad-hoc networks requires significant organisational effort when real mobile ad-hoc networks with several mobile terminals are considered. For fine tuning and optimizing ad-hoc protocols, network simulators like the OpNet Modeller, NS2, or GloMoSim are most suitable. However, for the comprehension of the mode of operations of ad-hoc network mechanisms and their effects on the communication flow of applications a real-time experiment environment is more preferable. In this paper, an emulation environment for mobile ad-hoc networks is presented, which operates within existing local area networks utilizing virtual overlay structures reflecting the characteristics of mobile ad-hoc networks. Routing protocols and applications are able to communicate in real-time over emulated mobile infrastuctures using the unmodified IP protocol stack. The emulator provides an extensible interface for specifying mobility patterns in order to define and to evaluate diverse mobile scenarios. Several ad-hoc specific mobility patterns have been integrated and evaluated.

1. Introduction

Mobile ad-hoc network (MANET) protocols and applications can be evaluated by simulation as well as by emulation. Simulators like OpNet Modeller, NS2, or GloMoSim support the definiton of mobile ad-hoc scenarios and the use of several mobility models for evaluating the behavior of ad-hoc routing protocols. Applications are usually modelled through their characteristic traffic patterns. In general, simulations can not be performed in real-time, because of the enormous computational effort to simulate the *real world*, for example the wireless transmission characteristics like signal degradation and multi-path propagation. In many cases, the gathered data is being analyzed and interpreted after performing the simulation. The results are often used for fine-tuning and optimizing the considered protocols.

When designing a new protocol it is often more important to evaluate the overall protocol behavior than to optimize specific parameters. Hence, during the concept and design phase an environment for rapid prototyping for "proof of concept" is preferable, which helps in gaining a first insight in the behavior of a developed protocol. The implementation and evaluation within a simulation environment might be extremely time consuming. The configuration of a simulation includes a high level of detail and the evaluation of protocols takes place in non real-time processes. Furthermore, in order to evaluate the real-time interaction with applications and the impact on their communication flow in mobile ad-hoc scenarios, the protocol implementation needs to be adjusted in order to work outside the simulation environment.

An appropriate real-time experiment environment for evaluating ad-hoc applications should provide a communication infrastructure reflecting the dynamic characteristics of mobile ad-hoc networks. It should also utilize the regular IP protocol stack for communication, which is used in the target network of the real world. In this way, no additional effort would be necessary to deploy the developed protocols and applications. In this paper, such an emulation environment is presented, which defines virtual overlay structures over existing local area networks. Several management tools are implemented for configuring, visualizing, and manipulating virtual ad-hoc networks interactively.

In the following, an overview of existing ad-hoc emulation environments is given. Then, the distributed architecture of our developed system as well as its management and visualization tools are presented. For the validation of its usability, our first protocol and application evaluations are discussed.

2. Emulation Environments

Emulation environments for mobile ad-hoc networks can be classified using the following categories [7]:

Central Control Emulators: All stations acting as mobile nodes are connected to a central emulation server. All

Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services (WONS'05)
0-7695-2290-0/05 $ 20.00 IEEE

traffic is directed via that server, which forwards the traffic to the destinations according to internal parameters characterizing the current emulated network scenario, i.e. reachability, connection quality, collisions and more (cf. *Jemu* [2]). However, passing all communication through a central server represents a bottleneck and affects the scalability. Further, each packet must be analyzed by the server whether to be forwarded or dropped. This might impact the real-time behavior of communication flows.

Simulator Combined Emulators: As with the Central-Control emulators, stations acting as mobile nodes are also connected to a central server. The server represents a network simulator, which computes a detailed model of the wireless environment. Communication traffic is directed via the central simulator. For instance, the *NS2* simulator includes an emulation mode [1, 3], which operates in that way. Packets are forwarded via the *NS2* server, which decides whether packets get forwarded, delayed, or dropped according to the simulated network conditions. The emulation of the entire ad-hoc network regarding topology changes and ad-hoc routing takes place within the simulation model. Hence, only the upper protocol layers can be evaluated. Ad-hoc routing protocols need to be integrated into the simulator environment. Interactions between several protocol layers can not be evaluated.

