NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ROUTE OPTIMIZATION FOR MOBILE IPV6 USING THE RETURN ROUTABILITY PROCEDURE: TEST BED IMPLEMENTATION AND SECURITY ANALYSIS

by

Ioannis Kandirakis

March 2007

Thesis Advisor: Geoffrey Xie
Second Reader: John Fulp

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2007	Master's Thesis

4. TITLE AND SUBTITLE: Route Optimization for Mobile IPv6 Using the Return Routability Procedure: Test Bed Implementation and Security Analysis	5. FUNDING NUMBERS
6. AUTHOR Ioannis Kandirakis	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Hellenic Navy General Staff Athens, Greece	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT

Mobile IPv6 is an IP-layer mobility protocol that is designed to provide mobility support, allowing an IPv6 node to arbitrarily change its location on the IPv6 Internet and still maintain existing connections by handling the change of addresses at the Internet layer using Mobile IPv6 messages, options, and processes that ensure the correct delivery of data regardless of the mobile node's location. Return Routability is an infrastructureless, lightweight procedure that enables a mobile IPv6 node to request another IPv6 node (maybe unaware of mobility) to test the ownership of its permanent IPv6 address in both its home network and its temporary address in the current IPv6 network; and authorizes a binding procedure by the use of a cryptographic token exchange.

The main objective of this research effort is to build a test bed for investigating the vulnerabilities of the Mobile IPv6 RR procedure. The test bed shall facilitate the enactment and analysis of the effects of specific threats on the hosts and the network. While this thesis is not about discovering new vulnerabilities or evaluating countermeasures, the resulting test bed and software shall lay the necessary groundwork for future research in those directions.

14. SUBJECT TERMS Mobile IPv6, Return Routability Procedure, Test Bed, Security, MIPL 2.0.2,SUSE LINUX 10.1	15. NUMBER OF PAGES 121
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ROUTE OPTIMIZATION FOR MOBILE IPV6 USING THE RETURN ROUTABILITY PROCEDURE: TEST BED IMPLEMENTATION AND SECURITY ANALYSIS

Ioannis Kandirakis
Lieutenant, Hellenic Navy
B.S., Hellenic Naval Academy, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2007

Author: Ioannis Kandirakis

Approved by: Geoffrey Xie
Thesis Advisor

John Fulp
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Mobile IPv6 is an IP-layer mobility protocol that is designed to provide mobility support, allowing an IPv6 node to arbitrarily change its location on the IPv6 Internet and still maintain existing connections by handling the change of addresses at the Internet layer using Mobile IPv6 messages, options, and processes that ensure the correct delivery of data regardless of the mobile node's location. Return Routability is an infrastructureless, lightweight procedure that enables a mobile IPv6 node to request another IPv6 node (maybe unaware of mobility) to test the ownership of its permanent IPv6 address in both its home network and its temporary address in the current IPv6 network; and authorizes a binding procedure by the use of a cryptographic token exchange.

The main objective of this research effort is to build a test bed for investigating the vulnerabilities of the Mobile IPv6 RR procedure. The test bed shall facilitate the enactment and analysis of the effects of specific threats on the hosts and the network. While this thesis is not about discovering new vulnerabilities or evaluating countermeasures, the resulting test bed and software shall lay the necessary groundwork for future research in those directions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

There are lots of people I would like to thank for a huge variety of reasons.

I would like to thank the Hellenic Navy for providing the opportunity to pursue my studies at the Naval Postgraduate School.

I am deeply indebted to my advisors Prof. Geoffrey Xie and Prof. John Fulp for their mentoring, inspiration and support throughout this work. Without their common-sense, knowledge and perceptiveness I would never have finished.

I would also like to thank all the rest of the Academic Staff of the Naval Postgraduate School and especially the Department of Computer Science for the knowledge that they provided me with a high sense of responsibility.

The greatest acknowledgement I reserve for my family, my wife Filio and my son Filippos, who endured this long process with me, always offering support, love and patience.

I dedicate this thesis to my beloved father whom I lost during my studies in NPS and he will never have the chance to see my diploma.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Mobile IPv6 (MIPv6) is a network layer protocol for enabling mobility in IPv6 networks. IP mobility technology has gained a significant amount of traction over the last few years, mainly due to the following factors [Soliman04]:

- Increasing dependence of society on information and the need to access it from any place and any time
- Wide spread deployment of high-speed wireless networks.
- Emergence of 3G wireless networks that support packet data services.
- Affordable mobile devices that are multifunctional and capable of services that go beyond just voice and SMS
- Inclusion of IP stacks in PDAs, mobile phones and portable PCs.

Mobile IPv6 is the network layer protocol developed to replace mobile IPv4. IPv6 has a larger address space and is expected to improve network performance and network security over that of IPv4. The intended improvements include both enhancements of existing IPv4 functionalities and new features. Most of the former category of improvements have been tested and analyzed during the operational period of IPv4; the new features; however, have not been equally tested. Some of them still have not been incorporated into popular operating systems, and some exist only as RFC specifications, with no actual implementation.

One of the new features of Mobile IPv6 is the Return Routability (RR) procedure, an infrastructureless solution to achieve Routing Optimization and avoid routing triangles.

This procedure is the subject of serious discussions concerning its security implications. Several problems have been identified, and solutions have been proposed [Johnson04]. A systematic implementation and analysis in a laboratory environment of the potential threats to the hosts and the network, during the execution of the RR process, will help in the evaluation of the proposed solutions and in research for new ones. Such analysis, together with the tools to gather the data to support that analysis, is the focus of this thesis.

A. OBJECTIVE

The main objective of this research effort is to build a test bed for investigating the vulnerabilities of the Mobile IPv6 RR procedure. The test bed shall facilitate the enactment and analysis of the effects of specific threats on the hosts and the network. The threats shall be implemented in software and validated using the test bed. While this thesis is not about discovering new vulnerabilities or evaluating countermeasures, the resulting test bed and software shall lay the necessary groundwork for future research in those directions. Thus, the following tasks will be accomplished.

1. Identify known security issues with the proposed Mobile IPv6 RR procedure.
2. Configure a suite of hardware components to investigate the susceptibility of the autoconfiguration protocol to the selected risks.
3. Implement attacks against the test bed and assess the performance of the protocol in the presence of malicious activity.

B. RESEARCH QUESTIONS

This thesis investigates the following specific issues.

1. Are there any OSs that support the proposed security functions of MIPv6, and if there are, to which extent?
2. How do the components of the MIPv6 secure their communication?
3. What are the possible threats to secure communication between the mobile IPv6 nodes?
4. What are the suggested solutions?
5. Are there any known exploits of the vulnerabilities of the MIPv6?
6. Are there any proposed threat mitigation solutions, and if so, what are they?

C. ORGANIZATION

This thesis is organized as follows.

Chapter I provides and introduction to the thesis and the rudiments of the Mobile IPv6 protocol.

Chapter II presents the need for transition to IPv6, the benefits of IP Mobility, and provides an overview of the Mobile IPv6 protocol. In addition, it describes the RR procedure and the assumptions made for the design implementation of the MIPv6 protocol. It is intended to be a high-level description that will introduce the MIPv6 terminology and help the reader comprehend how the MIPv6 protocol and its RR procedure work.

Chapter III presents the layout and configuration process of the implemented MIPv6 test bed. A very limited number of published host operating systems are advertised to have support for MIPv6. Worse, all the available MIPv6 capable OS releases are experimental in nature and still going through rigorous validation tests. As such, a

significant amount of effort for this thesis was spent determining a working combination of OS version and MIPv6 extension in a “trial and error” manner. The experience is documented in this chapter. This chapter is intended to provide sufficient details so that it can be used as a “how-to” guide for deploying a MIPv6 test bed using open-source software.

An evaluation of the RR procedure is provided in Chapter IV that is based on the experimental results from both Chapter III and the research of [Aura06], which was published during this research.

Conclusions and recommendations for threat mitigation are presented in the final chapter, along with suggestions for future work on the analysis and evaluation of the proposed solutions.

II. BACKGROUND

This chapter briefly presents the argument for transition to IPv6, the rudiments of the IP Mobility protocol, and the Mobile IPv6 protocol in particular. Finally, it describes the RR procedure and the assumptions made for the design implementation of the MIPv6 protocol. It is intended to be a high-level description that will introduce the Mobile IPv6 terminology (MIPv6) and help the reader comprehend how the MIPv6 protocol and its attendant RR procedure work.

A. THE NEED FOR TRANSITION TO IPV6

The **Internet Protocol** (**IP**) is a data-oriented protocol used for communicating data across a packet-switched internetwork. IP is a network layer protocol in the internet protocol suite and is encapsulated in a data link layer protocol (e.g., Ethernet). As a lower layer protocol, IP provides the service of communicable unique global addressing amongst computers. This implies that the data link layer need not provide this service. Ethernet provides globally unique addresses except it is not globally communicable (i.e., two arbitrarily chosen Ethernet devices will only be able to communicate if they are on the same bus). The difference is that IP is concerned with the final destination of data packets. Ethernet is concerned with only the *next* device (computer, router, etc.) in the chain. The final destination and next device could be one and the same (if they are on the same bus), but the final destination could be on the other side of the world [http://en.wikipedia.org/wiki/Internet_Protocol Last visited on February 2, 2007].

The current version of the IP protocol (IPv4) has not changed a lot since RFC 791, which was published in 1981. It is common belief that IPv4 served us well for over 25 years and still does.

However, the initial design of IPv4 did not anticipate contemporary issues such as [Soliman04]:

- The exponential growth of the Internet and the impending exhaustion of the IPv4 address space.
- The need for simpler and more automatic configuration of addresses and other settings that do not necessarily rely on the administration of a DHCP infrastructure.
- The requirement for security at the IP level.
- The need for better support for real-time delivery of data.
- The emergence of IP-capable mobile devices.
- The need of society to access information from any place and at any time.

To address not only these, but also many proposed methods for improving IPv4, the IETF has developed a suite of protocols and standards known as IP version 6 (IPv6) with the following features [Davies02]:

- New header format
- Larger address space
- Efficient and hierarchical addressing and routing infrastructure
- Stateless and stateful address auto configuration
- Built-in security
- Better support for quality of service (QoS)
- A new protocol for neighboring node interaction
- Extensibility

In addition, the internals of the IPv6 protocol have been designed with scalability and extensibility in mind.

This will allow many different kinds of devices besides PCs, like cell phones and home appliances, to more easily join the Internet in future [http://wireless.about.com/od/networkprotocolsip/g/bldef_ipv6.htm Last visited on February 5, 2007].

B. IP MOBILITY

IP Mobility is defined as the change in a node's IP address due to the following reasons:

- Change of its attachment point within the Internet topology.
- Change in the topology itself, which causes a node to change its address.

Mobility is considered to be an important issue, and the need for an IP mobility management solution is motivated by the following [Soliman04]:

- Users would like to have the choice of using certain technologies over others.
- Hosts need to be reachable independently of their "normal" (home) physical origin.

Mobile IPv6 is designed to handle the mobility management on the IP layer for the emerging IPv6 protocol.

The solution to IP mobility is the Mobile IP protocol, designed to allow mobile device users to move from one network to another while maintaining reachability via their permanent/home IP address. Defined in RFC 2002, Mobile IP is an enhancement of the Internet Protocol (IP) that adds mechanisms for forwarding Internet traffic to mobile devices (known as mobile nodes) when they are connecting through other than their home network.

[http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci849848,00.html Last visited on February 5, 2007]

C. MOBILE IPV6 TERMINOLOGY

In order for the reader to better understand the description of the MIPv6 protocol and the RR procedure, its concomitant terminology and message transactions are presented in this section:

Home link: The home link is the link that is assigned the home subnet prefix. The mobile node uses the home subnet prefix to create a home address.

Home Address (HA): A unicast routable address assigned to a mobile node, used as the permanent address of the mobile node. This address is within the mobile node's home link. Standard IP routing mechanisms will deliver packets destined for a mobile node's home address to its home link. Mobile nodes can have multiple home addresses, for instance when there are multiple home prefixes on the home link.

Home Agent (HA): The home agent is a router on the home link that maintains an awareness of the mobile nodes of its home link that are away from home and the addresses that they are currently using. If a mobile node is on the home link, the home agent acts as a normal IPv6 router, forwarding packets addressed to the mobile node. If the mobile node is away from home, the home agent tunnels data sent to the mobile node's home address to the mobile node's current (remote) location on the IPv6 Internet.

Mobile node (MN): A mobile node is an IPv6 node that can change links/networks, and therefore addresses, and yet continue to maintain reachability using its home address. A mobile node has “awareness” of its home address and the

global address of its current link address, and indicates its home address to the home agent and IPv6 nodes with which it is communicating.

Foreign link: A foreign link is a link that is not the mobile node's home link. A foreign link is assigned a foreign subnet prefix.

Care-of Address (CoA): the temporary, network-specific IP address for routing messages to the mobile node’s current location. The association of a care-of address with a home address for a mobile node is known as a binding. Correspondent nodes and home agents keep information on bindings in a binding cache.

Correspondent Node (CN): A correspondent node is an IPv6 node that is capable of communicating with a mobile node while it is away from home. A CN can also be a mobile node.

Cookie: random number used by a mobile node used to prevent spoofing by a bogus CN in the RR procedure.

Care-of init cookie: a cookie sent to the CN in the Care-of Test Init message, to be returned in the Care-of Test message.

Home init cookie: a cookie sent to the CN in the Home Test Init message, to be returned in the Home Test message.

Keygen Token: a number supplied by a CN in the RR procedure to enable the MN to compute the necessary binding management key for authorizing a BU.

Nonces: random numbers used internally by the CN in the creation of keygen tokens related to the RR procedure.

Binding management key (Kbm): Key used for authorizing a binding cache management message (e.g., BU and BACK messages).

Binding Update (BU): Used by a mobile node to notify other nodes of a new care-of address. It can also be used to delete old bindings.

Binding Acknowledgement (BA): Used to acknowledge receipt of a Binding Update.

Binding Refresh Request (BRR): Used by the CN to inform the mobile node that the binding is (or is going) stale.

Binding Error (BE): It is used by the CN to signal an error.

D. MOBILE IPV6

Mobile IPv6 grew out of experiences with Mobile IPv4; itself an attempt to enable IP attached devices to migrate between physical networks without having to change the publicly visible IP address by which they were uniquely known to the rest of the Internet.

When a node moves from one access network to another or switches between access technologies, it acquires a new IPv6 address and cannot be reached directly via its old IPv6 address due to its router’s ingress filtering. This implies that all current communications (for example streaming video from the Internet or a TCP session) are stopped and will have to be restarted by the user or the application.

The Mobile IPv6 protocol (RFC 3775) has been defined to address those issues and allow the node to be always reachable at the same IPv6 address whatever the access network it uses. It allows the host to move transparently for the applications and the users, without the need to reset all the current connections each time the host moves to another access network.

Its design aims to solve two problems:

- To allow transport layer sessions (TCP connections and UDP-based transactions) to continue even if the host(s) move and change their IP addresses.
- To allow a node to be reached through a static IP address; that is, a home (of) address (HoA).

E. BASIC MOBILE IPV6 PROCESS-TUNNELING MODE

The basic idea in Mobile IPv6 is to allow a home agent (HA) to work as a stationary proxy for a mobile node (MN). Whenever the mobile node is away from its home network, the home agent intercepts packets destined to the node and forwards the packets by tunneling them to the node's current address, the care-of address (CoA). The transport layer (e.g., TCP, UDP) uses the home address as a stationary identifier for the mobile node.

With Mobile IPv6, a host has two addresses while moving in the Internet topology: one permanent address that identifies the host, and the other representing the location in the Internet topology. The Mobile IPv6 protocol takes care of the binding between these two addresses (thanks to a Home Agent), and ensures that the host is always reachable at its permanent address even if it moves in the Internet topology.

Mobile IPv6 adopts a new strategy for securing a MN that roams around the Internet. A MN needs to keep getting new local IP addresses (CoA) and keep his HA informed that he's moved and where he has gone.

There are two possible modes for communications between the mobile node and a CN in MIPv6. The first mode, bidirectional tunneling, does not require Mobile IPv6 support from the CN and is available even if the mobile node has not registered its current binding with the CN. Packets from the CN are routed to the home agent and then tunneled to the mobile node. Packets to the CN are tunneled from the mobile node to the home agent ("reverse tunneled") and then routed normally from the home network to the CN. The roaming device is authenticated through its home address, and all communications to that device pass through the home address before being sent to the temporary location (CoA).

Bidirectional tunneling is responsible for triangle routing. Triangle routing may incur unnecessary latency, which is not desirable for real time traffic such as VoIP. Also it impacts on reliability since a longer data path is more likely to break due to a link failure.

In a nutshell, the bidirectional tunneling is described by the following steps:

1. The MN uses its HoA when it is in its home network. A datagram sent from CN to MN, will be sent to MN’s HA.
2. HA delivers the datagram to MN at its HoA.
3. MN moves to a visiting network and acquires a temporary IP address, CoA from the agent (local router) of the visiting network.
4. The MN registers its CoA to its HA.

5. The CN sends a datagram to the MN, unaware if it is in its home network, to the only address that it can reach the MN, its HoA.
6. The HA forwards the datagram to MN, at its CoA.
7. The MN sends datagrams to CN, tunneling them through its HA due to ingress filtering.

The above procedure is illustrated in Figure 1 below.

