A Portable Real-time Emulator for Testing Multi-Radio MANETs

Weirong Jiang[1] and Chao Zhang[2]

[1]*Tsinghua University*
Research Institute of Information Technology
Beijing 100084, China
jwr2000@mails.tsinghua.edu.cn

[2]*Tsinghua-TCB Institute of*
Applied Communication Systems
Beijing 100084, China
chao.zhch@gmail.com

Abstract

In building a real-life mobile ad-hoc network (MANET), network emulation has been appraised as an efficient approach for testing the real implementations of routing algorithms and protocol stacks. Most existing MANET emulators can hardly support both real-time scene construction for proof-of-concept test and real-time traffic recording for performance evaluation simultaneously. They also lack the ability to emulate the multi-radio environment. This paper presents a flexible TCP/IP-based real-time MANET emulator that can be portably deployed to facilitate the development of real multi-radio MANET routing protocols. It friendly provides visual interaction of topology control and rich configuration of emulation conditions to enable a real-time and comprehensive examination of protocol implementations.

1. Introduction

With a plethora of novel techniques such as multi-radio introduced into developing real-life mobile ad-hoc network (MANET) systems, comprehensive test and evaluation of MANET protocols are necessary for their success in real world use. Existing approaches include field testbed, network simulation and emulation. They are compared from the view of TCP/IP protocol stack as shown in Figure 1.

1) Field Testbed. Tightly coupled to the physical circumstances, field testbed is neither scalable nor reproducible, though it is theoretically the best test environment for its realism. The deployment is also cumbersome and expensive.

2) Network Simulation. Widely used in MANET research, network simulators [1,2,4] can easily produce a controllable and repeatable simulation environment. But protocol and algorithm modules in these simulators are computer-modeled rather than real implementations. Designers have to implement the protocols twice, once for simulation, and again for deployment.

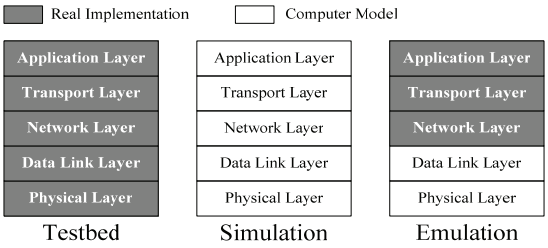

Figure 1. MANET Protocol Test Approaches.

3) Network Emulation. As a combination of network simulation and testbed, network emulation can offer a fully controlled and reproducible network environment while retaining some realism. The key advantage of emulation is that the implementations of protocols and services will be tested and evaluated without any conversion and modification. Its cost-effectiveness and scalability also bring itself much praise.

During our efforts to develop a routing protocol for multi-radio MANETs, the test and evaluation process is divided into two main phases:

Phase 1: Proof-of-concept test for debugging;

Phase 2: Performance evaluation for optimization.

The two phases pose different real-time requirements for MANET emulators, i.e. real-time scene construction for proof-of-concept test and real-time traffic recording for performance evaluation. According to the discussions in the following section, how to simultaneously satisfy the both requirements comes to be a challenge for designing a MANET emulator, especially to emulate a highly dynamic multi-radio environment.

The multi-radio multi-channel technique [12] has recently been considered as a promising avenue for improving the capacity of MANETs. When applying it to the MANET routing protocol design, there is a need

to develop an emulator able to emulate the multi-radio environment.

Furthermore, to gain a quick and straightforward insight in the behavior of a developed routing protocol, a GUI-based emulator that can replay the scenario after emulation and can be portably deployed without excessive efforts to modify the operating system kernel or the hardware driver code will be preferred.

In this paper, such a visual and flexible TCP/IP-based real-time emulator for testing multi-radio MANETs is proposed, named **PoEm**. It supports:

- ✓ **Real-time scene construction** by friendly visual interaction of topology control and circumstance configuration in the central server;
- ✓ **Real-time traffic recording** by parallel time-stamping in the peripheral clients;
- ✓ **Multi-radio environment** emulation by channel-ID indexed neighbor tables.

