Enabling Mobile IP during the Transition from IPv4 to IPv6

Kuljaree Tantayakul

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2009

Thesis Title Enabling Mobile IP during the Transition from IPv4 to IPv6

Author Miss Kuljaree Tantayakul

Major Program Computer Engineering

Major Advisor	**Examining Committee:**
[illegible]	[illegible] Chairperson
(Mr. Robert Elz)	(Dr. Nittida Elz)
	[illegible]
Co-Advisor	(Mr. Robert Elz)
[illegible]	[illegible]
(Assoc. Prof. Dr. Sinchai Kamolphiwong)	(Assoc. Prof. Dr. Sinchai Kamolphiwong)
	[illegible]
	(Assoc. Prof. Thossaporn Kamolphiwong)
	Panita Pongpaibool
	(Dr. Panita Pongpaibool)

The Graduate School, Prince of Songkla University, has approved this thesis as partial fulfillment of the requirements for the Master of Engineering Degree in Computer Engineering

[illegible]

(Assoc. Prof. Dr. Krerkchai Thongnoo)
Dean of Graduate School

ชื่อวิทยานิพนธ์	การทำให้ Mobile IP สามารถทำงานได้ระหว่างการเปลี่ยนจากเครือข่าย IPv4 เป็น IPv6
ผู้เขียน	นางสาวกุลจรี ตันตยกุล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	2551

บทคัดย่อ

ปัจจุบันนี้ความต้องการใช้งานอินเตอร์เน็ตมีความต้องการมากขึ้น และมีบทบาทอย่างมากในชีวิตประจำวัน รวมทั้งการพัฒนาความสามารถของอุปกรณ์สื่อสารไร้สายและคอมพิวเตอร์แบบพกพา ส่งผลให้มีการเพิ่มความสามารถในการใช้งานอินเตอร์เน็ตในอุปกรณ์เหล่านี้ ซึ่งทำให้ IETF (Internet Engineering Task Force) ได้ออกแบบโปรโตคอลที่สนับสนุนการเคลื่อนที่ของผู้ใช้งานที่เรียกว่า Mobile IP

ขณะที่กำลังมีการเปลี่ยนแปลงการใช้งานจาก IPv4 ไปเป็น IPv6 เครื่องโมบายโหนดหรืออุปกรณ์พกพาอาจจะมีการเคลื่อนย้ายจากเครือข่าย IPv4 ไปยังเครือข่าย IPv6 หรือเคลื่อนที่จากเครือข่าย IPv6 ไปยังเครือข่าย IPv4 ซึ่งมีผลทำให้ไม่สามารถรักษาการเชื่อมต่อระหว่างเครื่องโมบายโหนดกับเครื่องอื่นๆที่กำลังติดต่ออยู่ได้

วิทยานิพนธ์นี้เสนอวิธีการที่ทำให้การเชื่อมต่อระหว่างเครื่องโมบายโหนดกับเครื่องอื่นๆสามารถดำรงอยู่ได้ขณะที่มีการเคลื่อนย้ายจากเครือข่าย IPv4 ไปยังเครือข่าย IPv6 และจากเครือข่าย IPv6 ไปยังเครือข่าย IPv4 โดยใช้กลไกการเปลี่ยนและทำงานร่วมระหว่าง IPv4 และ IPv6 ที่มีอยู่

คำสำคัญ: IPv4, IPv6, Mobile IP, Mobile IPv4, Mobile IPv6, MN, CN, NAT-PT, DNS-ALG

Thesis Title	Enabling Mobile IP during the Transition from IPv4 to IPv6
Author	Miss Kuljaree Tantayakul
Major Program	Computer Engineering
Academic Year	2008

ABSTRACT

During the transition from IPv4 to IPv6 a mobile node may be required to move from an IPv4 only to an IPv6 only network, or vice versa. This causes problems for mobile IP nodes. In this paper we examine those problems and suggest some possible solutions to permit a mobile node to move between IP protocol versions.

Keywords: IPv4, IPv6, Mobile IP, Mobile IPv4, Mobile IPv6, MN, CN, NAT-PT, DNS-ALG

ACKNOWLEDGEMENT

I would like to thank my advisors, Mr. Robert Elz. He has devoted a lot of time to me and he has always been very patient with my questions. My co-advisors, Assoc. Prof. Dr. Sinchai Kamolphiwong, have helped and given several good opportunities for my work.

I would like to thank Assoc. Prof. Thossaporn Kamolphiwong, Dr. Nittida Elz and Dr. Panita Pongpaibool to review and give the comment this work to be better.

I would like to thank the Graduate School and the staff of Computer Engineering Department to give the scholarship and the convenient for me.

I would like to thank my friends in CNR and Graduate student room for their supports and assistances.

Finally, I would like to thank my parent for their love and everything.

Kuljaree Tantayakul

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

LIST OF ABBREVIATIONS AND SYSBOLS

BA	Binding Acknowledgement
BU	Binding Update
CN	Correspondent Node
CoA	Care-of Address
DHCP	Dynamic Host Configuration Protocol
HA	Home Agent
HoA	Home Address
ICMPv4	Internet Control Message Protocol version 4
ICMPv6	Internet Control Message Protocol version 6
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
KAME	KArigoME[1]
MH	Mobility Header
MIP	Mobile IP
MIPv4	Mobile IPv4
MIPv6	Mobile IPv6
MN	Mobile Node
PDA	Personal Digital Assistant
RA	Router Advertisement
RH	Route Header
RIPng	Routing Information Protocol next generation
RO	Route Optimization
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

[1] KArigoME means "turtle" in Japanese

CHAPTER 1

INTRODUCTION

This thesis began as an examination of the IPv4 to IPv6 transition issues, hoping to find ways to make this a simpler process. During that examination it was noticed that transition for Mobile IP was an issue largely forgotten. A mobile node may become connected to either an IPv4 or IPv6 network as it roams. To operate correctly, Mobile IP must support a mobile node moving between networks running different versions of IP. Thus this work became one of suggesting a possible solution to this problem. The improvement, or simplification of the transition process is as a result is achieved by enabling Mobile IP to function universally.

1.1 Significance and Reasons

The Internet uses Internet Protocol (IP) [1] addresses which are required to identify each host, network device, mobile phones, electronic devices and other equipment for enabling a variety of communications as required between people, between people and devices, and between devices. The use of the IPv4 protocol in mobile networks has considerably increased the number of hosts that can potentially access the global Internet. Because of the rapid growth of the Internet, the identifiers used to label nodes, that is IPv4 addresses, are running out which is becoming a problem for Internet. These problems are reduced by using Network Address Translator (NAT) [2] and Classless Inter-Domain Routing (CIDR) [3]. However, NAT breaks end-to-end connectivity in several applications and reduces flexibility. CIDR had an effect, but only in allowing a little more time before all IPv4 addresses are used. Other IPv4 problems for Internet are lack of security and the rapid growth of the size of the routing tables.

The Internet Engineering Task Force (IETF) designed the next generation protocol, now called Internet Protocol version 6 (IPv6) [4][5], to solve these problems and replace the current version of Internet Protocol, IPv4.

However, the migration of IPv4 to IPv6 cannot happen overnight. Rather, there will be a period of transition when both protocols are in use over the same infrastructure. The transition is not easy to accomplish because the Internet has a large number of hosts and network devices. Equipment and systems, and application software, must be extended to support IPv6. Because IPv6 addresses are longer than IPv4 addresses, a change in application data structures that embed IP addresses is required. In general, the current applications written for IPv4 need to rewrite or bridged to support IPv6. Training costs, plus uncertainty of risk, also inhibit universal IPv6 deployment, further slowing the transition.

Currently, the use of wireless mobile communication and mobile devices such as laptops and handheld devices is increasing. We can call each of these a Mobile Node (MN). Every MN needs to have mobility support. Security also needs to be considered because mobile devices often use wireless networking technique that is inherently less secure than wired communication, and because the act of moving is equivalent with replacing one node by another - it is important that this happens only when the node really has moved, and not merely because some other node claims to be the original node in a new location. MNs use the Mobile Internet Protocol (MIP) for mobility support [23]. Mobile IPv4 (MIPv4) [23] was designed for Internet Protocol version 4 (IPv4) then it was adapted to become Mobile IPv6 (MIPv6) [24] for IPv6. MIPv6 shares many ideas in mobile IPv4, but introduces some new features derived from new facilities available in IPv6. In the period of transition from IPv4 to IPv6, it is possible that an MN on a network which supports only IPv4 mobility, or only IPv6 mobility, might move to another network that supports only IPv4 mobility, or only IPv6 mobility, or both types of mobility. When a MN from a network that supports only one IP version moves to the other network, it needs to receive a new address from the foreign network to be its Core-of-Address (CoA) to use in the registration process with its Home Agent (HA). However as mobile IPv6 is not backward compatible with mobile IPv4, and the mobility issues with IPv4/IPv6 interconnected networks are largely ignored in the present standards (RFCs) and proposals (Internet Drafts) when a MN moves between network versions, its communications with its Correspondent Nodes (CNs) are likely to fail.

Therefore in this thesis, we propose a method to solve these problems. The proposed method is used to enable mobility support for mobility in both IPv4 and IPv6 networks while the MNs are communicating and to permit hand off into a different network version. We make use of an existing protocol translation mechanism to convert packet formats. We also use the services of a Domain Name System Application Layer Gateway (DNS-ALG) to provide address assignment services.

1.2 Objectives

1) To investigate and implement the IPv4/IPv6 Transition Mechanisms.
2) To investigate and implement Mobile IPv4 and IPv6
3) To suggest a possible solution to permit a Mobile Node to move between networks supporting only different IP protocol versions.
4) To design and implement an experimental network topology for testing and demonstrating the feasibility of this solution.

1.3 Advantages

1) It is useful to discover the available IPv4/IPv6 transition strategies.
2) The solution devised can be proposed as an alternative solution for IP mobility during the IP transition period, and used in the real world.
3) It could be a case study of mobility mechanisms between MIPv4 and MIPv6 studying the inter-operation of protocol translation and address assignment.
4) It may be lead to new ideas for developing mobility mechanisms between MIPv4 and MIPv6 by using existing translation techniques.

1.4 Scope of work

1) To survey and investigate the IPv4/IPv6 transition mechanism.
2) To investigate MIPv4 and MIPv6.
3) To find out a possible solution to permit MNs to move between IP protocol versions by using an existing transition mechanism.
4) Only a prototype implementation is expected, tested using just one translation mechanism and one DNS ALG implementation.
5) To verify and validate that the solution designed will enable the MNs to retain their connections and communication while they move between interconnected

networks supporting only different versions of the IP protocol.

1.5 Work plan

1) To survey and investigate the existing IPv4/IPv6 transition mechanism.
2) To survey and investigate MIPv4 and MIPv6.
3) To implement and test the functionality of MIPv4 and MIPv6.
4) To find and analysis the inter-operation of MIPv4 and MIPv6 protocols.
5) To design a possible solution for taking existing transition mechanisms and using them to allow connectivity of a MN when it moves to a different network protocol.
6) To design a test case for verification of this solution.
7) To setup the test bed network and implement the functionality of this mechanism in the mobile network components.
8) To analyze the result and form conclusion.
9) To write this report.

1.6 Outline

This document is composed of seven chapters: the Introduction, a Survey of IPv4/IPv6 Transition Mechanisms, an overview of Mobile IP, the design outline and explanation of the mobility extension proposal designed, detailed protocol design and explanation, a report on the implementation and testing performed, and Discussion and Conclusions.

The first chapter, this chapter, the Introduction, provides the motivation for the work performed and discusses the purpose and scope of this work. In addition, it presents a brief work plan used to schedule the work required. It also contains an outline of the remainder of the thesis, that is, this section.

The second chapter, IPv4/IPv6 Transition Mechanisms, gives an overview of transition mechanisms designed for the migration from IPv4 to IPv6, after a brief introduction to those protocols and a description of their differences. It describes the scope of each transition mechanism and analyzes each of them to discover its applicability and whether it can be useful to aid in the solution to our problem.

The third chapter, Mobile IP Overview, describes the principles of MIPv4 and MIPv6, including explaining the security requirements and mechanisms for mobile IP. It presents the problem of inter-operation of MIPv4 and MIPv6 domains, also a comparison of MIPv4 and MIPv6 in as much as that is relevant this work.

The fourth chapter, Mobility Mechanism between MIPv4 and MIPv6, presents the problem statement of the work, and then examines the issues that arise from the problem, the solutions adopted, and why, and what problems each of those solutions causes in turn. It presents a brief description of the solution proposed, concentrating particularly upon the reasoning that led to the result achieved.

The fifth chapter, Design, sets out the algorithms and procedures used to permit MNs to move between IP protocol versions. It gives details of packet formats and encoding methods that comprise the solution proposed here.

The sixth chapter, Implementation and Testing, provides details of the prototype implementation developed to test the solution designed and demonstrate its feasibility, and described the experimental network which is used for implementation and testing, It also presents the results of this work.

The seventh, and final, chapter, Discussion and Conclusion, analyzes and describes the limitations of the work, and makes suggestions for future work that is required to better make use of the solution proposed here, and to extend it to a better solution.