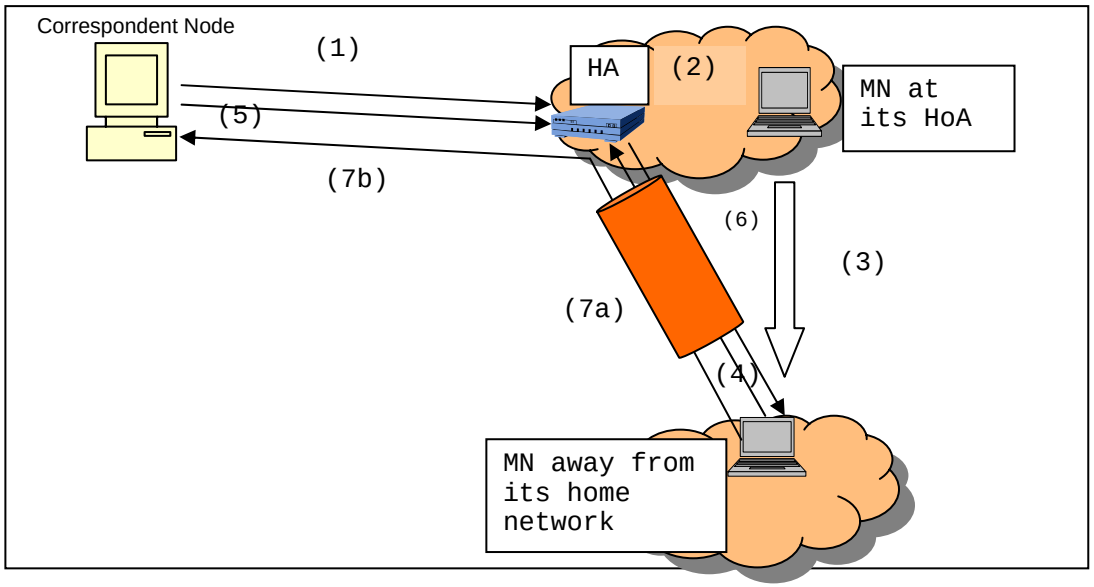

Figure 1. Bidirectional Tunneling of Mobile IPv6

This is the basic mode of function of Mobile IPv6 in absence of any optimization and is called triangle routing because every message between MN and CN has to route via the MN's Home Agent.

Triangle routing may create delays, caused by a long trip time that affects real time traffic such as VoIP. Also, it impacts on reliability since the longer path may have broken links.

Route optimization is an optional feature of Mobile IPv6 that eliminates triangle routing. It is a mode of operation that allows the mobile node and its peer, a CN, to exchange packets directly, bypassing the home agent completely after the initial setup phase.

When route optimization is used, the mobile node sends its current care-of address to the CN, using binding update (BU) messages. The CN stores the binding between the home address and care-of address into its Binding Cache. One way to achieve route optimization is the implementation of the RR procedure, an infrastructureless solution in which the MN requests the CN to test its ownership of the HoA and CoA and authorizes a binding procedure by the use of a cryptographic token exchange.

F. OVERVIEW OF RETURN ROUTABILITY (RR) PROCEDURE

Mobile IPv6 Route Optimization verifies a mobile node's authenticity through a routing property. H. Soliman in Chapter 5 of his book, [Soliman04], describes the Return Routability (RR) procedure with great detail. The essence of the RR procedure is that the MN requests that the CN test its ownership of its HoA and CoA. This is done by sending two independent messages: the Home address Test Init (HOTI) and Care-Of address Test Init (COTI). The CN creates two tokens that only the CN can create (encrypt with a secret key Kcn that is known only to CN) and sends one token to each address (home and care-of addresses) in two separate messages: HOme Test (HOT) and Care-Of Test (COT).

The mobile node uses both of these tokens to create a key (Kbm) that can be used to authenticate a binding update message to the CN. Since the CN knows all the information needed to produce the key, it can reproduce it when the binding update is received, and so authenticate the message. The same key is used to authenticate the binding acknowledgment.

The HOTI message is sent by the mobile node to request a test of the home address. The source address used in the IPv6 header is the mobile node’s home address and the destination is the CN’s address. Hence, this message has to be tunneled to the home agent (since the home address is not topologically correct in the visited network), which decapsulates the message and forwards it to the CN. The HOTI message is transported inside a mobility header type 1. This message contains a cookie (called home init cookie) generated by the mobile node and later returned by the CN. The cookie is a random number that has no significance; it is included to ensure that the entity responding to the HOTI message has actually received it. This message is protected on the mobile node–home agent path by ESP in tunnel mode.

The home agent verifies the ESP header and forwards the internal message to the CN. In this case the home agent is not provided with a home address option in the outer header (unlike the binding update message) to use in order to locate the right security association in the SAD. In this scenario, the home agent’s SPD is configured to treat the mobile node’s care-of address as a security gateway address. The implication of this configuration is that the home agent can associate a security association entry in

the SAD with a specific tunnel interface, identified by the mobile node’s care-of address. Hence, the home agent will be able to identify the security association based on the interface from which it was received. This message (and the HOT message) is treated differently by not including the home address option. The reason is that the binding update is sent before establishing the tunnel. Therefore, no tunnel interface can be used to identify the security association.

Almost simultaneously, the mobile node can send a COTI message. The COTI message is sent from the mobile node’s care-of address directly to the CN. It is transported in a mobility header type 2. The message contains another random cookie (called care-of init cookie). The COTI cookie is a random number used to ensure that the responder to a COTI message has actually received the original (COTI) message.

When the CN receives the HOTI message, it generates a 64-bit home keygen token (the token generated is based on the home address). The home keygen token is generated by taking the first 64 bits of the output of a message authentication code function using Kcn and is then computed on the concatenation of the home address and a nonce generated by the CN as follows:

Home keygen token = First (64, HMAC_SHA1(Kcn, home address|nonce|0))

where First(n, j) represents the first n bits in j. HMAC_SHA1(Kcn, info) means a hashed message authentication code (or a keyed hash) based on the SHA1 hash algorithm and uses Kcn to key the function, which operates on info. The 0 is used to distinguish the home keygen token from the care-of keygen token, shown later.

The CN then constructs a HOT message and sends it to the mobile node. This message contains the home init cookie originally sent by the mobile node and the home keygen token. Since the CN generates nonces frequently, it needs to be aware of the nonce used to generate a particular cookie. Nonces are stored in an indexed list. Therefore, a CN only needs to know the index corresponding to a particular nonce to be able to generate the home keygen token again. The nonce index is included in the HOT message. This will be needed later by the CN to authenticate the binding update.

The message will be intercepted by the home agent and tunneled to the mobile node’s care-of address. A secure tunnel (ESP) is used to forward this message to the mobile node.

A similar operation is done when the CN receives the COTI message. It generates a care-of keygen token, where

Care-of keygen token = First(64, MAC (Kcn, care-of address |nonce|1))

The nonce used in this operation might not be the same nonce used to create a home keygen token, depending on when the COTI message was received (the CN might have generated a new nonce). Therefore, the nonce index should be sent to the mobile node in the COT message.

This message concludes the RR procedure. At this point, the CN has not yet stored any more information than it had at the beginning of this procedure: Kcn and an indexed list of nonces. The CN stores neither the home keygen token nor the care-of keygen token. When needed, these tokens can be regenerated, given the nonce indices originally used to generate them.

After receiving the HOT (tunneled from the home agent) and the COT message, the mobile node is in a position to generate a binding management key, Kbm. This is done as follows:

Kbm = SHA1 (home keygen token|care-of keygen token)

The mobile node can now construct the mobility header used for the binding update message. The mobility header includes the binding update, a nonce indices option, and a binding authorization data option. The nonce indices option contains the two indices received in the HOT and COT messages.

The authentication data are calculated as follows:

Auth_data = First (96, MAC(Kbm, Mobility_data)

where

Mobility_data = care-of address| final dst| Mobility header data

The mobility header data includes the content of the mobility header with the exception of the authorization data option itself. The final destination is the packet’s final destination, that is, the CN’s address. If the CN were also a mobile node, a routing header type 2 (containing its home address) would be included in the packet. Since the routing header is processed before the mobility header, the final dst field should contain that CN’s home address.

Since the CN does not keep state for any mobile nodes during the RR procedure, the mobile node needs to include its home and care-of addresses in the binding update. The home address is included in a home address option (in a

destination options extension header), which precedes the mobility header. If the care-of address were different from the packet's source address, it should be included in the alternate-care-of address option; otherwise, the packet's source address is assumed to be the care-of address. In any case, the care-of address should always be the one used in the source address field of the COTI message; otherwise, the wrong care-of keygen token will be used to generate Kbm when the binding update is received at the CN.

After the binding update message is constructed, the mobile node sends it to the CN.

When the CN receives the binding update, it looks into the nonce indices option and finds the corresponding nonces. The CN will be able to regenerate Kbm as follows:

1. Generate home keygen token: First (64, MAC (Kcn, home address| nonce|0)). The home address is taken from the home address option.
2. Generate care-of keygen token: First (64, MAC (Kcn, care-of address| nonce|1)). The care-of address is taken from the alternate care- of address option when present; otherwise, the source address is used.
3. Generate Kbm: Hash (home keygen token|care-of keygen token).
4. Calculate Auth_data: First (96, MAC(Kbm, Mobility_data).
5. If Auth_data is equal to the content of the binding authorization data option, accept the binding update.

If an acknowledgment is requested, the CN must send a binding acknowledgment. The binding acknowledgment should also contain the binding authorization data option.

The binding refresh advice option informs the mobile node about the time when a new binding update is needed.

The advantage of the RR procedure is that it is lightweight and does not require pre-shared authentication material. It also requires no state at the CN. On the other hand, the two reachability tests can lead to a handoff delay unacceptable for many real time or interactive applications such as Voice over IP (VoIP) and video conferencing. Also, the security that the Return-Routability procedure guarantees might not be sufficient for security-sensitive applications. And finally, periodically refreshing a registration at a CN implies a hidden signaling overhead that may prevent mobile nodes from hibernation during times of inactivity [Arkko06].

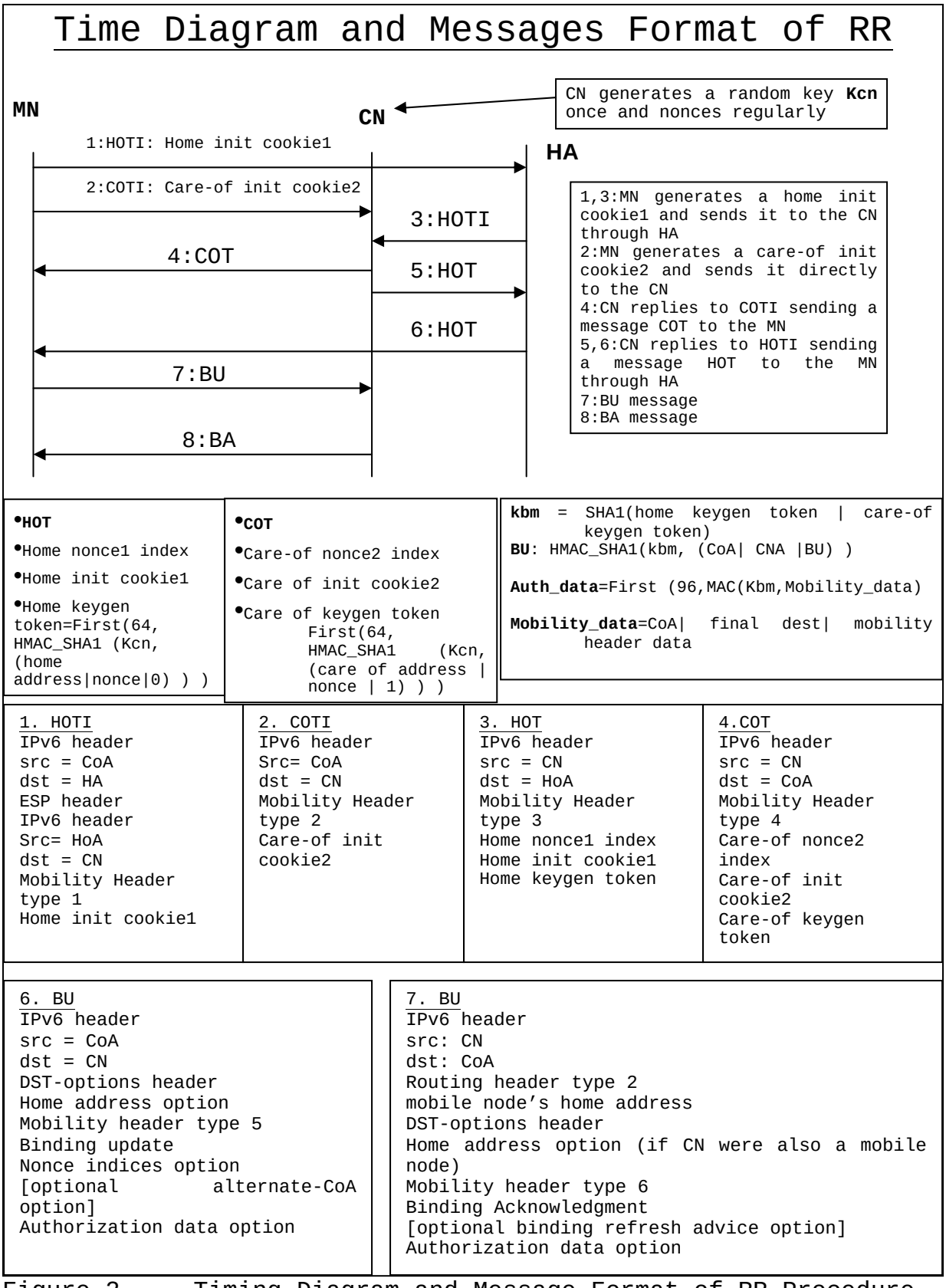

Figure 2. Timing Diagram and Message Format of RR Procedure

G. PRIOR EVALUATIONS OF MIPV6 PROTOCOL

One important base assumption is that the routing prefixes available to a node are determined by its current location, and therefore the node must change its IP address as it moves. In current IPv6 operational practice the IP address prefixes are distributed in a hierarchical manner. This limits the number of routing table entries each individual router needs to handle. An important implication is that the topology determines what globally routable IP addresses are available at a given location. That is, the nodes cannot freely decide what globally routable IP address to use; they must rely on the routing prefixes served by the local routers via Router Advertisements or by a DHCP server. In other words, IP addresses are just what the name says, addresses (i.e., locators) [Nikander05].

Furthermore, in the current Internet structure, the routers collectively maintain a distributed database of the network topology and forward each packet towards the location determined by the destination address carried in the packet. To maintain the topology information, the routers must trust each other, at least to a certain extent. The routers learn the topology information from the other routers, and they have no option but to trust their neighbor routers about distant topology. At the borders of administrative domains, policy rules are used to limit the amount of—perhaps faulty—routing table information received from the peer domains. While this is mostly used to weed out administrative mistakes, it also helps with security. The aim is to maintain a reasonably accurate idea of the network topology even if someone is feeding faulty information to the routing system [Nikander05].

In the Mobile IPv6 security design, different approaches were chosen for securing the communication between the mobile node and its home agent and between the mobile node and its CNs. In the home agent case, it was assumed that the mobile node and the home agent know each other through a prior arrangement, such as a business relationship. In contrast, it was strictly assumed that the mobile node and the CN do not need to have any prior arrangement, thereby allowing Mobile IPv6 to function in a scalable manner without requiring any configuration at the CNs [Nikander05].

The Return-Routability procedure was designed with the objective of providing a level of security that compares to that of today's non-mobile Internet. As such, it protects against impersonation, denial of service, and redirection-based flooding attacks that would not be possible without Route Optimization. This approach is based on an assumption that a mobile Internet cannot become any safer than the non-mobile Internet [Nikander05].

The goal of the current Mobile IPv6 route optimization security has been to produce a design with a level of security close to that of a static IPv4-based Internet, and with an acceptable cost in terms of packets, delay, and processing. The result is not what one would expect. It is definitely not a traditional cryptographic protocol. Instead, the result relies heavily on the assumption of an uncorrupted routing infrastructure and builds upon the idea of checking that an alleged mobile node is indeed reachable through both its home address and its care-of address. Furthermore, the lifetime of the state created at the

corresponded nodes is deliberately restricted to a few minutes, in order to limit the potential threat from time shifting [Nikander05].

Moreover, given the typically limited bandwidth in a wireless medium, resources ought to be spent in an economic matter. This is especially important for the amount of signaling that a mobility protocol requires [Arkko06].

Additionally, applications that require a security level higher than what the Return-Routability procedure can provide are generally advised to use end-to-end protection such as IPsec or Transport Layer Security (TLS) [Arkko06].

RR protects certain signaling messages, exchanged between a mobile node and its home agent, through an authenticated and encrypted tunnel. This prevents unauthorized nodes on that path, including eavesdroppers in the mobile node's wireless access network, from listening in on these messages [Soliman04].

Given that a pre-existing end-to-end security relationship between the mobile node and the CN cannot generally be assumed, this protection exists only for the mobile node's side. If the CN is immobile, the path between the home agent and the CN remains unprotected. This is a path between two stationary nodes, so all types of attacks that a villain could wage on this path are already possible in the non-mobile Internet. In case the CN is mobile, it has its own home agent, and only the path between the two (stationary) home agents remains unprotected [Arkko06].

RFC 3775 fails to conceal a mobile node's current position as route-optimized packets always carry both home and care-of addresses. Both the CN and a third party can

therefore track the mobile node's whereabouts. A workaround is to fall back to bidirectional tunneling where location privacy is needed. Packets carrying the mobile node's care-of address are thus only transferred between the mobile node and the home agent, where they can be encrypted through IPsec ESP. But even then, the mobile node should periodically re-establish its IPsec security associations so as to become untraceable through its SPIs [Arkko06].

The RR procedure implicitly assumes that the routing infrastructure is secure and trusted. Thus, it is appropriate to design a protocol to secure the binding update as long as it is no less secure than the underlying routing infrastructure. In other words, if a packet is sent to a particular destination, the routing system delivers it to that destination. If an attacker compromises the routing infrastructure and manages to control one or more routers, several serious attacks can be launched independently of RR procedures [Soliman04].

The RR procedure protects Binding Updates against all attackers who are unable to monitor the path between the home agent and the CN. The procedure does not defend against attackers who can monitor this path [Aura06].

Another assumption made by RR is that it is difficult for an attacker to be located on two different paths at the same time and receive both tokens needed to generate Kbm. This could happen if an attacker is sharing a link with the CN; he would be able to see all of the RR packets, construct a binding update message, send it to the CN, and receive all of the CN’s traffic addressed to the mobile node. However, an attacker does not need to go through all this trouble to hijack the CN’s connections with the mobile

node if he shares a link with the CN; he can simply pretend to be a router by stealing the default router's link-layer address and sending a fake router advertisement to the CN. Alternatively, he can send a Neighbor Discovery redirect message to the CN requesting that all its traffic be sent to his link-layer address. Thus, an attacker sharing a link with the CN can cause serious harm without Mobile IPv6; that is, Neighbor Discovery messages are the weakest link when an attacker is sharing a link with the CN [Nikander05].