The rest of this paper is organized as follows. Section 2 reviews the existing MANET emulators and their real-time performance. Section 3 introduces the overall architecture of PoEm. Then some key issues are discussed in Section 4. Section 5 presents the preliminary implementation of PoEm and Section 6 conducts two sample experiments to simply validate the emulator. In the end we conclude the paper and outline the future work in Section 7.

2. MANET Emulator

According to their architectures, existing MANET emulators can be roughly classified into two categories: centralized and distributed emulators.

2.1. Centralized Emulator

In the centralized emulators such as JEmu [7] and Seawind [9], all workstations acting as mobile nodes connect to a central emulation server. All traffic is directed via that server, which forwards the traffic to the destinations following the current network scene being emulated, i.e. topology, link quality, collisions etc.

Centralized emulators provide high degree of detail in the emulation of irregular mobility and volatile circumstance since the central server offers plentiful convenience to set arbitrary scenes in real time during the test period. But processing all traffic by a central server without any assistance from clients represents a bottleneck so that recording the traffic by one server in real time will be bounded by the server processing power (e.g. NIC capacity, CPU speed, etc.). On the other hand, the performance statistics is based on the time-stamping on each packet and the non-real-time traffic recording may result in an inaccurate evaluation.

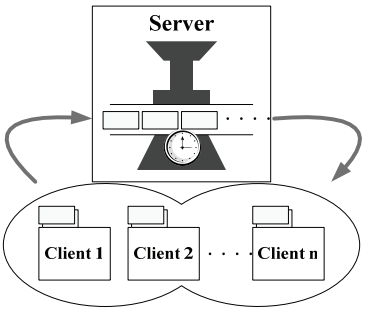

Figure 2. Centralized Emulator.

The fundamental reason for centralized emulators failing to obtain real-time traffic recording is the contention for the unique source of the incoming interface in the central server, as Figure 2 shows. Several emulation clients generate packets simultaneously but in the view of the server these packets are sent at different time due to the serial reception and subsequent processing. The error may be prevented by utilizing multiple interfaces to time-stamp the packets in parallel, which however will bring a negative effect on the system scalability and implementation cost.

2.2. Distributed Emulator

As for the distributed emulation systems such as MobiEmu [8], EMWIN[10] and MASSIVE [3], each station acting as a mobile node is responsible for directing and forwarding traffic in a peer-to-peer manner or via a fast switch. A central control instance governs the overall network topology and regulates the configuration of each mobile node by broadcasting scene messages, e.g. setting some node's current neighbors, lowering some link's bandwidth, etc.

Nevertheless, for keeping a consistent view of the global scene which is constructed in real time, these emulators need a presumption that, either each station has comparable processing power to receive the broadcasted scene messages and update its local information in the same step, or the emulated scene varies little to produce few broadcasting messages to relieve the strict requirement for homogeneity of stations' performance. But this presumption tends to be unreal for a scalable emulator consisting of diverse ends to emulate a large-scale multi-radio MANET with irregular high mobility and volatile circumstance, e.g. switching the channel, changing the radio range, moving out some nodes and lowering link bandwidth (to emulate a military attack), etc. at any time, which probably arouse a broadcast storm of scene messages.

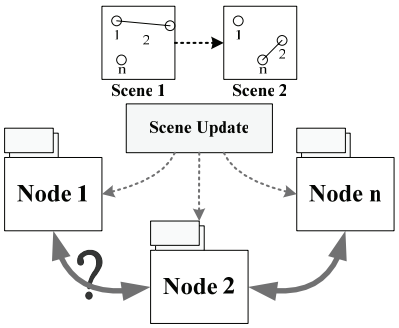

Figure 3. Distributed Emulator.

In fact, the conflict between numerous update messages and capacity heterogeneity of distributed stations will result in the asynchronism of the scene update when the distributed emulator attempts to emulate a highly dynamic multi-radio MANET under real-time scene construction. As Figure 3 shows, if the global scene updates inconsistently, real-time scene construction may confuse some emulation nodes to direct their traffic following the expired scene.

