Experimentation of a MANET Routing Algorithm on the GENI ORBIT Testbed

Kang Chen, Ke Xu, Steven Winburn, Haiying Shen and Kuang-Ching Wang
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {kangc, kxu, sawinbu, shenh, kwang}@clemson.edu

***Abstract*—This paper proposes a systematic procedure for experimentation of Mobile ad hoc networks (MANETs) on the ORBIT testbed. MANETs have attracted significant research interests in recent years. Most of routing or file sharing algorithms in MANETs were only evaluated by theoretical analysis or simulations because of the requirement of large scale networks. However, due to the distinctive properties of MANETs, such as mobility and decentralized structure, it has been non-trivial to deploy a real testbed for the verification. The Global Environment for Network Innovations (GENI) project sponsored by the National Science Foundation (NSF) provides an exploratory environment for academic real-world experiments, such as the ORBIT testbed. A stable and repeatable procedure for experimentation on real testbeds is necessary and important to assure the validity of results. In this paper, a MANET routing algorithm, namely LORD, was tested on the ORBIT testbed, using the proposed procedure. Specifically, we first configure the wireless interface on each node to enable the communication between each pair of nodes. Then a set of methods are adopted to construct the MANETs scenario for test. The network status is monitored throughout the entire duration of experiments. Finally, the experiment results of LORD on the GENI ORBIT testbed are demonstrated.**

I. Introduction

Mobile ad hoc networks (MANETs) have become popular research topics in the past decades. Various issues in MANETs, such as hardware design, communication protocol and routing, have been widely studied by researchers. Among all topics, routing is an important issue since it is the key factor for the development of MANETs applications (i.e., file sharing, data collection, etc.). In MANETs, there is usually no centralized infrastructure, which means nodes communicate in a peer to peer mode when within each other's communication range. Also, network topology keeps changing due to node mobility. Such properties make traditional routing algorithms no longer adapt to the MANET scenario. Fortunately, many effective algorithms [1]–[5] have been proposed recently.

However, the deployment of large scale MANET testbed is non-trivial for the evaluation of MANET routing algorithms due to the previously mentioned natures of MANETs (i.e., nodes keep moving and decentralized structure). Also, large scale deployment requires high budget investment. Consequently, most routing algorithms are only evaluated by theoretical analysis or through simulators such as the ONE simulator [6], NS-2 [7], MobiREAL [8], and GloMoSim [9]. Though simulators try to emulate the MANET scenario in a realistic way, researchers still wish to evaluate the proposed algorithms in real testbeds with large scale to make the algorithms suitable for real deployment. Moreover, a real testbed can further help detect any practical problems for various MANETs applications.

To realize this goal, we propose a way to use the GENI [10] ORBIT testbed [11], which is a laboratory based network testbed, as a MANET testbed. The ORBIT testbed consists of 400 indoor nodes, each of which is equipped with 802.11 wireless card. Though the ORBIT testbed satisfies the requirement for large scale, it has no mobility and communication range limit, which are two important features of MANETs. A simple but effective way is then used to enable the two features in ORBIT. To enable stable and reliable testing environment, some monitoring scripts are developed. Finally, a set of procedures are proposed to evaluate MANETs routing methods in ORBIT.

Specifically, we first configured the 802.11 interface on each node in the same sub-net. Then nodes can communicate with any other node by knowing its IP address. After this, we added one layer in the LORD algorithm to simulate the mobility and communication range limit. In detail, each node is configured with a virtual position, which is dynamically changed according to the mobility model. Also, we let each node drop packets from nodes whose distance to the node is larger than the communication range. After this, we can regard the ORBIT as a MANET environment. We then evaluated the performance of a MANET routing algorithm, namely LORD [3], in ORBIT. The LORD algorithm is generally a geographical routing algorithm but requires no GPS information. During our experiment, some shortcomings of ORBIT were identified. For example, some nodes failed to remain stable. The wireless interfaces on them can easily stop working. We developed some scripts to solve node instability issue and to pick stable nodes in ORBIT to reduce the adverse effect brought about by the ORBIT testbed. In summary, our contributions are twofold:

(1) We proposed a set of procedures and scripts to test MANET routing algorithms on the GENI ORBIT testbed, which contains three steps: a) Test scenarios design, b) Initial wireless configuration and network status monitoring, and c) Experiment start and data collection.
(2) We identified the stability issue of wireless interfaces in supporting large scale MANET scenario on the ORBIT testbed.