Distributed Emulators: As opposed to the centralized approaches each station acting as mobile node is responsible for directing and forwarding traffic. Usually, a central control instance governs the overall network topology and regulates the configuration of each mobile node, e.g., setting the current neighbor stations within direct radio range. Examples for distributed emulators are the *MobiEmu* [6] and the *EmWin/EmPower* [7, 8] emulators. Usually, the emulation follows a predefined execution flow without any possibility for interactive manipulation. Further, *EmWin/EmPower* also pre-calculates multi-hop routes before the emulation process runs. Within a dedicated testbed, only a single emulation scenario can be managed at a time.

Only distributed emulators are able to support real-time evaluation of topology-related protocols. Centralized emulators primarily support the evaluation of end-to-end network conditions, but provide a higher degree of detail in the emulation of the wireless medium. Except *MobiEmu*, the combined evaluation of ad-hoc routing protocols and applications is not supported. *MobiEmu* on the other hand, does not support the real time intervention in running experiments. Our research focus is to study how applications are affected by the underlying network protocols and middleware. For the development of protocols and applications with respect to mobility characteristics of MANETs our group has designed and implemented a MANET emulation environment. The emulator presented in this paper follows the distributed concept, supports interactive manipulation of the emulation process and allows the coupled evaluation of routing protocols and applications. It does not focus on simulating wireless MAC layers and environmental conditions as accurately as possible.

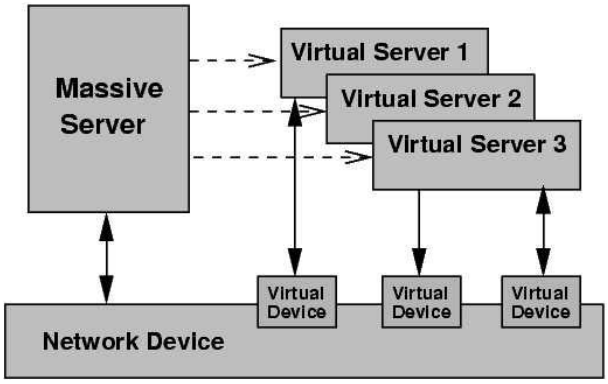

Figure 1. *Virtual Server Architecture.*

3. The Architecture of MASSIVE

Our emulation environment is called MASSIVE: *MAnet Server Suite Incorporating Virtual Environments*. The system design follows a distributed concept and addresses the important aspects needed to analyze the interworking of ad-hoc network protocols and applications:

- Interactive manipulation of emulation processes regarding network topology and routing.
- Use of the regular IP protocol stack in order to utilize the evaluation of routing protocols, applications and their interactions.
- Simple transfer of protocols and applications from the emulation environment to real ad-hoc networks.
- Concurrent management and emulation of multiple virtual ad-hoc networks.
- Visualization of topology and routing mechanisms.
- Management tools for configuring and controling virtual ad-hoc networks.

The emulation environment can run on a local area network with workstations running the Linux operating system.

3.1. Server Implementation

On each station of the local infrastructure a MASSIVE server is placed. A virtual mobile ad-hoc network (VMANET) is emulated through cooperating MASSIVE servers. Several VMANETs can be emulated concurrently. Every MASSIVE server hosts a virtual mobile node for each configured VMANET by instantiating a customized virtual

