JAIST Repository
https://dspace.jaist.ac.jp/

Title	On Testbed Environment for Evaluating Security in Mobile Ad hoc Network
Author(s)	NGUYEN, TRAN TRUNG
Citation	
Issue Date	2010-09
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/9147
Rights	
Description	Supervisor:Professor Yoichi Shinoda, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

On Testbed Environment for Evaluating Security in Mobile Ad hoc Network

By Nguyen Tran Trung

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Yoichi Shinoda

September, 2010

On Testbed Environment for Evaluating Security in Mobile Ad hoc Network

By Nguyen Tran Trung (0810203)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Yoichi Shinoda

and approved by
Professor Yoichi Shinoda
Research Associate Professor Ken-ichi Chinen
Associate Professor Azman Osman Lim

August, 2010 (Submitted)

Copyright © 2010 by Nguyen Tran Trung

Abstract

Doing research on mobile ad hoc network (MANET) security involves several constraints, some of which are wireless environment's attributes, such as bandwidth and delay, and the correctness of network topology. Besides, since testing and evaluating using real wireless devices are costly and somehow impractical; many tools have been developed in order to create media by which researchers can deeply investigate different aspects of network security. On software perspective, simulators recently have gained much attention because they provide very detailed low-level models to network behaviors; however, they lack of some physical wireless characteristics.

On the other hand, hardware view, emulators emerges as alternative to solve the shortcomings posed by simulators by providing a virtual wireless network at low level, and to support executable real code pieces that can be executed. In this thesis, we propose a testbed supporting a highly realistic wireless environment for evaluation of MANET security. One of the features which best distinguishes our testbed from the others is that it provides ability to emulate a MANET from layer 2 up while others cannot. Furthermore, our testbed is flexible configurability and deployment. At last, the quality of our testbed is also shown in its ability to offer readiness in deploying various kinds of attack methods thus easier for us to evaluate jobs done.

Keyword: ad hoc wireless network, network emulator, ad hoc network security, testbed

Contents

List of Figures

List of Tables

Chapter 1.
Introduction

1.1 Mobile Ad hoc Network

1.1.1 MANET Overview

A mobile ad-hoc network (MANET) is a non-infrastructure network that consists of self-configuring mobile nodes connected by wireless links. In this network, mobile nodes are free to move in an open environment. Two nodes can communicate directly as long as they are within the radio communication range of each other, but when the two nodes are far apart, they require the help of other intermediate nodes to relay their traffic.

Wireless ad-hoc networks can be classified into two kinds: single hop and multi-hop networks. Single hop means no intermediate nodes between the source and the destination such as in Bluetooth, or Infrared communication. While we call a multi hop network if the source bases on the other nodes to transmit his packets to the destination the network. The characteristics of MANET are defined as noted from [24]:

- **Autonomous**: Each node in a MANET is autonomous and works as both router and host.
- **Distribution**: A MANET is distributed in its operation and functionalities, such as routing, host configuration and security. For instance, unlike wired network, MANET cannot have a centralized firewall.
- **Multi-hop**: Since a node wants to send a message to a destination node which is out of its radio range, the source node will ask one or more nodes to transmit its message to the destination.
- **Dynamic topology**: Nodes are mobile and can join or leave the network at any time; therefore, the topology is dynamic.

- **Unstable link bandwidth**: The stability, capacity and reliability of wireless links are always inferior to wired links because of wireless environment and node's mobility,
- **Power constraint**: The mobile nodes are often lightweight, with less powerful CPU, memory and power.

Originally, MANET was developed for military purposes, as nodes were scattered on a battlefield for surveillance mission. In recent years, it is not only used in commercial but also in civilian applications. For example, a group of people in a conference can use ad hoc to communicate with each other to share documents or files. Another example is in hospital where each of patients wears a sensor that can log some information such as temperature, blood pressure … and then send them to doctors.

1.1.2 Security Issues

As nodes in a MANET are capable of moving in a wide range and cooperate with other ones while these encounters are not known whether harmful, ensuring security for this type of network is becoming one of the most important issues. In a survey [2], the authors classify the security attack on MANET in two groups: passive attacks and active attacks.

- **A passive attack** obtains data exchanged in the network without disrupting the operation of communication.
- **An active attack** involves information interruption, modification or fabrication, thereby disrupting the normal functionality of MANET.

In addition, these authors also list attacks on MANET in below tables:

Layer	Attacks
Application Layer	Repudiation, data corruption
Transport Layer	Session hijacking, SYN flooding
Network Layer	Wormhole, blackhole, Byzantine, flooding, resource consumption, location disclosure attacks
Data link Layer	Traffic analysis, monitoring, disruption MAC (802.11), WEP weakness
Physical Layer	Jamming, interceptions, eavesdropping
Multi-layer attacks	DoS, impersonation, replay, man-in-the-middle

Table 1: Security Attacks on Protocol Stacks

Table 2: Cryptography Primitive Attacks

Cryptography Primitive Attacks	Examples
Pseudorandom number attack	Nonce, timestamp, initialization vector (IV)
Digital signature attack	RSA signature, ElGaml signature, digital signature standard (DSS)
Hash collision attack	SHA-0, MD4, HAVAL-128, RIPEMD
Security handshake attacks	Diffie-Hellman key exchange protocol, Needham-Schroeder protocol

1.1.3 Security Evaluation Challenges

The obstacles of MANET security studies are establishing trustworthy testing environments that strongly affects on obtained results. There are some ways to investigate security in MANET, through simulation, emulation, or by setting up a real-world testbed. Simulation, often conducted on one or a small group of computers, heavily depends on software, thus suffers all shortcomings that simulators may have. Limitations found in simulators are bugs while implementing the program, instability of the program, slowness along with scalability and many more. Real-world testbed is a key to solve these problems of simulation. This testbed supports a realistic environment, and facilities to extract the best results. In spite of its advantages, real-world testbed also has some limitations. For example, it is hard to choose appropriate hardware, to keep relevant parameters when a scenario is repeated, and to find errors and evaluate algorithms. For those reasons, sometimes simulation results are not reliable enough to rely on; moreover; real-word testbed is expensive to build, hence this gap is the place where emulation comes in. Emulation is a technique where we use a system, one or a group of devices, to imitate another system's behaviors. Emulator's advantages are reducing cost, experiment repeatability, supporting for large-scale network, and real programs. Nevertheless, recent emulators do not fully support for security evaluation, especially on low layer network security. This table below summarizes their advantages and disadvantages.

Table 3: Real-world, Simulation, and Emulation Comparison

	Simulation	Real world	Emulation
Real-time execution	NO	YES	USUALLY
Control level	HIGH	LOW	HIGH
Condition range	LARGE	SMALL	LARGE
Result realism	LOW	HIGH	MEDIUM
Experimentation cost	LOW	HIGH	MEDIUM
Ease of use	HIGH	HIGH	MEDIUM

1.2 Motivation and Target Application Domain

Mobile ad hoc network-based applications will become important services in near future. Therefore, studies on MANET have to ensure these applications are not only reliable but also secured. As above discussion, existing testbed environments still have some limitation on supporting for security evaluation. We realized that in order to get prominent and robust results in emulations, some of the requirements as follows should be guaranteed:

- A good testbed has to be able to provide architecture for reproducible experiments.
- A testbed should provide a realistic environment that helps us get precise and reliable results as we may have if is deployed in a real environment.
- A testbed should be flexible and powerful enough to integrate different characteristics of emulated nodes such as mobility and power energy.

Because of these requirements and the necessary of an emulator for security MANET evaluation, we have developed an emulator, which provides a real-world-liked MANET environment for researchers who need to prove their research on practical way.

1.3 Purpose of This Thesis

We noticed that some existed emulators are not suitable for mobile ad hoc network security research. This is due to limitation in their approaches that just emulate some characteristic of this network, or are not flexible to extend in large-scale network.

The requirements motivate us to propose a testbed called eBATMAN (emulator-BAsed Testbed for studying Mobile Ad-hoc Network security) that provides not only a highly realistic environment, but also means to study security in MANETs. Our testbed has five major components: an emulator, an experiment controller, an action executor, a report collector, and a graphical user interface (GUI) application. These components constitute a highly realistic environment that supports various wireless communications specifications (IEEE 802.11a/b/g) overlaid on a wired network. Deployments on this testbed are facilitated with automatically generated attacking scenarios, and researchers can execute real security tools or trigger security events to make experiments. Furthermore, our testbed is also promising in providing an emulator with scalability.

In this thesis, at first we do a survey on MANET implementations and MANET security. In the second chapter, we introduce our testbed architecture in detail. The next is evaluating testbed performance, making some attack scenarios and doing some assessment. At the end of this thesis, we would like to give instructions to deploy, compile, and use the testbed in system.

Chapter 2.
Mobile Ad hoc Network: Security and Implementation

2.1 MANET Vulnerabilities

2.1.1 Wire network-inherited vulnerabilities

Generally, wireless ad hoc network are developed from wired-network, so most of protocols and standards on wireless are inherited from wired. As a result, wireless devices continue facing wired attacking methods such buffer overflows, session hijacking, TCP-SYN flood, man in the middle (layer 4), repudiation, data corruption and virus and worm attacks.

2.1.2 Its own vulnerabilities

In existence studies, security threats in MANET are deeply researched on theory; however some of them have become vulnerable in real environment. In this session, I only mention on some types of attacking methods that are tested in testbeds or real world by introducing what they are, and how they are exploited by using tools or emulating in testbeds.

a. Physical attacks

Mobile nodes in MANET share the wireless media with other nodes, so naturally they broadcast information into environment. Hence, attacker easily intercepts and reads the messages and conversations by using monitoring-mode receiver. Moreover, attacker can inject faked messages into the network. Another physical attack method is that attacker uses a powerful transmitter that is able to create such strong signal that the target's communication is disrupted. This attack method is called jamming attack, and the equipments are available.

Some packet constructors are such **commview for wifi** [21], **libnet** [7] , and **hping2** [13] used to inject packet. And, jamming attack is clearly evaluated in [29]. In this test, the attack is called the smart jamming attack in which a wireless network adapter is used as a jammer. Attacker nodes use high transmission power to make a busy medium, so other nodes are not able to transmit data.

b. Spoofing MAC and IP address

This attack is simple, yet hard to be detected. After collecting MAC and IP addresses broadcasted on the network, attacker will mimic the victim.

To change IP address, it simplifies to configure IP address in Windows or Linux by using ipconfig or ifconfig command. Besides, using tools such as **macchanger** [12], **smac** [14], or configuring in Windows register.

c. Dropping packet attack (blackhole attack)

Black hole attack is one in which attacker exploits the mobile ad hoc routing protocol such as AODV or OLSR. Firstly, he advertises neighbors that he is handling the valid shortest route to a destination node. However, he may not take this route. Secondly, when packets routed through him, he can drop all the packets or selectively drop some.

This type of attack is tested in detail in [29], and [5]. In [36], the authors add some functions for the testbed to implement this attack such check_drop, drop_q, drop_packet.

d. Modification attack

In sort of attack, attacker tries to forge the information in a packet such as header, or data in order to poison routing cache of other nodes or DNS cache, disturb route discovery and cause nodes to misroute packets. Seriously, attacker can modify node's certificate to become his own one.

There are a number of testing emulations for this attack in testbed such [29] in which the authors have shown the effect of the attack in the network. And additional, the authors in [36] help attacker by creating a tool to modify the header information. Last hop external flag is changed in the route reply option to make this route less interesting for the initiator of the route discovery. Or, attacker node removes itself from the route reply option. Another way is removing the next hop in route cache of source node in route error option part. In real world, the tools are such Cain&Abel [18], Winsock Packet Editor, paros proxy…

e. Fabrication attack

In this attack, attacker sends forged packets such as ROUTE ERROR packets, spoofed ROUTE REQUEST packets and ROUTE REPLY packets to victims in order to poison routing cache, remove the routes in routing table, consume bandwidth and energy, or cause congestion. In other cases, attack could forge ARP, DNS or DHCP cache of victim.

The attacking method on routing protocol was tested in [36]. ARP, DNS and DHCP attacks are well known in wired network security vulnerability, thus tools for these attacks are easy to find such Cain&Abel, arppoison, dsniff …

f. Wormhole attack

In this attack, attacker tries to record packets at one location in the network, tunnel them to another location by many ways like transferring packets in wired network, and then throws them again to victims. As the result, routing path can be disrupted when routing control messages are tunnels. Wormhole attack is usually used to against on-demand routing protocols such as DSR or AODV.

This attack is emulated in [6] and [3]. The authors of [30] developed a tool for in-band wormhole attack against OLSR protocol. This tool creates a tunnel to forward the OLSR messages by using vtun utility. Packets are captured by libpcap, and then they are injected into the tunnel to wormhole endpoint. This test is emulated on mobility environment on 19 IBM laptops.

g. Weakness of 802.11 WEP

WEP (wired equivalent privacy) is one of method to increase the privacy in IEEE 802.11 by encrypting radio signals. In additional, another purpose of WEP is supporting authority for accessing WLAN 802.11 standard with WEP cryptographic keys of 40 bits, 104 bits, or 128 bits. However, it is well known that WEP has a number of vulnerabilities and is easily to attack.

Some weaknesses are listed below:

- WEP protocol does not specify key management
- The initialization vector (IV) is just 24-bit field and sent in clear. The reuse of IV leads to analytic attacks.
- The combined use of a non-cryptographic integrity, CRC 32, with the stream cipher is a security risk.

The method attack is studied in [28] such the FMS attack, the KoreK attack, PTW attack, and the chopchop attack. The best tool for attacking WEP is aircrack-ng [11].

h. Location disclosure attack

Location disclosure attack is the quite same with foot printing method in wired network. Attacker collects the information about the nodes and network structure. Some interesting information is such route map, MAC address, location information and etc, which are necessary for planning further attack scenarios**.**

There are a number of tools to collect the information about existing nodes and route paths such netstumber, Omnipeek, tracert, airopeek,…

i. Denial of Service

Denial of service attack contains many methods of attack that aims to deplete resource of victim such as energy, CPU, memory, provided services, and transmitting signal. This attack can be launched from various layers. In the physical layer, attacker can employ signal jamming as mentioned above. In the network layer, routing process can be interrupted through routing control packet modification, dropping packet, table overflows, land attack, fragmentation attack, and so on. At transport and application layers, SYN flooding, session hijacking and malicious program such as virus, worms are well known recently.

In the paper [5], the authors tested DOS by flooding HELLO messages to neighbors. There are many developed tools for DOS attack such as MDK3, tcpsyn, httpflood, …

2.2 Security Challenges

A central vulnerability of MANET comes from Peer-to-Peer architecture in which each node acts like a router to forward packets to other nodes. Moreover, these nodes on network share the same opened environment that gives opportunity for malicious attackers. As the results, we cannot apply a single security solution to prevent from attacking. In [35] and [41], these authors explain the challenges for MANET security as follows:

- **Lacking of central points**: because of characteristics of MANET such lacking gateways, routers, etc, the mobile nodes in this network just know some neighbors in its range. As the result, this introduces new difficulties for security designs such as facing with the change of network topology, resource constraint…
- **Mobility**: MANET nodes can leave, join, and roam in the network on their own will, so the topology of network is changed frequently… Therefore, some proposed techniques security solutions such as intrusion detection system (IDS), or intrusion protection system (IDP) will be adapted with the change of topology. However, this also raises new problems for these systems.
- **Wireless link**: In wireless environment, a plenty of collision occurred when nodes send and receive the packets. The wireless channel is also subject to interferences and

errors, exhibiting volatile characteristics in terms of bandwidth and delay. In addition, some services such as routing protocols, broadcast services have to communicate with others in real-time, this can flood the network traffic.