2.3. Summary and Motivation

Real-time scene construction prefers a centralized architecture to guarantee a consistent scene anytime for each emulation node while real-time traffic recording favors a distributed architecture to time-stamp each packet in parallel. Existing MANET emulators do not satisfy the both two requirements due to their rigid architecture.

Towards addressing the problem, we are developing a **Po**rtable real-time **Em**ulator for testing multi-radio MANETs, called **PoEm**, which combines the parallel time-stamping into the centralized architecture.

Moreover, to our best knowledge, few emulators support the multi-radio emulation and post-emulation replay. PoEm is among them.

Table 1 compares PoEm with JEmu as a typical centralized emulator and with MobiEmu as a distributed one.

Following sections are to give a further discussion over PoEm.

Table 1. Feature Comparison.

Emulator	Real-time scene construction	Real-time traffic recording	Multi-radio environment	Post-emulation Replay
PoEm	√	√	√	√
JEmu	√			
MobiEmu		√		

3. PoEm Architecture

PoEm adopts centralized architecture as the main frame for its real-time scene construction and portable deployment. It covers following features:

- ✓ To test and evaluate real implemented protocols without any modification.
- ✓ Real-time scene construction.
- ✓ Real-time traffic recording.
- ✓ Emulating multi-radio environment.
- ✓ Post-emulation replay.
- ✓ Running independent of system platforms.
- ✓ Scalable in the number of emulated nodes.
- ✓ User-friendly for visualization and interaction.

3.1. Overview

PoEm is a software environment for testing and evaluating real MANET routing protocols. It operates in a client/server structure and runs on several workstations. Both the server software and the client software can run on any hardware platform since they are connected through TCP/IP connections independent of low layers. Clients connect to the server and each client is mapped into a Virtual MANET Node (VMN) in the server to build an emulated MANET. It is scalable that several clients can run in one workstation to emulate multiple MANET nodes. Figure 4 illustrates the overall structure of PoEm and following sections will discuss the emulation server and client separately.

3.2. Emulation Server

To mimic a real-life MANET environment, PoEm emulation server accepts connections from emulation clients and forwards the packets to their corresponding clients according to the emulated network scene. The emulation server creates the desired network scene by controlling the topology and configuring the wireless circumstance parameters based on a variety of models.

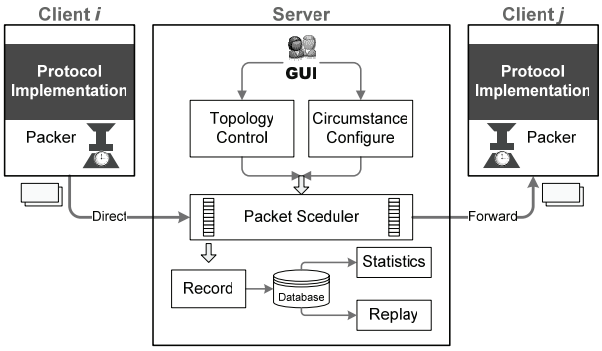

Figure 4. Overall Structure of PoEm.

Users can do those operations on the GUI in real time to set an arbitrary scene for tests, e.g. dragging and dropping VMNs anywhere, double-clicking the VMN to activate configuration dialogue-boxes anytime, etc.

For each incoming packet, PoEm server operates in parallel **multiple threads** through following steps.

Step1. Receives the packet from an emulation client.

Step2. One scheduling thread searches the channel-ID indexed neighbor table to find the destinations where the packet should be forwarded.

Step3. After judging whether to drop the packet or not, from the receipt time **that is stamped by clients**, the scheduling thread calculates when to forward the packet according to the link model which will be discussed further in Section 4.

$$t_{forward} = t_{receipt} + delay + \frac{packet\_size}{bandwidth}.$$

Step4. The thread lists the packet into the schedule.