The remaining of the paper is arranged as below. Related work is presented in Section II. Section III introduces the background on the GENI ORBIT testbed and the LORD algorithm. After this, we describe the design of our experiment in ORBIT in Section IV. Then, the experiment results are presented in Section V. Section VI concludes the paper with remark on future work.

II. Related Work

There are some MANET simulators developed by researchers [6]–[9]. The ONE [6] simulator is designed for the simulation of delay tolerant networks (DTN) and focuses on application layer message forwarding and storing. It can drive node mobility from real-world traces. The NS-2 [7] is a discrete event simulator for networking research. It can simulate almost all wired and wireless network scenarios. For MANETs, it can also import node mobility data from outside traces. Unlike the ONE simulator, it also supports the simulation of all lower layers (i.e., link layer and physical layer), thereby can provide more insight on wireless communications in the network. MobiREAL [8] is a realistic network simulator for MANET with support on realistic mobility of humans and automobiles. GloMoSim is a parallel discrete-event based simulator for wireless communication. Similar to NS-2, it also supports the simulation on all layers.

Kropff *et al.* [12] surveyed a list of real world and emulation testbeds for MANETs. It introduces and compares these testbeds in three key aspects: mobility modeling, wireless medium modeling, and testbed architecture. Since our focus is not to investigate testbeds, we introduce some of them in the three aspects. The ORBIT [11] platform has 400 physical nodes with IEEE 802.11 connection. It can realize real mobility through antenna switching. EWANT [13] is similar to ORBIT but has only 4 physical nodes. Hiyama [14] realized a real MANET testbed on stairs environment with 6 laptops, which cannot support large scale experiments too. MobiNeT [15] and NET [16] have 200 and 64 physical nodes, respectively. However, their wireless connections are simulated through wired connections. In summary, current testbeds except ORBIT are either in a small scale or fail to provide real wireless connectivity. This validates our selection of ORBIT for the evaluation of MANET routing algorithms. Liu implemented declarative policy-based adaptive MANET routing protocols on the ORBIT testbed [17]. This work focuses on the design of protocols and used 33 nodes on the testbed.

III. Background

In this section, we introduce the background on the GENI ORBIT testbed and the MANET routing algorithm (LORD) that we are going to evaluate.

A. GENI ORBIT testbed

The GENI ORBIT testbed is a network testbed operated by the WINLAB at Rutgers University. It has a large two-dimensional (20x20) grid of nodes equipped with 802.11 wireless cards. Each node is basically a Linux PC. It is available for remote access for all registered users. However, the grid is allocated by time slots, and each user can only use the resource during its allocated time slots. The main components of the ORBIT experiment service are the node handler on the Console and the Node Agent located on each node. The node handler is the controller of experiments. It sends commands to selected nodes at pre-defined time and tracks the execution. The node agent listens and executes the commands from the node handler. The ORBIT system also supports the function of disk image saving and loading in order to improve the experiment efficiency. When a user is granted the access, he/she can log on the console and consequently can log on all nodes.

To run an experiment, the standard way is to write a Ruby script and transfer it to the node handler, which uses it to start and control the experiment. The script should describe following items

(1) Required resources.
(2) Their initializations and optional associations.
(3) Their operation throughout the experiment duration.

However, we adopted a simple way to develop and run a shell script on the console node, which logs on each test node and launches our programs. We will elaborate this step in detail in Section IV.