- **Limited resources**: The mobile nodes like laptop, PDA or wireless sensor are generally constraint in battery power, processing speed, storage, and memory capacity. As the result, the operation of security solutions can be reduced the accuracy, efficiency such dropping packets, a numerous time for computation.
- **Cooperativeness**: MANET is a mobility network, so nodes have to communicate with other nodes by using some kinds of MANET routing protocol such AODV, DSR…Therefore, this can make these protocols to become a target of new attacks..

2.3 Security Layers

Due to unclear defense line in MANET, a good security solution should combine multiple defense layers to prevent completely from attackers.

a. Physical layer

Physical layer of MANET is easily susceptible to signal jamming, DOS attack, and some passive attacks than other layers. In addition, protecting solutions for this layer is too complicated because of the nature of attacks. Some authors proposed frameworks or algorithms to improve the weakness of recent ones; however, these are not effective to deploy in practice. Spectrum technology can be used to make challenge to detect and jam signal. FHSS (Frequency Hopping Spread Spectrum) make the signal jumping unpredictably among spectrums. And, DSSS (Direct Sequence Spread Spectrum) represents each data bit by multiple bits in transmitted signal. In spite of reducing capturing or jamming signal, these mechanisms are secure only when attackers cannot predict the hopping pattern or spreading code. In practice, by using some tools discussed on previous section, we can break the signal easily.

b. Link layer

Link layer security is protecting connectivity between two direct neighbors in the same signal region. Vulnerabilities on this layers presented in previous section usually come from design of this standard. But recently some security extensions to 802.11 are like WEP, WPA and WPA2. Unfortunately, these protocols are not completely suitable with MANET; thus some proposed protocols are developing such as LLSP.

c. Network layer

Network layer is more vulnerable than other layer in MANET. The weakness usually comes from routing protocols which are peer to peer routing. The first line of defense is using secured

routing protocol. To prevent modification attack, data in packets are encrypted or signed by digital signature. Some same methods provides this purpose such message authentication code (MAC), hashed MAC (HMAC), one-way hashed MAC key chain. IPSec is another mechanism to protect to packet from modification. Security routing protocol named ARAN protects to various attack like modification of sequence number, hop count, routing source, and etc. In addition, there are many papers that propose some routing protocols; however, these protocols are not proved in practice.

Wormhole attack has become famous in MANET security. Many recent papers have discussed to find a good way to detect wormhole such as using directional antenna, time synchronization…

d. Transport layer

Much vulnerability is related to TCP/IP protocol. To enhance security for this layer, some protocols have developed to protect data transmitting on the network. For example, socket secure layer (SSL), transport layer security (TLS) use public key cryptography system to hidden real data in the channel. TLS/SSL also protects from some attacks such as man in the middle, replay attack, rollback attack. However, on mobile ad hoc network, public key system also faces a challenging issue that is authenticating a public key of a node.

e. Application layer

This layer is a rich land for many malicious programs such as worm, virus, trojan and etc. Because of delicate codes, attackers can exploit these ones in order to control all of system. Therefore, first of all is making sure that our system be updated. This is import work to prevent unpredictable bugs of software and operating system. The second is hardening the network service by using permissions and access control. The next is applying some solutions to protect the system. Firewall is one of the best solutions. It provides a mechanism to defend and filter malicious packets to our system. Besides of that, firewall today is integrated a mechanism to block malicious program on computer trying to connect to Internet or other computers in the same network. Another necessary solution is antivirus, and anti spyware software which prevent our system from virus or malwares. Another mechanism is intrusion detection system (IDS) which effectively detects specific attacks gaining unauthorized access to services.

2.4 Survey on MANET Implementation

2.4.1 Real world Implementation

These are some real world implementations to demonstrate that even simple protocols do not behave as predicted in simulations. These implementations was surveyed in [27]

Table 4: Real-world Implementations

Name	Introduction
ABR at Georgia Institute of Technology	The ABR protocol was studied in a static four-node network using a chain topology. ABR belongs to the class of reactive routing protocols. It uses flooding for route discovery and beacons for route maintenance.
Radio characterization at Pervasive Computing and Networking laboratory / University of Pisa/IIT Institute Pisa, Italy	Interesting of this experiment is the measurements of the communication distance with respect to parameters such as data rate and ground height with 802.11 network interfaces. The authors determine the communication distances for their 802.11 equipped laptops at different data rates in an outdoor setting under optimal conditions.
Routing protocol evaluation at Dartmouth College/Colorado School of Mines/University of Illinois at Urbana Champaign / Bucknell University, Lewisburg	An experimental comparison of four MANET routing protocols (APRL, AODV, ODMRP and STARA) can be found. The network consisted of 40 laptops equipped with 802.11b cards running at a fixed rate of 2 Mb/s .
DSR at University of Colorado, Boulder	The MANET examined in this experiment is composed of 10 nodes out of which some are mounted on remote controlled miniature airplanes. The nodes are composed of single board computers equipped with 802.11b network interfaces and GPS, routing is performed with DSR. The authors demonstrate in this work that it is possible to combine airborne and ground nodes in a MANET.
Ad-hoc networking with directional antennas at BBN Technologies, Cambridge	This experiment is a system for ad-hoc networking with directional antennas. The implementation of this system used the same routing code for the real experiment as for the simulation. For routing, the link-state routing protocol HSLS was used. An experiment was conducted with 20 nodes (cars) that drove around a 4 · 3 km area.

2.4.2 Simulator Testbed

Besides real-world testbed, simulator is software that allows handling MANET as whole in the easily using and monitoring ways. Moreover, experimentation is described as scenario files which can be reusable and debug able. However, because of the complex nature of MANETs, simulation faces with many challenging issues such as accuracy, speed, scalability, usability, etc.

In [26], authors have a summary on MANET simulator.

Table 5: MANET Simulators

Name	Granularity	Metropolitan mobility	Parallelism	Interface	Popularity
ns-2	Finest	Support	No	C++/OTCL	88.8%
DIANEmu	Application-level	No	No	Java	< 0.1%
Glomosim	Fine	Support	SMP/beowul	Parsec(C-based)	4%
GTNets	Fine	No	SMP/beowul	C++	0.13
J-Sim	Fine	Support	RMI-based	Java	0.45
Jane	Application-level	Native	No	Java	< 0.1%
NAB	Medium	Native	No	OCaml	0.48%
OMNet++	Medium	No	MPI/PVM	C++	1.04%
OPNet	Fine	Support	Yes	C	2.61%
pdns	-	-	beowulf	C++/OTCL	< 0.1%
QualNet	Finer	Support	SMP/beowul	Parsec(C-based)	2.49%
SWANS	Medium	-	No	Java	0.3%

2.4.3 Emulator Testbed

Testbeds are vital for emulation MANET in the case that real world experiments could not be tested. In the same above paper, authors have survey many testbeds in classification as below.

Testbed	Architecture	Mobility Modeling	Wireless Medium Modeling	Virtuality	Reported Size
JEMU	centralized control	logical connectivity	on wired, centralized collision detection on frame level	1:1	12 physical
Lin et al.	centralized control	logical connectivity	on wired, centralized	1:1	4 physical

			bandwidth adaptation to IEEE 802.11		
MobiNet	centralized control	logical connectivity	on wired, centralized physical, link, and routing layer emulation	m:n (hybrid)	200 virtual
MASSIVE	distributed control	logical connectivity	on wired, no bandwidth adaptation	1:1	13 physical
MobiEmu	distributed control	logical connectivity	on wired, no bandwidth adaptation	1:1, m:n (hybrid)	50 physical
EMWIN / EMPOWER	distributed control	logical connectivity	on wired, wireless MAC emulation	m:n (hybrid)	48 virtual
NET	distributed control	logical connectivity	on wired, distributed bandwidth adaptation, distributed MAC emulation	m:n (hybrid)	64 physical, 1920 virtual

Table 6: Summary and Classification of Real World and Emulation Testbeds for MANETs

2.4.4 MANET Software

These programs support for MANET. Most of them are developed under Linux. These are referred in the website [10]

Software	Introduction	Website
ZRPd	ZRPd is a full implementation of the Zone Routing Protocol (ZRP) for Linux. It is a compromise between proactive and reactive routing protocols for ad-hoc Wi-Fi networks.	http://www.zrp.be/
Hipercom	Optimized Link State Routing is one of	http://hipercom.inria.fr/OO

Optimized Link State Routing	several protocols under investigation by the IETF for use in ad-hoc wireless networks, where not only the end users are mobile, but also the routers, services, etc	LSR/
Kernel AODV	Kernel AODV is a loadable kernel module for Linux. It implements AODV routing between computers equipped with WLAN devices.	http://w3.antd.nist.gov/wctg/aodv_kernel/
Qolyeste	Qolyeste is a C++ implementation of the OLSR protocol for mobile wireless ad hoc networks. It is meant to be enhanced with QoS features from the QOLSR research group	http://www.olsr.org/
AODV-UU	(Ad-hoc On-demand Distance Vector Routing, from Uppsala University) is a routing protocol under investigation by the IETF for use in ad-hoc networks where both end-users and routers are mobile. This implementation supports IPv6 and multicasting and is compliant with AODV Draft v.13.	http://core.it.uu.se/core/index.php/Main_Page
MIPL Mobile IPv6 for Linux	Mobile IPv6 for Linux is an implementation of Mobility support in IPv6. Mobility support allows a mobile device to be tracked as it migrates between networks or even ISPs, allowing packets to be forwarded to where the device is currently located. The software is now mostly compliant with RFC 3775.	http://tldp.org/HOWTO/Mobile-IPv6-HOWTO/index.html
Ad-hoc Wireless Distribution System	A layer 2 routing protocol for wireless mesh networks. It provides transparent Ethernet-like access to all participating nodes, thus easily allowing the employment of different higher level protocols like IP (with DHCP), IPv6, AppleTalk, etc.	http://awds.berlios.de/
B.A.T.M.A.N.	Better approach to mobile ad-hoc networking is a routing protocol for multi-hop ad-hoc mesh networks	http://www.open-mesh.net/
MANET Implementation	A survey of MANET software implementation in Linux	http://www.comnets.unibremen.de/~koo/manet-

Software		impl.html

Table 7: MANET Software

2.5 MANET Emulator for Security Evaluation

2.5.1 TeaLab

TEALab [25] is a testbed that aims to provide a representative environment in order to study attacks and attack recognition in ad hoc network. This testbed consists of a collection of MANET host communicating via common network environment such as Ethernet. To emulate wireless environment, TEALab controls network-to-link layer filtering to determine whether one host receives packets from other host or not. This is accomplished by the use of netfilter / iptables project [16].

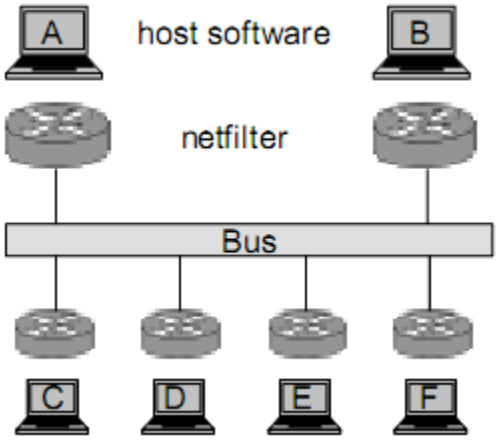

Figure 1:TEALab Architecture [25]

In addition, TEALab provides some components such as scenario generation, traffic generator, topology management, topology visualization, and a set of 33 attacker tools. Advantages of this system are facilities to study MANET security, easy configuration and scenario generation. However, because of the approach of this emulator, wireless propagation is unable to be simulated. As the result, most of attacks tools only run on network layer, or have a specific target.

2.5.2 SWOON

SWOON [39] is a testbed secure wireless overlay network. Its architecture is build on DETER [37] testbed that provides an infrastructure for repeatable experiment in computer security. By add new features to support wireless environment, SWOON enables to study security issues in wireless environment. Its architecture consists of DETER server, the control client, application-

shadow node pairs, and secure virtual links to communicate with DETER experiment nodes. DETER server provides facilities to control experiments in testbed, and creates the network topologies by setting up VLANs in switches. Control client provides a graphical user interface tool to configure desired network topologies, and sends commands to shadow nodes. Application-shadow node pairs comprise two kinds of node. One of them is application node on which real application can run. The other node is shadow node acting as a virtual wireless interface for an application node. Each node in the same VLAN can listen to other nodes talking. The 802.11 packets in emulating network are converted to 802.3 by adding a new 802.3 broadcast header. Virtual link is used for isolated private network from public network to prevent from malicious attacks.

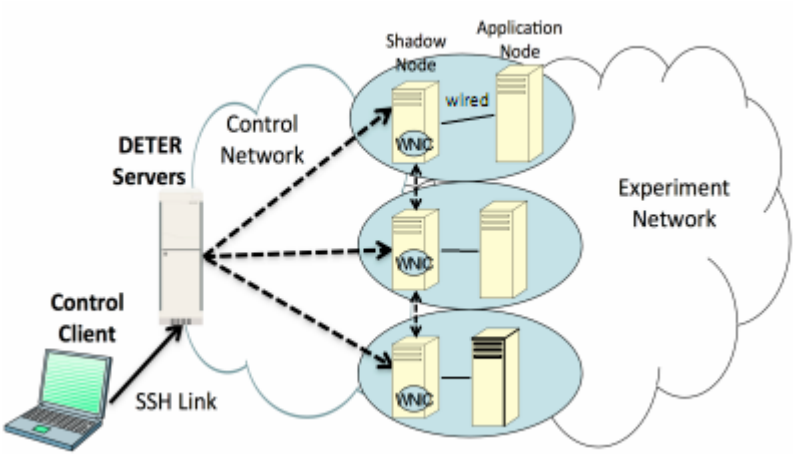

Figure 2: SWOON testbed [39]

This testbed provides a high realistic wireless environment for many purposes, especially security evaluation purposes. However, this architecture requires a complex in network configuration and computers when it has a large-scale testbed, for instance over 50 nodes. Besides of that, with a high topological changing network, it needs a series of reconfiguration for switches, so it takes a lot of time to do that.

2.5.3 S-MobiEmu

S-MobiEmu [40] is a software platform supported routing protocol and attack library for security solution study. This platform software bases on a public wireless emulator called MobiEmu to create a wireless experiment environment. In addition, the authors also add an attack emulator layer, called Basic Ad-hoc Security Routing (BASR) as a common abstraction layer for attack injection and for test case development.

The wireless emulation approach of [42] has the same idea with TeaLab. This means that MobiEmu also uses packet filtering, Netfilter/iptables in Unix to control packets from source nodes by adding and removing rules. Fig 3 shows the architecture of MobiEmu Testbed

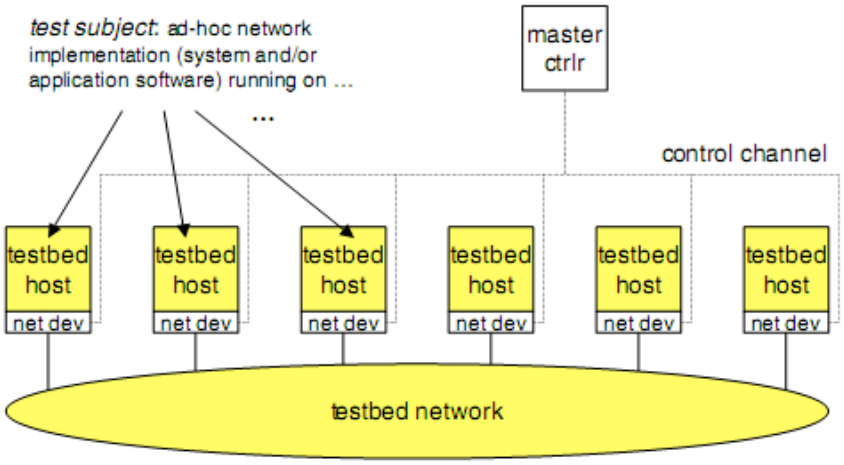

Figure 3: MobiEmu Architecture [42]

This emulator shares the problem with Tealab that cannot emulate a signal propagation environment and bandwidth adaptation. As the result of that, this emulator suits for studying security on network layer and upper layers.