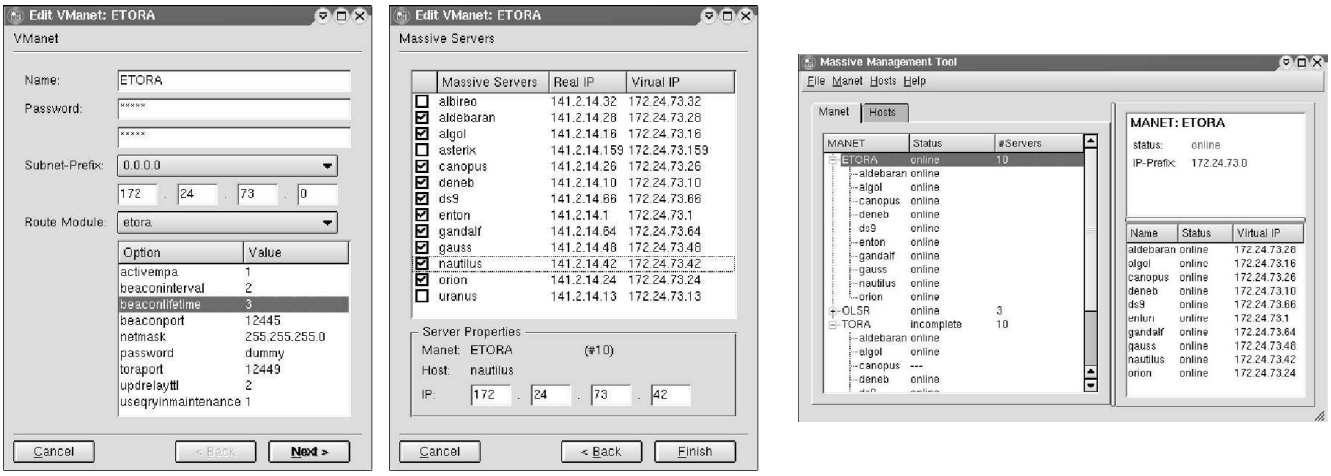

Figure 2. vMANET *Creation Wizard (left and middle) and Manipulation Tool (right).*

server (vSERVER) (see figure 1). A vMANET consists of a subset of available MASSIVE server stations and defines a virtual subnet within the local network. The vSERVER is responsible for the realization of the vMANET environment conditions regarding the emulated mobile node. A vSERVER configures and controls the corresponding virtual network device, which is mapped over the real network device of the server station. The virtual device is used for the communication within the vMANET.

The MASSIVE server provides an interface for the external access to control and status information methods. The interface is based on remote procedure calls (RPC). The specific interface methods are introduced in the following sections. MASSIVE limits access to privileged system calls, e.g., for manipulation of routing entries, based on password protection.

3.2. Neighborhood Control

The dynamic characteristics of ad-hoc networks are modelled through controlling the reachability of the virtual stations within the vMANET. The neighborhood of virtual mobile nodes within a vMANET is managed via *IPTables*. Our system uses the Linux packet filtering mechanism similar to the filter concept in the *MobiEmu* environment. The reachability of LAN stations regarding a virtual device is controlled by deploying filters in the kernel. The packet filtering uses a combination of IP and MAC addresses in order to bind filter rules to specific IP domains. In this way, several virtual ad-hoc networks can be managed concurrently within a single local area network.

A similar approach for MANET emulations has been carried out in [6] with the *MobiEmu* emulator. However, *MobiEmu* only considers MAC addresses for filtering. The implication is that all packets from a station are accepted resp. dropped regardless of the IP domain. Therefore, *MobiEmu* can only emulate a single virtual MANET at a time, whereas in our approach several vMANETs can be emulated concurrently. As opposed to *MobiEmu*, a password protected interface for manipulating the reachability of virtual terminals has been implemented. In this way, the efficiency of kernel-space packet filtering can be achieved.

3.3. Route Management

The MASSIVE server provides an interface for multi-hop route requests and cancelings: *routeRequest* and *routeClear*. However, the functionality of multi-hop route discovery is not part of the MASSIVE server itself. Requests for multi-hop route discovery are delegated to external ad-hoc routing daemons, which need to register their routing services with the MASSIVE server.

3.4. Status Information

MASSIVE provides a multi-featured user interface to gather current status information about the conditions of the emulated mobile ad-hoc network like real as well as virtual IP addresses, current neighbourhood condition, and route entries. MASSIVE further provides an extensible event-based subscription mechanism. Clients can subscribe with the MASSIVE server to obtain update information triggered by change events.

4. Management Tool

The vMANETs management tool provides a graphical user interface for setting up vMANETs, for assigning stations to them, and for choosing the routing protocol.

4.1. Configuration

First, the stations that should be included in the emulation process need to be selected. The managment tool provides a *scan* dialog, which probes the local network for available stations. The management tool also provides a two-step wizard to configure vMANETs (see figure 2). In the first step of the wizard the main properties of the vMANET can be specified: vMANET name, the access password, and the subnet prefix of the virtual ad-hoc network. Furthermore, an external routing engine can be chosen and customized. In the second step a list of active MASSIVE servers is presented, which can be added to the vMANET emulation. After these two steps a vMANET is configured and can be installed.