Since the main goal of the RR procedure is to ensure that securing route optimization does not make things worse than they are in today's Internet, the above case can be ignored. However, it is worth noting that this type of attack will become significant as soon as a mechanism is devised to secure Neighbor Discovery messages. When this happens, the RR procedure will become the weakest link [Soliman04].

An attacker can be located on the mobile node–CN path. In this location, he would only be able to see the care-of keygen token, which would not allow him to construct Kbm correctly to steal the mobile node's traffic.

The attacker might also send a large number of HOTI and COTI messages to try to consume the CN's resources in a way that makes it unable to process legitimate requests from real mobile nodes. The RR procedure is designed to allow CNs to be protected from memory-exhaustion attacks; a CN would only keep state when it receives an authenticated binding update from a mobile node. Clearly, this procedure cannot protect against an attacker aiming at using up the CN's link bandwidth by sending a very large number of

HOTI/COTI messages. However, this attack can be launched without RR by simply sending a large number of bogus messages. It is worth noting though, that the CN can simply decide to not receive any HOTI/COTI messages if it detects that it is being attacked. That is, the CN can “turn off” route optimization; communication with mobile nodes will still take place through the home agent [Soliman04].

Moreover, it is assumed that CN is able to implement the RR algorithm and maintain a cache of MNs.

One of the most important advantages of the RR procedure is that it does not require any manual configuration or infrastructure support. This feature assists with the quick deployment of Mobile IPv6 and encourages vendors to support route optimization, which would have been much harder if route optimization came with the burden of infrastructure support or the unrealistic assumption of manual configuration. However, it is important to note that this comes at the cost of having weak authentication compared to the more traditional applications of public key cryptography [Arkko06].

THIS PAGE INTENTIONALLY LEFT BLANK

III. MIPV6 TEST BED CONFIGURATION

This chapter presents the layout and configuration process of the implemented MIPv6 test bed. A very limited number of published host operating systems are advertised to have support for MIPv6. Worse, all the available MIPv6 capable OS releases are experimental in nature and yet going through rigorous validation tests. As such a significant amount of effort for this thesis was spent determining a working combination of OS version and MIPv6 extension, in a “trial and error” manner. The experience is documented in this chapter.

This chapter is intended to provide sufficient details so that it can be used as a “how-to” guide for deploying a MIPv6 test bed using open-source software.

A. PUBLISHED IMPLEMENTATIONS OF MIPV6

The most known implementations of MIPv6 are: “MIPL” (Mobile IPv6 for Linux [http://www.mipl.mediapoli.com/ Last visited on January 10, 2007]), “KAME” project (Mobile IPv6 for BSD based Oss [http://www.kame.net Last visited on January 11, 2007]) and “USAGI” (Mobile IPv6 for Linux based Oss [http://www.linux-ipv6.org/ Last visited on February 8, 2007]).

Mobile IPv6 for Linux (MIPL) is an implementation that was originally developed as part of a software project course in the Helsinki University of Technology (HUT), with the goal to create a prototype implementation of Mobile IPv6 for Linux. After the course, the implementation was further developed in the context of the GO/Core project at HUT Telecommunications and Multimedia Lab. It is an open

source implementation, released under the GNU GPL license and freely available to anyone(http://www.mobile-ipv6.org/software/). The MIPL implementation has been tested in interoperability and conformance testing events such as the ETSI IPv6 Plugtests and TAHI Interoperability events.

The "KAME" and "USAGI", projects are working on research and development on the implementation of the IPv6 and IPsec protocols, which operates on BSD based OSs for the "KAME" project and on a Linux based OS for the "USAGI" project. Accuracy of the implementation is now widely accepted and is being incorporated into BSD based OSs (FreeBSD, NetBSD, OpenBSD and BSD/OS) and Linux version 2.6 for the provision of an environment enabling the easy use of IPv6 to a large number of users [http://www.wide.ad.jp/about/research.html Last visited on January 15, 2007].

The “KAME” project was a joint effort of six companies in Japan to provide a free suite of IPv6, IPsec, and Mobile IPv6 protocols for BSD variants. Particularly, a mobile IPv6 implementation for the FreeBSD and NetBSD platforms has been developed under this project. The code is implemented as part of the kernel. In addition, several user space programs have been developed for MIPv6 control, for extracting MIPv6 statistics and for dynamic home agent discovery. The implementation follows RFC 3775 and includes functionality for HA, MN and CN (mandatory for an IPv6 implementation that claims to be IPv6 compliant). It also supports authentication of messages between a MN and its HA using IPsec [M. Dunmore, “Final MIPv6 Support Guide,”

February 8 2005, 6net, [http://www.6net.org/publications/deliverables/D4.1.4.pdf Last visited on January 15, 2007].

The “USAGI” Project (UniverSAl playGround for IPv6 Project) aims to provide a better IPv6 environment for Linux in conjunction with the WIDE, KAME, and TAHI projects. It includes Linux kernel extensions, IPv6 related libraries, and IPv6 applications.

The “TAHI” project [http://www.tahi.org/ Last visited on February 22, 2007] is aiming at providing a means of high-level verification of these technologies.

B. CHOOSING MIPV6 SOFTWARE

In the beginning, the FreeBSD OS developed by the KAME project was chosen for the MIPv6 test bed. The main reason for this choice was that all MIPv6 functionality was included in the OS kernel and no patch was required. Following the instructions for a similar project based on the 4.9 Version of FreebSD [Lawrence04] and using the current version (6.2) as well as the detailed instructions of [Blanchet06] it was made an attempt to configure and build a MIPv6 test bed. However, this attempt was unsuccessful. During my research, there were contradictory information about the compatibility and functionality of the current version with the Mobile IPv6 functionality. Pressed by time, a decision was made to switch and use a Linux OS and the MIPL implementation.

Specifically, SUSE Linux 10.1 was used and the experience suggested that the Linux option has several advantages over the FreeBSD option:

1. SUSE Linux has a very well-designed and full-featured system configuration tool, YAST, which is a complete control center for system administration. SUSE Linux proved to be easy to install and configure it in depth during the installation time. Moreover, Novell, the company behind SUSE, offers great on-line technical support and documentation.
2. Under Linux whenever a software module was needed for the test bed, the only thing to do was to invoke YAST to search and verify if the module (called package in Linux) was installed or not. If it wasn't, a simple mouse click on the module was sufficient and YAST assumed the responsibility to install, configure and resolve all dependencies automatically.
3. The MIPL project was the most recent release for MIPv6 implementation (released on 14 June 2006) and fully RFC 3775 compliant.

C. TEST BED DESCRIPTION

1. Test Bed Layout Description

The implemented network test bed consists of five computers. Two of them assume the roles of the CN and MN respectively. The other three are configured as IPv6 capable routers. PC-based software router implementation is used instead of commercial IPv6 routers in order to have more flexibility for the addition of new IPv6 features and fine tuning of network parameters such as the router advertisements' intervals [M. Dunmore (6net) Final MIPv6 Support Guide February 8, 2005]. Table 1 presents the main hardware characteristics of the PCs used.

Role	**Make/Model**	**CPU/speed**	**RAM size**
MN	DELL Optiplex GX620	Intel(R) Pentium(R)4 3.40 GHz	2 GB
CN	DELL Optiplex GX620	Intel(R) Pentium(R)4 3.40 GHz	2 GB
HA router	DELL Precision 340	Intel(R) Pentium(R)4 2.40 GHz	256 MB
Frouter	DELL Precision 340	Intel(R) Pentium(R)4 1.8 GHz	512 MB
CNrouter	DELL Precision 340	Intel(R) Pentium(R)4 2.40 GHz	512 MB

Table 1. Hardware Characteristics of MIPv6 Test bed Components

All the components of the network are connected via Netgear dual speed hubs (model DS104) running at 10 Mbps so as to facilitate packet sniffing for debugging purposes.

Handoffs between networks for the MN are simulated by unplugging the Ethernet cable to which the MN is currently attached and replace it with a cable from the network we wish to move into.

Figure 3 shows the physical layout for the implemented test bed.

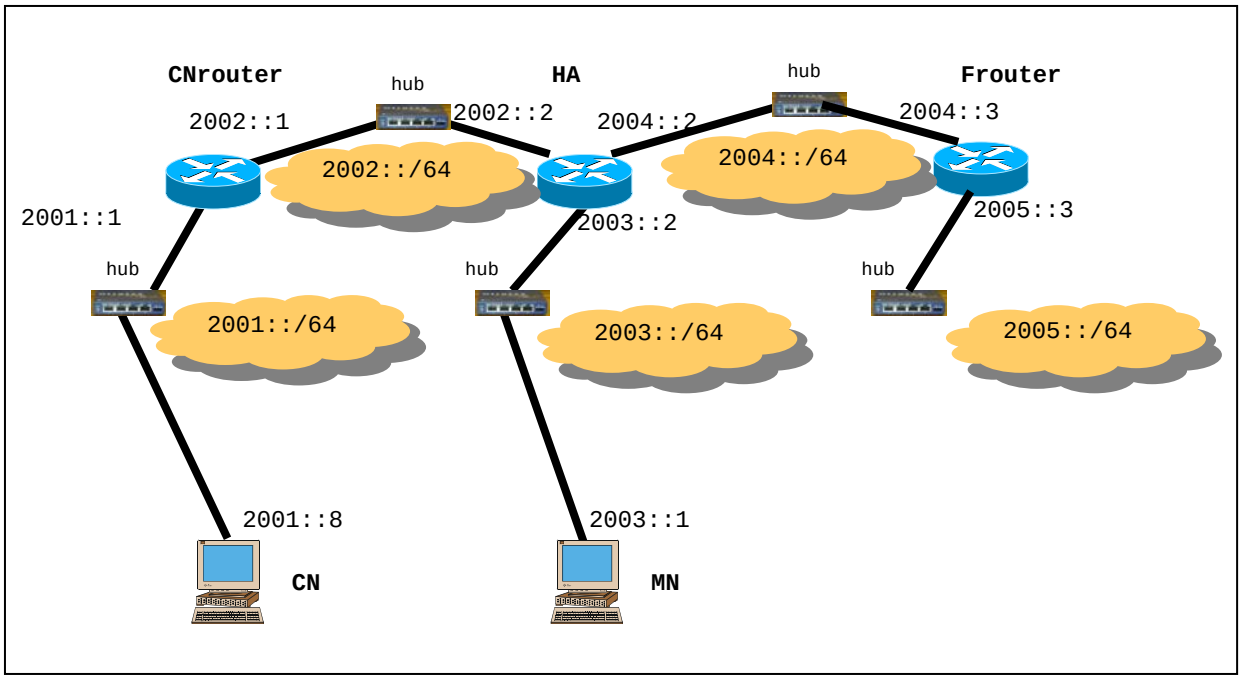

Figure 3. Physical Layout of MIPv6 Test bed

The home network of the mobile node (MN) is the 2003::/64. The home agent (HA) is installed on the HA router. The home network of the CN is the 2001::/64. During the experiments, the MN was moved between the home network and a foreign network, 2005::/64 which is advertised by the Frouter.

All systems run the boxed distribution SUSE 10.1 as their OS with Linux kernel 2.6.16.13-4 except the HA, the MN and the CN which have been recompiled with Linux kernel 2.6.16 patched with the MIPv6-2.0.2-linux-2.6.16.patch to provide the Mobile IPv6 features. The OS and the patch were downloaded from

ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.tar.bz2 and

http://mobile-ipv6.org/software/download/mipv6-2.0.2-linux-2.6.16.patch.gz, respectively.

In Table 2 are presented the interfaces of the Components of the test bed network along with their MAC and IP addresses.

Node	Interface	MAC	IP address
HA	eth0	00:04:75:b5:a6:32	2003::2
	eth1	00:0b:db:25:69:61	2004::2
	eth2	00:40:f4:5f:a9:13	2002::2
MN	eth0	00:12:3f:ae:20:5b	2003::1
CNrouter	eth0	00:0a:5e:00:49:1b	2002::1
	eth1	00:0b:db:25:73:68	2000::1
	eth2	00:40:f4:5a:5b:cc	2001::1
Frouter	eth0	00:08:74:41:5e:3f	2004::3
	eth1	00:09:5b:0a:5d:b3	2005::3
CN	eth0	00:12:3f:ae:21:c2	2001::8

Table 2. Test Bed IP and MAC Addresses

2. Configure-Patch-Build and Install the MIPv6 Kernel at HA, MN and CN

For the configuration of the implemented MIPv6 network components (HA, MN and CN), the following excellent tutorials were used:

- "How To Compile A Kernel - The SuSE Way," [http://www.howtoforge.com/kernel_compilation_suse Last visited on February 2, 2007].
- "Linux Mobile IPv6 HOWTO," [http://gnist.org/~lars/doc/Mobile-IPv6-HOWTO/Mobile-IPv6-HOWTO.html Last visited on February 10, 2007].

- “Mobile IPv6 Mini HOWTO,” [http://www.ipt.etsi.org/mini_howto.htm Last visited on February 12, 2007].

The first site describes the procedure of compiling a kernel on SuSE systems. It describes how to build a custom kernel using the latest unmodified kernel sources from [http://www.kernel.org/ (vanilla kernel) so that the user could be independent from the kernels supplied by his distribution.

Another reason for choosing this tutorial was because its goal was to build a kernel rpm package that could be used not only for installation of the MIPv6 capable kernel on the specific system, but also on the other SuSE systems that are used in the test bed and demand the same configuration.

The tutorial also shows how to patch the kernel sources if additional features are needed, like the MIPv6 patch for the Mobile IPv6 functionalities.

More specifically, the following steps were followed to install and patch a Linux kernel. (The tutorial provides more detailed screenshots of the installation.)

a. Install **ncurses-devel** which will be needed by the **make menuconfig** command which will be used later on:

```
# yast -i ncurses-devel
```

b. Modify a few tools that will be needed to build the new kernel:

```
# cp /usr/lib/rpm/find-provides.ksyms
/usr/lib/rpm/find-provides.ksyms_orig
```

```
# cp /usr/lib/rpm/find-requires.ksyms
/usr/lib/rpm/find-requires.ksyms_orig
# cp /usr/lib/rpm/find-supplements.ksyms
/usr/lib/rpm/find-supplements.ksyms_orig
```

c. Open each of these scripts and replace ***kernel-*) *is_kernel_package=1;;*** with ***kernel*) *is_kernel_package=1*** :

```
# vi /usr/lib/rpm/find-provides.ksyms
# vi /usr/lib/rpm/find-requires.ksyms
# vi /usr/lib/rpm/find-supplements.ksyms
```

Next, move to /usr/src in order to download the desired kernel (2.6.16) to /usr/src *directory*.

```
# cd /usr/src
```

d. Go to http://www.kernel.org/ and select the desired for installation kernel, in this case, linux-2.6.16.tar.bz2. The Kernel can be downloaded to directory /usr/src like this:

```
# wget http://www.kernel.org/pub/linux/kernel/v2.6/
linux-2.6.16.tar.bz2
```

e. Unpack the kernel sources and create a symlink *linux* to the kernel sources directory:

```
# tar xjf linux-2.6.16.tar.bz2
# ln -s linux-2.6.16 linux
```

Check that the linux is symlinked with the desired Kernel:

```
# ls –l
```

It should be seen: linux→linux-2.6.16. If the linux is still connected with the previous kernel, implement the commands:

```
# rm linux

# ln -s linux-2.6.16 linux
```

f. Change directory and download the patch found in http://mobile-ipv6.org/software/download/mipv6-2.0.2-linux-2.6.16.patch.gz to the Kernel source and uncompress it:

```
# cd /usr/local/src

# wget http://mobile-ipv6.org/software/download/mipv6-2.0.2-linux-2.6.16.patch.gz
```

g. Move again to /usr/src/linux in order to test the patch before apply it:

```
# cd /usr/src/linux

# zcat /usr/local/src/mipv6-2.0.2-linux-2.6.16.patch.gz | patch -p1 --dry-run
```

This command is just a test, it does nothing to sources. If it doesn't show errors, the following command should be executed which actually applies the patch. Don't do it if the first command shows errors:

```
# zcat /usr/local/src/mipv6-2.0.2-linux-2.6.16.patch.gz | patch -p1
```

h. Configure The Kernel

The configuration of the current working kernel will be used as a basis for the new kernel. The existing configuration is copied to /usr/src/linux:

```
# make mrproper
# cp /boot/config-`uname -r` ./.config
```

i. Run

```
# make menuconfig
```

This command brings up the kernel configuration menu. Go to **Load an Alternate Configuration File** and choose **.config** (which contains the configuration of the current working kernel) as the configuration file. Then browse through the kernel configuration menu and make your choices. Make sure that you get inside **Networking→** and load all the necessary functionalities of MIPv6. I chose them all. Make sure a kernel version identification string is specified, under **General Setup --->** (-default) Local version - append to kernel release (in my configuration I named it MIPv6).

j. When this step is finished, select **Exit** and answer the following question (Do you wish to save your new kernel configuration?) with Yes.

k. Install the user space MIPv6 tool. Change directory (/usr/local/src), download the latest Linux MIPv6 source code (mipv6-2.0.2) from http://mobile-ipv6.org/software/download/mipv6-2.0.2.tar.gz and uncompress it:

```
# cd /usr/local/src

# wget http://mobile-ipv6.org/software/download/mipv6-2.0.2.tar.gz

# tar zxfv mipv6-2.0.2.tar.gz
```

l. Change directory:

```
# cd mipv6-2.0.2
```

m. Configure, compile and install the source code including the **--enable-vt** option to **configure**, which will enable a virtual terminal listening on localhost port 7777 and can be used later on to provide with helpful information.

```
# CPPFLAGS=-I/usr/src/linux/include ./configure --enable-vt

# make

# make install
```

n. Before the kernel is being built, it is of vital importance to check if it is MIPv6 ready. There are two ways to verify it:

The first one is to go to directory that you have installed the MIPv6 user space source code

```
# cd /usr/local/src/mipv6-2.0.2
```

and execute the following command:

```
# ./chkconf_kernel.sh  /usr/src/linux
```

If the response is the following:

```
Checking kernel configuration...

Using /usr/src/linux/.config

All kernel options are as they should.
```

a correct configuration has taken place. Otherwise, make the corrections suggested and continue.