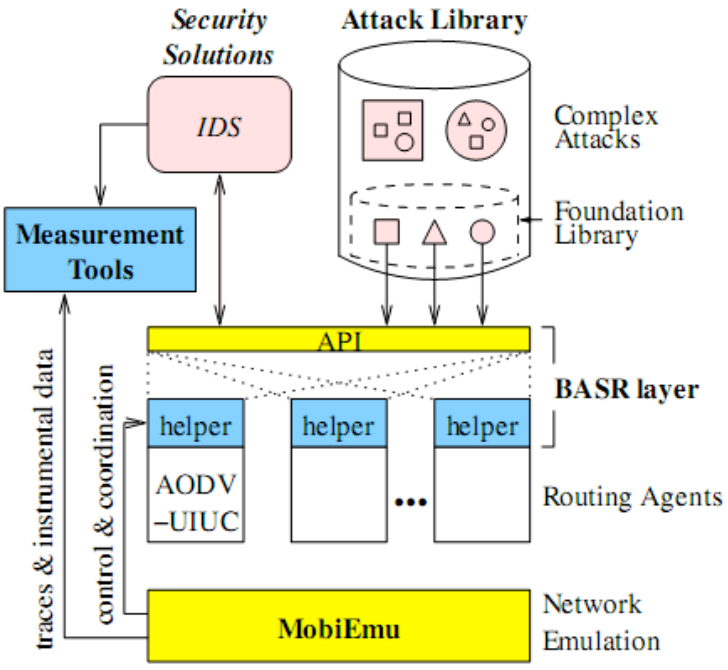

Figure 4: S-MobiEmu Architecture [40]

The role of S-MobiEmu provides facilities to implement attack on MANET. A set of API is developed for some type of common routines such as capturing and intercepting packets, overhearing traffic, and accessing routing table entry. Although these API are helpful to create security tools, they are also obstacles to extend flexibly security software.

2.5.4 QOMET

QOMET [1], designed and developed by Razvan et al, is a multi-purpose wireless emulator. It can be used on real network testbed to perform experiments related to 802.11 a/b/g WLAN network based on wired-network. For some scenarios, especially in wireless ad hoc network, QOMET is able to emulate node's movement and position in a genuine map. To create a wireless environment, the authors proposed a scenario-driven architecture. In the first stage, real-world scenario representation is calculated in order to generate quality degradation (ΔQ) description. In the second stage, the ΔQ description is converted in to emulation configuration to be applied for user-defined scenarios on wired network.

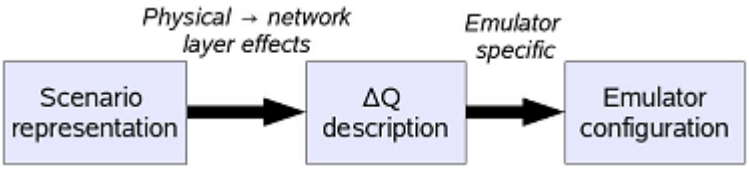

Figure 5: QOMET processing [1]

For making an experiment of mobile wireless ad hoc network, authors also provide a tool called do_wireconf[32] which applies QOMET output file to Dumynet[33] that controls connections' condition between nodes. With this approach, QOMET successfully create a wireless ad hoc emulator for various purposes. However, emulated network is just based on conditions of connections, so QOMET is a link-based wireless ad hoc emulator, like TEALab emulator. Therefore, in some evaluations such as data link layer security, or wireless propagation testing, QOMET does not provide such environment for these purposes.

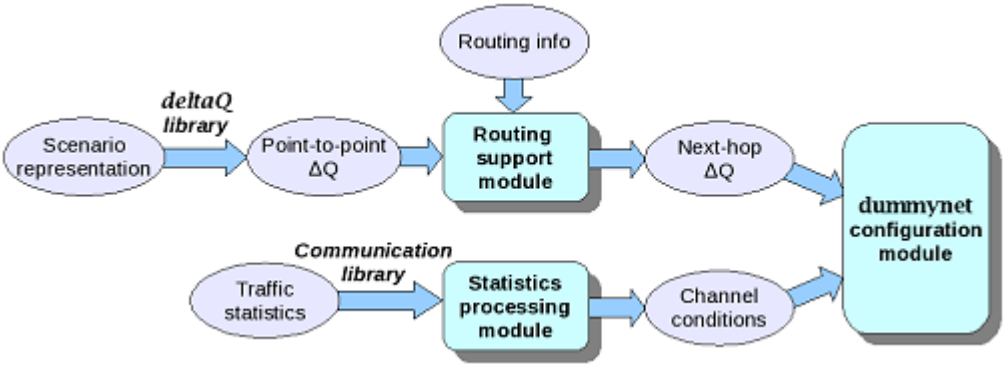

Figure 6: QOMET operation [32]

Taking all the limitations into account, we aim for a system which is highly realistic, flexible, scalable, and be able to work with different scenarios as well as different network layers and network conditions.

Chapter 3.
eBATMAN : An emulator–BAsed Testbed for studying Mobile Ad hoc Network security

3.1 Overview

As described on above sections, to appraise abilities of attack or defense strategies on mobile ad hoc network, a testbed should provide a real-world liked environment and facilities to get a credible results. This motivates us to develop an emulator-based testbed that satisfies the requirements on introduction section. Our test is called eBATMAN, an emulator based testbed for studying mobile ad hoc network.

As indicated its name, this testbed contains an emulator that creates wireless environment over a wired network, and facilities to measure security. Fortunately, our group has developed an emulator called QOMET, which emulates such wireless environment. However, as discussed its limitations, we propose a tool called BroadEnvi on which QOMET and other programs can run. This tool is responsible to establish a virtual wireless propagating environment, and dispatches packets in and out to the network. Several facilities such GUI application, Action Executor, and Report Collector help us to reduce time and cost for generating large scenarios. And additional, we will evaluate our emulator on StarBED[20], and make some sample evaluating on attack and defense on our testbed. Before showing these results, we will analyze in detail our system how it is constructed, and how it works.

3.2 Architecture

Our testbed contains five main parts: Emulator, Experiment Controller, Action Executor, Report Collector, and GUI application. A general view and relationship between these components is shown in Fig 7. Following are the descriptions for all components' functions:

- **Emulator:** we use QOMET as the core emulator. Besides, we also provide a wireless propagation environment overlay on a wired network. Therefore, our emulator affords a highly realistic environment for several types of testing on security.
- **Experiment Controller**: we use SpringOS, available for StarBED, to control conducted experiments. Spring OS was exclusively developed for StarBED project, in which a thousand computers are connected together usable for multiple purposes.
- **Action Executor** is a program to help an arbitrary node perform predefined actions in an experiment. This program receives time reported from QOMET and do appropriate actions within these assigned time slots. We will also create a set of action scripts to be called from Action Executor.
- **Report Collector** collects reports from nodes in a long experiment where hundreds of nodes run. Since this task has many things to do related to performance and data explosion, we propose an algorithm to collect reports efficiently.
- **GUI application** helps build scenarios and test testbed configuration for emulated networks.

How these parts work together will be discussed in the next section.

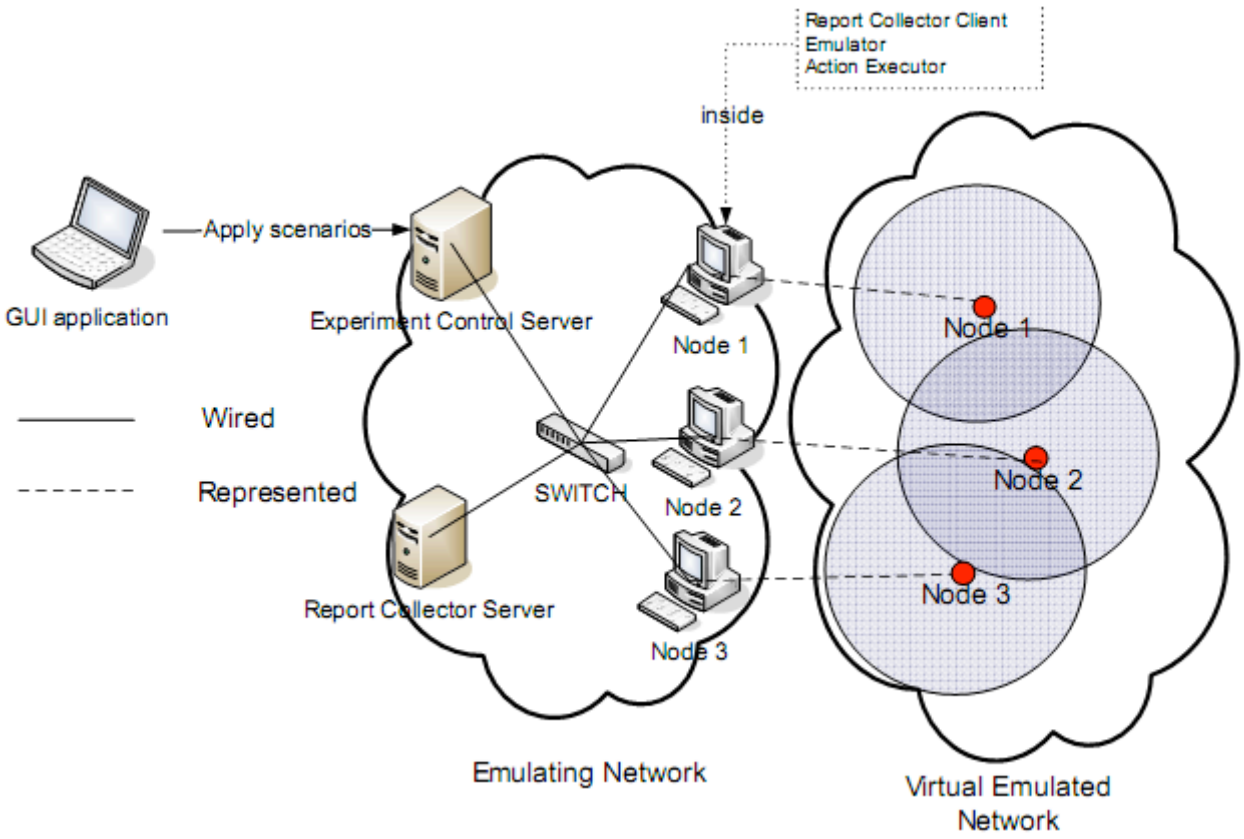

Figure 7: eBATMAN Architecture

3.3 System Elaboration

3.3.1 Experiment Controller

As provided aims, we had a plan to deploy eBATMAN on StarBED, which is a facility that contains a thousand of computers to verify the implementation for real environment of large scale environment. To reduce time and cost when build an experiment, SpringOS has been developed to manage not only hardware but also software in StarBED.

SpringOS is not an operating system. But it is rather set of software that stays on server and clients to assist in automatic experiment execution. The functionalities of SpringOS and StarBED as follow:

- **Resource management:** resource management assigns nodes to users, following the criteria described in configuration file. VLAN is required to separate more than one experiment at the same time, and same physical network.
- **Booting nodes: Wake up LAN** is a mechanism used to boot up nodes.

- **Software installation on nodes**: in all experiments, a disk image is created to spread to all nodes. Swipe out program is responsible for installing the image to nodes in one by one.
- **Building topology for experiment**: experiment topology can be built automatically by applying configuration file with VLAN IDs. Virtual networks are set up by configurable switches.
- **Driving scenario:** to run an experiment, a predefined scenario is created and is applied to all nodes before starting. Master will transmit commands to all clients installed on each PC. By this way, master has ability to manage all nodes at the same time. In addition, using message passing, clients can synchronize with others.

This Fig 8 illustrates the SpringOS' behavior

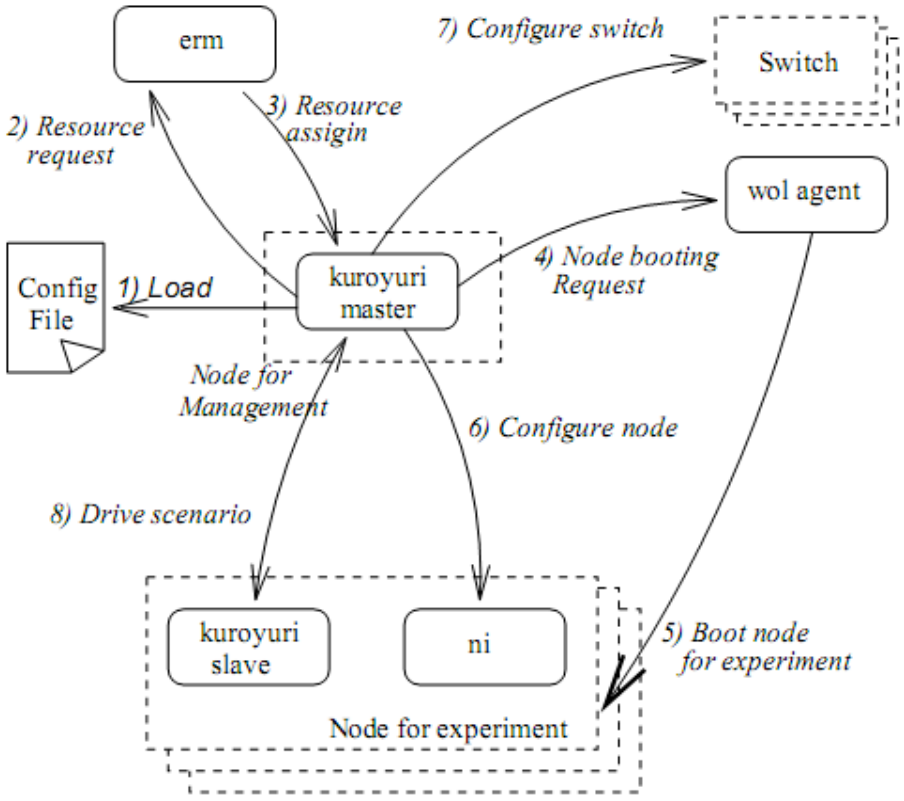

Figure 8: SpringOS Behavior [20]

These are steps to make and run an experiment on StarBED using SpringOS

- Prepare management node
- Set up experiment nodes
- Prepare experiment scenario file
- Execute experiment

3.3.2 Emulator

In our architecture, one physical computer in emulating network presents one logical node. All nodes in emulating work run and communicate with other ones in a virtual wireless environment. The general view of the emulating and virtual emulated network is displayed in Fig 9.