4.2. Distributed Emulation Control

The management tool provides a graphical control interface for the distributed emulation environment. It allows to control both the MASSIVE servers and the vMANET settings (see figure 2). vMANETs and MASSIVE servers are displayed using a tree structure. For each configured vMANET its state and the number of included vSERVERs are shown. Starting and stopping an entire vMANET initiates the appropriate RPC calls at each of the remote MASSIVE servers.

5. Visualization and Topology Control

For modelling the reachability and the dynamics within a vMANET a visualization and control tool has been developed. It provides a graphical user interface for arranging virtual mobile nodes within a vMANET interactively. The movement of the virtual mobile nodes can be managed manually (via drag and drop) or by using pre-defined mobility patterns. Several mobility patterns can be assigned (e.g., random, directed) for emulating the dynamics of ad-hoc networks. Furthermore, additional mobility patterns can be specified using a script based programming language. A convenient interface to the route management is also provided.

5.1. Visualization

Each vMANET node is drawn as a circle, which represents a specific vMANET instance on the corresponding MASSIVE server. The properties of a vMANET instance can be requested from the MASSIVE server. It includes some static parameters for the vMANET such as the vMANET reference name, its participating vMANET nodes, the address configuration of the virtual station and of the host station. Dynamic properties such as the current neighbors and

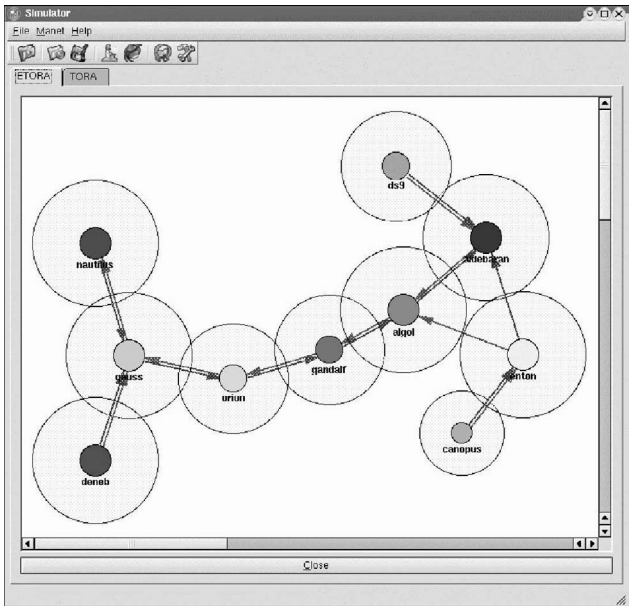

Figure 3. *Visualization and Topology Control Tool.*

the currently managed routes are also presented. Dynamically changing environment conditions can be updated periodically.

The range of the radio transmitters of the virtual mobile nodes is modelled by circles around the nodes. For visibility reasons the circles indicate half of the radio range. An overlapping of two different radio range circles denotes that the corresponding mobile nodes are within direct radio range. According to the type of a specific emulated mobile node (e.g., Laptop or PDA) the radio range can be adjusted individually (cf. different radio ranges of mobile nodes in figure 3).

5.2. Mobility Patterns

The topology of a vMANET can be modified by assigning mobility patterns. This allows to construct specific adhoc scenarios to observe the behavior of protocols and applications as a result of specific changes in the environment.

There are two categories of mobility patterns - creation patterns and movement patterns. Creation patterns specify where and when nodes should appear and disappear. Movement patterns specify the direction and the speed of nodes. Patterns can also be assigned to entire groups of nodes. Further, positive or negative values for “attractive forces” can be specified. This parameter can be assigned to nodes as well as to specific way points in the emulated area. It affects the movement of nodes within a specific vicinity of the source of an “attractive force”.

Creation and movement patterns can be specified using a scripting language embedded in the emulator. The script-

ing language provides powerful building blocks for creating complex patterns. For example, rules can be specified, which are executed based on events triggered by environmental changes within the emulation context. Furthermore, smaller (micro) movements can be grouped to form larger movements (macro). In the example pattern of figure 4, a node is switched on, stays in its place for 10 time units, then picks up a random node to follow for 50 time units. After that, it moves along a direction selected by random for another 20 time units.

```
switchOn()
while True:
    stay( 10 )
    anId = random( 20 )
    if anId != myId and existsNode( anId ):
       followNode( i, 50 )
    setRandomDirection()
    move( 20 )
```

Figure 4. *Micro movement pattern.*

Using creation and movement patterns, complex behavior models can be emulated, i.e., nodes forming *hot spot* areas with higher node density, nodes following other nodes, or nodes trying to efficiently scan a large area.