Another way to check if the configuration is correct is to use an editor (vi,pico,gedit,etc) and verify that in the **.config** file in /user/src/linux, the following options have been chosen:

CONFIG_EXPERIMENTAL=y

CONFIG_SYSVIPC=y

CONFIG_PROC_FS=y

CONFIG_NET=y

CONFIG_INET=y

CONFIG_IPV6=y

CONFIG_IPV6_MIP6=y

CONFIG_XFRM=y

CONFIG_XFRM_USER=y

CONFIG_XFRM_ENHANCEMENT=y

CONFIG_IPV6_TUNNEL=y

CONFIG_IPV6_ADVANCED_ROUTER=y

CONFIG_IPV6_MULTIPLE_TABLES=y

The Mobile Node also needs:

CONFIG_IPV6_SUBTREES=y

CONFIG_ARPD=y

In case that IPSec is desired to be enabled, it is also needed:

CONFIG_INET6_ESP=y

CONFIG_NET_KEY=y

CONFIG_NET_KEY_MIGRATE=y

o. Build the kernel, simply executing this command:

```
# make rpm
```

p. Install The New Kernel

After the successful kernel build, a *src.rpm* and an *rpm* package have been created. The *src.rpm* package can be found in the */usr/src/packages/SRPMS/* directory. Verify its name by running:

```
# ls -l /usr/src/packages/SRPMS/
```

On my system it was called:

kernel-2.6.16MIPv6-1.src.rpm.

The rpm package can be found, depending on the architecture, in one of the following directories:

/usr/src/packages/RPMS/i386/, */usr/src/packages/RPMS/i586/*, */usr/src/packages/RPMS/i686/*, */usr/src/packages/RPMS/x86_64/*, etc.,

On my system it was located in */usr/src/packages/RPMS/i386/*, and by running

```
# ls -l /usr/src/packages/RPMS/i386/
```

I found out that its name was: *kernel-2.6.16MIPv6-1.i386.rpm*.

q. Install the kernel rpm package like this:

```
# cd /usr/src/packages/RPMS/i386/
# rpm -ivh kernel-2.6.16MIPv6-1.i386.rpm
```

(The created kernel rpm package can be transferred and installed to other SuSE systems without having to compile the kernel there again.)

r. Create a ramdisk for the new kernel, because otherwise the system will most likely not boot our new kernel:

```
# mkinitrd
```

(This command will create new ramdisks for all installed kernels.)

s. Configure the GRUB boot loader so that the new kernel gets booted when the system is restarted. Instead of modifying /boot/grub/menu.lst directly, run yast to do it.

```
# yast
```

Go to **System ->** Boot **Loader:**

On the next screen you will be seen the new existing GRUB records. Go to **Add** to add a new one: Select **Clone Selected Section** to clone one of the working GRUB records:

Enter a name for the new kernel, e.g. **SUSE Linux 2.6.16-MIPv6**, and go to **Kernel -> Browse**:

The contents of the */boot* directory and the location of the new kernel are visible. Select the new kernel which typically begins with *vmlinuz* (in my case *vmlinuz-2.6.16-MIPv6*)

Next, go to **Initial RAM Disk -> Browse:**

See the contents of the **/boot** directory. Select the appropriate ramdisk for the new kernel which typically begins with **initrd** (e.g. *initrd-2.6.16-MIPv6*).

A new GRUB record for the new kernel will be seen. Mark it and hit **Up** until it is the first in the list. Then hit **Set as Default** to make the new kernel the default one, hit **Finish** and Select **Quit** to leave YaST: Now reboot the system:

```
# shutdown -r now
```

If everything goes well, it should come up with the new kernel. Check the version of the kernel by running:

```
# uname -r
```

This should display something like: 2.6.16MIPv6 (The name that the user gave to the kernel).

3. Setup of HA, MN, CN, and routers

a. HA

The HA is a router with additional functionalities which enable it to be the HA of the MN in the test bed network. Issue the following commands, since HA is a router:

```
# echo 1 >/proc/sys/net/ipv6/conf/all/forwarding
# echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
# echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
# echo 0 > /proc/sys/net/ipv6/conf/all/accept_redirects
```

IPv6 routing was established by Zebra software which manages TCP/IP based routing protocols.

Zebra had been chosen because, unlike traditional monolithic architectures, and even the so-called "new modular architectures" that remove the burden of processing routing functions from the cpu utilizing special ASIC chips; it instead offers true modularity.

Zebra is unique in its design in that it has a process for each protocol [http://www.zebra.org/what.html] Ripng (RIP next generation) was chosen to be the routing protocol for the IPv6 tesbed, provided by ripngd deamon, which provides IPv6 routing services in accordance with the configuration file ripngd.conf.

The daemons zebra and ripngd are activated by issuing the commands:

```
# /etc/init.d/ripngd start
# /etc/init.d/zebra start
```

The router advertisement daemon (radvd) is run by Linux or BSD systems acting as IPv6 routers. It sends Router Advertisement messages, specified by RFC 2461, to a local Ethernet LAN periodically and when requested by a node sending a Router Solicitation message. These messages are required for IPv6 stateless autoconfiguration. In order for the MN to be able to discover when it returns home, the HA must be configured to send out router advertisements. Router Advertisements were implemented via the radvd daemon which provides Router Advertisement services in accordance with the configuration file radvd.conf.

The daemon radvd is activated by issuing the command:

```
# /etc/init.d/radvd start
```

The user can verify that RAs are being sent by running:

```
# radvdump
```

Authentication and authorization of Mobile IPv6 messages between the MN and HA with IPsec is possible in 2.6.16 kernel. In order to provide IPsec between the binding messages of the HA and the MN, sa.conf file was created. In order to activate the Security Association in both the HA and MN, ipsec-tools should have been installed using Yast and implement the following command after editing the file sa.conf:

```
# setkey –f /etc/sa.conf
```

The functionality of MIPv6 is provided by the daemon mip6d, based in the configuration file mip6d.conf.

In case that a service is not provided, the user can always find the location of the daemon and run it locally. For example:

```
# whereis mip6d

mip6d:/etc/mip6d.conf /usr/local/sbin/mip6d
```

That means that the location of the daemon is at /usr/local/sbin/mip6d. We change directory:

```
# cd /usr/local/sbin
```

Now the daemon can be run locally:

```
Host_Name:/usr/local/sbin# ./mip6d start
```

Exact copy of the files of the HA can be found in Appendix A.

b. MN

The MN is a host at which only the mip6d daemon should be run. It is provided manually with its global IPv6 address and the necessary configurations:

```
# ifconfig eth0 inet6 add 2003::1/64

# echo 0 > /proc/sys/net/ipv6/conf/eth0/forwarding

# echo 1 > /proc/sys/net/ipv6/conf/eth0/autoconf

# echo 1 > /proc/sys/net/ipv6/conf/eth0/accept_ra

# echo 1 > /proc/sys/net/ipv6/conf/eth0/accept_redirects

# route –A inet6 add default gw 2003::2
```

Note that MN host will be given an additional IPv6 address based on the HA's prefix and its MAC address. MN needs also the file sa.conf which should be positioned in /etc/ directory. Exact copy of the files of the MN can be found in Appendix B.

c. CN

CN is an IPv6 host at which only the mip6d daemon should be run. It is provided manually with its global IPv6 address and the necessary configurations:

```
# ifconfig eth0 inet6 add 2001::8/64
# echo 0 > /proc/sys/net/ipv6/conf/eth0/forwarding
# echo 1 > /proc/sys/net/ipv6/conf/eth0/autoconf
# echo 1 > /proc/sys/net/ipv6/conf/eth0/accept_ra
# echo 1 > /proc/sys/net/ipv6/conf/eth0/accept_redirects
# route –A inet6 add default gw 2001::1
```

Note that CN host will be given an additional IPv6 address based on the CNrouter's prefix and its MAC address. It does not need the file sa.conf. Exact copy of the files of the CN can be found in Appendix C.

d. CNrouter

The CNrouter is an IPv6 router.

Issue the following commands, since CNrouter is a router:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
# echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
# echo 0 > /proc/sys/net/ipv6/conf/all/accept_redirects
```

As with HA, IPv6 routing was established by Zebra software and ripngd daemon, which also assign IPv6 addresses to the interfaces of the router along with radvd daemon. Moreover RAs are managed via radvd daemon and can be verified by radvdump command. An exact copy of the files of the CNrouter can be found in Appendix D.

e. Frouter

The Frouter is an IPv6 router. Its configuration follows the principles of CNrouter. An exact copy of the files of the Frouter can be found in Appendix E.

D. VERIFYING THE CONFIGURATION

In order to verify the MIPv6 network configuration the following scenario, divided into three phases, has been enacted:

- Phase 1: MN resides in its Home Network (attached to the 2003::/64 link).
- Phase 2: MN moves to a Foreign Network (2005::/64) and acquires a new CoA.
- Phase 3: MN returns to the Home Network.

The above scenario was performed with and without using IPsec between the HA and the MN.

The procedure that will be followed in order to observe the RR is that the MN will start ping the CN from its Home Network and then follow Phases 2 and 3.

The tools that will be used to verify the configuration are:

- “Sniffers” at the links HA-MN, HA-FRouter, Frouter-MN with PCs equipped with Wireshark/Ethereal software.

- Virtual terminal (# telnet local host 7777) at the MIPv6 capable nodes, which provide the user with information about the binding caches and the binding update lists at them (HA, MN and the CN):
- The following debugging commands (at the MN) may be used to provide the user with ad hoc debugging information:

```
# watch ifconfig eth0

# watch route -A inet6

# tcpdump -i eth0 -vv ip6 or proto ipv6
```

In order to be able to follow better the RR procedure the following table which describes very briefly the types of mobile headers and the routing of messages.

MOBILITY HEADER	**Message**	**SENDER→RECEIVER**
MH Type 1	HOTI	MN to CN via HA
MH Type 2	COTI	MN to CN
MH Type 3	HOT	CN to MN via HA
MH Type 4	COT	CN to MN
MH Type 5	BU	MN to HA or MN to CN
MH Type 6	BA	HA to MN or HA to CN

Table 3. Table of Mobility Header Types

1. Scenario without the Use of IPsec

a. Phase 1: MN Is At Its Home Network

Mobile IPv6 is being started in all nodes with the following order: HA, MN, CN, being in the directory that mip6d resides, executing the following command:

```
#./mip6d –c /etc/mip6d.conf
```

The output of the predefined execution at HA and MN nodes are shown in Figures 4 and 5:

```
HArouter:/usr/local/sbin # ./mip6d -c /etc/mip6d.conf
mip6d[12901]: MIPL Mobile IPv6 for Linux v2.0.2 started (Home Agent(1))
main: MIPL Mobile IPv6 for Linux started in debug mode, not detaching
from terminal
conf_show: config_file = /etc/mip6d.conf (2)
conf_show: vt_hostname = localhost
conf_show: vt_service = 7777
conf_show: mip6_entity = 2
conf_show: debug_level = 10
conf_show: PolicyModulePath = [internal]
conf_show: DefaultBindingAclPolicy = 0
conf_show: NonVolatileBindingCache = disabled
conf_show: KeyMngMobCapability = disabled
conf_show: UseMnHaIPsec = disabled (3)
conf_show: MnMaxHaBindingLife = 262140
conf_show: MnMaxCnBindingLife = 420
conf_show: MnRouterProbes = 0
conf_show: MnRouterProbeTimeout = 0.000000
conf_show: InitialBindackTimeoutFirstReg = 1.500000
conf_show: InitialBindackTimeoutReReg = 1.000000
conf_show: UseCnBuAck = enabled
conf_show: DoRouteOptimizationMN = enabled (4)
conf_show: MnUseAllInterfaces = disabled
conf_show: MnDiscardHaParamProb = disabled
conf_show: SendMobPfxSols = enabled
conf_show: OptimisticHandoff = disabled
conf_show: SendMobPfxAdvs = enabled
conf_show: SendUnsolMobPfxAdvs = enabled
conf_show: MaxMobPfxAdvInterval = 86400
conf_show: MinMobPfxAdvInterval = 600
conf_show: HaMaxBindingLife = 262140
conf_show: DoRouteOptimizationCN = enabled (5)
xfrm_cn_init: Adding policies and states for CN
xfrm_ha_init: Adding policies and states for HA
ha_if_addr_setup: Joined anycast group 2003:0:0:0:fdff:ffff:ffff:fffe
on iface 8
```

Figure 4. Running Output of HA mip6d when MN is at Home Network

Important information from the output of HA mip6d:

(1) The node is a Home Agent.

(2) The configuration file is the /etc/mip6d.conf

(3) IPsec is not enabled

(4) RO is enabled between HA and MN

(5) RO is enabled between HA and CN

It follows the output of the MN mip6d:

```
MN2:/usr/local/sbin # ./mip6d -c /etc/mip6d.conf
mip6d[16697]: MIPL Mobile IPv6 for Linux v2.0.2 started (Mobile Node(1)
)
main: MIPL Mobile IPv6 for Linux started in debug mode, not detaching
from terminal
conf_show: config_file = /etc/mip6d.conf (2)
conf_show: vt_hostname = localhost
conf_show: vt_service = 7777
conf_show: mip6_entity = 1
conf_show: debug_level = 10
conf_show: PolicyModulePath = [internal]
conf_show: DefaultBindingAclPolicy = 0
conf_show: NonVolatileBindingCache = disabled
conf_show: KeyMngMobCapability = disabled
conf_show: UseMnHaIPsec = disabled (3)
conf_show: MnMaxHaBindingLife = 262140
conf_show: MnMaxCnBindingLife = 420
conf_show: MnRouterProbes = 0
conf_show: MnRouterProbeTimeout = 0.000000
conf_show: InitialBindackTimeoutFirstReg = 1.500000
conf_show: InitialBindackTimeoutReReg = 1.000000
conf_show: UseCnBuAck = enabled
conf_show: DoRouteOptimizationMN = enabled (4)
conf_show: MnUseAllInterfaces = disabled
conf_show: MnDiscardHaParamProb = disabled
conf_show: SendMobPfxSols = enabled
conf_show: OptimisticHandoff = disabled
conf_show: SendMobPfxAdvs = enabled
conf_show: SendUnsolMobPfxAdvs = enabled
conf_show: MaxMobPfxAdvInterval = 86400
conf_show: MinMobPfxAdvInterval = 600
conf_show: HaMaxBindingLife = 262140
conf_show: DoRouteOptimizationCN = enabled (5)
xfrm_cn_init: Adding policies and states for CN
xfrm_mn_init: Adding policies and states for MN
conf_home_addr_info: HoA address 2003:0:0:0:0:0:0:1
conf_home_addr_info: HA address 2003:0:0:0:0:0:0:2
__tunnel_add: created tunnel ip6tnl1 (21) from 2003:0:0:0:0:0:0:1 to
2003:0:0:0:0:0:0:2 user count 1 (6)
conf_home_addr_info: Home address 2003:0:0:0:0:0:0:1
flag_hoa: set HoA 2003:0:0:0:0:0:0:1/128 iif 21 flags 10 preferred_time
4294967295 valid_time 4294967295
conf_home_addr_info: Added new home_addr_info successfully
__md_discover_router: discover link on iface eth0 (6)
md_change_default_router: add new router fe80:0:0:0:204:75ff:feb5:a632
on interface eth0 (6)
mn_addr_do_dad: DAD succeeded!
mn_move: 1535
mn_move: in home net (7)
mv_hoa: move HoA 2003:0:0:0:0:0:0:1/64 from iface 21 to 6
md_change_default_router: add new router fe80:0:0:0:204:75ff:feb5:a632
on interface eth0 (6)
md_expire_router: expiring router
fe80:0:0:0:204:75ff:feb5:a632 on iface eth0 (6)
```

Figure 5. Running Output of MN mip6d when MN is at Home Network

Important information from the output of HA mip6d daemon:

(1) The node is a Mobile Node

(2) The configuration file is the /etc/mip6d.conf

(3) IPsec is not enabled

(4) RO is enabled between HA and MN

(5) RO is enabled between HA and CN

(6) A tunnel has been created between HA and MN

(7) MN is at Home Network

Issuing the command

```
#ifconfig –a
```

at the MN, provides the output illustrated at Figure 6, proving that the tunnel (ip6tnl1) is up and ready for connections.