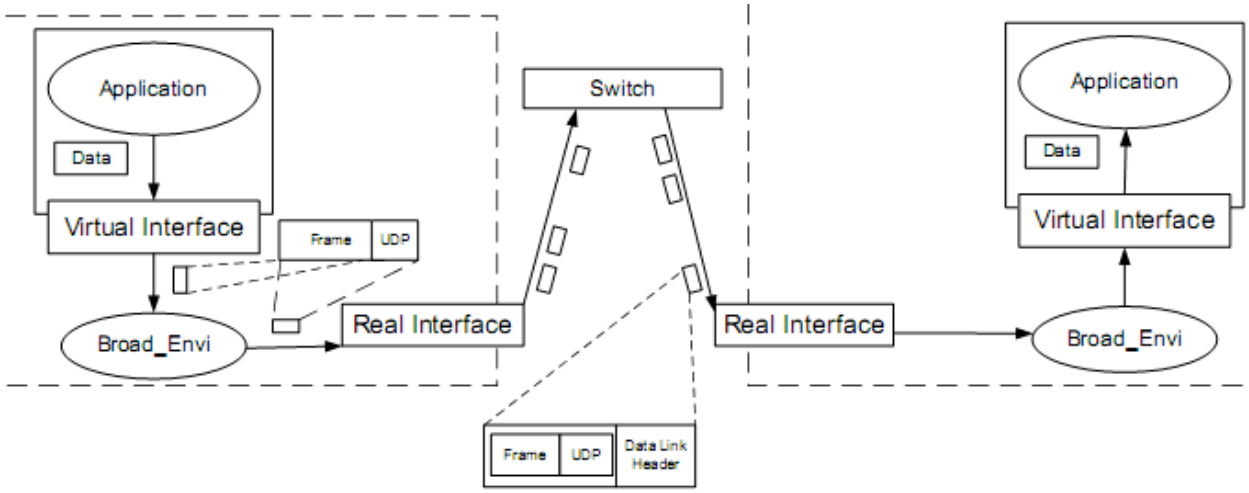

Figure 9: BroadEnvi Operation

To build up the system, we use a virtual interface acting as a wireless device to send and receive packets on the emulating network. When a packet from the application layer comes to the virtual interface, then is forwarded to BroadEnvi, a packet dispatcher. This tool presents both in the source and the destination node. In the source node, BroadEnvi gets the packets, wraps them, and then broadcasts to all neighbors of the residing node. On the destination side, the sent packets will be unwrapped by BroadEnvi, and come into the virtual emulated environment. BroadEnvi also checks MAC addresses on the packets in order to verify whether those come for this node. Nevertheless, when packet sniffers, or network-monitoring tools run in application layer, BroadEnvi will bypass all packets.

Having an aim to make our testbed be able to emulate security insurance in wireless networks from layer 2, a wireless propagation environment is a really important task for us. We propose two different approaches to create a wireless propagation environment. One uses UDP broadcast packets to send to all nodes in the network, while the other uses UDP unicast packets to neighbors. In the first approach, UDP broadcast packet is used to wrap the raw frame. One might think it is simple just to put the frame in a broadcast frame in data link layer. However, approaching this way, we have to modify the operating system kernel to remove the necessary header before it come into packet processing path, and it will become very complicated. Therefore we will stick with UDP broadcast. Advantages of UDP broadcast are as follows: (1) each node just sends a packet one time, and others receive it at the same time; (2) with a high

capacity core network, this approach can support around 50 nodes. However, because of the broadcasting usage, network throughput is shared among every node, so this limits the number of nodes in an experiment. In addition, it also requires all computers to reside in the same network segment. Since the first approach has those limitations, we test the second approach of using unicast packets. The second idea is that each packet is sent one by one to all neighbors. This idea solves the problem of an overworked shared environment, but it increases throughput and resources required at each node. Nonetheless, BroadEnvi will have to synchronize with QOMET to determine who its neighbors are. Additionally, this idea meets difficulties when a node's number of neighbors increases. When a number of neighbors increase, a node's throughput is decreased slightly.

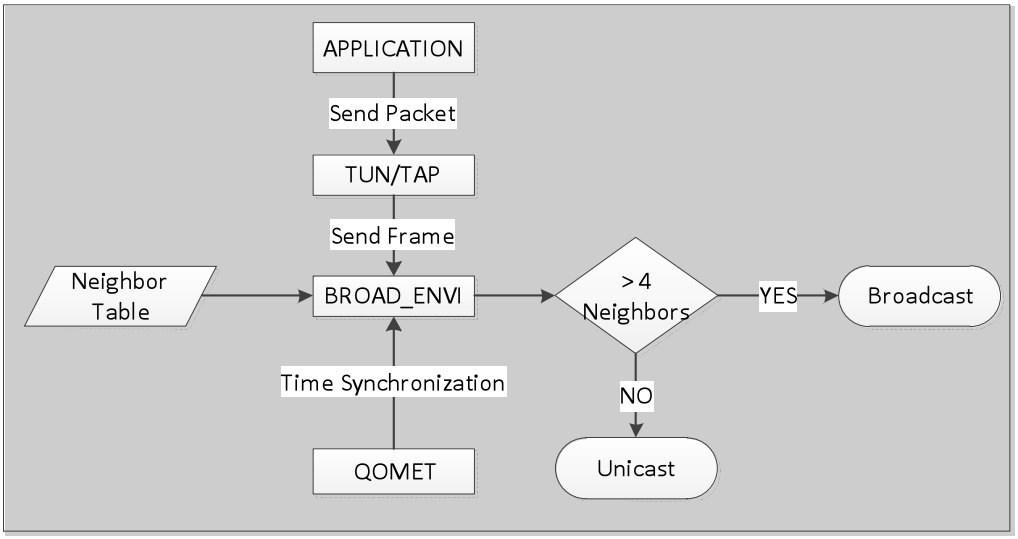

Figure 10: Packet Dispatcher

For advantages and disadvantages found in each approach, in our implementation, we combine two approaches into BroadEnvi to provide scalability to the overall system. Depending on the throughput and a node's capability, a node will check for the number of neighbors it has, if this number is big, say bigger than 4, broadcast will be used. We come to this number after several experiments that show a good balance around number 4. The evaluation of two approaches is discussed in the next part.

3.3.3 Action Executor

In our testbed, one of the most interesting features is that we can trigger attacks from any node or shield a protection on any node. We observe that the length of the action, either attack or protection, affects the successfulness of the applied security policy. For that reason, we develop and deploy an Action Executor on each node to make this possible. Fig 11 illustrates Action Executor's operations.

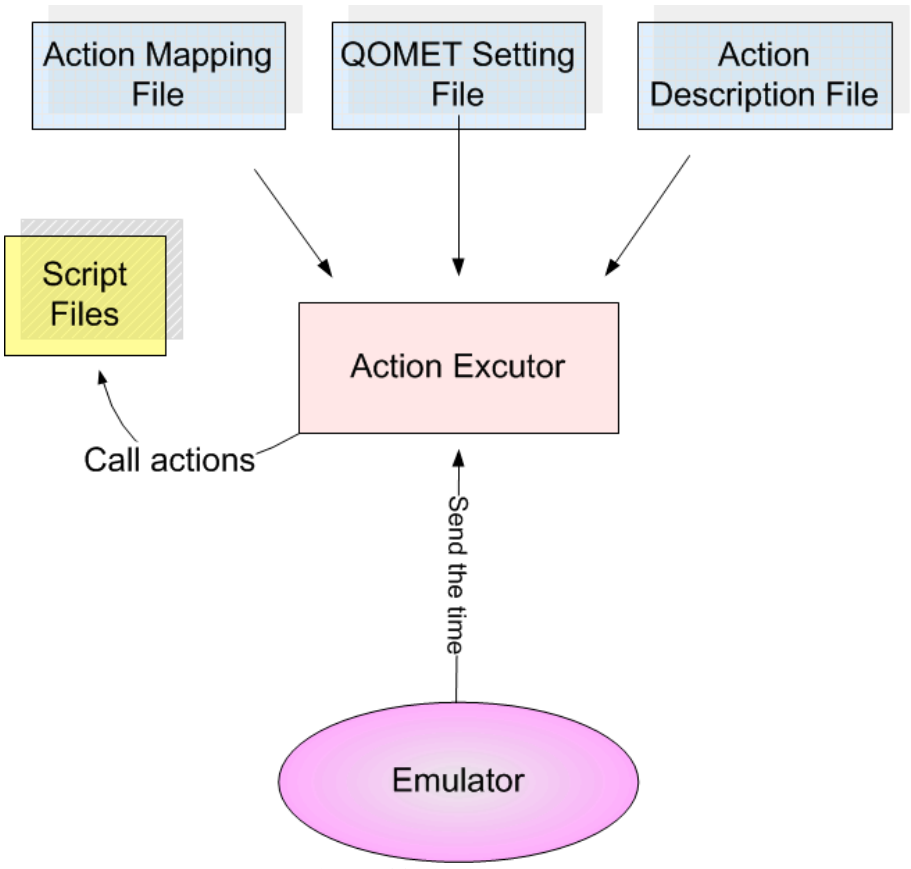

Figure 11: Action Executor

An Action Executor is responsible for revoking actions, either attack or protection. To support the meta-action, attack and protection, we also implement several minor operations such as capturing all possible packets, and monitoring network activities. To make an action carried out on time, the Action Executor is synchronized with the emulator to ensure the correctness of the predefined scenario. Additionally, we also define many action scripts to support revoking actions on each node. With this approach, users can create their own action scripts and easily integrate those into the system.

3.3.4 Report Collector

The Report Collector is responsible for collecting reports from all nodes in an experiment. A report file can contain routing table, operating system, IP address information, and other things that a node has. The problem is that how we can efficiently collect report from nodes in a network if the size of the network is huge. It is very likely that with a big number of nodes, or a huge amount of report files, the data server will be flooded and unexpectedly halt. This bottleneck problem motivates us to propose an algorithm to collect report efficiently to prevent the server from overworking. In our algorithms, nodes are arranged in order by their ID in a virtual circle as you can see in Fig 12.

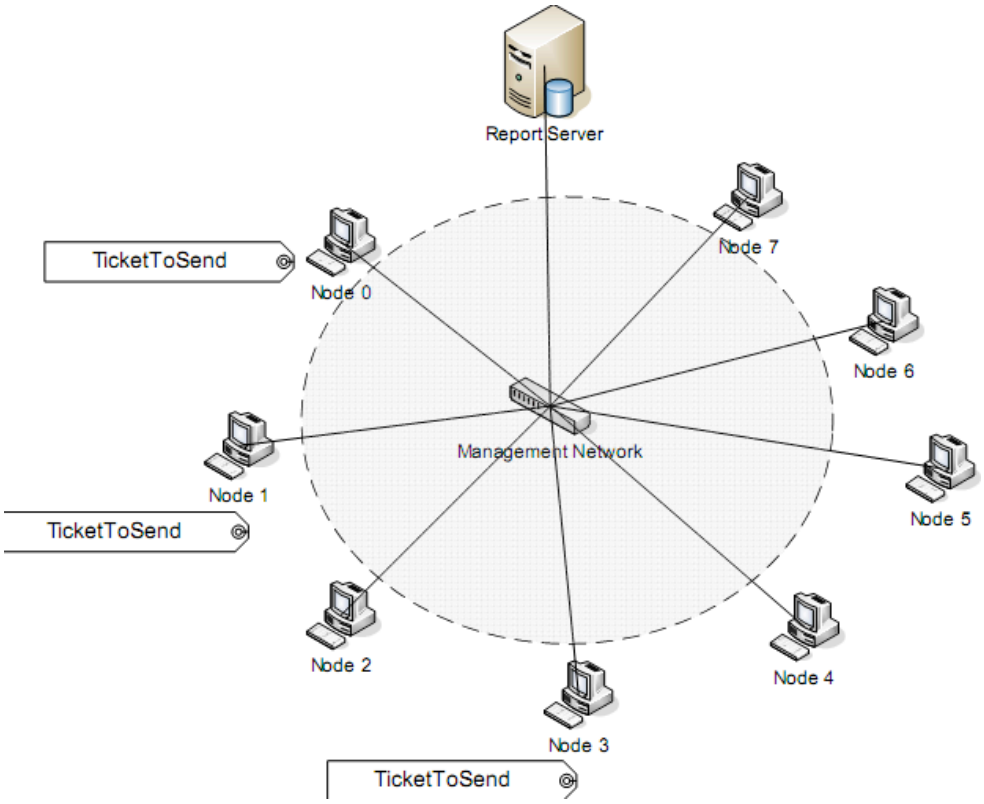

Figure 12: Virtual ring of nodes

According to server performance and workload, we define the number of nodes allowable for uploading reports simultaneously. To begin report collection, the Report Collector server sends a request ticket to the first node in the circle with the maximum number of uploading nodes indicated in a ticket field. When the first node receives this ticket, it intermediately forwards this ticket to the next node in the circle. The second node checks the maximum hop count field and increases the hop count field in the ticket. If these two fields are equal, this node stops forwarding the ticket. After, when a node finishes uploading its report, it automatically sends a request ticket to the next with maximum hop count of 1, and hop count is 0. (Note that the hop count on a ticket is only increased when the ticket is kept at one node rather than forwarded). The process starts over again for other nodes left in the circle. If uploading processing is running, and a request ticket comes in, then it will forward this ticket to next hop. When all tickets come back to the first nodes, it will drop them. Any ticket coming to the first node will be dropped and thus that stops the uploading process.

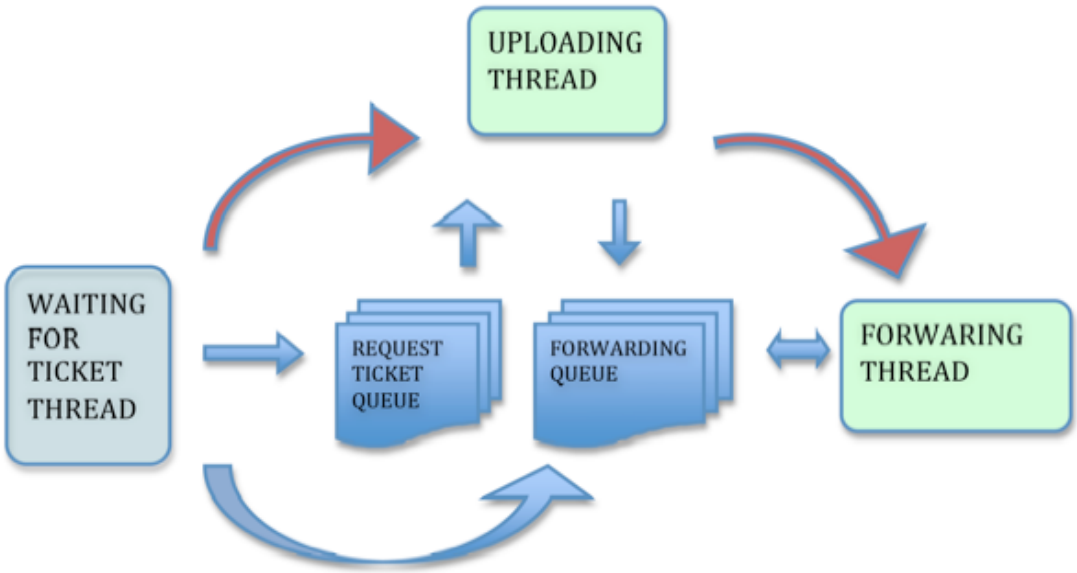

Figure 13: Report client process

The detail processes occur in Report Collector as follows. At first, after initializing global parameters, three threads are created, namely Wait Ticket Thread, Uploading Thread and Forwarding Thread. At Wait Ticket Thread state, a server is set up for listening tickets from other nodes. When this server receives a ticket, it puts this ticket to uploading queue, and passes a signal to Uploading Thread. When receiving a signal, Uploading Thread reads queue, and gets a ticket. If this ticket has hop count more than one, hop count will be decreased one. After that the ticket is pushed to forwarding thread, and a signal is sent forwarding thread… Concurrently, uploading state is checked if it works or not. In the case spare state, uploading state is come, and Uploader will collects files at a predefine folder. This will trigger Connect To Server. At this state, the program tries to connect to server that is defined in this ticket. If the server cannot be connected, the ticket is sent back to Request Queue. Otherwise, program uploads all files to the server. Finishing upload process, a ticket with one hop count is generated and put to Forwarding Queue. Forwarding Thread receives a signal to read its queue, and then it tries to connect to the next neighbor. If neighbor ID is smaller, ticket will be dropped. Otherwise, the ticket is transmitted to this neighbor.