B. The LORD Algorithm

The LORD algorithm is a locality-based distributed data sharing system for MANETs. It has two major components: DHT-based data index and retrieval protocol and 2) locality-aware regional-based data routing protocol. It represents each data item by a metadata which records the keywords of the data and the actual location of the node holding the data. To facilitate the data storage and retrieval, LORD divides the entire MANETs area into a number of geographic regions. Each metadata is then mapped to a region through the locality sensitive distributed hash function and is copied to all nodes in the region. When nodes move, they update the metadata information held by them accordingly. Then, when a query is generated, it is first transferred to the mapped region, which is obtained through the same mapping process of storing the metadata (i.e., using the same hash function). When the query arrives at the region, it tries to find the metadata which has higher similarity with the query than a predefined similarity threshold. If one metadata is found, the query goes directly to the node holding the actual file based on the location information in the metadata. In the query forward and response process, a region-based geographic routing algorithm is proposed, which is the key of the LORD algorithm. Therefore, we introduce the routing algorithm with more detail below.

Recall that the LORD divides the whole area into several small regions and each region is labeled with a landmark. During the routing, each node uses such information to select next-hop for the query/data forwarding. Specifically, each region has left-side and right-side angle range with respect to another region. The left-side angle range is the angle between

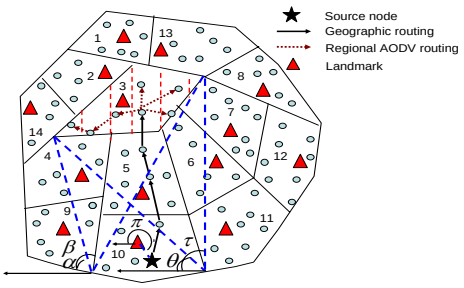

Fig. 1. Demonstration of the routing process in LORD.

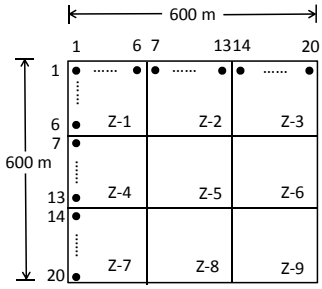

Fig. 2. Test area of ORBIT.

the most left vertex of the region to the leftmost and rightmost vertices of another region. Similarly, the right-side angle range is that of the most right vertex. For example, in Figure 1, region 10 and region 3 have left-side angle range $[\alpha, \beta]$ and right-side angle range $[\theta, \tau]$. The two angle ranges serve as the direction for query/data forwarding. If the node currently holding the query is on the left side of the landmark, it selects the furthermost node in the left-side angle range as the next-hop. Similarity, if the node is on the right side of the landmark, it selects the furthermost node in the right-side angle range as the next-hop. With such a design, the query/data is forwarded to the destination region gradually. When the query/data arrives at the destination region, it uses broadcasts if it is in the metadata storage stage or uses AODV if it is in the data transmission process to find the destination node.

IV. Experiment Design on the ORBIT testbed

We introduce how we test the LORD algorithm on the ORBIT testbed in this section. Specifically, we describe how we create the test scenario from the fixed ORBIT grid, how we configure the wireless interfaces on ORBIT nodes, and how we launch the experiment and collect experiment results.

A. Test Scenario Design

The ORBIT grid contains 400 (20x20) nodes in total. It has neither area size nor node position information. Therefore, we have to design that to fit our test requirement. Recall that LORD is suitable for relatively dense MANETs. Then, considering the total number of nodes is limited to 400 in ORBIT and not all nodes are available during the test (we will explain this later), we defined a mid-sized square area with side length equals to 600m, as shown in Figure 2. Also, similar to the design in LORD that the whole area is split into several areas, we divided the whole area into 9 equal size zones, which are labeled as Z-1 to Z-9 in Figure 2.