Step5. One scanning thread keeps watching the schedule and initiates a sending thread once the emulation clock meets the time to forward.

Step6. The sending thread sends out the packet via the corresponding connection.

Step7. One recording thread collects the complete information of every incoming/outgoing packet to the database for later statistics and replay.

Another recording thread gathers the detailed information of the varying scene for post-emulation replay.

3.3. Emulation Client

Developed routing protocols are embedded in the clients. All traffic originated from protocol implementations will be packed, **time-stamped** and then directed to the server via TCP/IP connections.

During emulation, all clients connect to the PoEm server and synchronize the local emulation clock with the server clock. The lightweight synchronization process is to be discussed in Section 4.1. Each client sends to the server the traffic generated by user interaction for test; and receives from its neighboring VMNs the traffic forwarded by the server.

On the GUI of emulation clients, users input the test set and configuration commands, observe the output results and information, and make judgments whether the developed protocol works correctly.

4. Key Issues

To implement the full function of PoEm as a GUI-based real-time emulator for testing multi-radio MANETs, several key issues should be figured out.

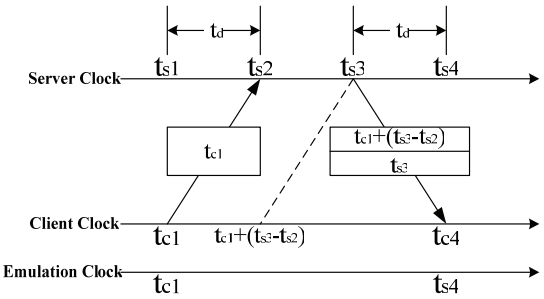

Figure 5. Emulation Clock Synchronization.

4.1 Emulation Clock Synchronization

Parallel time-stamping makes real-time traffic recording feasible. But its final success relies on the emulation clock synchronization between clients and the central server.

To accomplish it, we adopts the server clock as the unique reference for emulation clock and designs a lightweight synchronization scheme as following steps, also shown in Figure 5.

Step1. The client connects to the server and sends a message recording its local clock time t_{c1}.

Step2. The server accepts the connection and receives the synchronization message at its time t_{s2}.

Step3. At the server time t_{s3}, the server sends back to the client a message recording the time t_{s3} and the time $(t_{c1}+t_{s3}-t_{s2})$.

Step4. The client receives the reply from the server at its local time t_{c4}.

Step5. Assuming that the transport delay t_d from the client to the server is equal to that in reverse, the client calculates $t_d = 0.5[t_{c4}-(t_{c1}+t_{s3}-t_{s2})]$ and estimates the current server clock as $t_{s4} = t_{s3} + t_d$.

Step6. The client uses t_{s4} as the current emulation time and pushes the emulation clock forward.

Each client synchronizes its emulation clock with the server clock when initializing the connection. How to set the synchronization frequency is determined by the user in consideration of the emulation duration, client homogeneity and real-time requirements.

4.2 Multi-Radio Emulation

In multi-radio environment, each MANET node has multiple radios to assign multiple channels to adjust neighbor connectivity with other nodes. That is, the neighborhood lies on not only the radio range but also the channel assignment. It increases the topology complexity and affects the efficiency to update the neighbor tables.

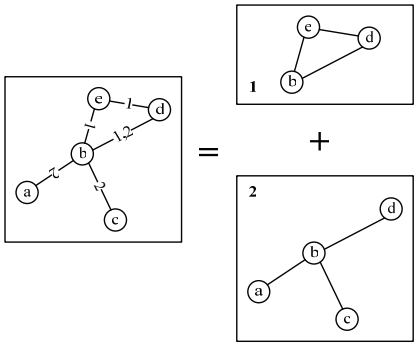

Figure 6. Channel-ID Indexed Neighbor Table.

PoEm server maintains multiple neighbor tables indexed by channel-IDs to emulate the multi-radio environment. Following expressions are defined to give a neighborhood model correspondingly.