3.3.5 GUI Application

The GUI application, as shown in Fig 14, provides some facilities such scenario generation, scenario visualization, and action description file creation. Scenario generation establishes a descriptive model for the network environment. This model defines various types of objects such

as buildings, roads, nodes, and every node's movement and actions. A GUI program drives a scenario in a graphical simulation to have an overview of nodes' movement behaviors. In addition, the GUI program also generates QOMET scenario descriptive file and Action Executor descriptive files for a transformation into binary files so that QOMET and the Action Executor can use.

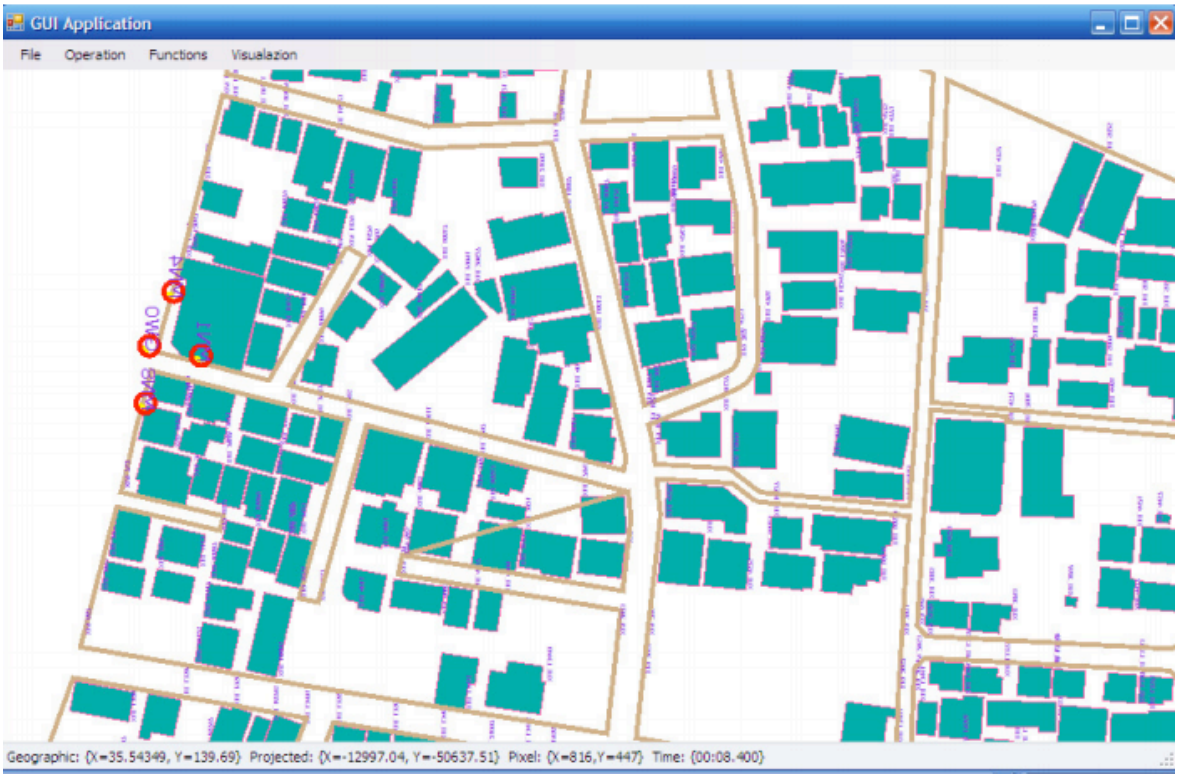

Figure 14: GUI

The Fig 15 presents the use case of GUI application. GUI application provides 3 functions for user: creating a scenario, visualizing QOMET output, and editing QOMET and Action XML files.

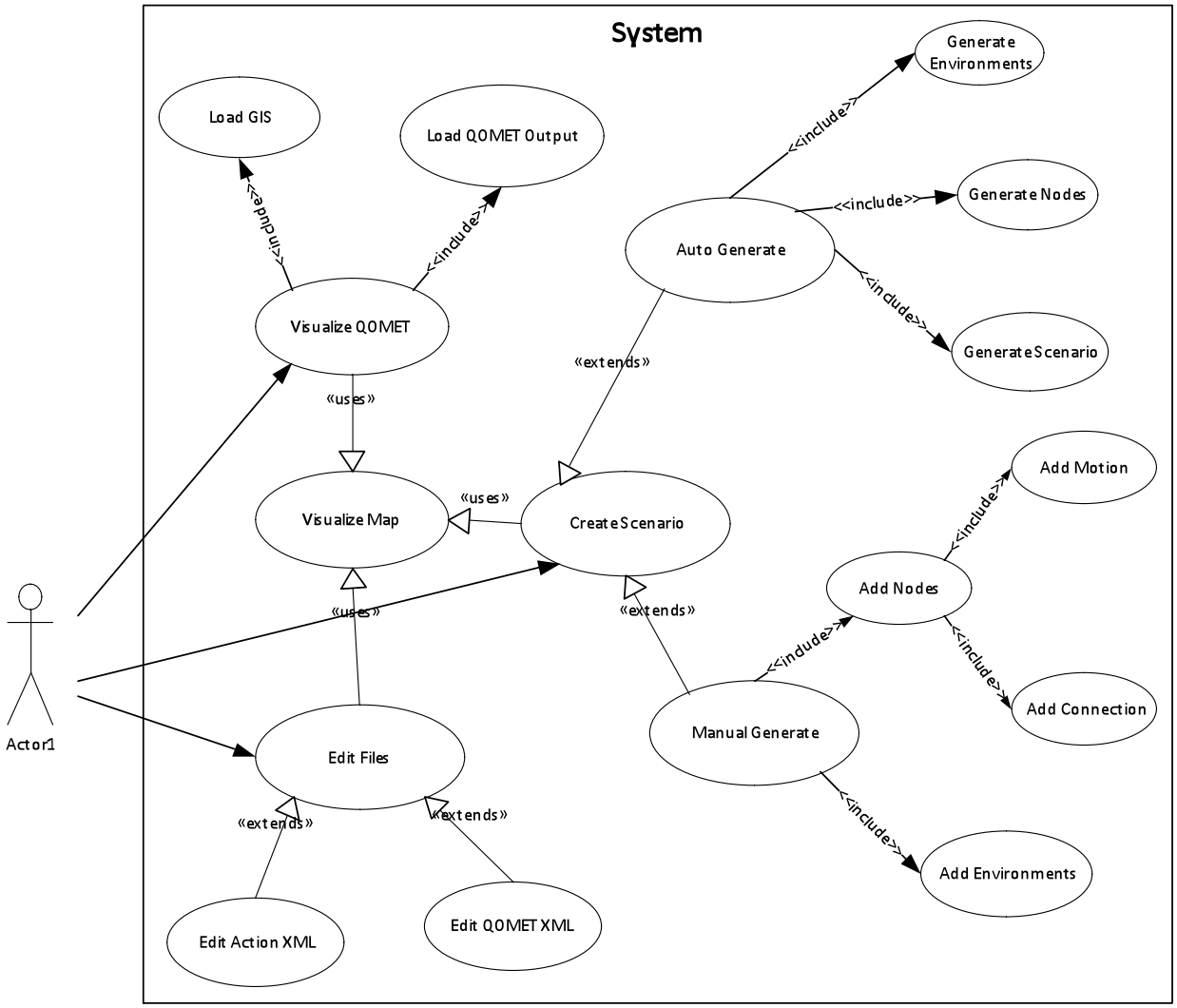

Figure 15: Usecase

The first function is **Create Scenario**. Users can choose either automatically generating function when they need a large scenario, eg. more than 50 nodes, or manually generating one. While automatic generator will randomly create map, nodes, environments and motions, manually generator has to add one by one node, and connection types, environment types, and both motion and connection for each node. However, to save time in these processes, users can define all properties of all nodes in the map in pre-defined function, and then if they want to change some things, they just select a node and change appropriate values. After finishing generating scenario, users can visualize map and nodes' mobility to recheck if they are correctly or not.

The second function is **Edit Files**. This function has an ability to edit pre-created files such as Action XML and QOMET XML files. The third function is visualizing map. Input file is a QOMET output file. This function provides a scenario graphical animation to recheck node's mobility before deploying it on testbed.

Chapter 4.
eBATMAN Assessment: Performance and Security Evaluation

4.1 Evaluating System

In our eBATMAN assessments in its performance and security evaluation, we use two systems. The first one is a small testing network; we build 4 computers each has specification of 700 MHz CPU, 256 MB RAM, and FreeBSD 5.4 OS. All computers are connected to via a fast Ethernet. To generate traffic in for the emulated network, we use Iperf v1.7 and choose OLSRD[17] for routing protocol in our system. This network is used for program's performance testing.

The second one is a big network on StarBED. We use 40 computers, which model is NEC Express5800 110Rc-1 with 1GHz pentium III CPU, 512Mb RAM, 2 Fast Ethernet, and FreeBSD 5.4 OS. We also choose OLSRD for routing protocol in our system. This system is used for attack evaluation.

Several of programs are used in these tests such as tcpdump, arp tool, rose.c, tcpsyn and snort.

4.2 System performance

In this section, we present some results that illustrate performance of BroadEnvi, our main contribution, in some scenarios.

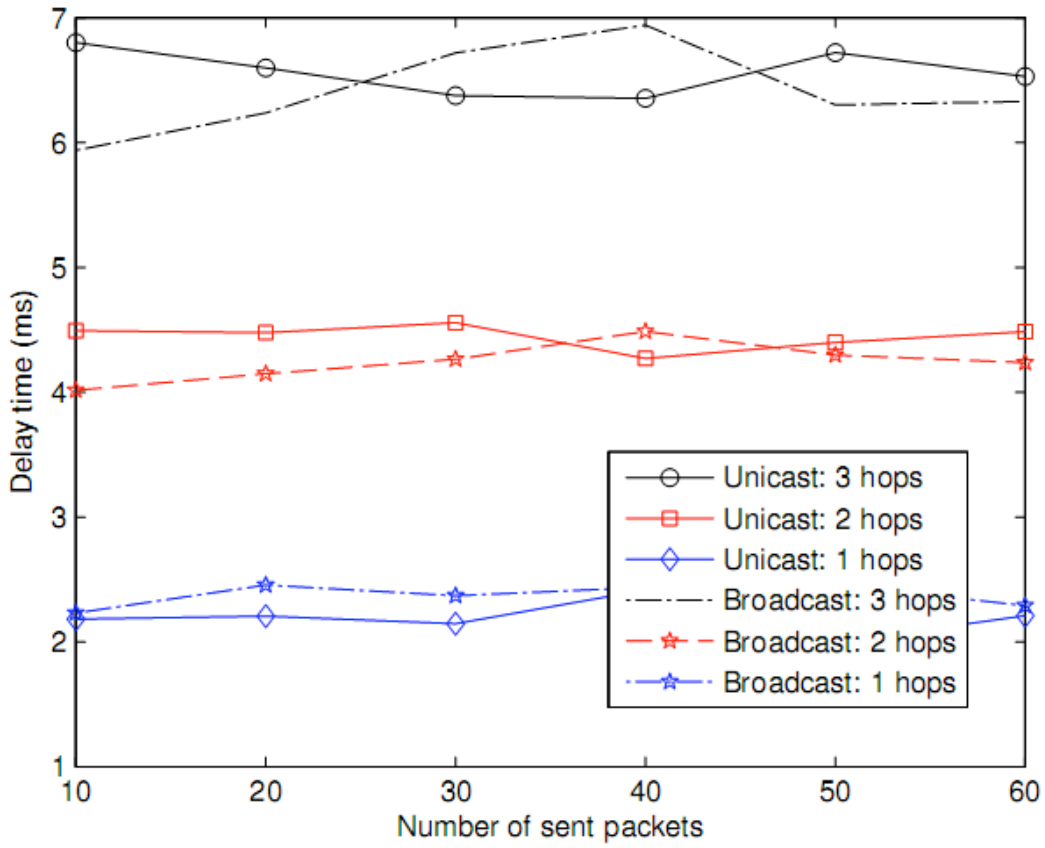

Figure 16: Delay Examination

Figure 16 exemplifies delay time of network packets in two approaches as mentioned: unicast and broadcast. When there are 1 or 2 hops, delay in the unicast approach is lower than in the broadcast approach in most cases. An explanation for this is that the feedbacks packets from the switch will be sent back to the source thus make the sending process slower. In the case of 3 hops, these values are relatively equal in average. The delay of BroadEnvi is low enough to expand quantity of network.

4.3 MANET Security Testing

4.3.1 TCP-SYN Denial of Service

We carry out experiments, where nodes are places according to Fig. 14 with 4 nodes to understand the effect that a Denial of Service (DoS) attack creates on performance of a target node.