Though there is no node position information, each node has a coordinator expressed as (x, y) with $x = 1, 2, \cdots, 20$ and $y = 1, 2, \cdots, 20$. We used such information to infer the zone a node belongs to and its virtual initial position. The zone id of a node is calculated as

$$Z = \lfloor y/7 \rfloor * 3 + \lfloor x/7 \rfloor + 1 \tag{1}$$

and the virtual initial position of each node (x_p, y_p) is scaled from its coordinator:

$$\begin{cases} x_p = x * M \\ y_p = y * M \end{cases} \tag{2}$$

where M is the scaling factor and was set to 30 in our test. The position information of each node is piggybacked in the beacon message of each node. When a node, say i, with position (x_{pi}, y_{pj}), receives a message from another node, say j, with position (x_{pj}, y_{pj}), node i calculates the distance between them as

$$d(i,j) = \sqrt{(x_{pi} - x_{pj})^2 + (y_{pi} - y_{pj})^2} \tag{3}$$

Based on the distance, we can realize the feature of communication range. If the distance between two nodes is larger than the defined range, the received messages on either node from the other node are dropped directly. Otherwise, messages are accepted for further processing.

We further defined node mobility based above designs. We used the random way point mobility model [18] that each nodes move randomly in its zone area. Specifically, each node first randomly picks a destination position in its zone area, then moves to that position with a certain speed. When it arrives at the destination position, it randomly selects another position as the new destination. Such a process repeats throughout the experiment.

B. Wireless Configuration

Before running the LORD algorithm to do experiments and collect data, the most important preliminary is to configure the ORBIT radio nodes and the network interfaces to satisfy the experimental requirements in the assumed scenarios. Actually, the effectiveness and stability of configuration is an important factor in deciding the consistency of experimental results, based on the experience of the authors and the ORBIT community in the past. In order to smoothly run the LORD algorithm and make data trustworthy, a standardized process of testbed configuration was developed and proven in this project, which also provides reference for other experimenters in the future. All the scripts and information can be found in our website [19].

The entire testbed configuration falls into three parts: (1) Operating System (OS) image loading, (2) wireless interface configuring, and (3) network status monitoring. The first two parts are completed before starting experiments, which

provide the required system and network environments. The third part is carried out during the whole experiment session, which actually guarantees that the experiments can be done effectively and stably without apparent influence from the performance fluctuation of the testbed, thereby ensuring the trustworthiness of collected data. The details of these three parts for testbed configuration are introduced below.

1) Operation System Image Loading: Each ORBIT node is a PC with a 1 GHz VIA C3 processor, 512 MB of RAM, 20 GB of local disk, two 100BaseT Ethernet ports, and two 802.11 a/b/g cards. Ubuntu is chosen as the OS used for running LORD due to its ease to deal with and popularity in the Linux family. Ubuntu 9.10 and Linux 2.6.31-19-generic kernel is used in this project. We simply use the loading command to load the saved image with the above OS configuration and necessary scripts developed by us. The OS image loading is a time-consuming process, especially when involving a large number of nodes in the grid.

2) Wireless Interface Configuring: The ORBIT testbed provides two modes for wireless configuration: ad hoc mode and infrastructure mode. In the former, nodes communicate with each other directly, while in the latter mode the messages exchanged between two nodes are actually relayed through a central access point (AP). Intuitively, the ad hoc mode is more close to the MANET scenario. However, after many trials and several rounds of discussions with the ORBIT community, we found the performance of ad hoc mode for a large scale network is unstable: lots of nodes lost connection after several minutes, and such issues involving driver implementations are difficult to debug and fix. Therefore, the infrastructure mode, which proves to be more stable for experiments after testing, was considered instead of ad hoc mode. In the infrastructure mode, one node is selected as an Access Point (AP) and it functions as a forwarding node in the network. All nodes use *ath5k* as the wireless driver for Atheros based wireless chipset in the Linux Kernel [20].

In order to make the AP work properly, *hostapd*, which is a user space daemon for AP, was installed. The *hostapd* implements IEEE 802.11 AP management, which is a daemon program that runs in the background and acts as the backend component controlling authentication [21].

The steps for configuring the AP and other nodes are the same, which are given by Table I. The last step is to create mapping entries of all the nodes in address resolution protocol (ARP) cache on each node, in order to reduce the number of broadcast packets in the network. Please notice that although the same steps are used for the configuration of AP and other nodes, the order is to first finish the configuration of the AP and start the *hostapd* in the background, and then configure all other normal nodes to make them successfully associate with the AP. In this experiment, all the nodes are in the same subnet and on the same channel. Eventually, the IP addresses of nodes successfully configured are saved in a log file called "success.txt".