When vMANET nodes move in or out of each others radio range, the appropriate RPC calls are initiated at the vMANET instances of the corresponding MASSIVE servers.

5.3. Route Control

In general, there are two classes of route algorithms for ad-hoc networks: on-demand and pro-active routing protocols. On-demand protocols establish and maintain routes only for specifically requested target nodes. Pro-active protocols establish and maintain routes to every reachable target node without any explicit triggering. Both approaches are supported by the visualization tool.

For utilizing on-demand routing protocols, the visualization tool provides GUI-based methods to request multi-hop route setups between vMANET nodes.

For utilizing pro-active routing protocols, no explicit route request mechanisms are needed. As those protocols calculate routes to every node in a vMANET, visualization filters are implemented in order to observe the maintenance mechansim for specific routes. Those filters control the visibility of multi-hop routes. Hence, route mechanisms can be observed in different levels of abstraction.

6. Ad-hoc Network Scenarios

The MASSIVE emulation environment allowed us to arrange several ad-hoc network experiments. Using the emulators visualization capabilities, ad-hoc mechanisms and their interworking with distributed ad-hoc applications have been evaluated and improved. An enhanced on-demand routing protocol (see [5]) and a collaborative E-learning application have been linked together via MASSIVE. The application used for this experiment has been developed by our group and is called ELAN: *E-Learning in Ad-hoc Networks* [4]. ELAN models a scenario often found at universities. Several students use their laptops or PDAs to gather information, to carry out tests, to exchange information with other students, or to discuss the currently presented contents of courses. One objective is to find suitable collaboration partners in mobile ad-hoc networks, i.e., students with similar educational objectives. Hence, the first step for initiating a collaboration is to notify students about common interests, common research areas, commonly attended lectures, etc. In order to find students with similar educational objectives, interest profiles need to be specified and propagated. Applications get notified, when overlapping profiles are found, which indicate similar interests of students. Triggered by such a notification, the middleware of ELAN contacts the extended routing engine and requests route establishment between their mobile nodes in advance. This accelerates communication processes.

Figure 5 shows the awareness notification. The system has found a student with correlating interests. The degree of correlation with the other student and with offered services is shown in percentage. It is now possible to contact the student or to use offered services.

The scenario of figure 5 also shows the interaction between the E-learning applications and the MASSIVE emulator. Both application instances are configured according to the virtual subnet, which emulates the ad-hoc network characteristics. The mobile ad-hoc network is emulated within MASSIVE and the effects on routing protocols and applications can be evaluated in real time.

7. Conclusion

Rather than simulating every conceivable effect of the wireless communication media in a time consuming process, the presented emulation environment focusses on visualizing the mechanisms within mobile ad-hoc networks and their effects on several layers of the protocol stack. With the developed ad-hoc network emulator MASSIVE the effects of mobility protocols on applications can be analyzed in an interactive manner. Complex behaviors can be modelled using an embedded scripting language for mobility patterns. Using feedback rules specified in the patterns the emulation process can react to dynamic changes in the virtual ad-hoc scenario. Such an emulation environment helps in gaining an in-depth understanding in the functioning of ad-hoc mechanisms. It also helps in developing cooperating protocols on different communication layers.