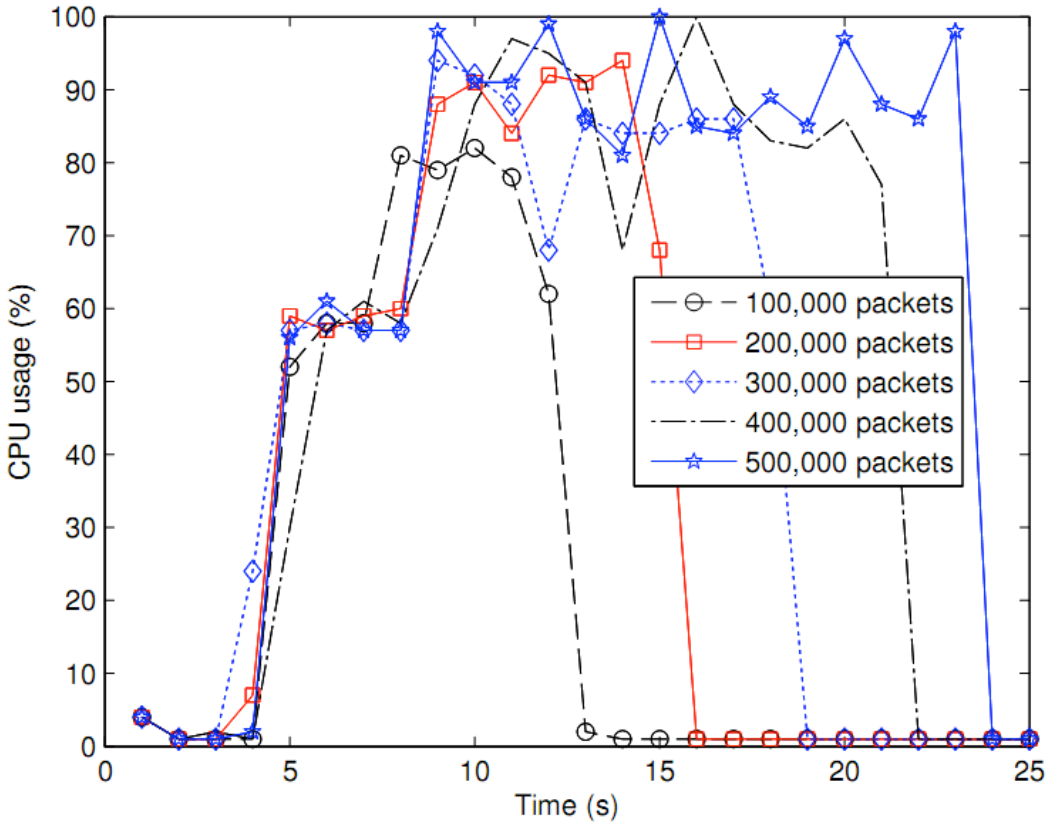

Figure 17: Performance of a attacked node during DOS attack

In this test, we execute a TCP_SYN DoS attack on a node in the network. During the first 5 seconds, since there is nothing running on the node, CPU usage is low, around 2%. At the 5th second, Iperf is turned on to check throughput between two nodes. This tool will automatically stop working if loss rate is high, and it takes around 50% of CPU usage. At the 7th second, attacks are executed. Each attack has 10 stages; the followed stage is one second after the previous one. We vary the number of packets send in each stage of during an attack from 10,000 to 50,000, therefore the total number of sent packets per one attack is from 100,000 to 500,000 packets. We see that CPU usage, when the victim encounters a big number of packets, drastically rises in all attacks. For the 100,000-packet attack, because the number of packets sent at each stage is not big enough to break down the system, CPU usage reaches maximum at 80%. In this case, Iperf stops working due to high loss rate at the 12th second. Next, we put in an attack with 20,000 packets per stage. We see the same trend of CPU usage and it is higher than that in the first case, but yet to reach 100%. In this attack, Iperf stops at the 11th second. Now we discuss attacks with the number of sent packets ranging from 300,000 to 500,000. We see that at some points, CPU usage reach nearly 100% and the high usage period lasts longer. It is noteworthy that after the time when Iperf stops, around the 11th second, the major portion of CPU usage at the victim is due to handling packets from attacker.

4.3.2 A study of Rose Attack on MANET

Rose Attack is a kind of denial of service attack types. Attacker floods packets to a victim to reduce its performance. Main mechanism of this attack is fragmenting packets, which means that packets will be sent to the victim in non-continuous sequence numbers. As the result of that, victim's operating system has to reserve recourses to arrange them in order. The characteristics of this attack tool are as follows:

- It can use either UDP or TCP to an attack facility.
- It changes source's IP address frequently.
- It can fake 00:00:00:00:00:00 for source MAC address.

In this section, we do some experiments to understand the attack's effect on ad-hoc network in both static and dynamic network. We study in a topology in which 40 nodes are randomly put on a map as below Fig 18. In this figure, nodes with solid color inside move randomly in dynamic network; in additional, they are also static nodes in the other network. Each scenario runs in 180 seconds, while attacking interval is around 50 seconds, and nodes start moving at 10^{th} second and stop at 150^{th} second. Attacking speed are around 50 packets per second. Minimum and maximum speed of each node are 0.5 and 1.0 meter per second respectively. In below figures, because of long we only demonstrate network state when the attack occurs.

In all scenarios, node 18 will attack node 27. We observe how many nodes are affected in this attack by watching their CPU usage and the packet lost rate of transmission between two nodes. In practice, this type of attack cannot influence other nodes' resources, because these packets will be dropped before they come into their OS kernels. However, bandwidth consumption and packet congestion still takes place in wireless propagate environment. As discussed in our testbed architecture, propagated packets will be sent to neighbors and dropped by BroadEnvi if they do not relate to this node; as a result, neighbors have to take an amount of CPU resources to treat them. Therefore, to find out which nodes are affected by the attack, we easily monitor nodes' CPU usage. At last, we will find out that how mobility affects to the success of attacks.

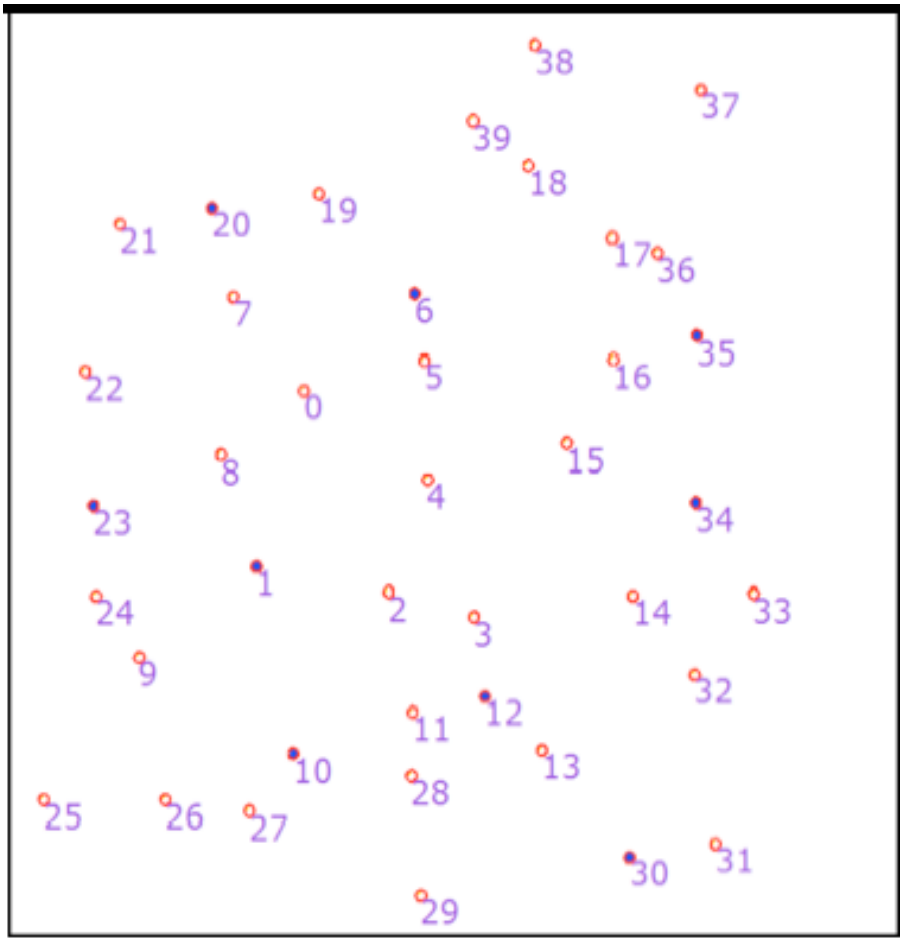

Figure 18: 40-nodes-scenario map

a. Attack and Mobility

Now, we discuss about the effect of attack between static and dynamic topology. In the dynamic topology, we have 9 nodes moving randomly in the network. In both cases, before starting attacking, to ensure routing path be created, 18 pings 27 in ten seconds.

This Fig. 19 describes CPU usage of static nodes during attacking.

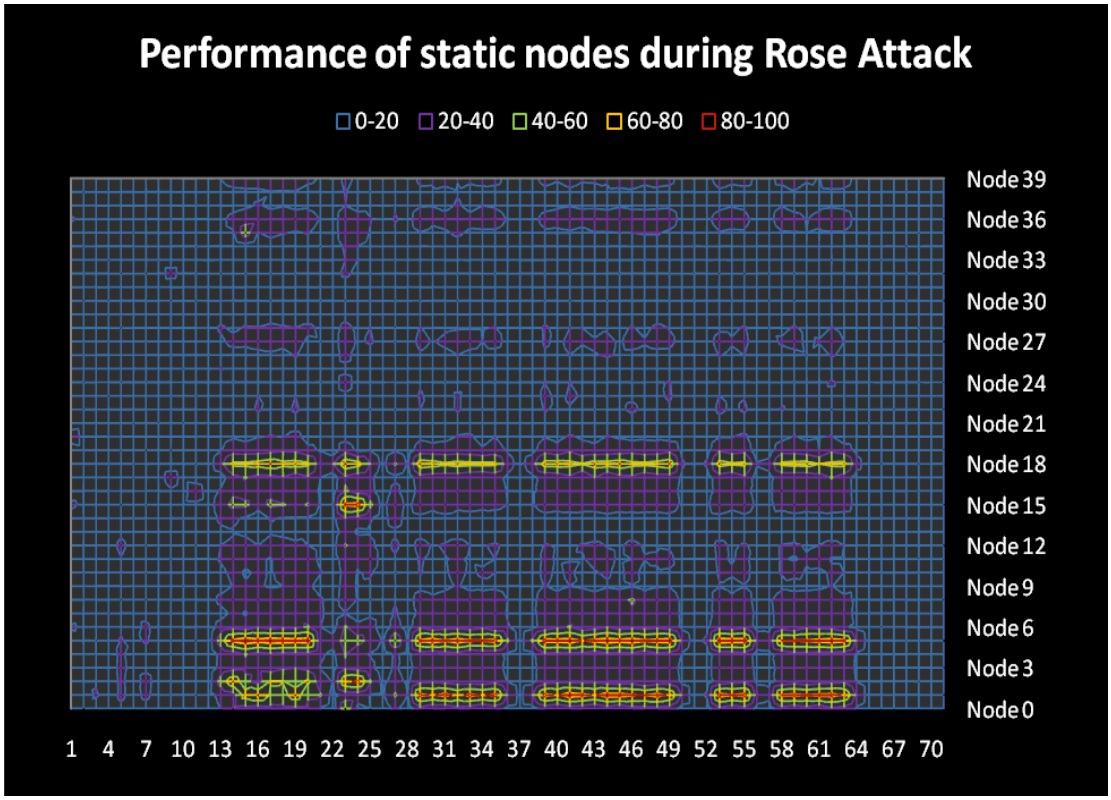

Figure 19: Performance of static nodes during Rose Attack

In this test, during the 13 first seconds, CPU usages of all nodes remain under 20%. Some of them jump to from 20 to 40 percent because they have many neighbors to send OLSRD packets. At the 13th second, the attack is started from node 18. Basing on the map and colors in Fig. 17, we can realize that attacking path is from 18 to 5, 5 to 1, and 1 to 27. Interestingly, although node 27 is the target, it is not as affected as node 1 and 5. The reason of this situation is that node 1 and 5 are intermediate nodes, so they receive a lot of packet from two neighbors, and packet loss rate is high enough to reduce the number receiving packets of node 27 (we will discuss about loss rate in the next part). While calling flooding packets at node 18, we catch on the screen announcements from OLSSRD which warns to unable to send packets. For that reason, at 22nd second, routing path is change to 15 and 2, but then it come back to 5 and 1. Additionally, our attacking tool is stop because of non-routing path to target. In the Fig. 18, these nodes are influenced by this attack like 0 to 8, 10 to 12, 15 to 19, 27, 36,38, and 39.

In conclusion, DOS attack in ad hoc network just strongly affects on intermediate nodes and routing protocol in the network. With faked source MAC address, a victim is difficult to investigate the attacker. And it is also an obstacle to defense because of faking address. We have installed Snort on each node to record network behavior. Snort logs are as follows

```
06/30-17:43:00.522115  [**] [1:528:4] BAD-TRAFFIC loopback traffic [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {TCP} 127.104.30.249 -> 10.8.0.27
06/30-17:43:00.522131  [**] [1:528:4] BAD-TRAFFIC loopback traffic [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {TCP} 127.173.7.206  -> 10.8.0.27
```

Figure 20: Snort Log

Fig. 21 depicts attacking in a topology with dynamical nodes. Attacking still start from node 18 to node 27. While in static topology, node 2 and 5 take over 80% CPU usage, these nodes remain under 80% in dynamic topology. A barrier to attacker is mobility of nodes, which causes routing table be updated frequently. Besides, as mentioned above, since DOS attack affects routing protocol, our attacking tool is often stopped. We can see that over 80 per cent of nodes in network receives attacking packets.

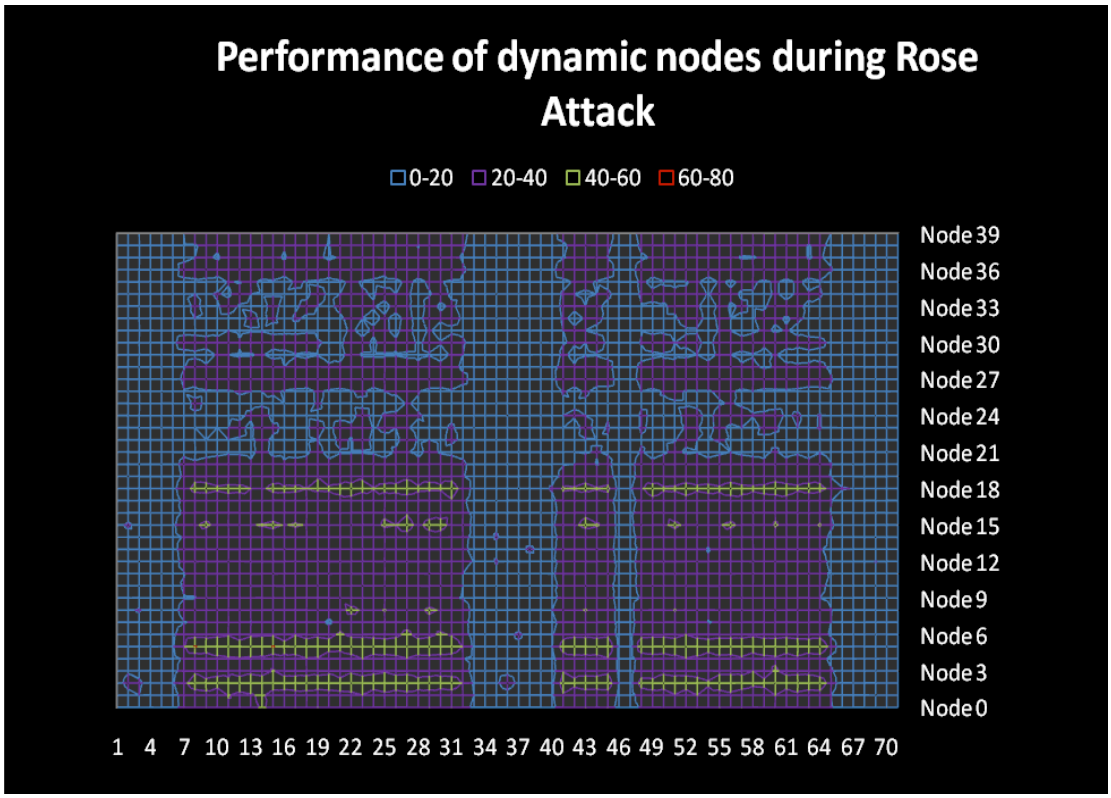

Figure 21: Performance of dynamic nodes during Rose Attack period

In summary, mobility is an obstacle for DOS attack on MANET to succeed. However, this also creates a side effect to overall network in which bandwidth of all nodes is reduced.

In next part, we study about packet loss rate during attacking in case both cases of dynamic and static topology.

b. Packet Loss Rate and Attacking.

In this test, we calculate loss rate between two nodes: 14 and 23 via UDP bandwidth test. Because Iperf will flood a huge number of packets into the network with high bandwidth testing, which causes the same result with DOS attack. Therefore, we examine this tool in low bandwidth testing to obtain loss rate of transmission. We test in both cases: static and dynamic topology to understand how mobility and attack associates with loss rate. The bandwidth for testing cases is 1Mb/s and 2Mb/s. With each of tests, we examine three times; and these below numbers are computed in average. Attacking speeds in both cases are around 50 packets/sec.