3) Network Status Monitoring: Once all the above steps are done, the testbed is ready for use. Although the testbed is well-configured for running the LORD algorithms, during an experiment session, network status monitoring is still necessary in order to avoid unpredictable influences on the testbed, thereby guaranteeing the validity of each experiment. The monitoring scripts run in the background, and check the wireless connection between nodes periodically. We define a node that can be connected through the wireless interface as a good node and a node that is not connectable as a bad node. The success of TCP connections (SSH are used in the scripts) between pairs of nodes are examined to determine whether the nodes are good or bad. The log file (i.e., good_num.txt) is created and updated, in which the information of good nodes in the network is saved. Once bad nodes which lose connection are found after analyzing the results in "good_num.txt" by monitoring scripts, the configuring script will be executed automatically to reconfigure those nodes. Based on our observation, the wireless links of more than 98% of the nodes (70 nodes are tested totally) still perform normally in a test lasting for more than one hour.

TABLE I
NETWORK CONFIGURATION COMMANDS.

Steps	Command
Add wireless driver	modprobe ath5k
Add IP address and subnet mask to the interface	ifconfig interface *ip_addr/prelixlen*
Set the hardware address of the interface	ifconfig interface hw ether *hw_addr*
Set the channel in the device	iwconfig channel #
Set the ESSID	iwconfig essid name
Create an ARP address mapping entry for IP address with hardware address	arp -s *ip_addr hw_addr*

The major problems of testbed configuration and solutions are summarized as below.

- The MadWifi driver is not stable based on several tests. Instead, *ath5k* is chosen as Linux kernel driver for WLAN, which is a new and emerging driver replacing MadWifi in the long run.
- Ad hoc mode blocks some wireless interfaces after several minutes, even if no experiments running on the testbed. Infrastructure mode is proven to be relatively stable.
- Broadcasting packets in the network after initial configuration block some wireless interfaces. Creating ARP entries on each node when configuring the wireless interfaces can solve this problem.
- Pinging or sending UDP packets are not suitable for monitoring the network status because of packet losses. TCP connection is proven to be a way to monitor the status of wireless links between pairs of nodes.

C. Experiment Initiation and Result Collection

The ORBIT tutorial introduces a systematical way to start experiments and collect results. It uses Ruby to write a script to let the node handler in the console control the

experiment execution by indicating which nodes are to be used, and when to start and stop the programs. During the test, experiment measurements and traces are collected and stored in a database.

However, considering that our experiment requires almost no control once it is started and the test results are all stored in the traces, we adopted a simple way to start our experiment and collect test results. Specifically, after completing all previously described settings and configurations, we first use a shell script to transfer our program to each selected node, and then run another shell script to start all the LORD program on selected test nodes. When all queries are done, we run the stop script to terminate all LORD programs and transfer the trace files back to the console. We then analyze the collected trace files to evaluate the measured metrics.

V. Experiments

With aforementioned design and network configuration, we conducted experiments on the GENI ORBIT testbed. We observed both the stability of the ORBIT testbed and the efficiency of LORD with different number of nodes.

A. Settings

In the wireless configuration step, we selected node (4,7) as the AP node and set the IP of a node as "10.13.x.y", where x and y represent the coordinator of the node. For the query generation, we let each node generate one query every second for totally 10 seconds. The destination area of each query was randomly selected from the nine zones shown in Figure 2. The communication range was set to 250m.

Though there are 400 nodes in the ORBIT testbed, we found the number of nodes that can be accessed stably through the wireless connection is less than 100, as shown in next sub-section. As a result, we conducted experiments with [30, 60] nodes. We tested the performance of LORD in scenarios with and without node mobility. In each test, we measured following two metrics:

(1) Success rate: the percentage of queries that arrive at the destination zone successfully.
(2) Average path length: the average number of hops that a successful delivered query has traversed.