NS(n): Node set indexed by channel *n*.

CS(A): Channel set of node *A*.

NT(A,n): Neighbor table of node *A* via channel *n*.

R(A,n): The radio range of *A* on channel *n*.

D(A,B): The distance between *A* and *B*.

Then, for channel *k*

$$\left.\begin{array}{c} k \in CS(A) \cap k \in CS(B) \Rightarrow A, B \in NS(k) \\ D(A,B) < R(A,k) \end{array}\right\} \Rightarrow B \in NT(A,k)$$

In contrast to the scheme that keeps one unique neighbor table with multiple channel-ID marked units, our scheme reduces the cost to update the neighbor table when the emulation scene has changed. As Figure 6 shows, taking *node* ***a*** as an example, unless it switches one of its radios to channel *1*, any change of *node* ***a*** won't cause the update between it and the nodes in the neighbor table indexed by channel *1* since its radio is on channel *2*. This scheme improves the update efficiency and relieves the server processor of heavy load especially when emulating dynamic large-scale multi-radio MANETs.

4.3 Configurable Models

As a user-friendly and interactive emulator, PoEm makes some revisions to the underlying models to ease the parameter configuration on the GUI.

4.3.1. Mobility Model. The VMN mobility model is generalized as a 4-tuple: <*pause_time, direction, move_speed, move_time*>, which is easy to be configured on the GUI. By setting their types {constant or random} and values {constant or variation range}, this mobility model can practically diverge to different 2-D entity mobility models [11]. For instance, this model turns to the Random Walk Mobility Model when:

$$\begin{cases} pause\_time = 0 \\ direction = rand[0, 360°] \\ move\_speed = rand[minspeed, maxspeed] \\ move\_time = time\_step \end{cases}$$

The formulas of the mobility model in PoEm can be presented as follows:

$$\begin{cases} x(t + \Delta t_{\{pause,move\}}) = x(t) + v(t)\Delta t_{move} \cos\theta(t) \\ y(t + \Delta t_{\{pause,move\}}) = y(t) + v(t)\Delta t_{move} \sin\theta(t) \end{cases}$$

4.3.2. Link Model. A link can be normally modeled as three parameters: *packet loss*, *bandwidth*, and *delay* [5]. PoEm adopts the packet loss model derived from [6]:

$$P_{packet\_loss} = \begin{cases} P_0, r \le D_0 \\ K_p (r - D_0)^{\beta} + P_0, r > D_0 \end{cases}$$

where $K_p = \dfrac{P_1 - P_0}{(R - D_0)^{\beta}}$ and r indicates the distance away from the source VMN. This model turns to the constant model once $P_1 = P_0$.

Distinct from the discrete expression in [5], the bandwidth is modeled as a Gaussian distribution: $Bandwidth = M \exp(-K_b r^2)$ where $K_b = \dfrac{\ln M - \ln m}{R^2}$. It turns to the constant model when $m = M$.

The model parameters $P_1, P_0, D_0, R, \beta, M, m$ are all configurable on the GUI.

5. Preliminary Implementation

We've implemented the preliminary version of PoEm on Windows OS with Microsoft Visual C++ 6.0. The central server and several client processes run as general applications on several workstations wired with a fast Ethernet LAN in a lab setting, as shown in Figure 7. They are connected via TCP sockets. All the records about packets and topology are logged into a SQL database on server through ODBC. Underlying data structures for server functioning are crosslinks for neighbor tables and queues for schedules.

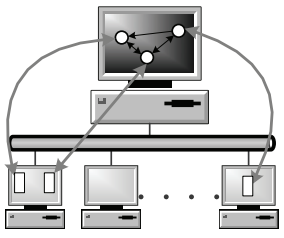

Figure 7. Hardware Platform of PoEm.