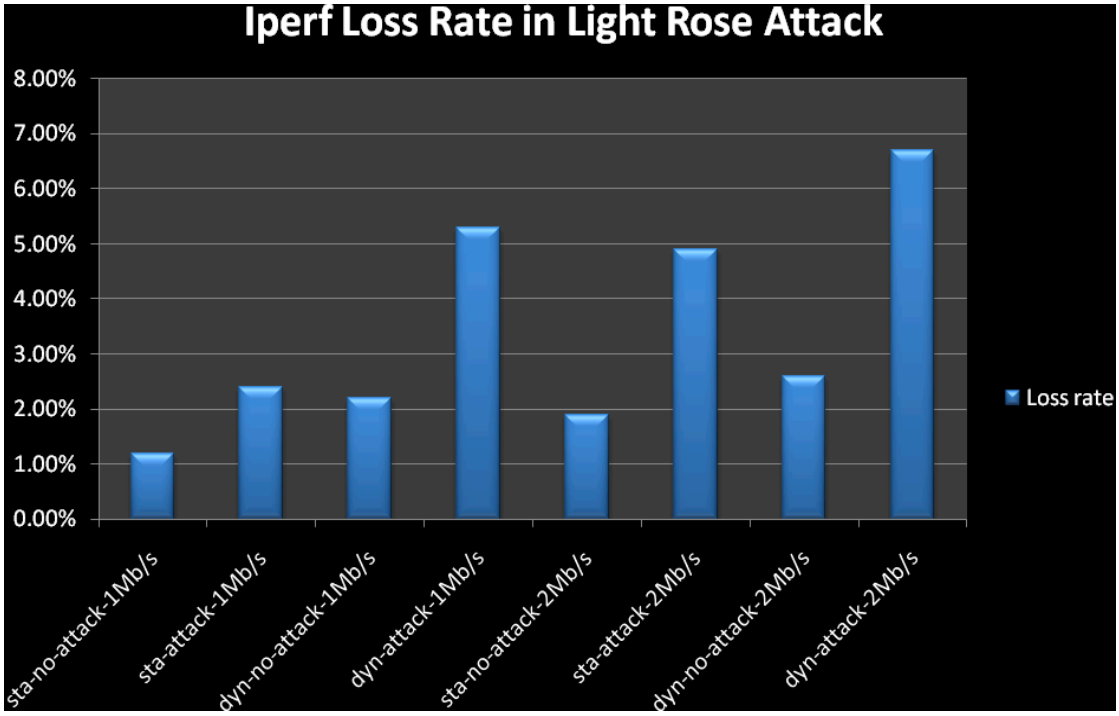

Figure 22: Iperf loss rate observation between two nodes

In the 1Mb/s bandwidth testing, while loss rate remains around 1% in static topology without attack, this rate in dynamic one is clearly higher. This means that node's mobility is an important factor to loss rate. In all of cases, the loss rate is double in scenarios with dynamic nodes. In addition, when attacking occurs in the network, loss rate also jumps up. In dynamic topology with attacking, loss rate of transmission significantly increases, more than 5%.

In the 2Mb/s bandwidth testing, we got the same form of results as previous tests. However, loss rate in the worst case rises up to approximate 7%. In practice, we agree that high-speed wireless transmission will get more highly packet loss because of signal collision in a large network. That is a reason why we put the low bandwidth in these tests to prevent trustable results from packet collision and over process of testing nodes in our system.

In all cases, loss rate depends on mobility and attacking. With this result, we go to explain what happens with node 27 in Fig.18. Because collision among intermediate nodes is too high, a number of packets to the target are significant decreased. Consequently, with high bandwidth-related DOS attack, attacker just kills all his neighbors, but not the victim.

c. Defense Rose Attack on MANET

We examine a method to prevent nodes from Rose Attack. We investigate on the captured logs, and find that MAC source of attacking packets is 00:00:00:00:00:00. Therefore, we propose a simple way is dropping all these packets via a rule of ipfw, a firewall on FreeBSD. Running again the same scenarios, we receive results as Fig. 23 below.

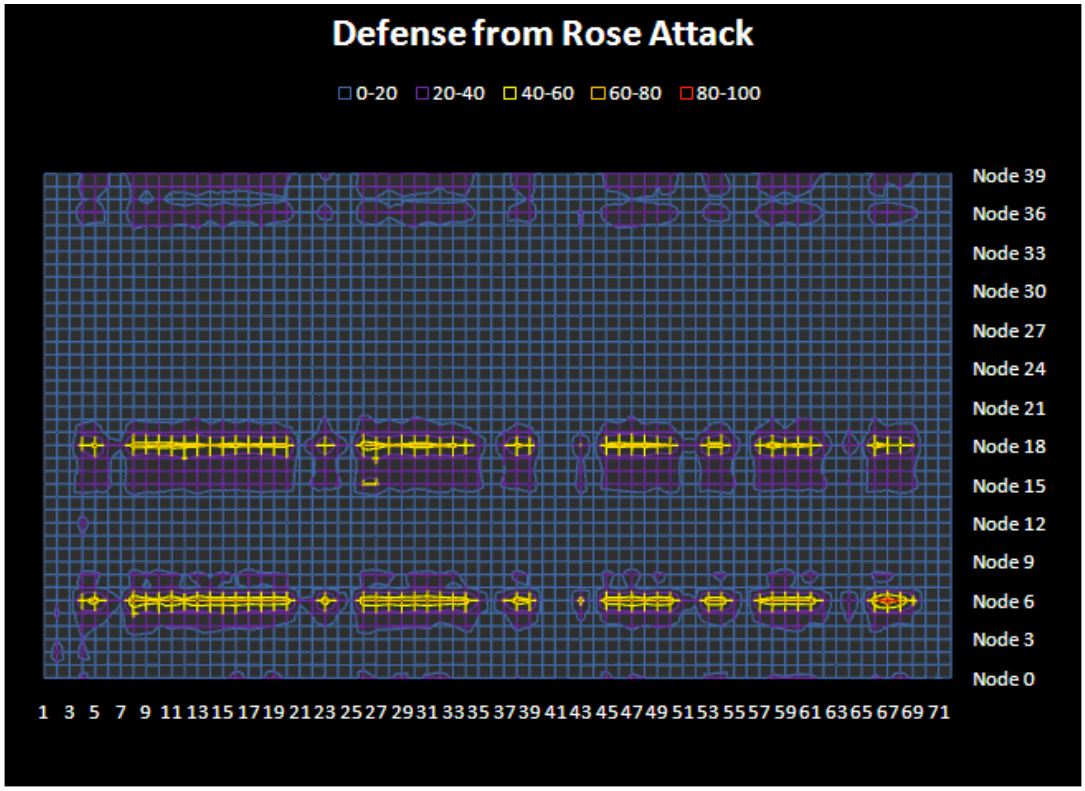

Figure 23: Defense Rose Attack on MANET

We are happy that the attack is blocked by the filter, but it still disturbs all the neighbors, especially node 6. An explanation for this case is that node 18 tries to flood packets to its victim,

and node 6 is the next hop on routing path from 18 to victim. As the result, node 6 has to receive these packets bringing its MAC address in destination field. After receiving them, it applies the rule to drop them, while other nodes drop these packets on their cards. That is the reason why node 6 must take a lot of resource for processing packets, while the others do not. From Fig 21, we can see node 6, 15,16,17, 36, 38 and 39 are neighbors of node 18.

In this test, we only demonstrate the defense of this attack on our system; it could be applied practice as well. However, real attackers will change MAC and IP address of packets frequently so that they cannot be detected, and bypass firewalls. A good countermeasure is combining IDSes and firewall in order to isolate attackers from the network.

4.3.3 A study on session stealing attack on MANET.

In this type of attack, attacker tries to mimic one side in two-party communication to steal their session. Another way to do this is man in the middle attack. But in this part, we concentrate on using spoofing attack to divert the communication between two nodes in mobile ad hoc network.

We study in two cases: attacking at first, then running the communication, and running the communication firstly, and then starting attack. Moreover, we conduct many scenarios in each case such as: faking only IP address, faking both MAC and IP address. Besides, we tested on if there is routing protocol running on attacker machine or not as well. In the case of non-routing protocol, we add some static routing entries for the attacker to send packets to victim. Although this type of attack is very simple, and is evaluated as critical vulnerability in MANET, we cannot succeed to bend the communications in practice. We will explain why as below.

In this testing, we try to bend the communication between two nodes and redirect it to attacker. The communication between two nodes is simply done by ping. We reuse the map above to build up scenarios. In all scenarios, node 3 will ping to node 37, and attacker, node 0, takes an effort to interfere the communication by mimicking either IP or MAC address, or both of them.

We play a role as an actual attacker in real-world. The first work is footprinting. We start tcpdump to capture all packets travelling in environment. After a minute, we obtain enough information to begin attacking. These are MAC and IP address of nodes which take part in communication between node 3 and node 37 as follows.

- 10.8.0.13 is at 00:bd:49:a0:01:00
- 10.8.0.15 is at 00:bd:ef:9f:01:00
- 10.8.0.27 is at 00:bd:68:9e:01:00

- 10.8.0.47 is unknown because this node is out of node 0's range.

Communicating path is from 3 to 5, 5 to 17, and 17 to 37. We repeat this first scenario and then only change IP address of node 0 to node 37. We can listen to all packets from 5 to 3, 5 to 17.

```
14:59:30.443504 00:bd:ef:9f:01:00 > 00:bd:68:9e:01:00, ethertype IPv4 (0x0800), length 98: IP (tos
0x0, ttl  63, id 53445, offset 0, flags [none], length: 84) 10.8.0.13 > 10.8.0.47: icmp 64: echo request
seq 3
14:59:30.445576 00:bd:ef:9f:01:00 > 00:bd:49:a0:01:00, ethertype IPv4 (0x0800), length 98: IP (tos
0x0, ttl  62, id 44217, offset 0, flags [none], length: 84) 10.8.0.47 > 10.8.0.13: icmp 64: echo reply
seq 3
```

Node 3 and 37 continue talking without any strange problem. However, when we enable OLSRD in node 0 in another scenario, this communication stops intermediately because incorrect entries in the routing table. In the case without running OLSRD, node 5 does not update its routing table that ip 10.8.0.47 is the neighbor. So this node still works in normal way. Nevertheless, when OSLSD starts, it causes routing table of node 5 changing and becoming failure.

```
15:15:07.730235 00:bd:ef:9f:01:00 > 00:bd:49:a0:01:00, ethertype IPv4 (0x0800), length 70: IP (tos
0x0, ttl  64, id 58823, offset 0, flags [none], length: 56) 10.8.0.15 > 10.8.0.13: icmp 36: time exceeded
in-transit for IP (tos 0x0, ttl   1, id 59503, offset 0, flags [none], length: 84) 10.8.0.13 > 10.8.0.47: icmp
64: echo reply seq 0
```

This is the reason why the communication is broken. Learning from this problem, our obstacle comes from OLSRD which controls routing table in all nodes. So now we try to poison it in the right way. We don't want to fully mimic node 17 because it is not our first purpose. We decide to fake MAC address of 17 and IP address of 37, and hope that routing path will change. We start the scenario again in two cases with and without OLSRD. Without routing protocol, we have to add static routing entry for operating system to guide sending packets. Try pinging node 5, surprisingly, echo reply messages do not come back. Taking some packets by tcpdump, we realize that node 5 have forwarded them to node 17 because destination address is 10.8.0.47. Now starting OLSRD to poison node's 5 routing table. After a half of minute for routing convergence, we ping node 5 again, and feel happy when it works. Node 3 begins sending packets to node 37. We can capture all packets, they do not still belong to us. MAC addresses of these packets are node 17's ones. It is unbelievable because they must come to us. Looking at routing table of node 5, we meet very strange things that node 5 has two entry as follows:

.

```
10.8.0.46/32      link#4          UC        0       0   tap0
10.8.0.47         10.8.0.27       UGH       0      38   tap0 =>
10.8.0.47/32      link#4          UC        0       0   tap0
10.8.0.48         10.8.0.16       UGH       0       0   tap0
10.8.0.49/32      link#4          UC        0       0   tap0
```

OLSRD calculates that the best path from 5 to 37 is always through 17. Maybe, it is not enough time for OLSRD updating this path. We let some more seconds for OLSRD update this information, and take an effort to communicate to node 3 first to make a routing path. This effort does not work, because of node's 5 routing table. In one of scenarios, we have received the reply messages from node 3 but our ping program cannot process these packets. We analyze all log files from experiments. We find that some time node 3 chooses another path to deliver the packets, through node 14. Or node 37 also choose another path through 17 and 34 like:

```
15:53:52.853792 00:bd:68:9e:01:00 > 00:bd:81:9c:01:00, ethertype IPv4 (0x0800), length 98: IP (tos
0x0, ttl  62, id 14182, offset 0, flags [none], length: 84) 10.8.0.13 > 10.8.0.47: icmp 64: echo request
seq 37
15:53:52.857246 00:bd:81:9c:01:00 > 00:bd:85:95:01:00, ethertype IPv4 (0x0800), length 98: IP (tos
0x0, ttl  61, id 14182, offset 0, flags [none], length: 84) 10.8.0.13 > 10.8.0.47: icmp 64: echo request
seq 37
15:53:52.857292 00:bd:85:95:01:00 > 00:bd:68:9e:01:00, ethertype IPv4 (0x0800), length 98: IP (tos
0x0, ttl  64, id 54343, offset 0, flags [none], length: 84) 10.8.0.47 > 10.8.0.13: icmp 64: echo reply seq
37
15:53:52.857967 00:bd:68:9e:01:00 > 00:bd:ef:9f:01:00, ethertype IPv4 (0x0800), length 98: IP (tos
0x0, ttl  63, id 54343, offset 0, flags [none], length: 84) 10.8.0.47 > 10.8.0.13: icmp 64: echo reply seq
37
```

We also recognize whenever starting communication firstly, victim usually break the session because of routing failure. We believe that if we have a long time enough to expire routing entries in node 5, we can obtain a good result.

In conclusion, stealing a session by mimicking victim in mobile ad hoc network is not easy to succeed in practice. Because of OLSRD, attacker need to infect him self to routing table of intermediate nodes without any incorrect information which could reach to influence to whole network. If the attacker manually controls communication of victim, he could successfully steal their session.

4.4 Discussion

We would like to discuss the works at which our testbed outdoes other ones. Compared to other testbed, as in Table 8, our testbed and SWOON are capable of emulating almost all network layers thus give researchers an opportunity to better reflect the real-world security issues. Our testbed surpasses SWOON at the point that it can support mobility models and even obstacles on a map as the real world. Moreover, our testbed helps create scenarios easily as well as making the work of scenario addition quite simple through the GUI application. As the table shows, our testbed not only possesses advantages from all other testbeds but also supports actual movement behaviors thus provide researchers with a more convenient facility to study security in MANETs.