```
MN2:/etc # ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:12:3F:AE:20:5B
          inet6 addr: 2003::1/64 Scope:Global (1)
          inet6 addr: 2003::212:3fff:feae:205b/64 Scope:Global (2)
          inet6 addr: fe80::212:3fff:feae:205b/64 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:155202 errors:0 dropped:0 overruns:0 frame:6
          TX packets:28259 errors:0 dropped:0 overruns:0 carrier:0
          collisions:9 txqueuelen:1000
          RX bytes:19807002 (18.8 Mb)  TX bytes:3921682 (3.7 Mb)
          Interrupt:169

gre0        Link encap:UNSPEC  HWaddr 00-00-00-00-20-5B-00-00-00-00-00-
00-00-00-00-00
          NOARP  MTU:1476  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

ip6tnl0    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00
          NOARP  MTU:1460  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

ip6tnl1    Link encap:UNSPEC  HWaddr 20-03-00-00-00-00-00-00-00-00-00-
00-00-00-00-00
          inet6 addr: fe80::212:3fff:feae:205b/64 Scope:Link
```

```
          UP POINTOPOINT RUNNING NOARP  MTU:1460  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1598 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1598 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:127230 (124.2 Kb)  TX bytes:127230 (124.2 Kb)

sit0      Link encap:IPv6-in-IPv4
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

tunl0     Link encap:IPIP Tunnel  HWaddr
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Figure 6. Running Output MN ifconfig when MN is at Home Network

Last, we start the mip6d daemon to the CN with the following output (Figure 7):

```
CN2:/usr/local/sbin # ./mip6d -c /etc/mip6d.conf
mip6d[31550]: MIPL Mobile IPv6 for Linux v2.0.2 started (Correspondent
Node)
main: MIPL Mobile IPv6 for Linux started in debug mode, not detaching
from terminal
conf_show: config_file = /etc/mip6d.conf
conf_show: vt_hostname = localhost
conf_show: vt_service = 7777
conf_show: mip6_entity = 0
conf_show: debug_level = 10
conf_show: PolicyModulePath = [internal]
conf_show: DefaultBindingAclPolicy = 0
conf_show: NonVolatileBindingCache = disabled
conf_show: KeyMngMobCapability = disabled
conf_show: UseMnHaIPsec = enabled
conf_show: MnMaxHaBindingLife = 262140
conf_show: MnMaxCnBindingLife = 420
conf_show: MnRouterProbes = 0
conf_show: MnRouterProbeTimeout = 0.000000
conf_show: InitialBindackTimeoutFirstReg = 1.500000
conf_show: InitialBindackTimeoutReReg = 1.000000
```

```
conf_show: UseCnBuAck = enabled
conf_show: DoRouteOptimizationMN = enabled
conf_show: MnUseAllInterfaces = disabled
conf_show: MnDiscardHaParamProb = disabled
conf_show: SendMobPfxSols = enabled
conf_show: OptimisticHandoff = disabled
conf_show: SendMobPfxAdvs = enabled
conf_show: SendUnsolMobPfxAdvs = enabled
conf_show: MaxMobPfxAdvInterval = 86400
conf_show: MinMobPfxAdvInterval = 600
conf_show: HaMaxBindingLife = 262140
conf_show: DoRouteOptimizationCN = enabled
xfrm_cn_init: Adding policies and states for CN
```

Figure 7. Running Output CN mip6d when MN is at Home Network

Figure 8 shows the kernel IP routing table of MN. It will be shown that the default route will be changed in the next phase (after the MN moves to the Foreign Network).

```
MN2:/etc/init.d # route -A inet6
Kernel IPv6 routing table
Destination                    Next Hop                   Flags Metric Ref    Use Iface
2001::/64                      2003::2                    UG    1024   9810     0 eth0
2003::/64                      ::                         U     256    109      0 eth0
fe80::/64                      ::                         U     256    0        0 ip6tnl1
fe80::/64                      ::                         U     256    0        0 eth0
ff02::1/128                    ff02::1                    UC    0      106      0 eth0
ff00::/8                       ::                         U     256    0        0 eth0
ff00::/8                       ::                         U     256    0        0 ip6tnl1
::/0                           fe80::204:75ff:feb5:a632   UGDA  1024   0        0 eth0
::1/128                        ::                         U     0      21       1 lo
2003::1/128                    ::                         U     0      175      1 lo
2003::212:3fff:feae:205b/128   ::                         U     0      0        1 lo
fe80::212:3fff:feae:205b/128   ::                         U     0      0        1 lo
fe80::212:3fff:feae:205b/128   ::                         U     0      1        1 lo
```

Figure 8. MN Kernel IP Routing Table before MN Movement

b. Phase 2: MN Moves to a Foreign Network

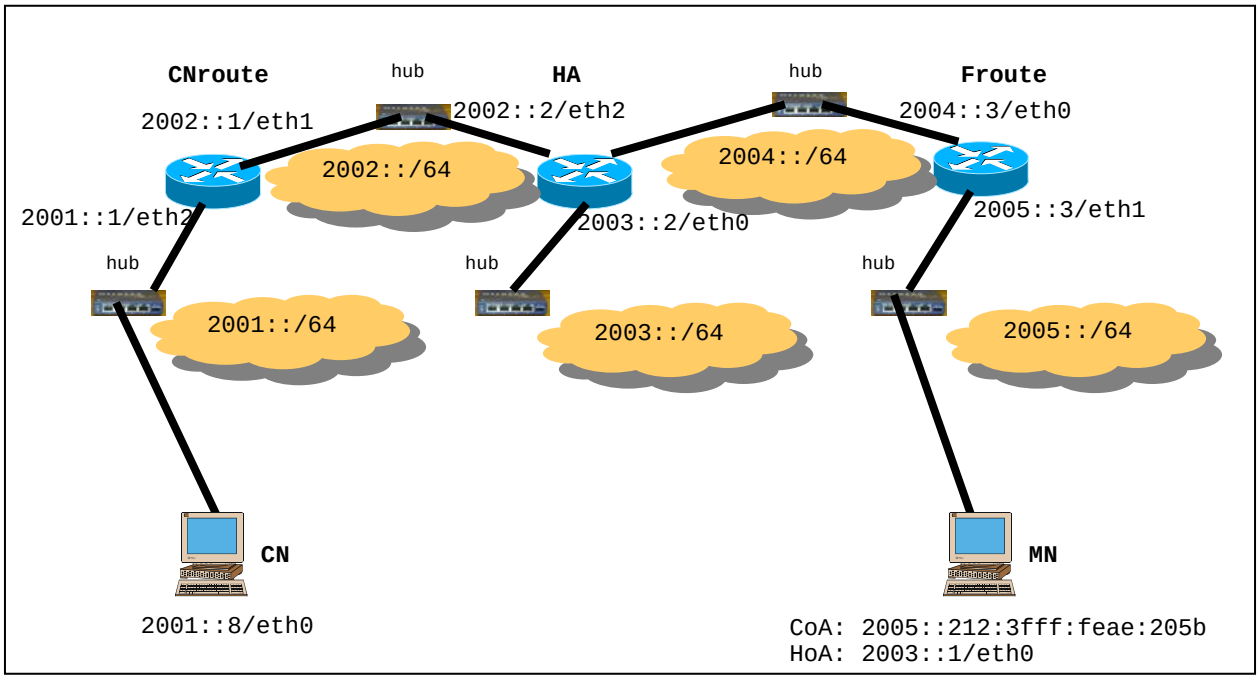

Figure 9. MN Moved to Foreign Network 2005::/64

When the MN detects that the default router (HA) cannot be reached, the MN sends out a Router Solicitation and the Foreign Router interface, where the MN has been attached, responds with its prefix. The MN reconfigures itself, creating a new IPv6 address, using the prefix of the Foreign Router and its MAC address (1) and its HoA appears in the created ip6tnl1 (2) , as shown in Figure 10.

```
MN2:/home/ikandira # ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:12:3F:AE:20:5B
          inet6 addr: 2005::212:3fff:feae:205b/64 Scope:Global
          inet6 addr: fe80::212:3fff:feae:205b/64 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:158180 errors:0 dropped:0 overruns:0 frame:7
          TX packets:30770 errors:0 dropped:0 overruns:0 carrier:0
          collisions:9 txqueuelen:1000
          RX bytes:20214758 (19.2 Mb)  TX bytes:4270328 (4.0 Mb)
          Interrupt:169

gre0        Link encap:UNSPEC  HWaddr 00-00-00-00-20-5B-00-00-00-00-00-
00-00-00-00-00
          NOARP  MTU:1476  Metric:1
```

```
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

ip6tnl0    Link encap:UNSPEC   HWaddr 00-00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00
          NOARP  MTU:1460  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

ip6tnl1    Link encap:UNSPEC   HWaddr 20-05-00-00-00-00-00-00-00-00-00-
00-00-00-00-00
          inet6 addr: 2003::1/128 Scope:Global
          inet6 addr: fe80::212:3fff:feae:205b/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP  MTU:1460  Metric:1
          RX packets:21 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:4 dropped:4 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1408 (1.3 Kb)  TX bytes:2016 (1.9 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1654 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1654 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:132076 (128.9 Kb)  TX bytes:132076 (128.9 Kb)

sit0      Link encap:IPv6-in-IPv4
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

tunl0     Link encap:IPIP Tunnel  HWaddr
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Figure 10. ifconfig of MN moved to the Foreign Network

After the MN acquires its new address (CoA), it informs its HA for its new CoA via a BU message and it receives a BA message, where the HA verifies that it is aware for the MN’s CoA.

Using the virtual terminal, it is observed that the HA has updated its binding cache (Figure 10) and the MN shows its binding update list (Figure 11):

```
HArouter:/etc/init.d # telnet localhost 7777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
mip6d> help
bc fancy hal nonce pl prompt quit thread verbose
mip6d> verbose yes
yes
mip6d> bc
hoa 2003:0:0:0:0:0:0:1 nonce 0 status registered
 coa 2005:0:0:0:212:3fff:feae:205b nonce 0 flags AH--
 local 2003:0:0:0:0:0:0:2 tunnel ip6tnl1 link eth0
 lifetime 11236 / 11992 seq 64365 unreach 0 mpa -96 / 657 retry 0
mip6d> hal
eth0 2003:0:0:0:0:0:0:2
 preference 20 lifetime 10000
mip6d> pl
eth0 2003:0:0:0:0:0:0:2/64
 valid 11998 / 12000 preferred 10000 flags OAR
```

Figure 11. Virtual Terminal Information Provided by HA

```
MN2:/etc # telnet localhost 7777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
mip6d> help
bc bul fancy nonce prompt quit rr verbose
mip6d> verbose yes
yes
mip6d> bul
== BUL_ENTRY ==
Home address    2003:0:0:0:0:0:0:1
Care-of address 2005:0:0:0:212:3fff:feae:205b
CN address      2003:0:0:0:0:0:0:2
 lifetime = 11992,  delay = 11392000
 flags: IP6_MH_BU_HOME IP6_MH_BU_ACK
 ack ready
 dev eth0 last_coa 2005:0:0:0:212:3fff:feae:205b
 lifetime 11707 / 11992 seq 64365 resend 0 delay 11392(after 11108s)
expires 11707
 mps 10515 / 10797
== BUL_ENTRY ==
Home address    2003:0:0:0:0:0:0:1
Care-of address 2005:0:0:0:212:3fff:feae:205b
CN address      2001:0:0:0:0:0:0:8
 lifetime = 420,  delay = 420000
 flags: IP6_MH_BU_ACK
 ack ready RR state ready
 dev eth0 last_coa 2005:0:0:0:212:3fff:feae:205b
care-of nonce index 4home nonce index 4
```

```
 lifetime 261 / 420 seq 56778 resend 0 delay 420(after 262s) expires
261

mip6d> rr
== Return Routability Entry (HOT_ENTRY) ==
 HoA 2003:0:0:0:0:0:1
 CN  2001:0:0:0:0:0:8
 CoA 2005:0:0:0:212:3fff:feae:205b
 Interface ip6tnl1
 resend 0 delay 210 (after 36 seconds) expires in 36 seconds
== Return Routability Entry (COT_ENTRY) ==
 CoA 2005:0:0:0:212:3fff:feae:205b
 CN  2001:0:0:0:0:0:8
 HoA 2003:0:0:0:0:0:1
 Interface eth0
 resend 0 delay 210 (after 36 seconds) expires in 36 seconds
```

Figure 12. Virtual Terminal Information Provided by MN

Moreover, the MN is changing its default route to a tunnel (Figure 13).