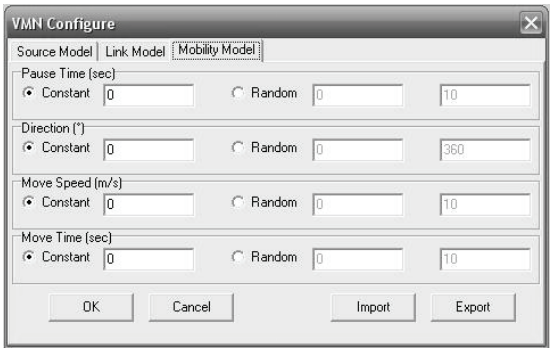

Figure 8. VMN Configuration.

Users configure VMN properties on GUI in real time to set an arbitrary scenario for testing, through double-clicking the VMN to do configurations anytime, shown as a multi-tab dialog box in Figure 8.

6. Sample Experiments

To validate the preliminary implementation of PoEm, we conducted two simple experiments to verify its ability for real-time scene construction and real-time traffic recording.

6.1 Proof-of-Concept Test

The emulated network is constructed as Figure 9 shows to test a hybrid MANET routing protocol developed by our group, which is combining the periodic-broadcasting and on-demand mechanisms to achieve high robustness for military applications.

Doing following operations, we inspect the routing table in VMN1 in real time as a part of the test. The results are listed in Table 2.

The results in the accordance with the expected definitely demonstrate the effectiveness of PoEm as a MANET emulator for proof-of-concept tests under real-time scene construction.

Table 2. Test Results

Operation	Routing Table in VMN1
Step1. Construct the network scene shown in Figure 8.	# of Routing Entries: 2 254.0.0.1-->254.0.0.2 254.0.0.1-->254.0.0.3
Step2. Shrink the radio range of VMN1 to exclude VMN3.	# of Routing Entries: 2 254.0.0.1-->254.0.0.2 254.0.0.1-->254.0.0.2-- >254.0.0.3
Step3. Set different channels for the radios on VMN1 and VMN2.	# of Routing Entries: 0

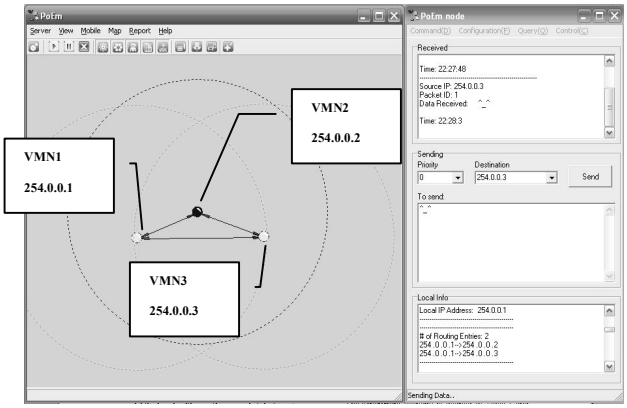

Figure 9. Emulated MANET.

6.2 Performance Evaluation

The experiment scenario is defined as Figure 10 and Table 3 describe. VMN1 with a radio on channel 1 sends CBR traffic of 4Mbps to VMN3 with a radio on channel 2 while VMN2 with two radios on channel 1 and 2 moves at the speed of 10 (unit)/s downwards. The receiver VMN3 is outside the radio range of the sender VMN1 and hence VMN2 plays a role of relayer inside. The traffic load of 4 Mbps is actually heavy in real-life large-scope MANETs, especially for most military use.

To compare the experimental performance with the expected, the packet loss rate is adopted as the metric for comparison due to its easy computation in theory. The packet loss in the test is purely caused by the link model settings since the two channels are assigned diverse channel IDs to avoid any collision.

According to the theoretical models, we drew both the expected real-time and non-real-time performance curves in advance.

The result shown in Figure 11 proves that PoEm is an effective real-time MANET emulator for the performance evaluation with sufficient-granularity. The minor error between the experimental and the expected real-time performance is analyzed as the result of the drift of random number generator and the overload of server computation.