CHAPTER 2

IPv4/IPv6 TRANSITION MECHANISMS

This chapter gives an overview of IPv4 and IPv6 with emphasis on mechanisms intended for transition from IPv4 to IPv6. The first section explains why the IPv4/IPv6 transition mechanisms are required. Some of the techniques involved with the IPv4/IPv6 transitions will be described in section 2.2. Then we introduce the concept of Application Layer Gateways, and in particular, that for the Domain Name System (DNS-ALG) section 2.3. This is followed by a summary of the IPv4/IPv6 transition mechanism. In the final section of the chapter, we note one significant missing feature with current transition plans, related to maintaining connectivity for mobile nodes using the Mobile IP protocols.

2.1 Introduction

Today almost every access network uses the Internet Protocol, version 4, IPv4 [1], which has proved to be robust, easily implemented and interoperable. IPv4 provides a basic datagram network providing best effort datagram delivery between any two connected nodes. The end nodes (End Systems or hosts) are responsible for all higher level functionality, including adding reliability to the communications. Intermediate systems (or routers) form the center of the network, and are responsible only for directing datagrams toward their ultimate destination, and in some recent cases, providing some agreed quality of service.

The success of IPv4 has now become its downfall. The address space of IPv4 is a 32 bit identifier, providing an absolute maximum of 2^{32} host addresses. In practice the number is significantly smaller, a non-trivial fraction of the available address space is reserved for various special purposes, and of that available to be assigned to end nodes, the practice of making assignments only in blocks containing a power of two number of addresses causes much of the address space to be "wasted" - that is, assigned to nodes that do not actually exist, and unavailable for assignment elsewhere.

In addition to this, the routing tables, used by the intermediate systems, to identify paths to the end nodes have grown unwieldy. Their size causes problems particularly when new paths need to be calculated.

To overcome these problems, a new version of the Internet Protocol, version 6, IPv6 [4][5], has been developed. It was designed to replace the exhausted IPv4. IPv6 has a 128 bit address space that can provide a huge number of addresses, even assuming that a large fraction are wasted. It has a number of other advantages over IPv4. IPv6 address are planned to be structured hierarchically to simplify address delegation and routing. The IPv6 packet header been simplified, with little used functionality moved to optional extension headers. IPv6 also provides a powerful autoconfiguration mechanism to ease installation, and improve router discovery and detection of dead routers or unreachable neighbors on the link.

IPv6 also mandates provision of authentication and confidentiality mechanisms, Mobile IP functionality, multicast, and more - all functionality developed after the initial IPv4 protocols were long deployed, and which consequently are not always available with typical IPv4 implementations.

Designing a new fundamental protocol is relatively easy, having it universally implemented and deployed is a more challenging task. This will not happen overnight. There will be a period of transition when both IP protocols coexist and communicate over the same infrastructure. The transition is not easy to accomplish because the Internet has a large number of hosts and network devices. They must be extended to support IPv6. Also current software, applications and Operating Systems (OS) need to be upgraded to support IPv6. Because IPv6 addresses are longer than IPv4 addresses, a change in application data structures that embed IP addresses is required. In general, applications written for IPv4 need to rewritten or bridged to support IPv6. Aside from the software, the users, operators, administrators, all need to be trained to understand IPv6 and its differences from IPv4. The problems of migration to IPv6 need to be investigated and understood before migration to IPv6. What factors we should be consider before migration to IPv6 and how should an organization or site choose the suitable method for migration of their network? The IETF IPv6 Working Group (ngtrans) [7] discussed and designed the IPv4/IPv6 Transition Mechanism for using during a period of transition. Before we discuss the transition methods, we need a brief comparison of IPv4 and IPv6.

2.1.1Comparison between IPv4 header and IPv6 header

The IP header is the fundamental aspect of the IP packet or datagram that contains control information about how the packet can be delivered from its source to its destination. The format of the IPv4 header is shown in Figure 2.1. The header uses 13 fields to identify various control settings. The IPv4 header's total length is 20 octets (8 bits per octet) or five 32-bit words. The IPv6 header, as is shown in Figure 2.2, has only 8 fields with a fixed length of 40 octets. IPv6 expanded the IPv4 32-bit address space to a 128-bit address space. That change accounts for all of the growth of the header size, and is mitigated by the removal of some of the header fields. Aside from the number of bits, interpretation of IPv4 and IPv6 addresses is essentially identical. Even though the IPv6 header is longer than the IPv4 header, IPv6 makes the IP header more efficient. In the following we give a brief comparison of the two headers, comparing the various header fields in three groupings: Equivalent fields, Modified fields and Missing fields.

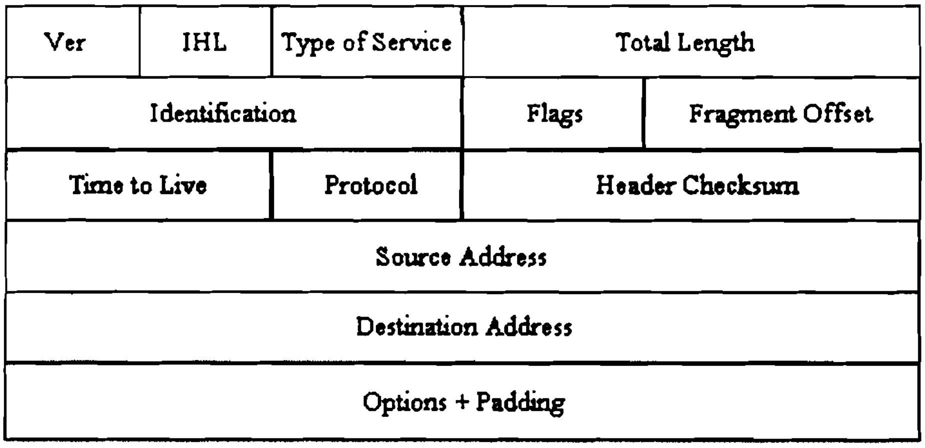


Figure 2.1 IPv4 header format

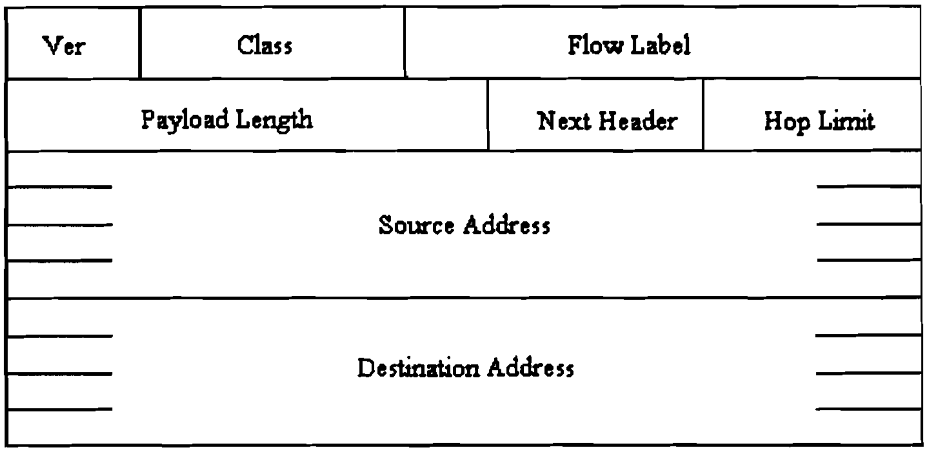


Figure 2.2 IPv6 header format

Equivalent fields

Version field

The Ver (version) field remains in IPv6. In IPv6 the version field's number is 6, and in IPv4 its number is 4.

Source and Destination field

The Source Address and Destination Address fields also remain in IPv6. These fields serve the same function in both IPv6 and IPv4 headers, but each field is 128 bits long in IPv6 instead of the 32 bit length from IPv4.

Modified fields

Total Length field

In IPv6 the Total Length field from IPv4 has been renamed Payload Length. These two fields are similar but not identical. The payload length is the length of the data the IPv6 packet carries after the IPv6 header, whereas IPv4's total length is the length of the IPv4 header plus the length of the data. Because the IPv6 header length is fixed at 40 octets, IPv6 can easily derive total length by adding 40 to the Payload Length. This modification was required as otherwise there would have been some, unlikely but possible, IPv4 payloads that IPv6 would have been unable to carry namely those with a payload length between 65496 and 65515 octets, which with IPv4 and a standard 20 octet header produce a total length of between 65516 and 65535 which can be carried in the 16 bit length field. With IPv6 the total length would have been between 65536 and 65555 for the same payload, too large for a 16 bit field. Rather than increasing the field size, simply omitting the fixed header size from the value carried in the packets allows all possible IPv4 payloads to be carried in IPv6 datagrams.

Time to Live field

IPv4 uses the Time to Live (TTL) field to specify how long a packet can remain alive when it travels in the network. This specification prevents the packet from becoming locked in an infinite loop. IPv4 expresses the TTL field in seconds, and decrements the field value every second. Because clocks are not assumed synchronized across the Internet, the specification also demands that every node decrease the TTL by at least one, this makes the field also a limit on the number of hops through which a packet may pass. When the time is up, that is a router decreases the TTL to zero, the network discards the packet. The IPv6 designers noted that in the modern Internet, packets almost never remain at any one node for a period greater than a second, so the TTL is almost never decremented by more than one. In spite of this, to be correct, an IPv4 implementation must maintain the ability to measure the time that each packet spends within it, and decrement the TTL again each time a second passes. To simplify implementations IPv6 changed IPv4's Time to Live field into a Hop Limit field, with all mention of time deleted. In practice this is not a change at all, though it seems like it is at first glance.

Protocol field

IPv6 renamed and enhanced IPv4's Protocol field to the Next Header field. The Protocol field in IPv4 designates a packet's transport-layer protocol (e.g., TCP or UDP) which specifies what type of transport data follows the IPv4 header. It was renamed as the Next Header field in IPv6 to allow extension headers between the IPv6 header and the transport data. Each extension header other than the transport header (such as the TCP header) includes its own Next Header field that specifies the following header's type. IPv6 initially defines six extension headers: Hop-by-Hop Options, Destination Options, Routing, Fragment, Authentication and Encapsulating Security Payload. Figure 2.3 shows an example of an IPv6 packet with extension headers.

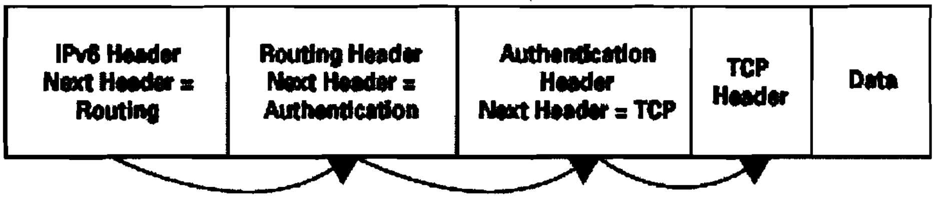


Figure 2.3 Example of an IPv6 packet's extension headers

TOS field

IPv6 includes in its header two new fields intended to support Quality of Service (QoS). These fields are the Traffic Class and Flow Label fields. The Class field replaces IPv4's Type of Service field, which these days is mostly interpreted as a Traffic Class field, when it is interpreted at all. It allows the originating host or the forwarding router identify the class or priority of the packet. The Flow Label field is new, and enables the source host group a sequence of packets that require common special handling by intermediate routers when the packets travel from source to destination. In IPv4 routers needed to understand and examine transport header contents to achieve equivalent functionality.

Missing fields

Options field

Because IPv6's extension headers replace IPv4's Options field IPv6 removed the Options field. This also allowed the Header Length (IHL) field to be removed, as in IPv6 the header is of fixed length. IPv6 also removed the three fields related to data fragmentation in IPv4: Identification, Flags, and Fragment Offset. In IPv6, the new Fragment extension header covers data fragmentation. As relatively few packets are fragmented, this avoids every other packet carrying meaningless unused header fields.

Header Checksum field

IPv6 eliminates the Header Checksum field which enables header error checking in IPv4. IPv6 noted that most link layers provide much more robust error detection methods, so in practice only errors in router processing would ever be detected by the header checksum. On the other hand, as the header contents are defined to change at every router, as at the very least the TTL in IPv4, or Hop Limit in IPv6 are altered, the IPv4 header checksum needed to also be adjusted at every intermediate router. This slows router processing. After an analysis of the various failure modes should an undetected header error occur, and finding nothing of concern, IPv6 decided to eliminate this field. Note that link layer error detection is still used, as is transport layer error detection for end to end reliability, the IPv4 header checksum only ever checked the IPv4 header itself.

The header design in IPv6 can improve packet transmission, particularly router performance. For example in IPv4, when the source host includes optional information that requires action from only the destination host, every intermediate router between the source and the destination must nevertheless examine the contents of the Options field. In IPv6, the source host can use the Destination Options extension header to carry optional information and only the destination host will check the information in that header. Therefore, routers forward packets faster because they don't waste time checking information that's irrelevant to them. Another improvement in packet transmission increases router efficiency. In IPv6, routers do not handle data fragmentation and reassembly. Fragmentation is carried out only at the source host, and reassembly, as in IPv4, is carried out only at the destination host. This change in IPv6 leads to better router performance, and simpler router implementations, with less opportunity for bugs in rarely executed code.