```
MN2:/etc/init.d # route -A inet6
Kernel IPv6 routing table
Destination                    Next Hop               Flags Metric Ref    Use Iface
::/0                           ::                      U     128    0        0 ip6tnl1
2001::8/128                    2001::8                 UC    0      69       0 ip6tnl1
2003::2/128                    2003::2                 UC    0      2        1 ip6tnl1
2001::/64                      2003::2                 UG    1024   9920     1 eth0
2005::/64                      ::                      UA    256    18       0 eth0
fe80::/64                      ::                      U     256    0        0 ip6tnl1
fe80::/64                      ::                      U     256    0        0 eth0
ff02::1/128                    ff02::1                 UC    0      19       0 eth0
ff00::/8                       ::                      U     256    0        0 eth0
ff00::/8                       ::                      U     256    0        0 ip6tnl1
::/0                           fe80::209:5bff:fe0a:5db3 UGDA  1024   24       2 eth0
::1/128                        ::                      U     0      21       1 lo
2003::1/128                    ::                      U     0      71       1 lo
2005::212:3fff:feae:205b/128   ::                      U     0      82       1 lo
fe80::212:3fff:feae:205b/128   ::                      U     0      0        1 lo
fe80::212:3fff:feae:205b/128   ::                      U     0      2        1 lo
```

Figure 13. MN Kernel IP Routing Table after Movement to Foreign Network

As expected, since IPSec was not used, all the messages were captured from Wireshark on the link Frouter to MN, and presented in Figures 14 to 19.

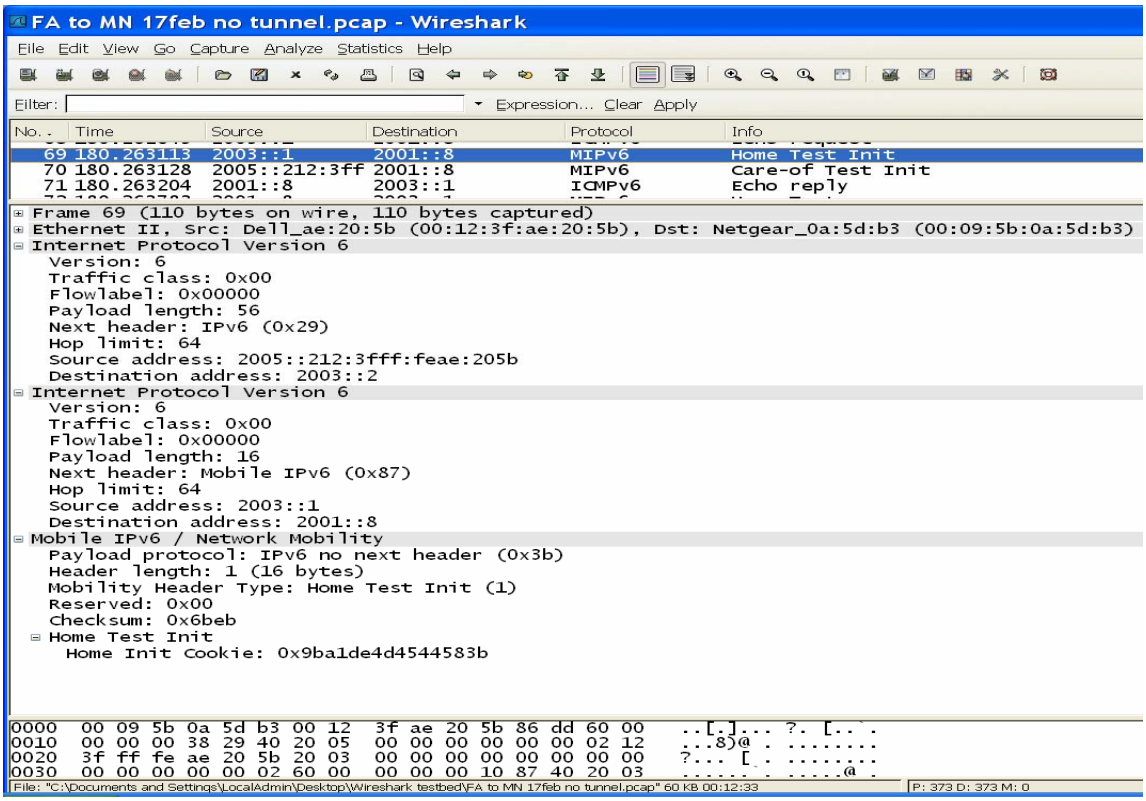

Figure 14. HOTI Message from MN to CN

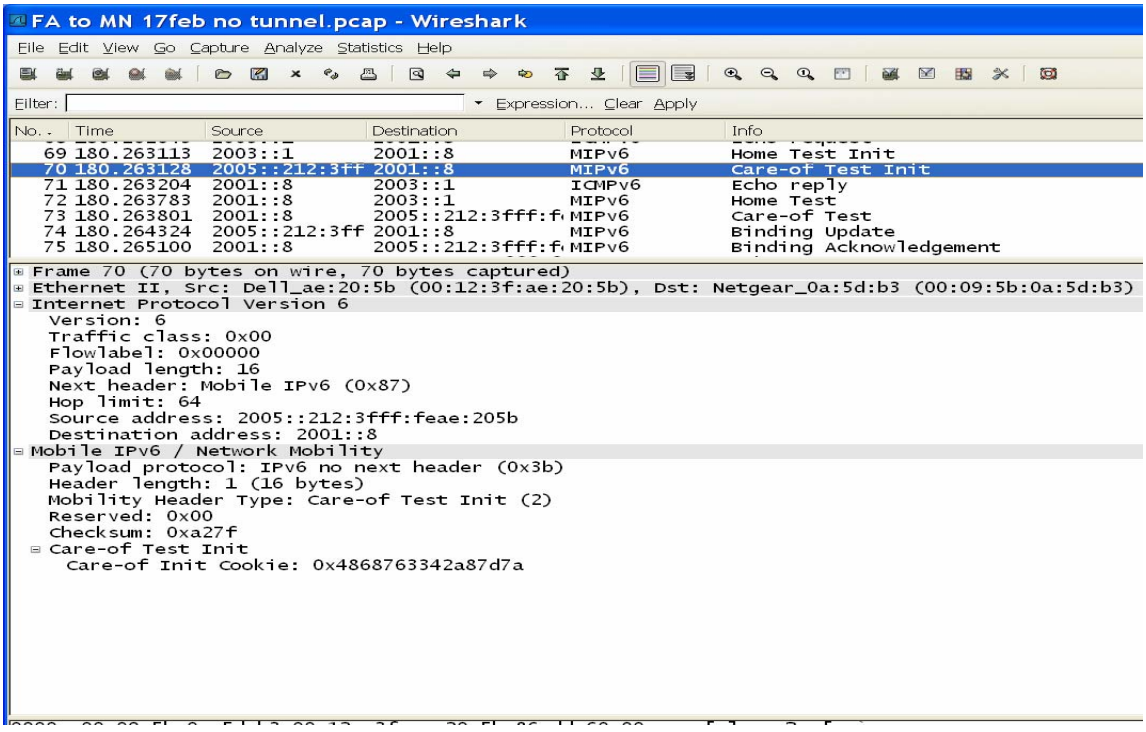

Figure 15. COTI Message from MN (CoA) to CN

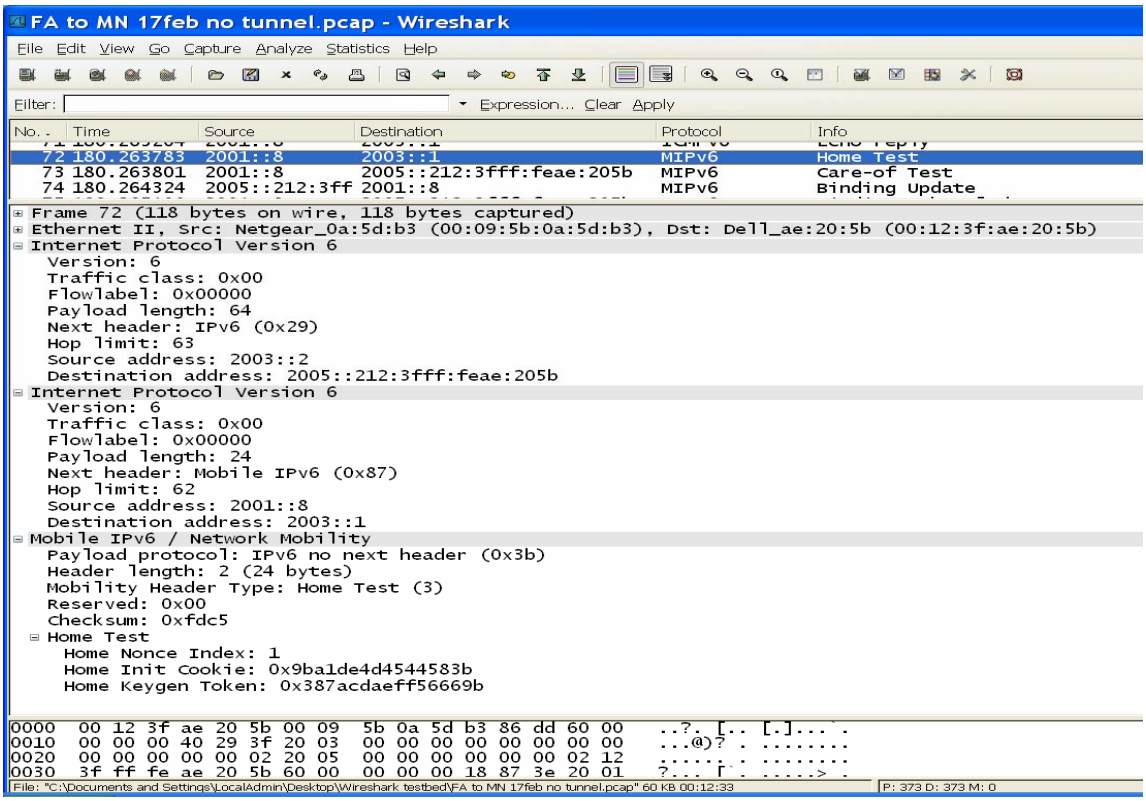

Figure 16. HOT Message from CN to CN (HoA)

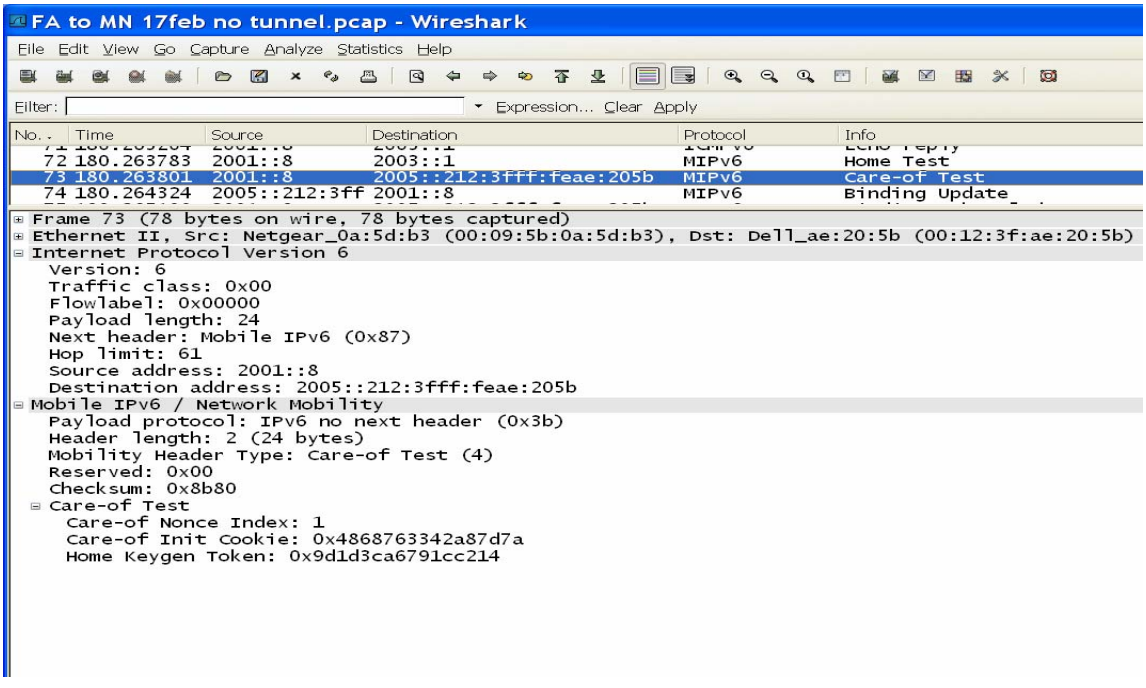

Figure 17. COT Message from CN to MN (CoA)

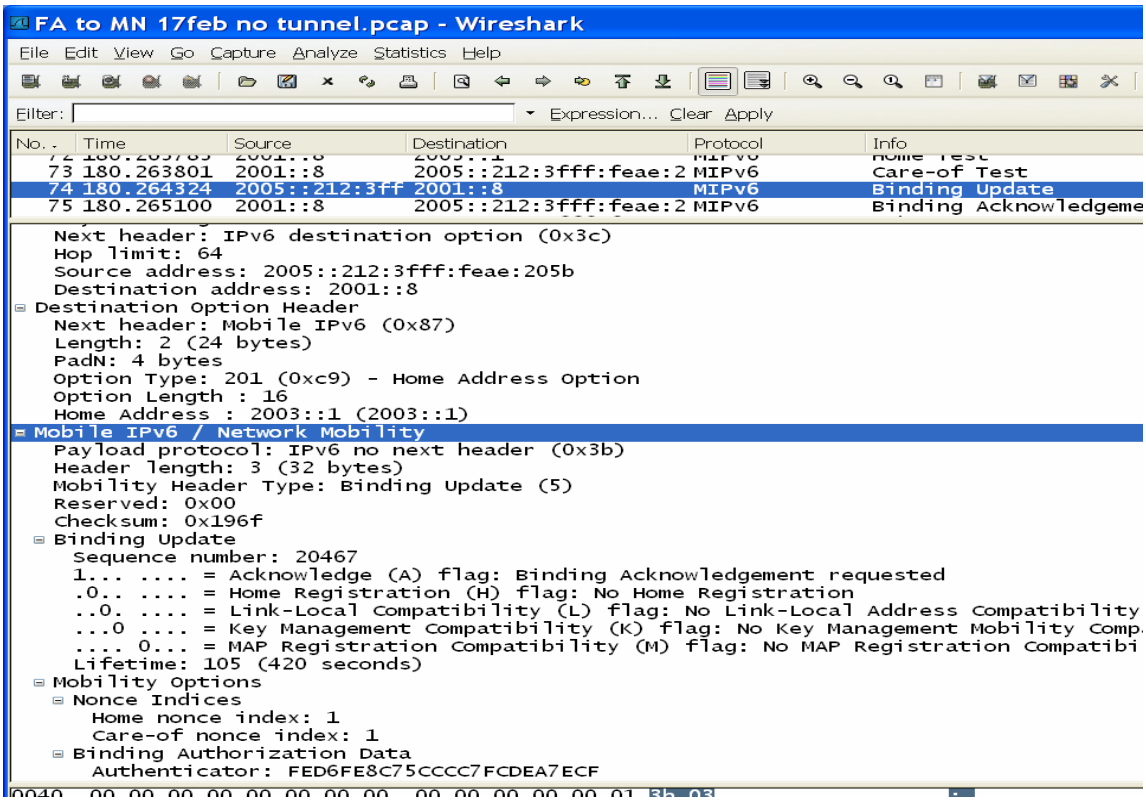

Figure 18. BU Message from MN(CoA) to CN

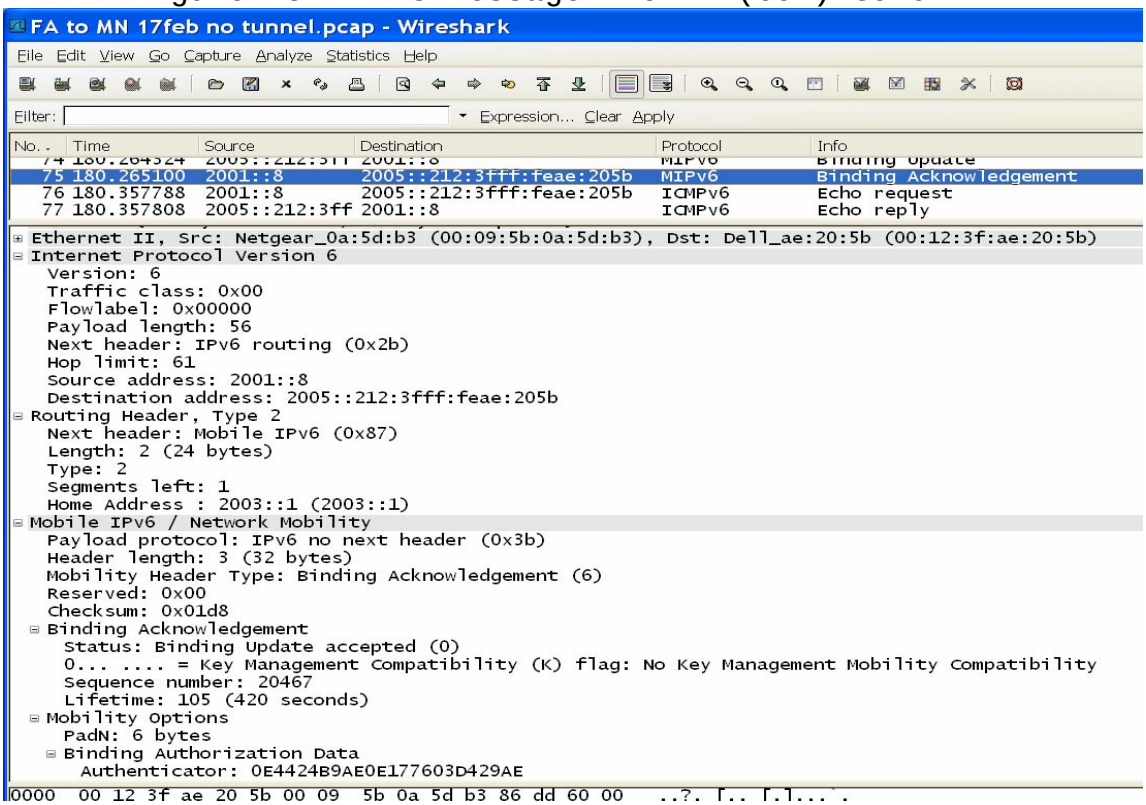

Figure 19. BA Message from CN to MN(CoA)

c. Phase 3: MN Returns to its Home Network

The MN assumes that it returned to its Home Network, since it receives the router advertisement with the HA-bit set from the interface of the HA. The HA flushes its binding cache information. This can be seen from the virtual terminal of the HA (Figure 20).

```
HArouter:/etc/init.d # telnet localhost 7777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
mip6d> help
bc fancy hal nonce pl prompt quit thread verbose
mip6d> bc
mip6d> hal
eth0 2003:0:0:0:0:0:0:2
 preference 20 lifetime 10000
mip6d> hal
eth0 2003:0:0:0:0:0:0:2
 preference 20 lifetime 10000
mip6d> pl
eth0 2003:0:0:0:0:0:0:2/64
 valid 11999 / 12000 preferred 10000 flags OAR
```

Figure 20. HA Virtual Terminal Output

2. Scenario with the Use of IPsec

IPv6 was designed with security in mind from the outset, mandating the support of authentication and encryption in all IPv6 implementations [Soliman04].

IPsec is a set of protocols to support a secure exchange of packets at the IP layer. It uses two protocols to provide traffic security services, Authentication Header (AH) and Encapsulating Security Payload (ESP).

The IP Authentication Header (AH) offers integrity and date origin authentication, with optional anti-replay features.

The Encapsulating Security Payload (ESP) protocol offers the same set of services, and also offers confidentiality (provide authentication and encryption only for the headers following the ESP header).

Currently only the ESP IPsec protocol is supported in MIPL.

IPsec supports two tunnel modes: Transport and Tunnel. Transport mode encrypts only the transport (layer 4) and data portion (application payload) of each packet, but leaves the original IP header in the clear. The more secure Tunnel mode encrypts both the original IP header and the upper layers. On the receiving side, an IPSec-compliant device decrypts each packet.

The protection offered by IPsec is based on requirements defined by a Security Policy Database (SPD) which is established and maintained by user or system administrator, or by an application operating within constraints established by either of the above.

In our case, it is the MIPL which establishes the SPD between HA and MN. An SPD cannot be dictated because the CoA of MN is not known in advance in the foreign network. Only the MIPL can manage the binding between CoA and HoA.

The Security Association Database (SAD) contains parameters that are associated with each established Security Association (SA). A SA is a simplex "connection" that provides security services to the traffic carried by it.

In order to enable IPsec in the test bed the following changes have to be made:

- IPsec should be enabled in mip6d.conf files of both the HA and MN (UseMnHaIPsec enabled;) and a set of IPsec policy should be added (see Appendices A and B for more details in files mip6d.conf)
- SAs should be set manually. The configuration file (sa.conf), must be the same on the MN and HA (see Appendices A and B for more details in files sa.conf (they must be the same))

One interesting observation is that, when the sa.conf files were implemented, the des-cbc “secret” should have exactly eight characters while the hmac-sha1 “secret” should have exactly 20 characters.

The set of commands that were used was the following:

Load the implemented configuration:

```
# setkey -f /etc/sa.conf
```

Verify the SAD:

```
# setkey –D
```

Flush the SPD:

```
# setkey –FP
```

Flush the SAD:

```
# setkey –F
```

Initially, when the MN is at Home Network, the SPD can be seen, executing the command:

```
# setkey -D
```

and the output of the command is:

```
MN2:/etc # setkey -D
2003::2 2003::1
        esp mode=tunnel spi=2005(0x000007d5) reqid=0(0x00000000)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 17 21:31:50 2007
        diff: 126138(s) hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=5 pid=20007 refcnt=0
2003::2 2003::1
        esp mode=transport spi=2003(0x000007d3) reqid=3(0x00000003)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 17 21:31:50 2007
        diff: 126138(s) hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=4 pid=20007 refcnt=0
2003::2 2003::1
        esp mode=transport spi=2001(0x000007d1) reqid=2(0x00000002)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 17 21:31:50 2007
        diff: 126138(s) hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=3 pid=20007 refcnt=0
2003::1 2003::2
        esp mode=tunnel spi=2004(0x000007d4) reqid=0(0x00000000)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 17 21:31:50 2007
        diff: 126138(s) hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=2 pid=20007 refcnt=0
2003::1 2003::2
        esp mode=transport spi=2002(0x000007d2) reqid=3(0x00000003)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 17 21:31:50 2007
        diff: 126138(s) hard: 0(s)      soft: 0(s)
```

```
        last:                              hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=1 pid=20007 refcnt=0
2003::1 2003::2
        esp mode=transport spi=2000(0x000007d0) reqid=1(0x00000001)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 17 21:31:50 2007
        diff: 126138(s) hard: 0(s)      soft: 0(s)
        last: Feb 16 15:26:37 2007      hard: 0(s)      soft: 0(s)
        current: 26352(bytes)   hard: 0(bytes)  soft: 0(bytes)
        allocated: 244  hard: 0 soft: 0
        sadb_seq=0 pid=20007 refcnt=0
```

Figure 21. MN SPD Output before MN Moves to the Foreign Network

When the MN moves to the foreign network, the SPD is updated via MIPL with the CoA of the MN, as can be seen in Figure 22.