Table 8: Comparison of different testbeds

	Layer			**Scenario Generating**	**Real mobility**
	2	3	4-7		
QOMET	x	o	o	x	o
TEALab	x	o	o	o	x
SWOON	o	o	o	o	x
MobiEmu	x	o	o	x	x
eBATMAN	o	o	o	o	o

Chapter 5.
Conclusion an Future work

In this thesis, we have presented eBATMAN, a framework which includes a highly realistic testbed along with many other facilities, as an aid to understand security in wireless ad hoc networks. The testbed we propose is flexible in setting configurations, fast and stable in conducting experiments and reliable in producing results. Furthermore, the testbed supports different types of attacks on different layers from data link to the application layer. Our framework also has a GUI application that helps create and add scenarios easier and monitor mobility of nodes in a more intuitive way. Taking all into account, the testbed can be used as a good facility for researchers to study MANET security by deploying real scenarios, real attacks, and real movements. Finally, we have demonstrated our testbed's results with some security-related scenarios.

Performance of the system is what we are going to improve in the near future. Furthermore, we continue improving GUI program and Report Collector to make it friendlier to user. And we intend to use the testbed to understand our lab members' study in MANET.

Acknowledgments

I would like to express my gratitude to all those who supported and gave me opportunities to complete this thesis.

At first, I would like to give a sincere thank to Vietnamese government, and Jaist staff for their financial support in one year in Japan.

Second, I would like to express my gratitude to my supervisor Prof Yoichi Shinoda, who has been giving valuable instructions and directions to me in my study. He taught me many things not only in science but also in life.

Next, I want to give special thanks to Mr Nam, Mr Lan , Mr Razvan, and Prof Nguyen Dinh Thuc who help and encourage me to over with all my obstacles in research and experiments.

To my friends, I want to say “thank you” for all they have done for me.

Finally, I am always indebt to my parents for giving my life and educating me in whole of life

Bibliography

1 R. Beuran, J. Nakata, T. Okada, L. T. Nguyen, Y. Tan, Y. Shinoda. "A Multi-purpose Wireless Network Emulator: QOMET." AINA2008. Okinawa, Japan, March 25-28, 2008.

2 B. Wu, J. Chen, J. Wu, and M. Cardei. "A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks." Wireless/Mobile Network Security. Springer, 2008.

3 Dan Lynch and Scott Knight, Maria A. Gorlatova, Yannick Lacharité, Louise Lamont, Ramiro Liscano, Peter C. Mason. "Providing Effective Security in Mobile Ad hoc Networks without Affecting Bandwidth or Interoperability." Army Science Conference, 26th. December 1-4,2008.

4 Farooq Anjum,Petros Mouchtaris. "Security for wireless ad hoc networks." Wiley-Interscience, 2006.

5 Giovanni Vigna , Sumit Gwalani , Kavitha Srinivasan , Elizabeth M. Belding-Royer , Richard A. Kemmerer. "An Intrusion Detection Tool for AODV-Based Ad hoc Wireless Networks." Proceedings of the 20th Annual Computer Security Applications Conference. December 06-10, 2004. 16-27.

6 Gorlatova, M.A,Mason, P.C.,Wang, M.,Lamont, L, Liscano, R. "Detecting Wormhole Attacks in Mobile Ad Hoc Networks through Protocol Breaking and Packet." Military Communications Conference. IEEE, 2006. 1-7.

7 http://libnet.sourceforge.net/.

8 http://mobiemu.sourceforge.net/.

9 http://sourceforge.net/projects/iperf/.

10 http://tuxmobil.org/manet_linux.html.

11 http://www.aircrack-ng.org.

12 http://www.alobbs.com/macchanger.

13 http://www.hping.org/.

14 http://www.klcconsulting.net/smac/.

15 http://www.l0t3k.org/security/tools/arp/.

16 http://www.netfilter.org/.

17 http://www.olsr.org/.

18 http://www.oxid.it/cain.html.

19 http://www.snort.org.

20 http://www.starbed.org/.

21 http://www.tamos.com/products/commwifi/.

22 http://www.tcpdump.org.

23 J. Hoebeke I. Moerman B. Dhoedt P. Demeester. "An overview of mobile ad hoc networks: applications and challenges." The Journal of The Communications Network,Vol. 3 (2004).

24 Li Y., Wei J.,. "Guidelines on Selecting Intrusion Detection Methods in MANET." ISECON,v21 (2004).

25 Little, M. "TEALab: A Testbed for Ad Hoc Networking Security Research." IEEE Military Communications Conference. Atlantic City, NJ, October 2005.

26 Luc Hogie,Pascal Bouvry,Frédéric Guinand . "An Overview of MANETs Simulation." Electronic Notes in Theoretical Computer Science . Elsevier Science Publishers, 2006 . 81-101 .
27 M. Kropff, T. Krop, M. Hollick, P. S. Mogre, R. Steinmet. "A Survey on Real World and Emulation Testbeds for Mobile Ad hoc Networks." Testbeds and Research Infrastructures for the Development of Networks and Communities. 2006. 6.
28 M.Beck, E.Tews. Practical attacks against WEP and WPA. 2008. <http://dl.aircrack-ng.org/breakingwepandwpa.pdf>.
29 Marko Jahnke,Jens Toelle,Alexander Finkenbrink ,Alexander Wenzel,Elmar Gerhards-Padilla,Nils Aschenbruck,Peter Martini. "Methodologies and Frameworks for Testing IDS in Ad hoc Networks." International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems. Chania, Crete Island, Greece: Proceedings of the 3rd ACM workshop on QoS and security for wireless and mobile networks, 2007.
30 P. Kruus, D. Sterne, et al. "In-Band Wormholes and Countermeasures in OLSR Networks." Proc. SecureComm. Baltimore, MD, August 2006.
31 P. Michiardi and R. Molva,. "Core: A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks,." Communication and Multimedia Security Conference (CMS'02). September, 2002.
32 R. Beuran, L. T. Nguyen, T. Miyachi, J. Nakata, K. Chinen, Y. Tan, Y. Shinoda. "QOMB: A Wireless Network Emulation Testbed." IEEE Global Communications Conference (GLOBECOM 2009). Honolulu, Hawaii, USA, November 30-December 4, 2009.
33 Rizzo, L. Dummynet FreeBSD network emulator. <http://info.iet.unipi.it/~luigi/ip_dummynet/>.
34 S.Buchegger and J.LeBoudec. "Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes-Fairness In Dynamic Ad hoc NeTworks)." Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02). June 2002. 226-336.
35 Sevil Sen, John Andrew, Clark. "Intrusion Detection in Mobile Ad Hoc Networks." Guide to Wireless Ad Hoc Networks. Springer, 2009. 427-454.
36 Sonja Buchegger, Cedric Tissieres, Jean-Yves Le Boudec. "A test-bed for misbehavior detection in mobile ad-hoc networks - how much can watchdogs really do." ic2003. 2003.
37 T.Benzel etal. "Experience with DETER: A Testbed for Security Research,." Tridentcom. IEEE, 2006.
38 Wolfgang Kiess, Martin Mauve. "A survey on real-world implementations of mobile ad-hoc networks." Ad Hoc Netw. April 2007. 324-339.
39 Y. L. Huang, et al. "SWOON: A Testbed for Secure Wireless Overlay Networks." Proc. of the Conf. on Cyber Security Experimentation and Test (CSET'08). San Jose, CA, USA, July 28–August 1, 2008.
40 Y. Zhang, Y.A. Huang, W. Lee. "An extensible environment for evaluating secure MANET." First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm,. 2005.pp
41 Yang, H, Luo, H Y, Ye, F, Lu, S W, Zhang, L. "Security in mobile ad hoc networks: Challenges and solutions." Postprints, Multi-Campus, 02-01-2004.
42 Yongguang Zhang and Wei Li. "An Integrated Environment for Testing Mobile Ad-Hoc Networks." Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02). Lausanne, Switzerland: June 2002. , n.d.

APPRENDIX

System Deployment

1 Requirements

a. OS:

All programs are coded and built on FreeBSD 5.4. Hence, we recommend using FreeBSD 5.4 to prevent unwanted errors. To execute QOMET, do_wireconf needs DYMMMYNET to adapt bandwidth and error rate of each pipe. Therefore, make sure that you had turned it on in OS kernel. Beside, usually by default, FreeBSD does not use ipfw and tun/tap device; so enable them in /etc/rc.conf file or default boot file. For compiling OLSRD and QOMET, they require gmake, automake . Thus, you should install them before compiling all system.

b. QOMET, OLSRD, and SpringOS

- Please read compiling guides that are contained in their source code.
- We use QOMET version 1.6 beta, and OLSRD version 0.6.

c. ACTION_EXECUTOR, BROAD_ENVI, REPORT_COLLECTOR

Libraries are required library for these programs:

- libxml2
- libOftp for REPORT_COLLECTOR
- pthread

d. Tips for compiling packages preparation

There are some issues when you compile the packages. If you meet errors related compiling process, you should see these tips to correct them.

- Installing libxml2: after unzipping the package and installing it, please copy all files in /usr/local/include to /usr/include, and libxml*.* in /usr/local/lib to /usr/lib.
- Enable tun/tap device: tun/tap module is default module in FreeBSD kernel; however, some systems do not load it in start up process. So, to enable it, you change this file boot/defaults/loader.conf like this:

```
if_tap_load="YES"       # Ethernet tunnel software network interface
if_tun_load="YES"       # Tunnel driver (user process ppp)
```

```
ipfw_load="YES"          # Firewall
pf_load="YES"
```

- With new version of OLSRD and QOMET, they are built on newer version of FreeBSD; thus when you meet some compiling errors related to kernel, you should read them and add some library in some header files in /usr/include/ .

2 Source Explanation

- Action_Executor_0.7: this is Action Executor program which calls script files to make actions
- Action_For_Node: this program converts action script xml file to action binary file for each node.
- Report_Collector_v1.1: this is report collector module in each PC to collect all files in /root/logfile folder.
- Broad_Envi_v0.28: this is packet dispatcher which creates experimental virtual network.
- Control_Program : this program manages Broad_Envi, Action_Executor and Olsrd to make sure that these program will be run in order, and be discarded when the experiment stops.
- Action_Script: this folder contains all scripts called by Action_Executor
- Neighbor_Table_v1: this program converts the QOMET binary file to the neighbor binary file which defines which node is this node's neighbor in a point of time.
- Read_Config: this folder contains configuration files for all programs such QOMET, Broad_Envi..
- Client_Scripts: this folder contains script files which are used for making experiments on SpringOS and StarBED

3 Source Codes Installation.

a. Olsrd:

- Please read README file in Olsrd source code.
- Install OLSRD and configure Olsrd.config in order that Olsrd runs on TAP0 .

b. Qomet

- Before compiling Qomet source code, you need to copy our modified do_wireconf.c, wireconf.c and wireconf.h to Qomet-1.6-beta\wireconf folder.
- Deleting do_wireconf and Qomet binary file if they exist.
- Installing Qomet as guidance in README file.
- Copying do_wireconf in wireconf folder to Qomet-1.5-beta folder.

c. SpringOS

- Please read SpringOS user guide.

d. Action Executor, Broad Envi, Report Collector

It is better to compile these programs in root permission and root folder. Before building all source codes, you need create a folder called Read_Config, and copy files in our source files in relevant folder to this. We provide a script file to build all programs. However, the script commands have fixed parameters, you should change the directory name to your directory's name. We also provide a script call "build" in each folder, thus you can use this file to compile the source by calling command: sh build or bash build.

4 Configuration Files

a. Action mapping file: action_mapping.xml

This file describes the mapping from action id to an absolute path of relative action script file.

b. Action descriptive file: action.xml

This file describes actions of each node in a point of times. In a specific time, each node can call many actions mapping to action_id in "duration" amount of time to des_node id.

c. Broadcast environment configuration file: be_config.xml

This describes on which interface broad_envi will run. Experimental interface is virtual interface on which QOMET and other program will stand, while control interface is control network on which packets come in and out the real device.

d. Configuration description: config_files.xml

This file contains absolute path of other configuration files such as setting file for QOMET, neighbor table binary file…

e. Program list: program_list.xml

This file is used by control program to call other program like Broad_Envi, do_action and Olsrd.

f. Report collector description: rc_config.xml

This file describes which interfaces Report Collector will run, and where Qomet setting file and data logs are. And addition, FTP servers are defined in data_server_list.txt file.

g. Qomet scenario file: scenario_file.xml

This is Qomet scenario file which contains node, connection, and other parameters definition in a experiment.

h. Other files

File name	Explain
data_server_list.txt	contains a list of FTP servers which receive log files from nodes. Each server is assigned a unique ID
ip_to_ip.txt	contain a list of pair of ip address. First address is ip of experimental interface, and the other is control interface. The order of this pair is important for the system
setting_file	contains a pair of id and experimental interface
setting_file_rc	contains a pair of id and control interface. This file is used for Report Collector Agent
makes_file_script	contains a list of command to make action binary file and neighbor table binary file for each node
kill_process.sh	is a shell script file called by do_action to kill children processes which called by him

5 Usage

a. Prepare a disk image

In some scenarios, we need more than twenty computers, and we get tired when installing a number of software and packages on each Pc. Fortunately, SpringOS supports us to distribute all software to these Computers. Before assigning to all computers, we need to build an image of one Pc in which we have install all needed software. Please reading SpringOS tutorial to build and swipe out an image in StarBED.

After that, we change ip address of control interface to match with our configuration files, and rebuild programs. SpringOS has a function to do that in batch mode.

b. Distribute disk image to Computers

To distribute an image to all Computers, we recommend that you should do it one by one, because in the worst cases, SpringOS does not work properly; hence it will repeat its process. However, we have written a shell script file to deploy our image to all computers and it bypasses one if it meets a trouble.

c. Run a experiment

To get easy, we write some scripts to run experiments. Following these steps to run an experiment.

- Test controlling program and QOMET on one PC to make them run correctly. If it gets trouble, check again configuration files and path of these programs.
- Restart Computers again to make them fresh. Some issues come out because of existed last configurations.
- Write a SpringOS scenario to run all program. In our experiment, at first we synchronize time between clients and master. Then, we call controlling program and put it in back ground by using & at the end of command. After that, we call do_wireconf. There is a warning for this step. Master and clients have to have the same time to shutdown. If master stop before clients, then clients can not communicate with it and do not stop all programs.
- After finishing an experiment, we should stop all programs by using SpringOS to ensure that no program still runs.
- Now, we start a SpringOS scenario to wake up Report Collector in each node. Then, we run a program call client_master that will send a ticket to the first node. In this step, we have to calculate how big of log files we have in each node and how much time to upload them to server. If server can receive more than 10 connections, we can change number of uploading nodes in the ticket in source code of client_master, and rebuild it.
- Eventually, we call a SpringOS scenario to clean all logging files in all nodes.

We also provide the example scenarios and script files in our source code. Note that, our programs such Broad_Envi and others do not require input parameters because they automatically detect these parameters in system. For this reason, we recommend that the system needs to be configured properly to avoid unpredicted issues.

6 Work with StarBED

StarBED is really impressive system that supports for network experimental evaluations. We conduct all testing scenarios on StarBED automatically by writing some scripts. With them, we save a lot of time to deploy and obtain results of many experiments. Our workings in StarBED are:

1) Install all needed programs on a machine.
2) Build an image of this machine and wipe out this to rest of nodes.
3) Manually configure some parameters on each machine to guarantee that all parameters are correct.
4) Write scripts for SpingOS master and client.
5) Call these scripts to start scenarios. We write some shell script to put required parameters automatically, then we just call these like “ sh run_test.sh”

6) After a scenario finish, we call “sh get_data.sh”, then run client_master program in Report Collector module to signal to all clients to upload reports. In this step, all report will be uploaded to FTP server in their directories on this server.
7) Save all reports to our computer, and call “sh clean_all.sh”. This script will call master to inform to client clean all reports in log_file directory and on FTP server.
8) Do step 5 again with another scenario.