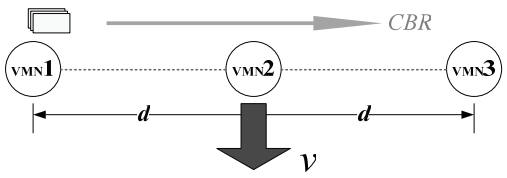

Figure 10. Experiment Scenario.

Table 3. Experiment Parameters

Parameter		Value
hop distance d		120 (unit)
radio range R		200 (unit)
CBR		4Mbps
moving speed v		10 (unit)/s
moving direction		90°
channel IDs	VMN 1-2	1
	VMN 2-3	2
packet loss model	P_0	0.1
	P_1	0.9
	D_0	50 (unit)
	β	2

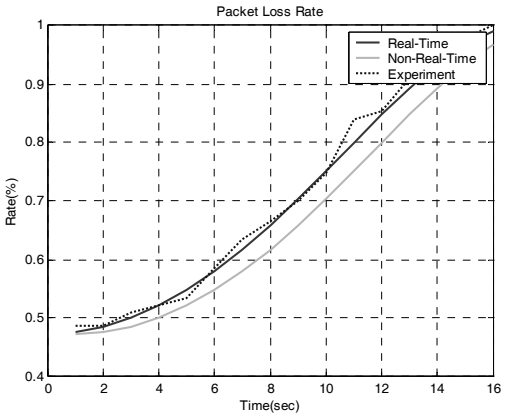

Figure 11. Experiment Result.

7. Conclusions and Future Work

This paper has presented a visual and flexible TCP/IP-based real-time emulator for testing multi-radio MANETs, named **PoEm**, which supports both real-time scene construction and real-time traffic recording by combining the parallel time-stamping into the central architecture. PoEm can facilitate the development of real multi-radio MANET routing protocols through portable deployment. It provides user-friendly interaction of topology control and rich configuration of emulation conditions to enable a real-time and comprehensive examination of protocol implementations.

Our future work is to expand the one server to a parallelized cluster to conquer the performance bottleneck so as to support fine-granularity performance evaluations driven by scenario scripts. Sophisticated underlying models such as power consumption, MAC algorithms and group mobility also need be added into our system to provide more precise examinations.

8. Acknowledgements

We would like to thank the colleagues in TCB for their inspirational discussions. We would also like to thank Prof. Jun Li for his helpful comments to improve the presentation of this paper.

References

[1] ns-2, http://www.isi.edu/nsnam/ns/
[2] QualNet, http://www.qualnet.com
[3] M. Matthes, H. Biehl, M. Lauer, and O. Drobnik, "MASSIVE: An Emulation Environment for Mobile Ad-hoc Networks," Proc. of the 2nd Annual Conference on Wireless On-demand Network Systems and Services, 2005.
[4] OPNET, http://www.opnet.com
[5] D. Herrscher, A. Leonhardi, and K. Rothermel, "Modeling Computer Networks for Emulation," Proc. of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications, 2002.
[6] W. Liu and H. Song, "Research and Implementation of Mobile Ad Hoc Emulation System," Proc. of the 22nd International Conference on Distributed Computing Systems Workshops, 2002.
[7] J. Flynn, H. Tewari, and D. O'Mahony, "JEmu: A Real Time Emulation System for Mobile Ad Hoc Networks," Proc. of the 1st Joint IEI/IEE Symposium on Telecommunications Systems Research, 2001.
[8] Y. Zhang and W. Li, "An Integrated Environment for Testing Mobile Ad-Hoc Networks," Proc. of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002.
[9] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen, "Seawind: a Wireless Network Emulator," Proc. of 11th GI/ITG Conference on Measuring, Modeling and Evaluation of Computer and Communication Systems, 2001.
[10] P. Zheng and L. M. Ni, "EMWIN: Emulating a Mobile Wireless Network using a Wired Network," Proc. of the Fifth International Workshop on Wireless Mobile Multimedia, 2002.
[11] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," Wireless Communications & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5):483–502, 2002.
[12] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," Proc. IEEE Infocom 2005.