```
MN2:/etc # setkey -D
2003::2 2003::1
        esp mode=transport spi=2003(0x000007d3) reqid=3(0x00000003)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 22 16:26:47 2007
        diff: 539835(s) hard: 0(s)      soft: 0(s)
        last: Feb 22 16:26:00 2007      hard: 0(s)      soft: 0(s)
        current: 520(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 13   hard: 0 soft: 0
        sadb_seq=5 pid=5823 refcnt=0
2003::2 2003::1
        esp mode=transport spi=2001(0x000007d1) reqid=2(0x00000002)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 22 16:26:47 2007
        diff: 539835(s) hard: 0(s)      soft: 0(s)
        last: Feb 22 16:25:58 2007      hard: 0(s)      soft: 0(s)
        current: 368(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 23   hard: 0 soft: 0
        sadb_seq=4 pid=5823 refcnt=0
2005::212:3fff:feae:205b 2003::2
        esp mode=tunnel spi=2004(0x000007d4) reqid=0(0x00000000)
        E: des-cbc  6d795f6b 65795f31
```

```
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 22 16:26:47 2007
        diff: 539835(s) hard: 0(s)      soft: 0(s)
        last: Feb 22 16:25:58 2007      hard: 0(s)      soft: 0(s)
        current: 132(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 1    hard: 0 soft: 0
        sadb_seq=3 pid=5823 refcnt=0
2003::1 2003::2
        esp mode=transport spi=2002(0x000007d2) reqid=3(0x00000003)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 22 16:26:47 2007
        diff: 539835(s) hard: 0(s)      soft: 0(s)
        last: Feb 22 16:26:00 2007      hard: 0(s)      soft: 0(s)
        current: 1092(bytes)    hard: 0(bytes)  soft: 0(bytes)
        allocated: 13   hard: 0 soft: 0
        sadb_seq=2 pid=5823 refcnt=0
2003::1 2003::2
        esp mode=transport spi=2000(0x000007d0) reqid=1(0x00000001)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 22 16:26:47 2007
        diff: 539835(s) hard: 0(s)      soft: 0(s)
        last: Feb 22 16:25:57 2007      hard: 0(s)      soft: 0(s)
        current: 42984(bytes)   hard: 0(bytes)  soft: 0(bytes)
        allocated: 398  hard: 0 soft: 0
        sadb_seq=1 pid=5823 refcnt=0

2003::2 2005::212:3fff:feae:205b
        esp mode=tunnel spi=2005(0x000007d5) reqid=0(0x00000000)
        E: des-cbc  6d795f6b 65795f31
        A: hmac-sha1  74686973 20697320 74686520 74657374 206b6579
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Feb 16 10:29:32 2007   current: Feb 22 16:26:47 2007
        diff: 539835(s) hard: 0(s)      soft: 0(s)
        last: Feb 22 16:25:58 2007      hard: 0(s)      soft: 0(s)
        current: 64(bytes)      hard: 0(bytes)  soft: 0(bytes)
        allocated: 1    hard: 0 soft: 0
        sadb_seq=0 pid=5823 refcnt=0
```

Figure 22. MN SPD Output after MN Moves to the Foreign Network

Figure 23 illustrates the whole RR procedure, protected by IPsec, as captured on the HA-FA link.

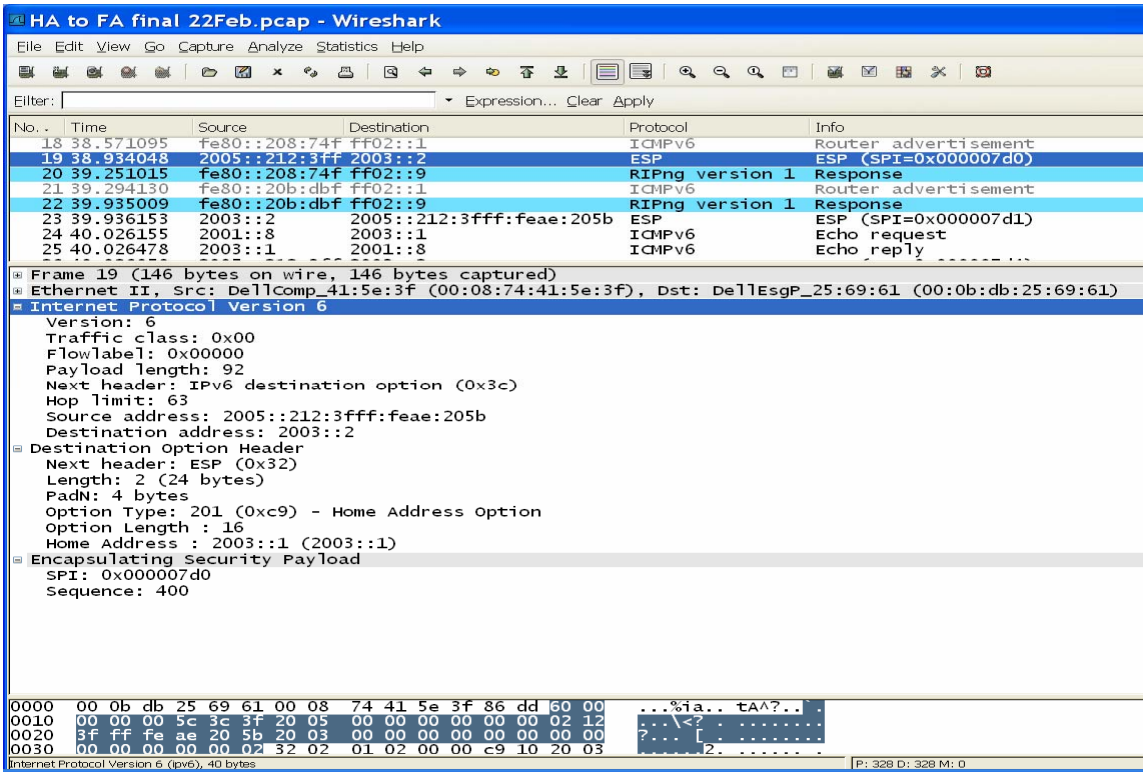

Figure 23. Ethereal Screen Capture of RR Procedure

Packet #19 is sent from the CoA of the MN to the HA and it is protected by ESP. This packet is the BU of MN for its new CoA.

In packet #23, the HA acknowledges the BU (BA).

Packet #26 is sent from the CoA of the MN to the HA and it is also protected by ESP. Almost at the same time, sends a COTI message (packet#27) to the CN. It can be safely assumed that packet #26 is the protected by ESP HOTI message.

Packet#28 is the protected by ESP, HOT message.

Packet#29 is the COT message.

Packet#30 is a binding update (BU) from the CoA of the MN to the CN.

Packet#30 is the Binding Acknowledge from the CN to the CoA of the MN.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SECURITY ISSUES OF MOBILE IPV6

This chapter presents possible threats against the Return Routability protocol and discusses the effectiveness of the security mechanisms provided by the protocol against these attacks. Alternative proposals for better securing the protocol are also described.

A. IDENTIFIED SECURITY THREATS AND MIPV6 PROTOCOL DEFENCE

The following table summarizes the possible attacks against the MIPv6 protocol and how the protocol mitigates these attacks as described in [Aura06]:

Threat	Solution provided by RR procedure [Aura06]
False BU Attack (attacker is neither on the CN-HoA route nor within the local network of CN)	Signaling messages between the MN and its HA are encrypted via IPsec and authenticated
Connection Hijack Attack	Session protected by the mobility header used for the BU message , that includes the BU, a nonce indices option and a binding authorization option.
Bombing Attack	Case 1: Target of the attack is an existent IP address The IP layer at target during the process of the decapsulation of the routing header encounters the “unknown” address of the attacker and drop the packets without passing it to the transport layer. Case 2: Target of the attack is a non-existent IP address The router of the subnet of the non-existent IP address should send an ICMP Destination Unreachable message to the sender of the unwanted packets.

Threat	Solution provided by RR procedure [Aura06]
Replay Attack	The Sequence Number of the ESP header provides anti-replay protection for the packet. The sequence number is 32-bit long starting from 1) which provides an unique id for each packet sent over the quick mode security association for the communication. The sequence number cannot repeat for the life of the quick mode security association. The receiver checks this field to verify that a packet for a security association with this number has not already been received. If one has been received, the packet is rejected. [http://technet2.microsoft.com/WindowsServer/en/library/c3a956bf-704b-4980-9655-762985e380f61033.mspx?mfr=true Last visited on March 15, 2007]
CPU Exhaustion	RR protocol uses relatively inexpensive encryption and one-way hash functions and the consumption of CPU power is not a major concern.
State-storage Exhaustion	The CN does not store a separate key for each MN. Instead it stores a single periodically-changing randomly generated master secret (Kcn) and computes the two keygen tokens with a one-way function from the master secret and from HoA and CoA Home keygen token = First (64, HMAC_SHA1(Kcn, home address|nonce|0)) and Care-of keygen token = First(64, MAC (Kcn, care-of address | nonce|1))
Amplification Attack (Force the CN to respond with a # of packets being significantly larger than that of the query.)	The CN waits to receive both HOTI and COTI messages from the MN. Then, it replies to MN with only two messages, HOT and COT, sending only as many messages as it receives, thus eliminating the amplification problem.
Reflection	The CN responds always to the same address from which it receives a message.

Threat	Solution provided by RR procedure [Aura06]
HOT, COT and BA spoofing	Usage of nonces (home init cookie-Care-of init cookie) in the HOTI and COTI messages, which the CN copies to the HOT and COT messages, respectively.
Insider attack from CN local network	Can be mitigated if the CN is also a MN of a HA. In that way all keygen tokens are protected by IPsec(ESP).

Table 4. Possible Threats and Defense Mechanisms provided by the RR Protocol

B. TEST BED SECURITY OBSERVATIONS

1. It is clearly stated in [Aura06] that the RR protocol does not defend against an attacker who can monitor the CN-HA data path. This threat can be mitigated if the CN is a MN itself.
2. A HA router can be easily identified, since it transmits on the clear (inside the router advertisements) its identity of being a HA.
3. The files that determine the security associations are stored in both the routers and the MN's file system. Routers are generally well protected, but the file security at MN depends on the user behavior. In cases where an attacker has broken into the MN's file system, the attacker can acquire the security association configuration file, which contains the security credentials and all the necessary information to implement any kind of attack against the MN.
4. The last 64 bits of the MN's IP CoA are predictable, since the CoA is configured according to the attached router advertisements due to Stateless DHCP (not a major concern since can be mandated to use statefull DHCP).

C. ATTACK TRAFFIC GENERATION WITH SCAPY6

Scapy [http://www.secdev.org/projects/scapy/ Last visited on March 8, 2007] is a powerful interactive packet manipulation program. Scapy6 was the IPv6 version of Scapy, which is now named mip6 (executable file mip6.py). It is written in Python and runs natively on Linux, and on most Unix systems with libpcap, libdnet and their respective

python wrapper. The scapy6.py and mip6.py files that can be used to construct mobile ipv6 packets can be found at [http://namabiiru.hongo.wide.ad.jp/scapy6/ Last visited on March 12, 2007]. Detailed examples of MIPv6 packet construction can be found in: [http://namabiiru.hongo.wide.ad.jp/scapy6/uts/mip6-test-report.html Last visited on March 12, 2007].

For this thesis, Scapy was used for the construction of one “bogus” BU message. Our goal was to inject a storm of fake BU to the CN from a machine that impersonates the MN during a MIPv6 MN-CN session.

Specifically, the attack machine was first attached to the 2000::/64 interface of the CNrouter, having in mind that the attack should be implemented from outside the MN-CN path. It then flooded the CN with 4000 fake BU’s, during a Mobile IPv6 session between the MN and the CN. The flood attack was unsuccessful, as it was expected. The performance of the ongoing session between the MN and the CN was not affected at all during the attack, demonstrating the MIPv6 protocol’s effectiveness against false BUs.

Details of how the fake BU was built using Scapy6 are given in Appendix F.

D. WORK IN PROGRESS FOR SECURING THE ROUTE OPTIMIZATION PROCEDURE FOR MOBILE IPV6

There is a lot of work in progress for enhancing the Mobile IPv6 Route Optimization procedure.

Vogt and Arkko in [Vogt07] describe and evaluate strategies to enhance Mobile IPv6 Route Optimization, on the basis of existing proposals, in order to motivate and guide further research in this context.

Dupont in [Depont 07] analyses some new kinds of reflection attacks, as known as 3rd party bombing introduced by Mobile IPv6 and makes some recommendations in order to protect the MIPv6 protocol.

Li, et al., in [Li07] puts forward a mechanism called ECC RRP, which enhances the key management for the Return Routability Procedure with anonymous Elliptic Curve Cryptography (ECC) Key Agreement Protocol. The proposed solution is not prone to attacks by nodes on the route from CN to HA in way that the RR process described in [RFC 3775] and reduces the latency related to handovers that require new binding updates.

Mun, et al., in [Mun06] propose an optimized scheme which performs Routing Optimization using the AAA infrastructure between the HA and a CN instead of the RR procedure.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

The main objective of this research effort was to build a test bed for investigating the vulnerabilities of the Mobile IPv6 RR procedure. The test bed should facilitate the enactment and analysis of the effects of specific threats on the Mobile IPv6 hosts and the network. While this thesis is not about discovering new vulnerabilities or evaluating countermeasures, the resulting test bed and software has laid the necessary groundwork for future research in those directions.

It must be mentioned that RR is not the only way to secure MIPv6 messages. Alternatively, MIPv6 can be secured using public keys and certificates or Cryptographic Generated Addresses (CGAs), having always in mind that there is always a trade-off between convenience and security.

A. CONCLUSIONS

The major findings from this thesis research are:

- When stateless DHCP is used, the last 64 bits of the MN's IP CoA is predictable (EUI64), since the CoA is configured according to the attached router advertisements. If an Attacker knows a node's MAC address, he can violate its location privacy (the prefix would give out its location in the Internet). Stateful DHCP is mandated for location privacy. Stateless DHCP is more convenient for private or campus networks that are protected by firewalls, covering the issue of location privacy.
- The RR procedure uses IPsec for authentication, with relatively inexpensive encryption and one-way hash functions. The consumption of CPU power is not a major concern (which is extremely important for mobile devices). There is always the solution of using PKI, but the processing

overhead of existing PKI solutions is particularly noticeable within the context of mobile handheld devices, such as PDAs (Personal Digital Assistants) or mobile/smart phones, which have certain limitations regarding storage and computational capacity because of their small size.

- The security of the MIPv6 protocol is yet to be “battle-tested”. Most existing MIPv6 implementations are still intended for research and development purposes. There are many different implementations, as indicated in Chapter 2, and the protocol itself is under continuous improvement and refinement. There is not yet any Microsoft OS implementation of MIPv6 protocol.

B. FUTURE WORK

The main remaining vulnerability in the RR protocol is that an attacker, in the same local network as the correspondent node, may be able to intercept and spoof all of the BU protocol messages [Aura06]. The present test bed can be extended in such a way that both MN and CN be MNs belonging to their respective home agents. Theoretically, such a configuration would protect all RR messages with IPsec. It would be interesting to see how the protocol would react and defend itself in such an occasion against the possible attacks. That way, the two communicating mobile nodes can securely optimize the routing between their care-of addresses regardless of any potential attacker on the current access networks [Aura06].

Currently, Scapy6 does not support the construction of HOTI, COTI, HOT, and COT packets. An extension to Scapy6 to support all kinds of RR messages would give the opportunity to MIPv6 developers to evaluate the security threats of MIPv6 protocol under realistic simulation conditions. The implementation of all identified threats against the RR

procedure, using Scapy6 or other software, would help evaluate the effectivenss of the proposed solution of IPsec for threat mitigation.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. CONFIGURATION FILES OF HA

The following table contains the configuration files of HA along with their locations.

Configuration File	Location	Functionality
radvd.conf	/etc/	Advertise the prefixes of the Router's interfaces
ripngd.conf	/etc/quagga/	Implements the standard Routing Information Protocol for IPv6
zebra.conf	/etc/quagga	Along with ripngd daemon provides IPv6 routing for the test bed
mip6d.conf	/usr/local/sbin	MIPv6 configuration file used by mip6d daemon
sa.conf	/etc/	Security Association configuration file

radvd.conf

```
#This is the radvd.conf file of the HA
#Location: /etc/radvd.conf
#This daemon advertises the prefixes of the interfaces of the HA
router;
#Note that only the interface 0 has the option AdvHomeAgentFlag on;
 interface eth0
     {
        AdvSendAdvert on;
        MaxRtrAdvInterval 3;
        MinRtrAdvInterval 1;
        AdvIntervalOpt on;
        AdvHomeAgentFlag on;
        HomeAgentLifetime 10000;
        HomeAgentPreference 20;
        AdvHomeAgentInfo on;
        prefix 2003::2/64
        {
          AdvRouterAddr on;
          AdvOnLink on;
          AdvAutonomous on;
          AdvPreferredLifetime 10000;
```

```
          AdvValidLifetime 12000;
        };
      };
  interface eth1
     {
        AdvSendAdvert on;
        MaxRtrAdvInterval 10;
        MinRtrAdvInterval 3;
        AdvIntervalOpt on;
        prefix 2004::2/64
        {
          AdvRouterAddr on;
          AdvOnLink on;
          AdvAutonomous on;
        };
      };
interface eth2
     {
        AdvSendAdvert on;
        MaxRtrAdvInterval 10;
        MinRtrAdvInterval 3;
        AdvIntervalOpt on;
        prefix 2002::2/64
        {
          AdvRouterAddr on;
          AdvOnLink on;
          AdvAutonomous on;
        };
      };
```

ripngd.conf

```
!This is the ripngd.conf file of the HA
!Location:/etc/quagga/ripngd.conf
!This daemon implements the standard Routing Information
! Protocol for IPv6
hostname quagga
password quagga
!
router ripng
  network eth0
  network eth1
  network eth2
  redistribute connected
  redistribute static
  redistribute kernel
!
```

zebra.conf

```
!This is the zebra.conf file of the HA
!Location:/etc/quagga/zebra.conf
!This daemon along with ripngd provide the
!tesbed with IPv6 routing
hostname quagga
password quagga
enable password quagga
```

```
log file /var/log/quagga/quagga.log
!
interface eth0
 ipv6 address 2003::2/64
 ipv6 nd prefix 2003::/64
 no ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface eth1
 ipv6 address 2004::2/64
 ipv6 nd prefix 2004::/64
 ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface eth2
 ipv6 address 2002::2/64
 ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface lo
!
interface sit0
 ipv6 nd suppress-ra
!
ip forwarding
ipv6 forwarding
ipv6 route 2001::/64  2002::1
ipv6 route 2005::/64  2004::3
!
line vty
!
```

mip6d.conf

```
# This is the mip6d.conf file of the HA
# Location: /etc/mip6d.conf
# Mobile IPv6 configuration file: Home Agent
# This file provides MIPv6 functionality to the HA
# via daemon mip6d

         # filename: /etc/mip6d.conf
         NodeConfig HA;
         ## If set to > 0, will not detach from tty
         DebugLevel 10;

         ## List of interfaces where we serve as Home Agent
         Interface "eth0";
         ##
         ## IPsec configuration
         ## Use when IPsec is not enabled
         ## UseMnHaIPsec disabled;
         ## Use when IPsec is enabled
          UseMnHaIPsec enabled;
         ## Define the set of IPsec Policy
          IPsecPolicySet {
                HomeAgentAddress 2003::2;
                HomeAddress 2003::1/64;
```

```
                IPsecPolicy HomeRegBinding UseESP 1 2;
                IPsecPolicy MobPfxDisc UseESP 3;
                IPsecPolicy TunnelMh UseESP;
        }
```

sa.conf

```
# This is the Security Association configuration  file of HA
# Note that an exact copy of this file resides also at MN
# To activate run: #setkey –f /etc/sa.conf
# Location:/etc/sa.conf
#------------------------------
# 2003::1 is home address of MN
# 2003::2 is address of HA
#des-cbc key should be 8 characters long
#hmac-sha1 key should be 20 characters long

flush;

# MN -> HA transport SA for BU
 add 2003:0:0:0::1 2003:0:0:0::2 esp 2000
       -u 1
       -m transport
       -E des-cbc "my_key_1"
       -A hmac-sha1 "this is the test key" ;

# HA -> MN transport SA for BA
 add 2003:0:0:0::2 2003:0:0:0::1 esp 2001
      -u 2
        -m transport
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# MN -> HA transport SA for MPS
 add 2003:0:0:0::1 2003:0:0:0::2 esp 2002
      -u 3
        -m transport
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# HA -> MN transport SA for MPA
 add 2003:0:0:0::2 2003:0:0:0::1 esp 2003
      -u 3
        -m transport
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# MN -> HA tunnel SA for HoTI
 add 2003:0:0:0::1 2003:0:0:0::2 esp 2004
        -m tunnel
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;
```

```
# HA -> MN tunnel SA for HoT
 add 2003:0:0:0::2 2003:0:0:0::1 esp 2005
        -m tunnel
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. CONFIGURATION FILES OF MN

The following table contains the configuration files of MN along with their locations.