2.2 IPv4/IPv6 Transition Mechanism

Transition mechanisms are the topic discussed by the IETF IPv6 Transition Working Group (ngtrans) [7]. The main issue is the transition from IPv4 to IPv6 and how both versions can coexist before complete migration. It was understood that the transition would take quite a long time but not forever. The are two basic aspects of transition, transition of the network, including end nodes and routers, and everything required to enable IPv6, and transition of the applications that use the network to enable use of IPv6 where before only IPv4 was possible. For the network, transition mechanisms can be divided into three categories as follows: Dual-Stack [8], Tunneling [9] and Translation. Each category has a different role to play in the overall transition environment. The transition process is complex as it has to deal with issues related to IPv4-IPv6 interoperability including routing, translation of names to suitable IPv4 or IPv6 addresses using the Domain Name System, DNS, error handling, etc. The major IPv4/ IPv6 transition mechanism are discussed in the following sections.

2.2.1Dual-Stacks

Dual-Stacks [8] are the basic transition mechanism. Dual-Stacks literally maintain both IPv4 and IPv6 protocol stacks in every network device and operate both protocols in parallel across every network link. The methodology of the transition this way is simple. In the period between when IPv6 was defined, and when IPv4 ceases to be viable - that is, when the last available IPv4 address has been consumed and not a single address remains available for assignment to new nodes - the network would be gradually modified to support both IPv4 and IPv6. Applications can choose their protocol stack. IPv4 applications use the IPv4 stack and IPv6 applications use the IPv6 stack. Applications able to use both IPv4 and IPv6 can use IPv6 whenever communication with a peer that also supports IPv6 over networks links that support IPv6, and fall back to IPv4 whenever IPv6 is not yet available. The default behavior a dual-stack host should observe is to attempt to resolve an IPv6 address first and if not available, or if communications using it fail, resolve an IPv4 address and use that if available. Also dual-stack capabilities in the network devices allow handling of both IPv4 and IPv6 packet types based on the IP header version field or link layer packet type identification. When running dual IPv4/IPv6 stacks, a host can access to both IPv4 and IPv6 resources. Routers run both protocols for forwarding the both IPv4/IPv6 packets to end nodes or destination hosts. Figure 2.4 illustrates dual Internet protocol stacks.

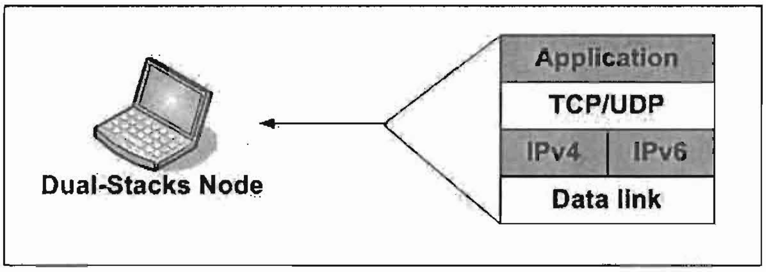


Figure 2.4 Dual Internet Protocol Stacks

The Dual-Stacks mechanism is the fundamental transition solution. The plan is simple. Over time, while IPv4 remained viable, expected to be many years when the transition plan was created, and with hindsight and careful management, even longer, IPv6 would be gradually deployed as systems were gradually upgraded to newer releases that supported IPv6. By the time there are any nodes that cannot obtain IPv4 addresses, and hence cannot use IPv4, IPv6 support should be available everywhere, and so be available to be used to communicate with IPv6 only nodes. However this alone is not sufficient for transition purposes, more is required for a complete solution.

A dual-stack host has both IPv4 and IPv6 addresses on each of its interfaces. The IPv6 addresses are assigned either statically or dynamically using the autoconfiguration protocol [19]. Each interface has an IPv6 link local address and any proper multicast addresses and also an unicast address. Interfaces also have IPv4 addresses, usually assigned by the Dynamic Host Configuration Protocol (DHCP) [6].

2.2.2Tunneling Mechanisms

An immediate problem faced by early adopters of IPv6 was how could they communicate with each other, using IPv6. Early IPv6 sites could not expect to be neighbors, or so share common links. Without an IPv6 path connecting them, use of IPv6 would be impossible. Tunneling mechanisms [9] are a technique used for communication between two hosts or networks which have the same protocol but are separated by a network that does not support that protocol - such as when no native IPv6 infrastructure exists between two points but there is IPv4 connectivity, tunneling of IPv6 in IPv4 can be used. This same technique had earlier been used when IPv4 multicast was first introduced, and nodes supporting multicast packets needed to forward those packets through infrastructure that did not understand multicast. It had also been used to allow Appletalk [42] packets to be carried over the IP Internet, and for that matter, to allow IP packets to be carried over native Appletalk networks. The need for tunnels is very common in the early stages of the transition process. For IPv4 to IPv6 transition, the tunneling techniques are communication between one IPv6 node and another IPv6 node by tunneling through the IPv4 infrastructure. The IPv4 infrastructure does not need to support to IPv6. IPv6 packets are encapsulated in IPv4 packets and delivered across the IPv4 infrastructure. The tunnel endpoints must support both IPv4 and IPv6 protocol stacks. A tunnel can be configured between border-routers or between a border-router and a host, or even between two hosts. When two IPv6 nodes that connected by using tunneling techniques have communication between them, the IPv6 packets are delivered to the tunnel endpoint which supports both IPv4 and IPv6 protocol stacks. The tunnel endpoint will encapsulate each IPv6 packet in an IPv4 packet and deliver it across the IPv4 infrastructure to the other endpoint. This endpoint will decapsulate and send the IPv6 packet to its destination. The general process of tunneling mechanisms can be shown in Figure 2.5.

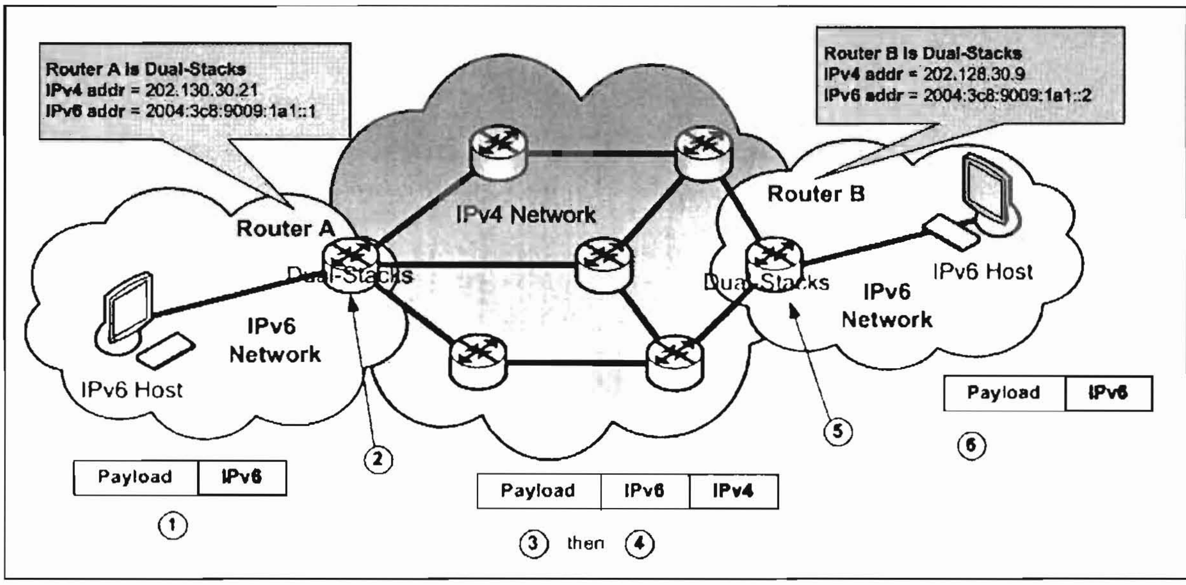


Figure 2.5 The general process of Tunneling Mechanism

2.2.2.1 General process of tunneling mechanisms

1) A packet with an IPv6 destination address arrives at Router A.
2) Router A identifies the destination in its routing table and finds that it can route this IPv6 address by sending it to Router B. It notes that its IPv6 link to Router B is via a tunnel through the IPv4 network. It finds that Router B's IPv4 address is, in this example, 202.128.30.9.
3) The IPv6 packet is encapsulated in an IPv4 packet, with Router B's IPv4 address as destination, and Router A's IPv4 address as source, and sent into the IPv4 network by Router A.
4) The IPv4 network routes the packet using the destination address as with any other normal IPv4 packet and the packet finally reaches Router B.
5) Router B processes the packet, which was addressed to it, and realizes that it is

carrying an IPv6 packet inside. It removes the IPv4 header and treats the enclosed IPv6 packet as if it had just arrived over its IPv6 tunnel link from router A. This packet is forwarded like any other arriving IPv6 packet, using the destination address from the IPv6 header to look up the next hop from its routing table. It finds, we assume, and in this example, that it can reach the IPv6 address destination in the IPv6 network to which it is connected.

6) Router B sends the IPv6 packet across its local IPv6 network to the destination node.

Tunneling mechanisms can be divided to three types as follows:

1) Configured Tunneling
2) Semi-Configured Tunneling
3) Automatic Tunneling

2.2.2.2 Configured Tunneling

A statically configured tunnel or manual tunnel is used for regular communication between two border routers, or between a host and a border router, or for connection to remote IPv6 networks such as the 6BONE. The border routers and hosts used as tunnel endpoints that are dual-stack devices and have to have tunnels manually configured. Installing a tunnel this way is no different than installing and configuring any other long distance network link, though it is usually cheaper, though less efficient. These tunnels are used between two points and require configuration of both the source (local) and destination (remote endpoint) addresses of the tunnel using addresses appropriate for the intermediate network, usually IPv4 for IPv6 tunnels.

Address Creation

In a manually configured tunnel, the IPv4 hosts must configure the tunnel by using a Pseudo (or software) Interface such as gif0 on NetBSD and FreeBSD. The interface requires both IPv4 and IPv6 addresses of both the local and remote endpoints.

2.2.2.3 Semi- Configured Tunneling

Tunnel Broker

Tunnels allow distant IPv6 networks to communicate. They operate well for linking co-operating networks. However, a new site that has just implemented IPv6 might not be able to readily locate an IPv6 peer site to which it can connect, and finding one willing to forward traffic to other IPv6 nodes can be even harder.

A Tunnel Broker [14] provides a solution to this problem. Tunnel Brokers are well known attachment points that allow tunnel connections from essentially anyone. Because of the large number of tunnels expected at the broker, manual configuration there would be burdensome, and slow, so instead a simple protocol is used by the client to cause a tunnel to be automatically established at the broker, or more correctly, at a server selected by the broker. A tunnel broker can be seen as a virtual IPv6 Service Provider (ISP) that provides IPv6 connectivity to users who are already connected to the IPv4 network. The tunnel broker is based on four important elements as follows.

Tunnel Broker (TB)

When a user desires IPv6 connectivity through the tunnel broker, the user must connect to the TB to register and activate the tunnel. The TB manages tunnel creation, modification and deletion on behalf of the user.

For scalability reasons the tunnel broker can share the load of network side tunnel end-points among several tunnel servers. It sends configuration orders to the relevant tunnel server whenever a tunnel has to be created, modified or deleted. The TB also assigns IPv6 address space to the user, as does any other IPv6 ISP, and may also register the user IPv6 address and name in the DNS.

A TB must have an IPv4 address. It may be dual-stack, that is, also be IPv6 addressable but this is not mandatory. The communication between the broker and the tunnel server (TS) can take place using either IPv4 addresses or IPv6 addresses.

Tunnel Server (TS)

The Tunnel Server or TS is a dual-stack router which is connected to the global Internet. Its operation is that upon receipt of a configuration order from the TB it creates, modifies or deletes the server side of each tunnel. After the tunnel is created it is used to forward the packets, as with a manually configured tunnel, except that it is perhaps more likely that a packet arriving over one tunnel will then be immediately re-encapsulated and forwarded via another tunnel.

The TS provides information that is necessary to configure the tunnel for the connected users. This information is an operating system (OS) specific script that establishes the IP tunnel to IPv6 network of the provider.

Tunnel Client (TC)

The Tunnel Client, or TC, is a node that requires to use IPv6 connectivity and is residing on a dual-stack host. A tunnel client can be either a host or a router.

Domain Name System (DNS)

The DNS is a global distributed database that keeps the details of the hosts, routers and the other network devices, and allows various associated information, such as their addresses, to be obtained. The DNS holds the user IPv6 addresses and names that the TB registered.

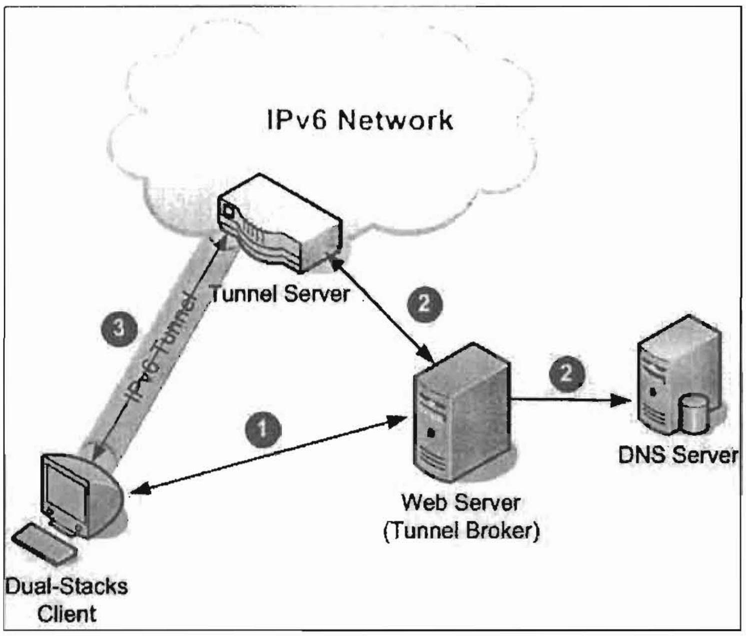


Figure 2.6 Tunnel Broker model

Figure 2.6 illustrates the establishment of a tunnel by using a tunnel broker. The concept of tunnel broker can be shown in the following steps.