B. Results on ORBIT Stability

Recall that we monitor the number of nodes in "good" status (i.e., connectable) after the wireless interface configuration step in the "good_num.txt". We ran tests for the LORD algorithms several times during the period in TABLE II, and each run took up to 3 minutes approximately. The logged information in one of our experiment is shown in Table II.

We find in Table II that the number of good nodes roughly lies in the range of [40,100]. By monitoring the network status, the number of good nodes before and after each run can be observed to determine the validity of results. We also find that with our experiment running, the number of good nodes decreases. Recall we mentioned in Section IV that the number of good nodes remains stable after configuration. But during that test, the LORD program was not launched on each node, which may be the reason for the decrease of good nodes. However, we are not clear what causes these phenomenon. Is it caused by the capacity of the AP node, the LORD program, or the inferences among all nodes? We will find the answer for this question in our future work.

TABLE II
The Number of Good Nodes.

Time	Amount
Fri Feb 3 18:04:23 UTC 2012	85
Fri Feb 3 18:22:30 UTC 2012	104
Fri Feb 3 18:40:31 UTC 2012	80
Fri Feb 3 18:58:40 UTC 2012	61
Fri Feb 3 19:16:56 UTC 2012	48
Fri Feb 3 19:34:58 UTC 2012	48

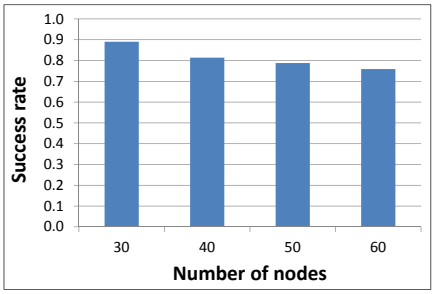

Fig. 3. Success rate without mobility.

C. Results on LORD

1) Without Node Mobility: Figure 3 and Figure 4 show the success rate and average path length of LORD when the number of nodes increases in the scenario without node mobility, respectively. We see that when the number of nodes increases, the success rate decreases while the average path length maintains almost in the same level. For the success rate, intuitively, when the number of nodes increases, more nodes would generate more query forwarding opportunities, which should increase the query success rate. However, we observe a decrease trend of the success rate in the figure. This is because 1) we let all nodes generate queries at the same moment, when there are more nodes in the system, the number of queries generated in a unit time also increases,

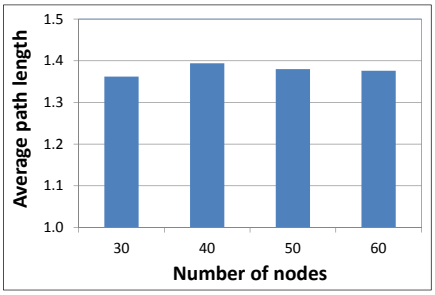

Fig. 4. Average path length without mobility.

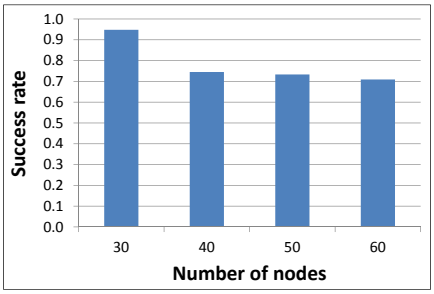

Fig. 5. Success rate with mobility.

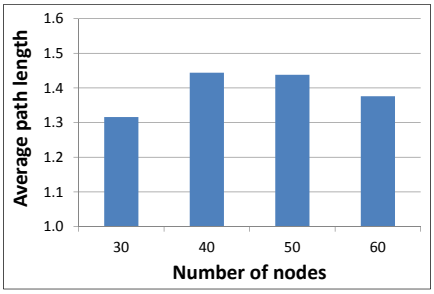

Fig. 6. Average path length with mobility.

making the LORD program on some nodes fail to respond or even crash; and 2) the communication range is large (250m). Such results reveal that the load in the network also affects the routing performance in a small network. For the average path length, since we set a relatively large communication range, most nodes can be reached within 2 hops, the average path length of successful queries remains stable when the number of nodes increases.