Configuration File	Location	Functionality
mip6d.conf	/usr/local/sbin	MIPv6 configuration file used by mip6d daemon
sa.conf	/etc/	Security Association configuration file

mip6d.conf

```
# This is the mip6d.conf file of the MN
# Location: /etc/mip6d.conf
# Mobile IPv6 configuration file: Mobile Node
# This file provides MIPv6 functionality to the MN
# via daemon mip6d
        #
        # filename: /etc/mip6d.conf

        NodeConfig MN;

        ## If set to > 0, will not detach from tty
        DebugLevel 10;
        ## Enable RO
        DoRouteOptimizationMN enabled;
        DoRouteOptimizationCN enabled;

        MnDiscardHaParamProb enabled;
        Interface "eth0";
        MnHomeLink "eth0" {
                HomeAgentAddress 2003::2;
                HomeAddress 2003::1/64;
        }
        ## IPsec configuration
        ##
        UseMnHaIPsec enabled;
        IPsecPolicySet {
                HomeAgentAddress 2003::2;
                HomeAddress 2003::1/64;
                IPsecPolicy HomeRegBinding UseESP1 2;
                IPsecPolicy MobPfxDisc UseESP 3;
                IPsecPolicy TunnelMh UseESP;
        }
```

sa.conf

```
# This is the Security Association configuration file of MN
# Note that an exact copy of this file resides also at HA
# To activate run: #setkey –f /etc/sa.conf
# Location:/etc/sa.conf
#------------------------------
# 2003::1 is home address of MN
# 2003::2 is address of HA
#des-cbc key should be 8 characters long
#hmac-sha1 key should be 20 characters long

flush;

# MN -> HA transport SA for BU
 add 2003:0:0:0::1 2003:0:0:0::2 esp 2000
       -u 1
       -m transport
       -E des-cbc "my_key_1"
       -A hmac-sha1 "this is the test key" ;

# HA -> MN transport SA for BA
 add 2003:0:0:0::2 2003:0:0:0::1 esp 2001
      -u 2
        -m transport
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# MN -> HA transport SA for MPS
 add 2003:0:0:0::1 2003:0:0:0::2 esp 2002
      -u 3
        -m transport
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# HA -> MN transport SA for MPA
 add 2003:0:0:0::2 2003:0:0:0::1 esp 2003
      -u 3
        -m transport
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# MN -> HA tunnel SA for HoTI
 add 2003:0:0:0::1 2003:0:0:0::2 esp 2004
        -m tunnel
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;

# HA -> MN tunnel SA for HoT
 add 2003:0:0:0::2 2003:0:0:0::1 esp 2005
        -m tunnel
        -E des-cbc "my_key_1"
        -A hmac-sha1 "this is the test key" ;
```

APPENDIX C. CONFIGURATION FILES OF CNROUTER

The following table contains the configuration files of CN along with their locations.

Configuration File	Location	Functionality
mip6d.conf	/usr/local/sbin	MIPv6 configuration file used by mip6d daemon

mip6d.conf

```
# This is the mip6d.conf file of the CN
# Location: /etc/mip6d.conf
# Mobile IPv6 configuration file: Correspondent Node
# This file provides MIPv6 functionality to the CN
# via daemon mip6d
NodeConfig CN;
## If set to > 0, will not detach from tty
DebugLevel 10;
## Enable RO
DoRouteOptimizationCN enabled;
UseCnBuAck enabled;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. CONFIGURATION FILES OF FROUTER

The following table contains the configuration files of Frouter along with their locations.

Configuration File	Location	Functionality
radvd.conf	/etc/	Advertise the prefixes of the Router's interfaces
ripngd.conf	/etc/quagga/	Implements the standard Routing Information Protocol for IPv6
zebra.conf	/etc/quagga	Along with ripngd daemon provides IPv6 routing for the test bed

radvd.conf

```
# This is the radvd.conf file of the Frouter
# Location: /etc/radvd.conf
# This daemon advertises the prefixes of the interfaces of the Frouter
        interface eth0
    {
          AdvSendAdvert on;
          AdvIntervalOpt on;
          MinRtrAdvInterval 3;
          MaxRtrAdvInterval 10;
          AdvHomeAgentFlag off;
          prefix 2004::/64
          {
              AdvOnLink on;
              AdvAutonomous on;
              AdvRouterAddr on;
          };
    };
    interface eth1
    {
          AdvSendAdvert on;
          AdvIntervalOpt on;
          MinRtrAdvInterval 3;
          MaxRtrAdvInterval 10;
          AdvHomeAgentFlag off;
          prefix 2005::/64
          {
              AdvOnLink on;
              AdvAutonomous on;
```

```
            AdvRouterAddr on;
        };
    };
```

ripngd.conf

```
!This is the ripngd.conf file of the Frouter
!Location:/etc/quagga/ripngd.conf
!This daemon implements the standard Routing Information
! Protocol for IPv6
hostname quagga
password quagga
!
router ripng
  network eth0
  network eth1
  redistribute connected
  redistribute static
  redistribute kernel
!
```

zebra.conf

```
!This is the zebra.conf file of the Frouter
!Location:/etc/quagga/zebra.conf
!This daemon along with ripngd provide the
!tesbed with IPv6 routing
hostname quagga
password quagga
enable password quagga
log file /var/log/quagga/quagga.log
!
interface eth0
 ipv6 address 2004::3/64
 ipv6 nd prefix 2004::/64
 ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface eth1
 ipv6 address 2005::3/64
ipv6 nd prefix 2005::/64
no ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface lo
!
interface sit0
 ipv6 nd suppress-ra
!
ip forwarding
ipv6 forwarding
ipv6 route 2001::/64 2004::2
ipv6 route 2002::/64 2004::2
ipv6 route 2003::/64 2004::2
!
line vty
!
```

APPENDIX E. CONFIGURATION FILES OF CNROUTER

The following table contains the configuration files of CNrouter along with their locations.

Configuration File	Location	Functionality
radvd.conf	/etc/	Advertise the prefixes of the Router's interfaces
ripngd.conf	/etc/quagga/	Implements the standard Routing Information Protocol for IPv6
zebra.conf	/etc/quagga	Along with ripngd daemon provides IPv6 routing for the test bed

radvd.conf

```
#This is the radvd.conf file of the CNrouter
#Location: /etc/radvd.conf
#This daemon advertises the prefixes of the interfaces of the CNrouter
interface eth0
    {
          AdvSendAdvert on;
          AdvIntervalOpt on;

          MinRtrAdvInterval 3;
          MaxRtrAdvInterval 10;
          AdvHomeAgentFlag off;

          prefix 2002::1/64
          {
              AdvOnLink on;
              AdvAutonomous on;
              AdvRouterAddr on;
          };
    };

interface eth1
    {
          AdvSendAdvert on;
          AdvIntervalOpt on;

          MinRtrAdvInterval 3;
          MaxRtrAdvInterval 10;
          AdvHomeAgentFlag off;
```

```
        prefix 2000::1/64
        {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
        };
    };

 interface eth2
    {
        AdvSendAdvert on;
        AdvIntervalOpt on;

        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        AdvHomeAgentFlag off;

        prefix 2001::1/64
        {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
        };
    };
```

ripngd.conf

```
!This is the ripngd.conf file of the CNrouter
!Location:/etc/quagga/ripngd.conf
!This daemon implements the standard Routing Information
! Protocol for IPv6
hostname quagga
password quagga
!
router ripng
 network eth0
 network eth1
 network eth2
  redistribute connected
  redistribute static
  redistribute kernel
!
```

zebra.conf

```
!This is the zebra.conf file of the CNrouter
!Location:/etc/quagga/zebra.conf
!This daemon along with ripngd provide the
!tesbed with IPv6 routing
!
! Zebra configuration saved from vty
!   2007/01/23 13:46:08
!
hostname quagga
password quagga
```

```
enable password quagga
log file /var/log/quagga/quagga.log
!
interface eth0
 ipv6 address 2002::1/64
 ipv6 nd prefix 2002::/64
 ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface eth1
 ipv6 address 2000::1/64
 ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface eth2
 ipv6 address 2001::1/64
 ipv6 nd prefix 2001::/64
 no ipv6 nd suppress-ra
 ipv6 nd ra-interval 10
!
interface lo
!
interface sit0
 ipv6 nd suppress-ra
!
ip forwarding
ipv6 forwarding
ipv6 route 2003::/64 2002::2
ipv6 route 2004::/64 2002::2
ipv6 route 2005::/64 2002::2
!
line vty
!
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. USING SCAPY6 FOR CONSTRUCTING A BU MESSAGE

This appendix describes the way that a BU message is constructed, using mip6.py (the IPv6 version of Scapy). It is impressing that a complicated message like a mobile IPv6 BU constructed with a few commands (/*Comments*/).

/* b1 is an IPv6 packet*/

```
>>> b1=IPv6()
```

/*The source address of the packet is */
/*2000::212:3fff:fead:f081*/

```
>>> b1.src='2000::212:3fff:fead:f081'
```

/*The destination address is 2001::8*/

```
>>> b1.dst='2001::8'
```

/*An IPv6 header has been constructed */

```
>>> b1.show()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= 0
  nh= No Next Header
  hlim= 64
  src= 2000::212:3fff:fead:f081
  dst= 2001::8 [Teredo srv: 0.0.0.0 cli: 255.255.255.247:65535]
```

/*An Extension Header (Destination Options Header) with */
/* the Home Address Option is needed*/

```
>>> b2=IPv6ExtHdrDestOpt(options=[HAO(hoa='2003::1')])
```

/*b3 is the concatenated result of b1 and b2 headers*/

```
>>> b3=b1/b2
```

/*This is the result of the concatenation*/

```
>>> b3.show()
###[ IPv6 ]###
  version= 6
```

```
  tc= 0
  fl= 0
  plen= 0
  nh= Destination Option Header
  hlim= 64
  src= 2000::212:3fff:fead:f081
  dst= 2001::8 [Teredo srv: 0.0.0.0 cli: 255.255.255.247:65535]
###[ IPv6 Extension Header - Destination Options Header ]###
     nh= No Next Header
     len= 0
     autopad= On
     \options\
      |###[ Home Address Option ]###
      |  otype= Home Address Option [11: discard+ICMP not mcast, 0:
Don't change en-route]
      |  optlen= 16
      |  hoa= 2003::1
```

/*Nonce Indices Option is created*/

```
>>> b4=NonceIndices( olen=4,hni=4, coni=4)
```

/*Binding Authorization Data is created*/

```
>>> b5=BindingAuthData( authenticator=2807)
```

/*A Mobility Header of Binding Update is created with */
/*the options of Nonce Indices and Binding Authorization*/
/*Data */

```
>>> b6=IPv6MobHdrBU( options=[b4,b5])
```

/*The final message is the concatenation of b3 */
/*and b5 Headers*/

```
>>> b_final=b3/b6
```

/*The final outcome of our MIPv6 BU construction*/

```
>>> b_final.show()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= 0
  nh= Destination Option Header
  hlim= 64
  src= 2000::212:3fff:fead:f081
  dst= 2001::8 [Teredo srv: 0.0.0.0 cli: 255.255.255.247:65535]
###[ IPv6 Extension Header - Destination Options Header ]###
     nh= Mobility Header
     len= 0
     autopad= On
     \options\
      |###[ Home Address Option ]###
```

```
      |  otype= Home Address Option [11: discard+ICMP not mcast, 0:
Don't change en-route]
      |  optlen= 16
      |  hoa= 2003::1
###[ IPv6 Mobility Header - Binding Update ]###
        nh= No Next Header
        len= 0
        mhtype= BU
        reserved= 0
        cksum= 0x0
        seq= 0x4242
        flags= AMR
        reserved= 0x0
        mhtime= 0x3
        autopad= On
        \options\
         |###[ Mobile IPv6 - Nonce Indices ]###
         |  otype= 4
         |  olen= 4
         |  hni= 4
         |  coni= 4
         |###[ Mobile IPv6 - Binding Authorization Data ]###
         |  otype= 5
         |  olen= 16
         |  authenticator= 2807
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[Arkko04] J. Arkko, et al., “Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents,” **RFC 3776**, June 2004.

[Arkko05] J. Arkko, Ed., J. Kempf, B. Zill, P. Nikander, “SEcure Neighbor Discovery (SEND),” **RFC 3971**, March 2005.

[Arkko06] J. Arkko, “A Taxonomy and Analysis of Enhancements to Mobile IPv6 Route Optimization,” **RFC4651**, August 2006.

[Aura05] T. Aura., “Cryptographically Generated Addresses (CGA),” **RFC 3972**, March 2005.

[Aura06] T. Aura, M. Roe,” Designing the Mobile IPv6 Security Protocol,” Microsoft Technical Report MSR-TR-2006-42, April 2006.

[Blanchet06] Marc Blanchet, “Migrating to IPv6: A Practical Guide to Implementing IPv6 in Mobile and Fixed Networks,” John Wiley & Sons, Ltd., January 2006.

[Davies02] Joseph Davies, “Understanding IPv6,” Microsoft Press, November 2002.

[Deering98] S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” **RFC 2460**, December 1998.

[Devarapalli05] V. Devarapalli, et al., “Network Mobility (NEMO) Basic Support Protocol,” **RFC 3963**, January 2005.

[Dunmore05] M .Dunmore (6net) “Final MIPv6 Support Guide,” February 8, 2005.

[Giaretta06] G. Giaretta, Ed. Patel, “Problem Statement for bootstrapping Mobile IPv6 (MIPv6),” **RFC 4640**, September 2006.

[Hinden03] R. Hinden, S. Deering., “Internet Protocol Version 6 (IPv6) Addressing Architecture,” **RFC 3513**, April 2003.

[Johnson04] D. Johnson, C. Perkins, J. Arkko, “Mobility Support in IPv6,” **RFC 3775**, June 2004.

[Keeni06] G. Keeni, et al., “Mobile IPv6 Management Information Base,” **RFC 4295**, April 2006.

[Kent05] S. Kent, “IP Authentication Header,” **RFC 4302**, December 2005.

[Kent05] S. Kent, “IP Encapsulating Security Payload (ESP),” **RFC 4303**, December 2005.

[Koodli05] R. Koodli, Ed.,”Fast Handovers for Mobile IPv6,” RFC 4068, July 2005.

[Kui04] R. Kui et al., ”Routing Optimization Security in Mobile IPv6,” March 2004.

[Lawrence04] Lawrence Stewart, Mai Banh, Grenville Armitage,” Implementing an IPv6 and Mobile IPv6 test bed using FreeBSD 4.9 and KAME,” March 2004.

[Nikander05] P. Nikander, J. Arkko, T. Aura, G. Montenegro, and E. Nordmark, "Mobile IP Version 6 Route Optimization Security Design Background," **RFC 4225**, December 2005.

[Nodmark05] E. Nordmark, T. Li, “Threats Relating to IPv6 Multi-homing Solutions”, **RFC 4218**, October 2005.

[Patel05] Patel, et al., ”Mobile Node Identifier Option for Mobile IPv6 (MIPv6),” **RFC 4283**, November 2005.

[Patel06] Patel, et al., “Authentication Protocol for Mobile IPv6,” **RFC 4285**, January 2006.

[Perkins06] C. Perkins, “Securing Mobile IPv6 Route Optimization Using a Static Shared Key,” **RFC 4449**, June 2006.

[Seo05] S. Kent, K. Seo, “Security Architecture for the Internet Protocol,” RFC 4301, December 2005.

[Soliman04] Hesham Soliman, “Mobile IPv6,” Addison-Welsey, April 2004.

[Soliman05] H. Soliman, et al., “Hierarchical Mobile IPv6 Mobility Management (HMIPv6)”, **RFC 4140**, August 2005.

[Thomson98] S. Thomson, T. Narten, “IPv6 Stateless Address Autoconfiguration,” **RFC 2462,** December 1998**.**

[Dupont07] F. Dupont, “A note about 3rd party bombing in Mobile IPv6,” draft-dupont-mipv6-3bombing-05.txt, Work in Progress, January 2007.

[Li07] Chunqiang. Li, Fuyou. Miao,.Madjid. Nakhjiri, “An enhancement of Mobile IPv6 Return Routability Procedure using Elliptic Curve Cryptography Key agreement Protocol,” draft-li-mipshop-mip6rrp-ecc-00.txt, Work in Progress, February 2007.

[Mun06] Youngsong Mun, Seonggeun Ryu, Jaehoon Nah, Seungwon Sohn, “An Enhanced Mobile IPv6 Handover for Roaming between Administrative Domains Based on AAA,” draft-mun-mipshop-emipv6-aaa-01.txt,Work in Progress, December 2006.

[Vogt07] C.Vogt, J. Arkko, “A Taxonomy and Analysis of Enhancements to Mobile IPv6 Route Optimization,” **RFC 4651**, February 2007.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Geoffrey Xie
   Naval Postgraduate School
   Monterey, California

4. Professor John Fulp
   Naval Postgraduate School
   Monterey, California

5. Neal Ziring
   National Security Agency
   Fort George G. Meade, Maryland

6. Matthew N. Smith
   National Security Agency
   Fort George G. Meade, Maryland