1) The user downloads the tunnel configuration script (information which is an operating system (OS) specific script that will establish the IP tunnel to IPv6

network of provider) from the tunnel broker web server.

2) Then the script runs, it connects to the broker, and requests a tunnel. The tunnel broker advises the tunnel server about the new tunnel. It also usually registers the user IPv6 address and name in the DNS.

3) The host establishes a dynamic tunnel to the tunnel server.

When operating a tunnel broker, it is desirable to periodically check the status of the tunnel client. If there are tunnels to destinations that are not reachable, the tunnel broker will free the associated resources.

2.2.2.4 Automatic Tunneling

Automatic 6to4 Tunneling

Tunnel brokers provide a simple solution to the connectivity problem for early IPv6 adopters. However the simplicity comes at a price, tunnels configured this way are unlikely to provide optimal network routing between end sites. For example, two nearby sites that communicate using IPv4 with a delay of a small number of milliseconds might both choose to connect to the IPv6 network using the same tunnel broker. Unfortunately, as there are very few of these brokers, the tunnel broker chosen is possibly on the other side of the world, leading to IPv6 communications between the two sites suffering delays of many hundreds of milliseconds. This gives the appearance of IPv6 traffic being much slower that the alternate IPv4 traffic, and also usually more likely to suffer packet loss.

6to4 tunneling [20][21] is an automatic tunneling technique that creates dynamic stateless tunnels over the IPv4 infrastructure to connect 6-to-4 domains to the IPv6 Internet for IPv6-only communication. 6to4 tunneling is a technique where the tunnel endpoint is determined by the globally unique IPv4 address embedded in the IPv6 6to4 address. A 6to4 IPv6 address is a combination of the well known unique routing prefix 2002::/16 with a globally unique 32-bit IPv4 address. The 6to4 address format is illustrated in Figure 2.7.

0x2002	IPv4 address	Subnet	Interface ID
16 bits	32 bits	16 bits	64 bits

Figure 2.7 The 6to4 address format.

Figure 2.7 illustrates the 6to4 address format. The prefix for the 6to4 address is 2002::/16. The following 32 bits contain the IPv4 address of 6to4 router. Together, this makes a 48 bit prefix, the same size IPv6 prefix as is typically assigned to an end site by a native provider of IPv6 connectivity (an ISP). As with any IPv6 address block, the remaining 80 bits are used to provide the subnet numbers, and then interface IDs of particular IPv6 end hosts.

6to4 IPv6 addresses are a different format from IPv4-compatible IPv6 addresses. IPv4-compatible IPv6 addresses are not used in 6to4 tunneling. 6to4 tunnels can be configured between border routers or between a border router and a host.

When IPv6 traffic is sent to a node with a 6-to-4 destination address, the packets will be encapsulated as with any other tunnel, but will then automatically find their destination because the IPv4 destination address used is taken from the IPv6 6-to-4 address. This allows the IPv4 packet to take the shortest available path directly to the intended recipient. Figure 2.8 illustrates the concept of the 6to4 tunneling mechanism.

The 6to4 tunneling consists three parts as following.

6to4 Router

The 6to4 router is an IPv6 router supporting a 6to4 pseudo-interface. It is normally the border router between an IPv6 site and the IPv4 Internet.

6to4 Relay Router

The 6to4 relay router is a 6to4 router configured to support transit routing between 6to4 addresses and native IPv6 addresses. It need for communication of the 6to4 hosts with IPv6 native hosts, that is with hosts that have IPv6 addresses that are not 6-to-4 addresses.

6to4 Site

A 6to4 site is a site running IPv6 internally using 6to4 addresses, therefore containing at least one 6to4 host and at least one 6to4 router.

6to4 host

A 6to4 host is an IPv6 host which has least one 6to4 address.

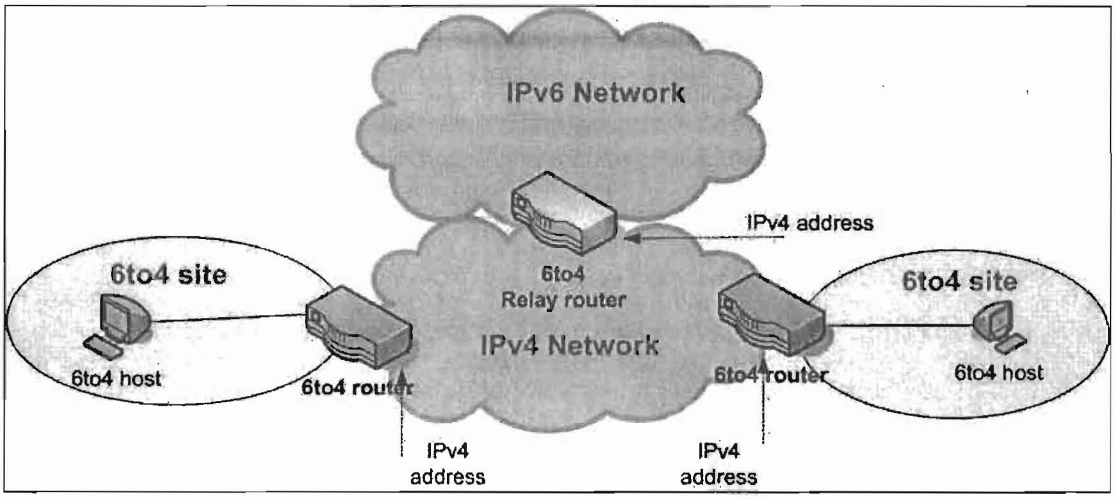


Figure 2.8 The 6to4 tunneling mechanism.

We can explain the method of the 6to4 tunneling mechanism in Figure 2.8, the communication can occur two scenarios as follows.

Scenario I: A 6to4 host communicates with a 6to4 host in another site.

When one 6to4 host sends an IPv6 packet to another 6to4 host in another site, the edge 6to4 router must handle and process this packet. The 2002:: prefix of the destination address of a packet arriving at a 6to4 router indicates that it is destined to another 6to4 site. It will be encapsulated as an IPv6 packet in an IPv4 packet and automatically tunneled to the destination 6to4 router across the IPv4 infrastructure. The destination IPv4 address of this encapsulated packet is derived from the embedded IPv4 address within the IPv6 destination 6to4 address prefix.

After the destination 6to4 router receives the encapsulated packet, it will decapsulate the packet and forward the enclosed IPv6 packet to destination 6to4 host.

Scenario II: The 6to4 host communicates with an IPv6-only host in the IPv6 network.

Here a 6to4 host connects to a native IPv6 host, or an IPv6-only host that is not within a 6to4 site. When the 6to4 border router receives the packet from 6to4 host, this packet will also be encapsulated and forwarded to a special purpose 6to4 "Relay" or 6to4 relay router.

After that the 6to4 relay router will handle and decapsulate this packet. Then the relay forwards the IPv6 packet to the destination host that resides in the native IPv6 network.

6to4 relay routers are identified like any other 6to4 router, using a 2002::/16 prefix, except they are all assigned the same well known IPv4 anycast address. This use of IPv4 anycast allows packets to the native IPv6 network to be routed to the nearest relay router to the sending node.

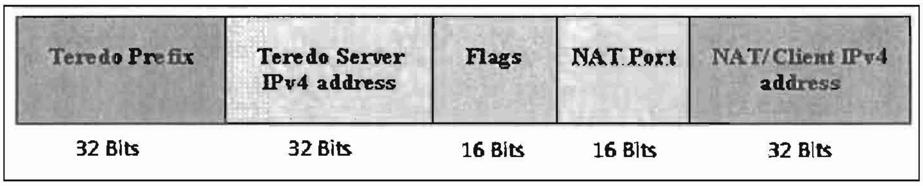
6to4 cannot be used with private IPv4 addresses. Therefore in an environment that relies on Network Address Translation, or NAT, 6to4 will work only if the NAT and 6to4 router functions are in the same system, so 6to4 can use the global address that NAT is also using.

Teredo

Teredo [22] is an address assignment and automatic tunneling technique that provides unicast IPv6 connectivity across the IPv4 infrastructure like the 6to4 technique. But Teredo automatic tunneling is capable of communicating through Network Address Translator (NAT) by encapsulation in IPv4 UDP packets. Teredo supports to "cone NAT", "restricted cone NAT" and "port restricted NAT" but doesn't support "symmetric NAT".

Teredo clients and the Teredo relay may send Teredo bubbles to force transmission along the path to create a mapping in a NAT. A Teredo bubble is a minimal IPv6 packet which is made of an IPv6 header and a null payload.

The Teredo address format is shown in Figure 2.9.



Teredo Prefix	Teredo Server IPv4 address	Flags	NAT Port	NAT/Client IPv4 address
32 Bits	32 Bits	16 Bits	16 Bits	32 Bits

Figure 2.9 Teredo address format

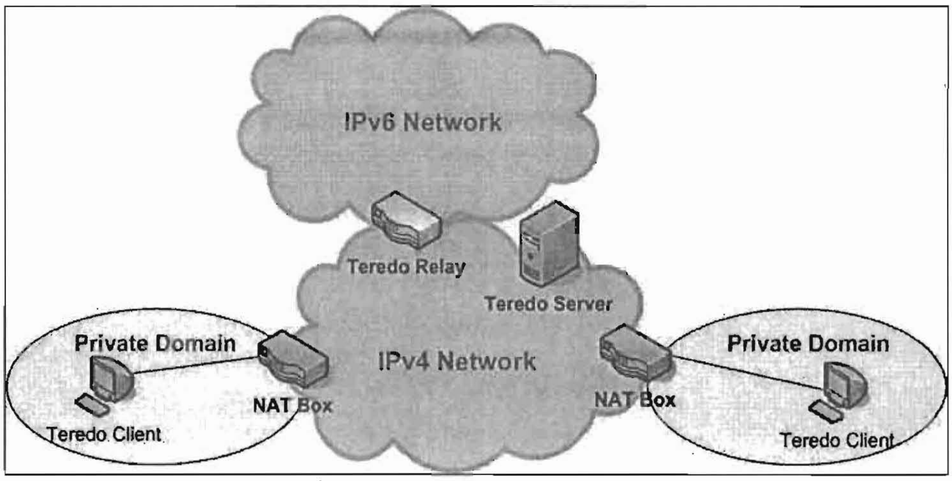


Figure 2.10 Teredo Architecture

The Teredo architecture can be illustrated as in Figure 2.10. It is based upon four components as follows.

Teredo Client

A Teredo client is a host that has some access to the IPv4 Internet and wants to gain access to the IPv6 Internet. It needs to support both IPv4 and IPv6. It receives an IPv6 prefix from a Teredo server and acts as tunnel endpoint.

Teredo Server

A Teredo Server is a host that has access to the IPv4 Internet through a globally routable address. It is used as a helper to provide IPv6 connectivity to Teredo clients by assigning IPv6 addresses to its Teredo clients. It listens on UDP port 3544 for incoming requests.

Teredo Relay

A Teredo relay is an IPv4/IPv6 router that will relay IPv4 UDP encapsulated IPv6 traffic between a Teredo client and native IPv6 hosts. That is, it will create tunnel end points for IPv6 packets tunneled over IPv4 UDP. Also a Teredo relay can provide interworking with 6to4.

"Plain" IPv6 nodes

"Plain" IPv6 nodes are the IPv6 native hosts in the IPv6 Internet or may be 6to4 hosts.

Intra-Site Automatic Tunnel Access Protocol (ISATAP)

ISATAP [15] is a transition mechanism which can be used within a site to connect isolated IPv4/IPv6 dual-stack hosts to the IPv6 internet. Once an ISATAP server/router has been set up the clients must be configured to connect to it. ISATAP is an automatic tunneling type.

2.2.3Translation Mechanism

The original transition plan assumed that IPv4 would remain viable until access to IPv6 was endemic. That is, it was assumed that IPv4 would still provide the basis for the network for long enough that IPv6 would become available everywhere before IPv4 addresses were exhausted. This was not an unreasonable assumption. Even though it was expected to take several years for vendors to have IPv6 available in their product lineup, and then many years after that for natural equipment upgrade and replacement cycles to gradually cause IPv6 to be available on all systems, planning for IPv6 had started early enough that it was expected that IPv4 would last long enough. Careful management of the remaining IPv4 address resources, including resumption and reallocation in smaller pieces of large chunks that had been inefficiently allocated, along with techniques such as Network Address Translation (NAT), has allowed IPv4 to continue even longer than initially planned, to the point where today it is rare to find any operating IPv4 device that is not IPv6 ready.