2) With Node Mobility: Figure 5 and Figure 6 show the success rate and average path length of LORD when the number of nodes increases in the scenario with node mobility, respectively. We find that the trends on the success rate and average path match those in the test without node mobility, shown in Figure 3 and Figure 4. The reasons are the same. However, comparing these results, we see that the overall success rate is lower and the overall average path length is higher in the test with node mobility. This is because node mobility may cause some forwarding opportunities to disappear during the test and some queries were dropped. Combining above results, we see that LORD is an effective routing algorithm for MANETs.

VI. Conclusion

In this paper, we introduced our procedure to test a MANET routing algorithm, namely LORD, in the GENI ORBIT testbed. The ORBIT testbed originally has a fixed grid consisting of 400 nodes equipped with 802.11 wireless cards. We designed node mobility and communication range over the ORBIT to simulate MANETs scenario. We developed a series of scripts to enhance the simplicity and effectiveness of our procedures. Experiments on the ORBIT testbed demonstrate the feasibility of our methods to simulate node mobility and the high efficiency of the LORD. In the future, we plan to design a method to control the distribution of nodes in the ORBIT grid and enhance the stability of wireless communications in ORBIT.

Acknowledgements

This research was supported by U.S. NSF grants CNS-0944089.

References

[1] "AODV IETF draft v1.3." http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt.
[2] "DSR IETF draft v1.0." http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt.
[3] Z. Li and H. Shen, "A mobility and congestion resilient data management system for mobile distributed networks," in *Proc. of MASS*, 2009.
[4] S. M. Das, H. Pucha, and Y. C. Hu, "Performance comparison of scalable location services for geographic ad hoc routing," in *Proc. of INFOCOM*, 2005.
[5] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, and L. Yin, "Data-centric storage in sensornets with ght, a geographic hash table," *Mobile Networks and Applications*, 2003.
[6] "The One Simulator," http://www.netlab.tkk.fi/tutkimus/dtn/theone/.
[7] "NS-2," http://www.isi.edu/nsnam/ns/.
[8] "MobiREAL," http://www.mobireal.net/.
[9] "GloMoSim," http://pcl.cs.ucla.edu/projects/glomosim/.
[10] "GENI project," http://www.geni.net/.
[11] "Orbit," http://www.orbit-lab.org/.
[12] M. Kropff, T. Krop, M. Hollick, P. S. Mogre, and R. Steinmetz, "A survey on realworld and emulation testbeds for mobile ad hoc networks," in *Proc. of TRIDENTCOM*, 2006.
[13] L. Li and H. Zhang, "Research on designing and implementing an experimental manet testbed," in *Proc. of ICCSN*, 2009.
[14] M. Hiyama, M. Ikeda, L. Barolli, E. Kulla, F. Xhafa, and A. Durresi, "Experimental evaluation of a manet testbed in indoor stairs scenarios," in *Proc. of BWCCA*, 2010.
[15] P. Mahadevan, A. Rodriguez, D. Becker, and A. Vahdat, "Mobinet: a scalable emulation infrastructure for ad hoc and wireless networks." *Mobile Computing and Communications Review*, vol. 10, no. 2, pp. 26–37, 2006.
[16] D. Herrscher, S. Maier, and K. Rothermel, "Distributed emulation of shared media networks," in *Proc. of SPECTS*, 2003.
[17] C. Liu, R. Correa, X. Li, P. Basu, B. T. Loo, and Y. Mao, "Declarative policy-based adaptive manet routing." in *Proc. of ICNP*, 2009.
[18] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. G. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols." in *Proc. of MOBICOM*, 1998.
[19] "Wireless configuration scripts," http://people.clemson.edu/kxu/publications/ORBIT_conf.tar.
[20] "Wireless driver," http://linuxwireless.org/en/users/Drivers/ath5k. [Accessd in Feb. 2012].
[21] "Hostpad," http://w1.fi/hostapd/. [Accessd in Feb. 2012].