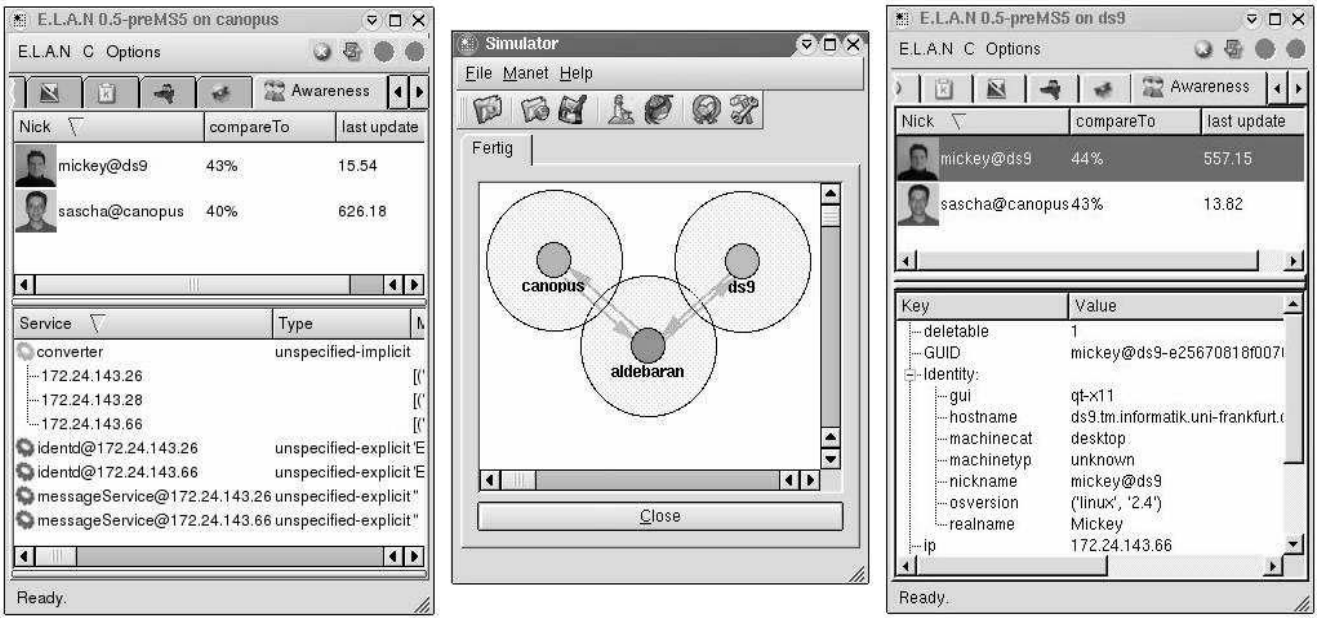

Figure 5. *Interaction between Applications and Routing Mechanisms using the* MASSIVE *emulator.*

The developed emulation environment presents a distributed server system. It includes several tools for configuring and maintaining the distributed servers as well as the virtual MANETs. Tools for the visualization of virtual MANETs and for topology control are also provided. The interworking of the emulation environment with ad-hoc routing protocols and applications has been presented. As a result of evaluation processes using the visualization tool, several protocol enhancements have been developed. Furthermore, the emulator has been proven as a valuable tool for educational lectures covering mobile ad-hoc networks and their complexity.

References

[1] K. Fall. Network emulation in the VINT/ns simulator. *In Proc. of the 4th IEEE Symposium on Computers and Communications (ISCC'99)*, July 1999.

[2] J. Flynn, H. Tewari, , and D. O'Mahony. Jemu: A real time emulation system for mobile ad hoc networks. *In Proc. of the First Joint IEI/IEE Symposium on Telecommunications Systems Research, Dublin, Ireland*, November 2001.

[3] Q. Ke, D.A. Maltz, and D.B. Johnson. Emulation of multi-hop wireless ad hoc networks. *In Proc. of the 7th International Workshop on Mobile Multimedia Communications (MoMuC'00), USA.*, Oct 2000.

[4] Michael Lauer, Michael Matthes, and Oswald Drobnik. A Framework for developing applications for adhoc environments. *In Proc. of the 4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, (SNPD'03), Luebeck, Germany.*, Oct 2003.

[5] Michael Matthes, Felix Apitzsch, Michael Lauer, and Oswald Drobnik. Application-oriented Routing for Mobile Ad-hoc Networks. *In Proc. of the fifth European Wireless Conference(EW04), Barcelona, Spain*, February 2004.

[6] Yongguang Zhang and Wei Li. An Integrated Environment for Testing Mobile Ad-Hoc Networks. *In Proc. of the third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), Lausanne, Switzerland*, June 2002.

[7] Pei Zheng and Lionel M. Ni. EMWIN: Emulating a Mobile Wireless Network using a Wired Network. *In Proc. of The Fifth International Workshop on Wireless Mobile Multimedia (WoWMoM 2002), Atlanta, Georgia, USA*, 2002.

[8] Pei Zheng and Lionel M. Ni. EMPOWER: A Network Emulator for Wireline and Wireless Networks. *In Proc. of IEEE InfoCom 2003, 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, USA*, 2003.