Unfortunately, the transition plan had one major flaw. It assumed that as IPv6 became available, it would actually be used. That is, that IPv6 would be enabled as soon as the local network supported it, or soon thereafter. Hindsight has shown us that this was a poor assumption. Even where all systems connected to a network support IPv6, or would if enabled, many network operators have deliberately chosen not to enable IPv6. The reasoning is that enabling IPv6 has costs, at the very least staff need to be trained to enable it, and then monitor its operation. On the other hand, there is no need to enable IPv6, as all destination systems needed, can still be reached using IPv4.

This reasoning is hard to contradict. Sites enabling IPv6 typically do so either out of a desire to be one of the early adopters, for whatever advantages that might bring, or from a sense of evangelism. Sites concerned by their bottom line financial concerns have no current rationale to enable IPv6, and will not have as long as IPv4 remains viable.

The inevitable conclusion from this is that when IPv4 finally allocates the very last IPv4 address, and the next site to join the network must use IPv6, or nothing, there will still be, probably many, sites that are only reachable using IPv4. This isn't because of any problem enabling IPv6, but because they have chosen not to do so. Until those sites experience difficulties communicating in a way that reduces their revenues, enabling IPv6 is not to be expected. Because of this, the dual stack transition plan, at least as originally envisioned, cannot succeed.

The obvious outcome would be for the Internet to split into two, overlapping, networks - an IPv4 network, and an IPv6 network, with many systems connected to both, but with communication only possible between systems sharing a common IP protocol version. This is not a desirable result. The alternative is to create a mechanism that allows a host that has only IPv4 to communicate with a host that has only IPv6. That is, IP protocol translation. Fortunately, the great similarity between IPv4 and IPv6 makes this a plausible solution.

Translation mechanisms have many algorithms which can be used to convert between the IPv4 and IPv6 protocols. Translation can occur at several layers in the protocol stack, including network, transport, and application layers. The basic role of translation in IPv4/IPv6 transition is the conversion of IP and ICMP packets. Example of a translation mechanisms, for different purposes, are The Stateless IP/ICMP Translation algorithm (SIIT), Network Address Translation - Protocol Translation (NAT-PT), Bump-In-the-Stack (BIS), Bump-In-the-API (BIA), SOCKS-Based IPv6/IPv4 Gateway, and SOCKS64.

2.2.3.1 The Stateless IP/ICMP Translation algorithm (SIIT)

The Stateless IP/ICMP Transition algorithm (SIIT) [10] is the algorithm upon which many translation schemes are based. SIIT provides the basic role of translation in IPv4/IPv6 transition is the conversion of IPv4 and IPv6 packet headers, as well as ICMPv4 and ICMPv6 messages. SIIT will ignore IPv4 options and IPv6 extension headers other than the fragment header. SIIT is used as the basis for BIS and NAT-PT which are discussed below.

2.2.3.2 Network Address Translation Protocol Translation (NAT-PT)

Network Address Translation Protocol Translation (NAT-PT) [11] is a stateful IPv4/IPv6 translator that is based upon the SIIT algorithm. NAT-PT translator translates the address, port and protocol of packets moving between IPv4 and IPv6 hosts. NAT-PT allows a large number of applications in IPv4 and IPv6 networks to inter-operate without requiring any changes to these applications. If an application works with IPv4 through NAT, which of necessity, most do today, and if it requires no extra assistance for NAT, there is a good chance that it will be supported by NAT-PT. The NAT-PT device is installed at the boundary between the IPv4 and IPv6 networks, and allocates a temporary IPv4 address to each IPv6 node (using a pool of IPv4 addresses), and acts as a communication proxy with IPv4 peers. The entire IPv4 network is mapped as a subnet of the IPv6 network's address space. NAPT-PT has been deployed into router functions to provide mapping address, port and protocol translation. NAT-PT can operate with static or dynamic translations.

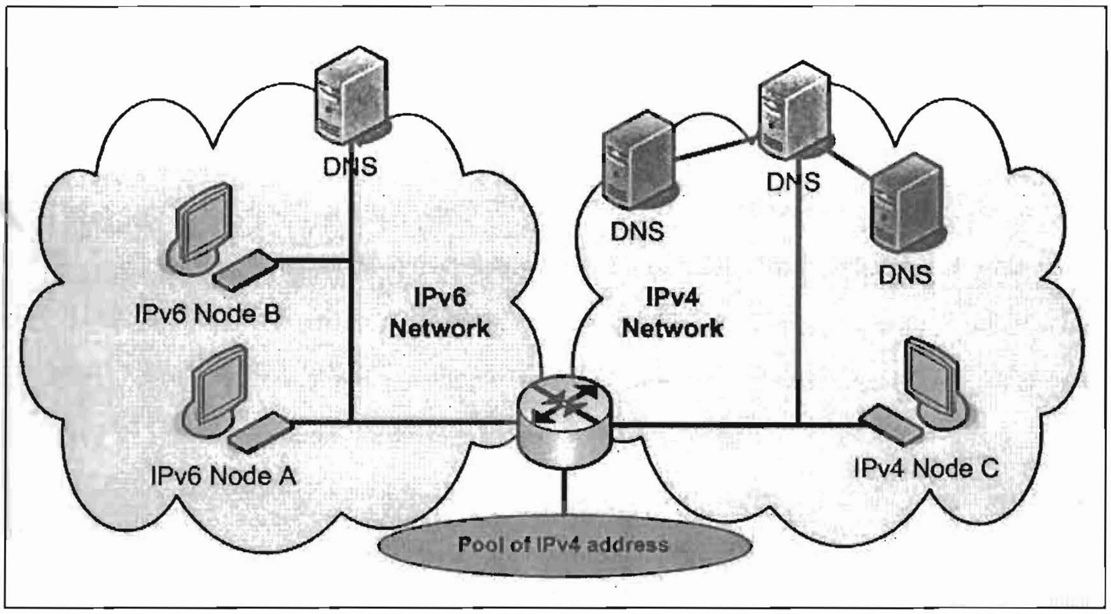


Figure 2.11 The network communication between IPv4 and IPv6 by using NAT-PT translator.

Basic NAT-PT operation is explained in Figure 2.12 and Figure 2.13. Figure 2.11 illustrates network communication between IPv4 and IPv6 using a NAT-PT translator. Assume that the IPv6 address of node A is FEDC:BA98::7654:3210, node B is FEDC:BA98::7654:3211 and node C is 132.146.243.30. Further assume that NAT-PT has been allocated the pool of addresses including the IPv4 subnet 120.130.26/24.

NAT-PT operation when the IPv6 node A wants to communicate with the IPv4 node C is shown in Figure 2.12.

1) IPv6 node A creates and sends a packet n which the source address is the IPv6 address of node A and the destination address is PREFIX:: IPv4 address of destination. PREFIX is the local subnet number, from the site containing node A's IPv6 address block, that has been assigned to map the IPv4 internet. Since IPv6 subnets allow 2^{64} addresses each, and the entire IPv4 internet is limited to 2^{32} addresses, this is a trivial mapping.
2) When the NAT-PT translator receives the packet, which it does by virtue of being on all network paths to PREFIX::/64, NAT-PT translates that packet to an IPv4 packet and sends it to its destination. The translated packet has a source address taken from the IPv4 address pool assigned for NAT-PT functions and has destination address set to the IPv4 address of node C, extracted from the original IPv6 destination address.
3) Node C receives the translated packet, which to it is simply an IPv4 packet, and eventually has a result packet to return to node A. The result packet has source address equal IPv4 address of node C and has destination address equal the received IPv4 address from the packet it received. That is, the source address of the received packet changes to be the destination address of the returned packet. This was the address allocated from the NAT-PT IPv4 address pool, which the NAT-PT translator has temporarily assigned to node A.
4) NAT-PT receives the result packet from node C. which it should, as the NAT-PT address pool, which includes the destination address of the packet, was assigned to it. NAT-PT processes the packet, that is, translates it to IPv6, and sends the result packet to node A. The translated packet has source IPv6 address which is PREFIX::IPv4-of-C and has destination address which is the IPv6 address of node A, which NAT-PT recovered from its mapping table, using the incoming IPv4 destination address as the lookup key.

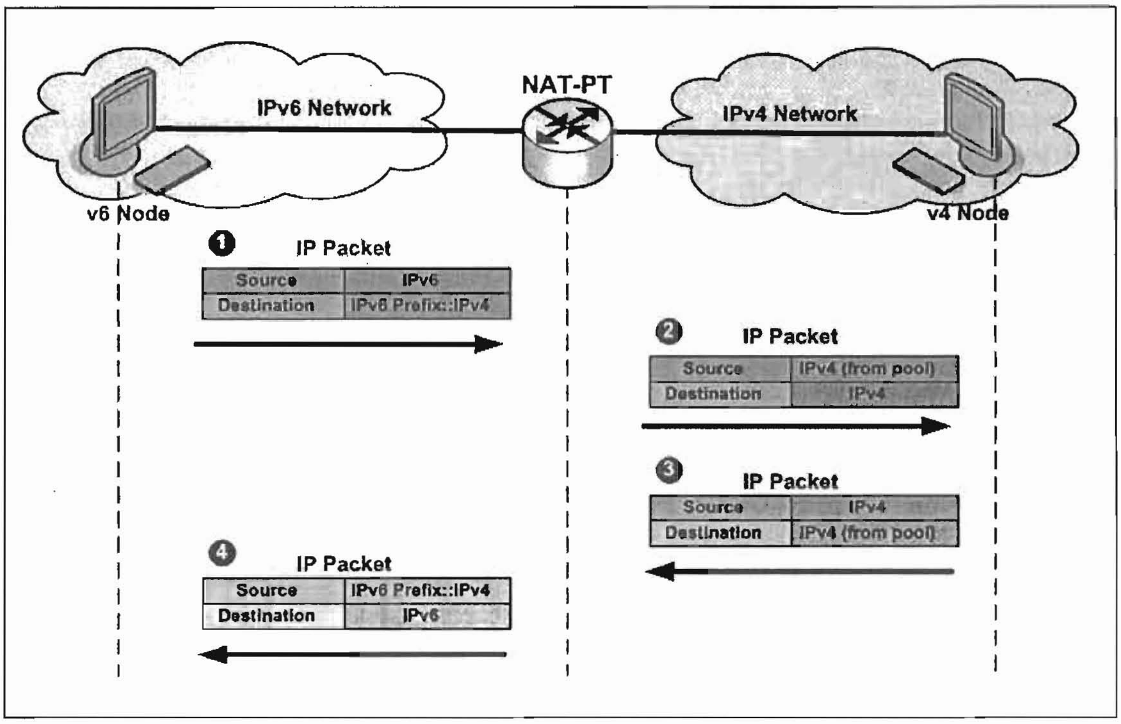


Figure 2.12 The NAT-PT operation when IPv6 node communicated to IPv4 node.

NAT-PT operation when the IPv4 node C wants to communicate with the IPv6 node A is shown in Figure 2.13.

1) The IPv4 node C creates and sends a packet where the source address is the IPv4 address of node C and the destination address is the assigned IPv4 address for node A from IPv4 pool maintained by the NAT-PT. The method by which node C obtains this address will be discussed later, see section 2.3.1.
2) When the NAT-PT translator receives that packet, NAT-PT translates the packet to an IPv6 packet and sends it to its destination, node A. The translated packet has source address equal PREFIX::IPv4 source address, and has destination address equal to the IPv6 address of node A. This process is identical to the processing of the reply packet in the previous scenario.
3) Node A receives the translated packet and returns a reply packet to node C. The reply packet has source address which is the IPv6 address of node A and has destination address equal the received IPv6 source address of the packet it received, which is PREFIX::IPv4 address of node C.
4) NAT-PT receives the reply packet from node A, translates it to IPv6, and sends

the result packet to node C. The translated packet has source address equal the assigned IPv4 address for node A from the NAT-PT IPv4 address pool, and has destination address equal IPv4 address of node C, extracted from the IPv6 destination address of the packet.

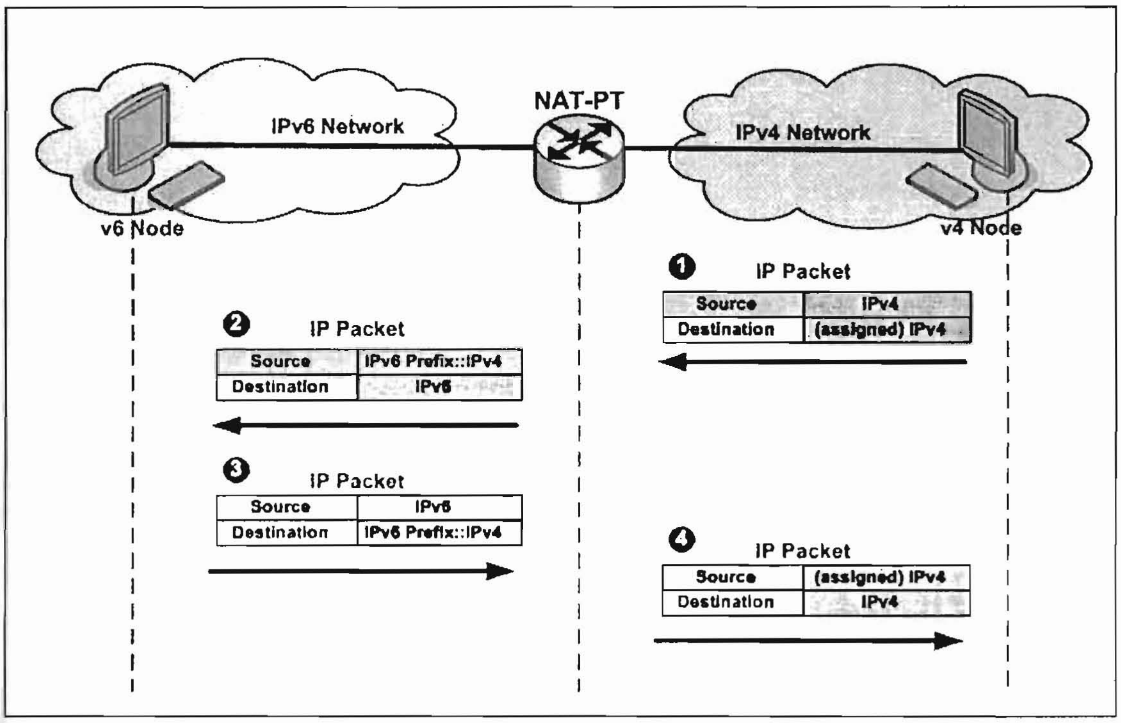


Figure 2.13 The NAT-PT operation when IPv4 node communicated to IPv6 node.

2.2.3.3 Dual-Stack Transition Mechanism (DSTM)

Dual-Stack Transition Mechanism (DSTM) [13] is a translation mechanism for dual stack hosts in an IPv6 domain that do not have an IPv4 routing infrastructure, but need to communicate with IPv4 systems or allow IPv4 applications to run on top of their IPv6 protocol stack. DSTM operation is based on the use of IPv4-over-IPv6 tunnels and the temporal allocation of a global IPv4 address to hosts requiring such communication.

2.2.3.4 Transport Relay Translator (TRT)

Another possibility to traverse between different IP versions is Transport Relay Translator (TRT) [24] which is used for translation between TCP/UDPv6 and TCP/UDPv4 sessions. Communication is initiated from the IPv6 side to a special destination address created by assign the prefix of destination IPv6 address by using the desired IPv4 address. Figure 2.14 illustrates the architecture of a TRT.

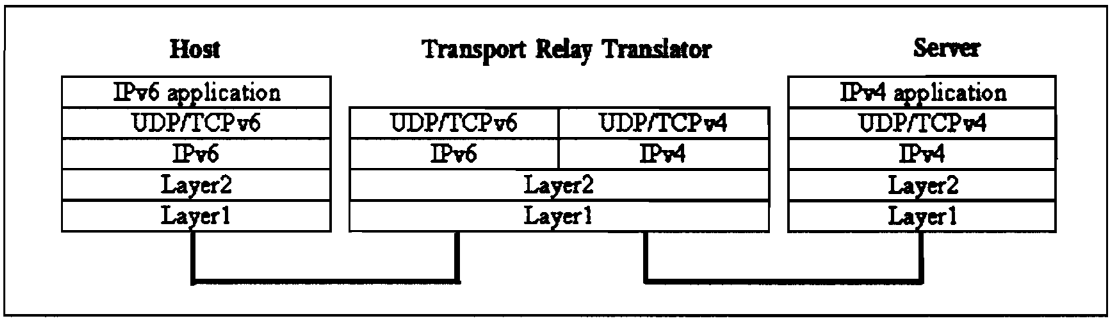


Figure 2.14 Transport Relay Translator

2.2.3.5 SOCKS-Based IPv6/IPv4 Gateway

The SOCKS-based IPv6/IPv4 gateway mechanism [12] is used for communication between IPv4-only and IPv6-only hosts. It consists of additional functionality in both the end system (client) and the dual-stack router (gateway) to permit a communications environment that relays two terminated IPv4 and IPv6 connections at the application layer.

2.2.4 Application Transition

Applications can obviously be rewritten to support IPv6 instead of, or in addition to, IPv4, as the communications network layer protocol. However, this can be a complex and costly task. The translation mechanisms described above can also be used within a host to enable applications written for one protocol to communicate with peers written for the other.

2.2.4.1 Bump-In-the-Stack (BIS)

Bump-in-the-Stack [16] is a translation mechanism used for communication between IPv4 applications on an IPv4-only host and IPv6 hosts. It is similar to taking the NAT-PT approach with SIIT and moving it to the OS protocol stack within each host. Unlike SIIT however, it assumes an IPv6 infrastructure. Whereas SIIT is a translation interface between the IPv6 and IPv4 networks, BIS is a translation interface between IPv4 applications and the underlying IPv6 network. The host stack design is based on that of dual stack host, with the addition of 3 modules, a translator, an extension name resolver, and an address mapper.

The translator translates outgoing IPv4 headers into IPv6 headers and incoming IPv6 headers into IPv4 headers (if applicable). It uses the header translation algorithm defined in SIIT. The extension name resolver acts as the DNS-ALG (see section 2.3.1). It uses a snooping module and an automatically allocated IPv4 address from a pool and works like a self-translator. It snoops IPv4 DNS queries and creates another query asking to resolve both "A" (IPv4) and "AAAA" (IPv6) DNS records. It then sends any returned "A" record back to requesting IPv4 application. If only "AAAA" records are returned, the resolver requests the address mapper to assign an IPv4 addresses and create an association between the IPv4 and IPv6 addresses. The BIS protocol stack is illustrated in Figure 2.15 .

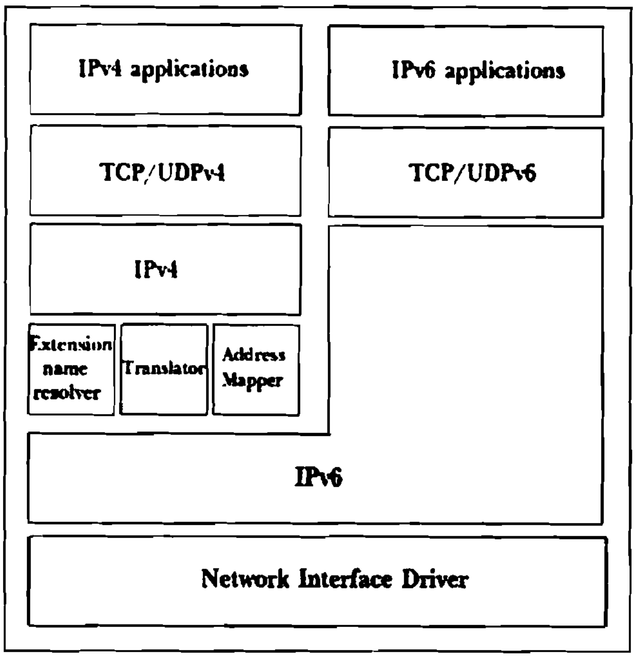


Figure 2.15 The BIS Protocol Stack

2.2.4.2 Bump in the API (BIA)

Bump in the APT (BIA) [17] is similar to BIS, and adds an API translator function between the socket layer and actual transport layer. BIA allows dual-stack hosts to use IPv4 applications to access IPv6 resources.

2.3 Application Level Gateway

An Application Level Gateway (ALG) [23] is an intermediary device located between two communicating hosts. It acts as an application proxy between IPv6-only clients and legacy servers that offer their services only via IPv4. Application Level Gateways are needed where the application protocol for an IPv6 application is not identical to the application protocol used with IPv4, and so packets for this application cannot simply have their header altered to the other protocol and expect to function correctly. There is application-specific information in the application layer at the ALG. One ALG example is for the File Transfer Protocol [23], needed as creating a file transfer data connection in IPv4 FTP involves sending IPv4 addresses as part of the application protocol. For IPv6 FTP those IPv4 addresses are obviously incorrect, and must be translated to the IPv6 FTP protocol equivalent, which happens not to usually include IPv6 addresses.

Another common ALG variant is the Domain Name System Application Level Gateway (DNS-ALG) [23] that is necessary for NAT-PT.

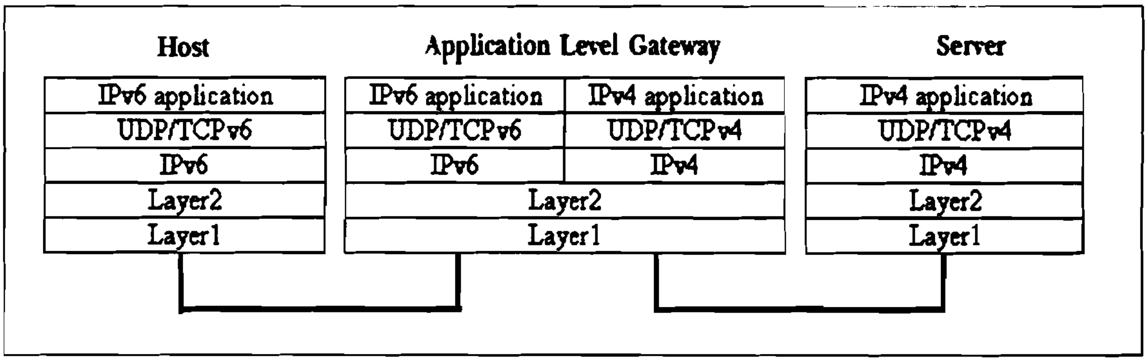


Figure 2.16 The architecture of an ALG.

Figure 2.16 illustrates the architecture of an ALG. The host in the example uses IPv6 and the server IPv4, although the opposite is just as possible. The host sends an IPv6 request towards the ALG. There, the ALG removes all IP and /TCP or UDP headers, translates the payload from IPv6 application format to the equivalent IPv4 application format and forwards the request using IPv4. The server sends the response back to the ALG, which translates the payload of the response and sends the response back to the client using IPv6.

2.3.1 Domain Name System-Application Level Gateway (DNS-ALG)

In Internet communication, applications generally begin with domain names rather than numerical addresses. The required address is obtained by looking it up in the Domain Name System (DNS), the distributed database containing name-address mappings, and more, for each internet domain name. Naturally, the IP address in a DNS answer cannot be used by a node which has only a different IP version. Just like NAT, NAT-PT cannot translate IP addresses inside the payload of a DNS packet. It needs a helper to adjust the destination address to be a suitable IP address. This helper is the Domain Name System-Application Layer Gateway (DNS-ALG) [23].

DNS-ALG can provide bidirectional address mapping between IPv6 and IPv4. It modifies the address in a reply DNS message to be the appropriate IP version. Then it informs NAT-PT about the mapping it has made between the original and the temporary addresses. The mapping is recorded into the list as a rule for address translation. The architecture of DNS-ALG is illustrated in Figure 2.17 and the procedures of interoperation between NAT-PT and DNS-ALG are shown in Figure 2.18.

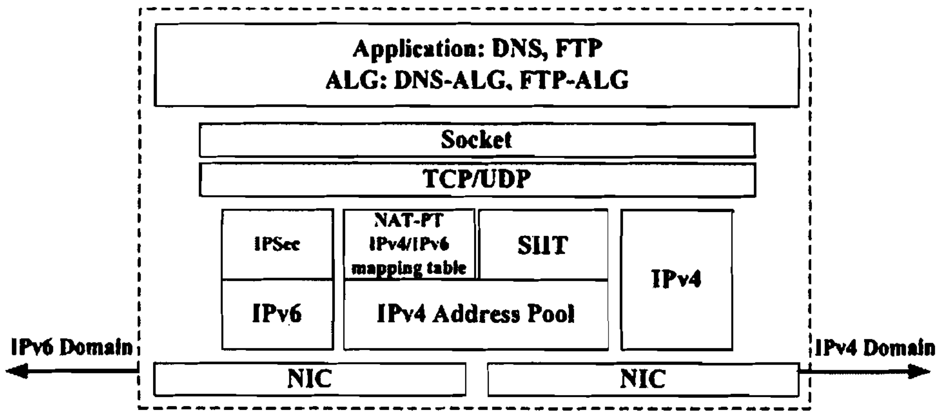


Figure 2.17 DNS-ALG communication

Figure 2.18 illustrates the signaling that occurs when an IPv6 node communicates with IPv4 node and, as usual, knows the v4 node by its domain name. That is, the v6 node may be sending e-mail to the v4 node, or doing a web lookup based upon a URL, or initiating communications for any of countless other reasons. The IPv6 node queries the DNS seeking an IPv6 address for its intended peer. An IPv6 address is needed as only IPv6 addresses are useful on the IPv6 network. The DNS-ALG intercepts the query, and does DNS queries for both the peer's IPv6 address (since all we know at this point is its domain name, we do not yet know if it is an IPv6 or an IPv4 node) and also seeks an IPv4 address for the peer.

If an IPv6 address is returned from the DNS, this is simply returned to the requesting node, and direct IPv6 communications proceed. If no IPv6 address is available, but an IPv4 address is returned, the DNS-ALG maps that to an available IPv6 address, which in this case simply means pretending the NAT-PT IPv6 subnet prefix to the IPv4 address, and return the result as the IPv6 address to the IPv6 node.

A similar process occurs when an IPv4 node initiates communications with a peer node. The v4 node queries for an IPv4 address, the only address form that is useful. The DNS-ALG detects the query, and performs both IPv4 and IPv6 queries. If only an IPv6 reply is available, indicating the peer is connected via IPv6 only, then the ALG obtains an available address from the NAT-PT IPv4 address pool, assigns that to map the IPv6 address obtained from the DNS, and returns that address as the IPv4 result of the original query. At the same time, it enters the mapping it has just made into the NAT-PT mapping tables, so packets later sent to the address it just selected will be correctly mapped, by NAT-PT, into the appropriate IPv6 address when the packets are being translated from v4 to v6.

2.4 Transition Summary

For network communications, Dual Stack was planned to be the basic transition mechanism to allow the Internet to gracefully switch from the old IPv4 network to its replacement, IPv6. Unfortunately, pragmatic, or commercial, considerations have arisen which have resulted in many segments of the IPv4 network making no immediate plans to enable IPv6. This defeats the dual stack transition plan. Furthermore, the only incentive that is likely to cause some of those network segments to enable IPv6, is the existence of significant numbers of IPv6 only nodes. In the period between when IPv6 only nodes exist for more than curiosity reasons, and when there are sufficient of them to cause the last IPv4 only segments of the network to finally turn the switch and enable IPv6, there will be nodes connected to the Internet which cannot communicate with each other. That is, the Internet will have fragmented.

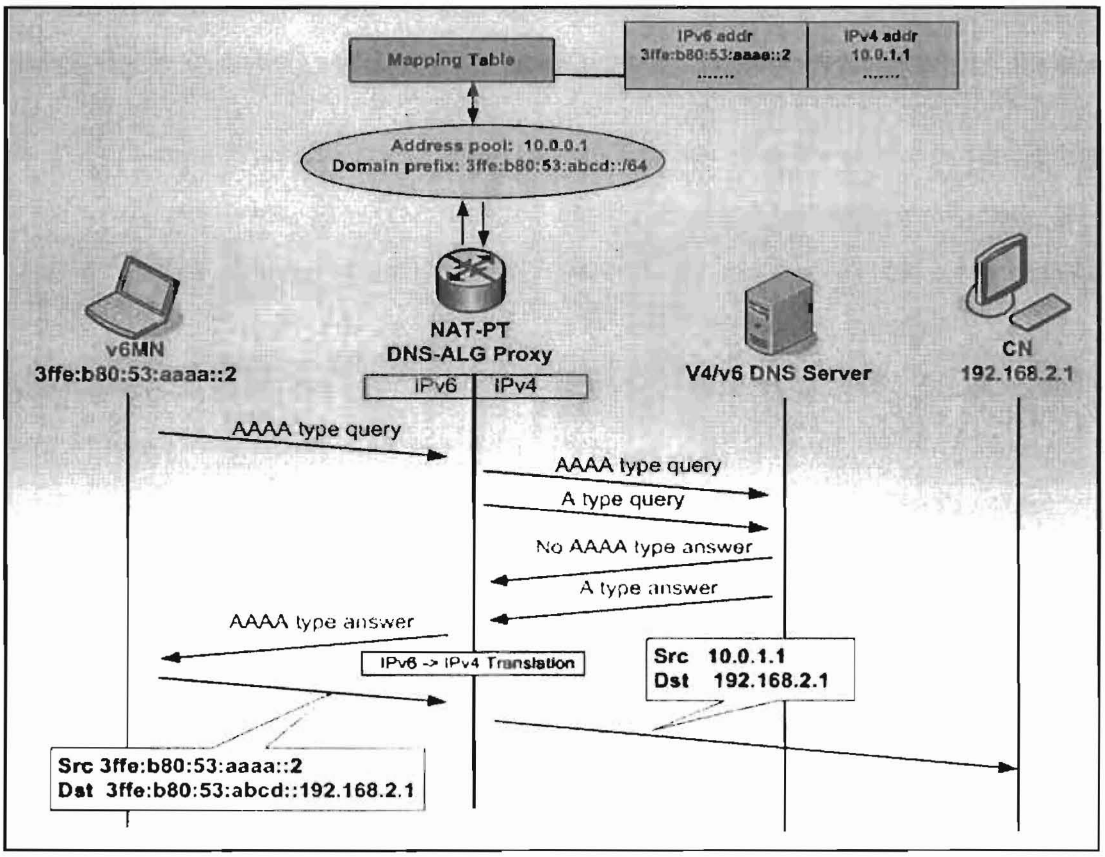


Figure 2.18 The procedure of interoperation between NAT-PT and DNS-ALG.

To avoid this result, protocol translation seems like the simplest solution for the period during which both protocols exist as sole use protocols in parts of the network. Protocol translation, sited at the boundary of the IPv6 only, or the IPv4 only, segments of the network, along with the appropriate ALGs, including certainly a DNS-ALG, should allow most common applications to function between IPv4 only and IPv6 only hosts.

2.5 A Missing Piece - Mobile IP

While protocol translation can handle most packets that move between the IPv4 and IPv6 only segments of the network, there is another kind of object that can also move between those segments that protocol translation cannot directly fix. That is, a mobile node. Both IPv4 and IPv6 support Mobile IP (MIP). Unfortunately, Mobile IPv4 (MIPv4) and Mobile IPv6 (MIPv6) are slightly different protocols, with different operating modes – MIPv6 makes use of the fact that all IPv6 nodes should understand it, whereas MIPv4 was a retrofit to

IPv4 and required to be able to interoperate with older IPv4 nodes that had never heard of MIPv4.

Mobile IP allows a node to move from one network segment to another, while retaining its existing communications connections. However, once we assume, or accept, that some network segments will be IPv4 only, and others IPv6 only, which is the conclusion of this chapter, then we must also accept that a mobile node might jump between those two types of network segments. Neither the mobile node, nor its user, is typically in the position to control which version of IP networking is installed in any particular network segment, and particularly not in a segment that is just temporarily being visited.

Unfortunately the differences between MIPv4 and MIPv6, along with some other factors, mean that no protocol translation device, not even aided by an ALG, can possibly allow a mobile node from one protocol universe to communicate when it is teleported into a network from a different universe.

Finding a solution to this problem is the objective of this thesis. Before we can discuss the mechanism designed, we must first discuss the basics of Mobile IP. Mobile IP, for both IPv4 and IPv6 is the subject of the next chapter. We will return to discuss the issue of moving between the two protocols in the subsequent chapter.

CHAPTER 3

MOBILE IP OVERVIEW

This chapter gives an introduction to the Mobile IP protocols [26], including explaining why Mobile IP is required in the current Internet and what problems Mobile IP was designed to solve. The first section introduces the rationale for, and problems that influenced the demand for and design of Mobile IP. The following section provides a taxonomy of some Mobile IP terminology for reference. Sections 3.4 and 3.5 describe the basic concepts of Mobile IP for both Mobile IP for IPv4 (MobileIPv4) [26] and Mobile IP for IPv6 (MobileIPv6) [27]. A comparison of MIPv4 and MIPv6 can be found in section 3.6, the chapter then concludes with a summary in section 3.7.

3.1 Introduction

As computing devices have become smaller, the desire to move them around to accompany their owners has grown. In the IP protocols, devices are identified by their IP address. The routing system is tasked with providing a path to any given address. However, routing cannot cope with managing the locations of every individual end system, rather it operates on units of clusters of systems, that is, upon networks. Even at that level the routing system is stressed, so the tend is to cluster networks, and manage routing at the global level only to the cluster, rather than to individual networks. Handing individual IP addresses would be beyond the capacity of the routing system, if there were more than a very small number. Large numbers of wandering portable devices can be expected, hence the routing system cannot be expected to handle paths to them.

The effect of this is that when an individual node changes its point of attachment to the network, it must acquire a new IP address that is related to its new attachment point. Packets to its old address will be sent to its old network connection point by the routing system. For devices that are inactive while moving, this is not a serious problem, once reattached, new connections are made using the appropriate local address. On the other hand, if a device moves while actively using the network, the change of address will upset its communications, as the address has the dual role of providing both location, and identification information. Changing the address to meet the locality of address requirements also changes the identity of a node, any peer wth whom it was communicating will not recognise this node as its partner.

This is the problem Mobile IP sets out to address. The desire is to allow nodes to move, in the IP sense, that is, to alter their addresses, while retaining extsting application communication sessions. The probable cause of the address change is that the node has moved to a new location, hence the name, Mobile IP, but any address change to an individual node, with or without actual motion, triggers this protocol. A simple answer to this problem would be to just inform the peer of the address change, and continue. Doing this would clearly require adequate security provisions to prevent connection hijacking, not a trivial matter, but when Mobile IP was being developed, a more debilitating problem was encountered.

Mobile IP for IPv4 was developed as an addendum to IPv4. There were many IPv4 systems already operating, with little prospect of ever moving or changing address, and no need for Mobile IP protocols. It was considered unlikely, at best, that many of the existing network nodes would be upgraded to understand Mobile IP. This would create a deployemt deadlock, without peers to communicate with that would understand an address change notification, assuming a secure way to make that notification was developed, and consequently, there would be no point in having mobile nodes wasting resources sending any such notifications. The Mobile IP solution goals required that no changes be demanded of typical static IPv4 nodes. The way that MobileIPv4 met this challenge will be explained in this chapter.

Mobile IP for IPv6 did not have the problem of existing nodes that drove the MobileIPv4 solution. MobileIPv4 existed already before IPv6 was defined, and the developers of IPv6 understood the need to handle mobile devices. IPv6 could, and did, demand that all conforming nodes understand Mobile IP signalling, hence a message notifying a peer of an address change becomes a possibility for MobileIPv6, providing the security issues can be addressed adequately. MobileIPv6 became a different protocol than MobileIPv4 because of this, and because IPv6 packet processing methods allowed some other ceoptimizations to be made. Mobile IPv6, and its differences from MobileIPv4 will also be explained.

3.2 Definitions

The following definitions are used in the discussion of Mobile IP:

- A Mobile Node (MN) is a node for which mobility is required.
- The Home Network is the network to which the MN is normaally attached.
- A Home Agent (HA) is a node, typically a router, on the MN\x92s home network that assists the MN retain its connectivity.
- A Foreign Agent (FA) is a router on the other network where the MN is connected when not at home. It can be used to assist with keeping the MN correctly communicating. Foreign Agents are optional and not often used in practice.
- A Correspondent Node (CN) is any node which is communicating with the MN.
- The Home Address (HoA) means the address of the MN on its Home Network. This is the address which should be reachable at all time, regardless of the MN's actual location.
- A Care of Address (CoA) is a new address which identifies the MN when not at home.
- A Foreign agent-based COA (FCOA) is an IP address assigned to the FA (packets are detunneled at the FA which sends them to the MN.)
- A Co-located COA (CCOA) is an IP address which belongs foreign network which is assigned to the MN via a registration process (packets are detunneled at the MN). The FA, if any, acts as the default router.
- A Tunnel is a virtual private channel using encapsulated packets.

3.3 Mobile IP

Mobile Internet Protocol (MIP), a standard proposed by IETF, is designed to solve the problem of moving nodes by allowing each mobile node to have two IP addresses and by transparently maintaining the binding between the two addresses. One of IP addresses is the permanent home address that is assigned at the home network and is used to identify communication endpoints. The other is a temporary address called a Care-of Address (CoA) which is obtained at each new location. Specifically, Mobile IP provides a mechanism for forwarding IP packets to mobile nodes which may be connected to any link, while using their permanent IP address as the node and connection identifier.

3.4 Mobile IPv4

Mobility support for IPv4 [26] defines a protocol that allows transparent routing of IP datagrams to mobile nodes as they move about from one network to another on the Internet. When a Mobile node moves into a foreign network, its communication activities are not disrupted. Instead, all the needed reconnection occurs automatically and without user interaction.

This section describes Mobile IP functionality. Typically, there are four main Mobile IP procedures. The first procedure is Movement Detection, the MN has to recognize if it has moved from one network to another network or just moved inside the network. In the latter case no address change is required, and MIP is not required. For the second, Location Discovery, the MN must discover where it is, whether local or foreign. The next is the Registration procedure: when the MN has moved to a foreign network, it must register its new Care-of Address with its HA to update the registration entry. The final procedure is data delivery, the MN establishes a tunnel to permit communication between the MN, HA and CN.

3.4.1 The procedure of Movement Detection

Movement detection is the process by which a Mobile Node (MN) detects that the Mobile IP (MIP) procedures need to be invoked - that is, that the MN has moved. The process ideally begins with a notification from the link layer that a new link layer association has been formed, but can operate without link layer involvement. After determining that movement might have occurred, the MN must determine if there was actual movement, or simply a reconnection.

There are two common methods for detection that the mobile node has moved, first by use of the advertisement lifetime and second by noticing a change of the network prefix carried in an Agent Advertisement, or assigned to the MN.

The first algorithm uses the lifetime.

The MN records the lifetimes in the advertisements it receives from each source of Agent Advertisements. The value of lifetime is updated each time a new advertisement arrives.

When the MN notices that the lifetime has expired, it must assume that it has lost contact with the current agent. The MN is then free to try registering with another agent from which it can receive advertisements. The new advertisements may indicate to the MN that it has moved, or may simply indicate that one FA on the LAN has been replaced by another.

The second algorithm is comparison of the network prefix.

The MN continuously listens for Agent Advertisements which, by so downing, allows it to observe the changing of the network prefix carried in each received Agent Advertisement message. When the MN detects that the network prefix has changed, it may assume that it has moved. The MN will also be monitoring any leased address it has received via DHCP [6] or other means. When it replaces one address with another, it has moved for MIP purposes, even if the prefix remains unchanged.

Then a MN detects that it has moved to a foreign network, it will commence the registration procedure with its HA to bind its home address with its new Care of Address.

Otherwise if the MN detected that it has returned to its home network, it must deregister from the HA so future packets can be received directly again.

3.4.2 The procedure of Agent Discovery

Each mobility agent periodically broadcasts or multicasts ICMP router advertisement messages. When the mobile node receives an Agent Advertisement it determines whether it is on its home network or a foreign network. Two cases may occur:

Case 1: The mobile node is on home network.

1. When the MN detects that it is on home network, using the movement detection functionality explained in the previous section, and where the MN was home previously, it operates without mobile service.
2. Otherwise if the MN is returning to its home network, the mobile node will deregister with its HA to delete the binding list kept by the HA, after which it will operate normally as if it had never moved.

Case 2: The mobile node is on a foreign network.

1. When the MN detects that it is on a foreign network, as determined by the movement detection function, it obtains a care of address from the Foreign Agent on the foreign network, if required.
2. The MN must register its new Care of Address with its HA by sending a Registration Request and receiving a Registration Reply message using the registration procedure via the Foreign Agent that is explained in the next section.

Agent Advertisement extension Format

4 bytes (32 bits)

Type	Length	Sequence Number
Registration Lifetime		R|B|H|F|M|G|V| Res
Zero or more Care of Address		

Figure 3.1 Agent Advertisement extension Format

The Agent Advertisement extension Format is illustrated in figure 3.1. The bit field within Agent Advertisement contains information to allow recipients to determine whether the mobility agent is a Home Agent or a Foreign Agent or both. Also these bits reveal supported encapsulation mechanisms, for which the common encapsulation is IP within IP.

The Registration Lifetime tells the maximum time, in seconds, that the mobility agent is willing to grant registration. The value zero (0x0000) indicates that a previous registration has expired. and all one bits (minus one in two's complement, 0xffff) indicates an indefinite registration time (infinity).

If the last field contains one or more Care of Addresses, these are the Foreign Agent's Care of Addresses that the mobile node within that foreign network can use to register a binding to its Home Agent.

The signaling Flow of Location Discovery

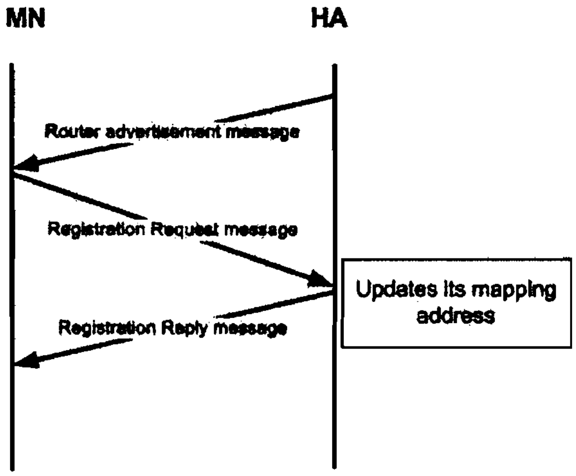

MN
HA
Router advertisement message
Registration Request message
Updates its mapping address
Registration Reply message

Figure 3.2 Case1: Location Discovery

==

In Figure 3.2 illustrates the signaling used for flow of location discovery when the MN moves or starts. When the HA is advertising the Router Advertisement message, then MN received it, MN can configured its home address from its HA and initial to register with its HA.

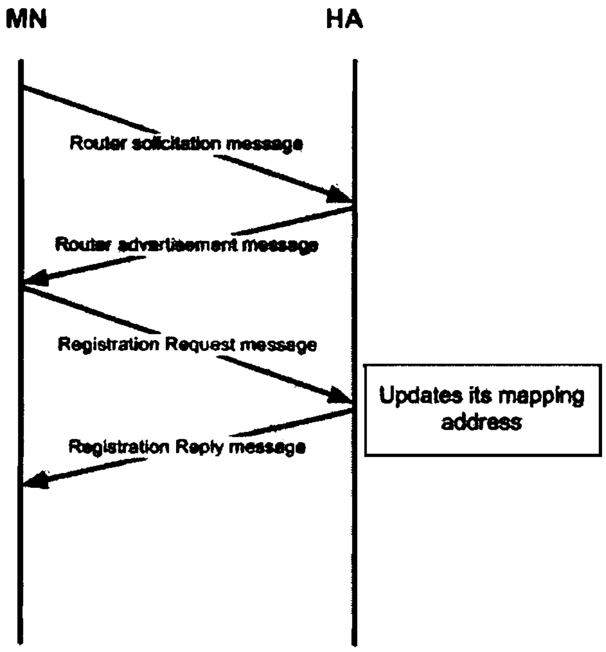

MN
HA
Router solicitation message
Router advertisement message
Registration Request message
Updates its mapping address
Registration Reply message

Figure 3.3 Location Discovery

In Figure 3.3 illustrated the signaling when the MN boot or start after the Home Agent advertised Router Advertisement Message, so the MN must send Router solicitation message for request the network information from HA. When the mobility HA received the request message, it sends the Router Advertisement to MN.

3.4.3 Registration Procedure

Once the MN has detected that it has moved to foreign network. The MN must inform its HA about its new location. Because if it does not do it, when the CN or the other MN sends packets to it. The HA is unable to tunnel these packets to the MN.

When the MN decides to register its new binding, the procedure can be described separate three cases that depend on it's a used Care of Address.

Case 1: The MN using a Co-located Care of Address (CCOA) and none of the received Agent Advertisement contained "R" bit. The mobile node can send a Registration Request message directly to its HA.

Case 2: The MN using a Foreign agent-based Care of Address (FCOA). The MN must instead send the registration to its Foreign Agent which will process and forward it to its HA.

Case 3: The MN using Co-located Care of Address (CCOA) and the "R" bit of Agent Advertisement was set. This case is recommended but not mandatory that the MN must send the registration via Foreign Agent.

All registration messages were sent by using UDP port 434 or contained in UDP datagram. The registration message can be separate to two types are the Registration Request Message and the Registration Reply message.

The Registration Request Message Format

The Registration Request message is sent from the MN to the HA that is used to register new Care-of Address. The Registration Request message may be relayed to the HA by the foreign agent or may be sent directly from the MN to the HA that depend on the kind of the CoA. The Registration Request message format can be illustrated in Figure 3.4

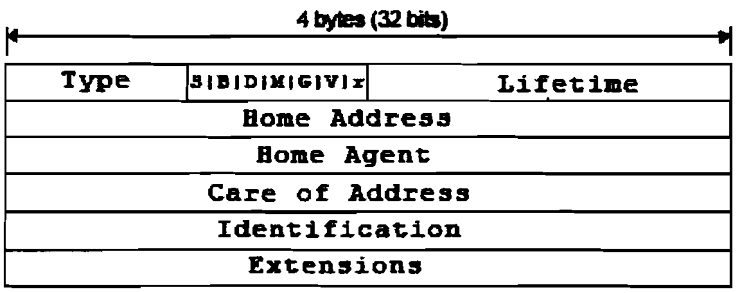


Figure 3.4 The Registration Request Message

Lifetime Field is preferred lifetime of binding which has two special values are 0x0000 and 0xffff used for deregistration and infinite lifetime in order.

Care of Address Field is used for distinguish different registration request and to provide help in authentication.

Identification is a 64-bit number that constructed by the MN and used for matching Registration Requests with Registration Replies. Also it used for protecting against replay attacks of registration messages.

Extension Field is used for the most common of them being the authentication extension.

The Registration Reply Message Format

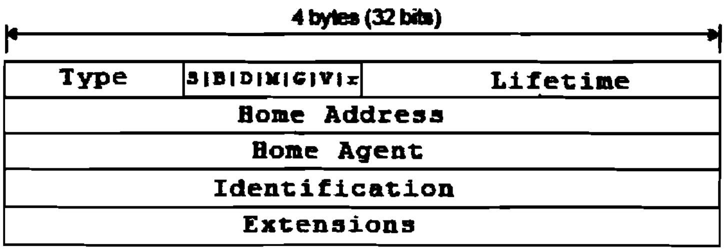


Figure 3.5 The Registration Reply Message

The Registration Reply message is sent from the HA to the MN that is used to inform the MN if the registration was successful or not. This is contained in UDP packet with source port 434 and a destination port set to the source port that found in the corresponding Registration message. The Registration Reply Message Format can be illustrated in Figure 3.5.

Type equal 3 for identification of Registration Reply.

Code Field tells whether the Registration succeeded or failed. If code field identifies that fail registration, the lifetime field should not be processed. Another field means same in Registration Request message.

The procedure of registration can be explained in the following steps.

1. The MN sent the Registration Request message (RReq) to its HA.
2. When its HA received the RReq message, HA process it and sends back a Registration Reply to the MN.

The HA can accept or deny the registration and procedure of Registration Reply to mobile node depends on the procedure of Registration Request. So that if the Registration Request is sent via Foreign Agent when HA sends reply, the Registration Reply also via Foreign Agent can be illustrated signaling in Figure 3.6.

Otherwise, if the MN sent registration request to its HA directly, HA will send the Registration Reply directly to MN that can be illustrated signaling in Figure 3.7.

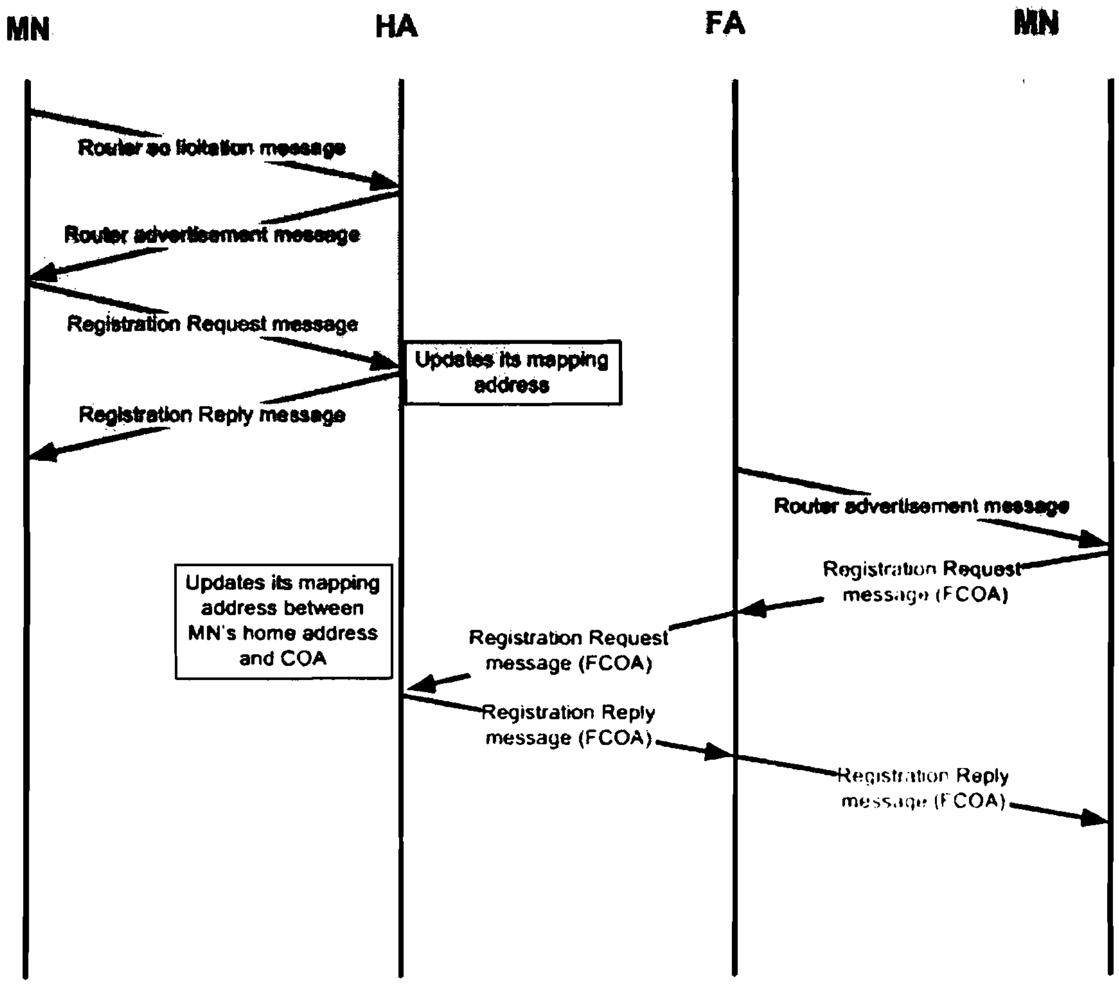

MN
HA
FA
MN
Router solicitation message
Router advertisement message
Registration Request message
Updates its mapping address
Registration Reply message
Router advertisement message
Registration Request message (FCOA)
Updates its mapping address between MN's home address and COA
Registration Request message (FCOA)
Registration Reply message (FCOA)
Registration Reply message (FCOA)

Figure 3.6 Location Discovery and Registration Process of Mobile IPv4 when MN uses FCOA

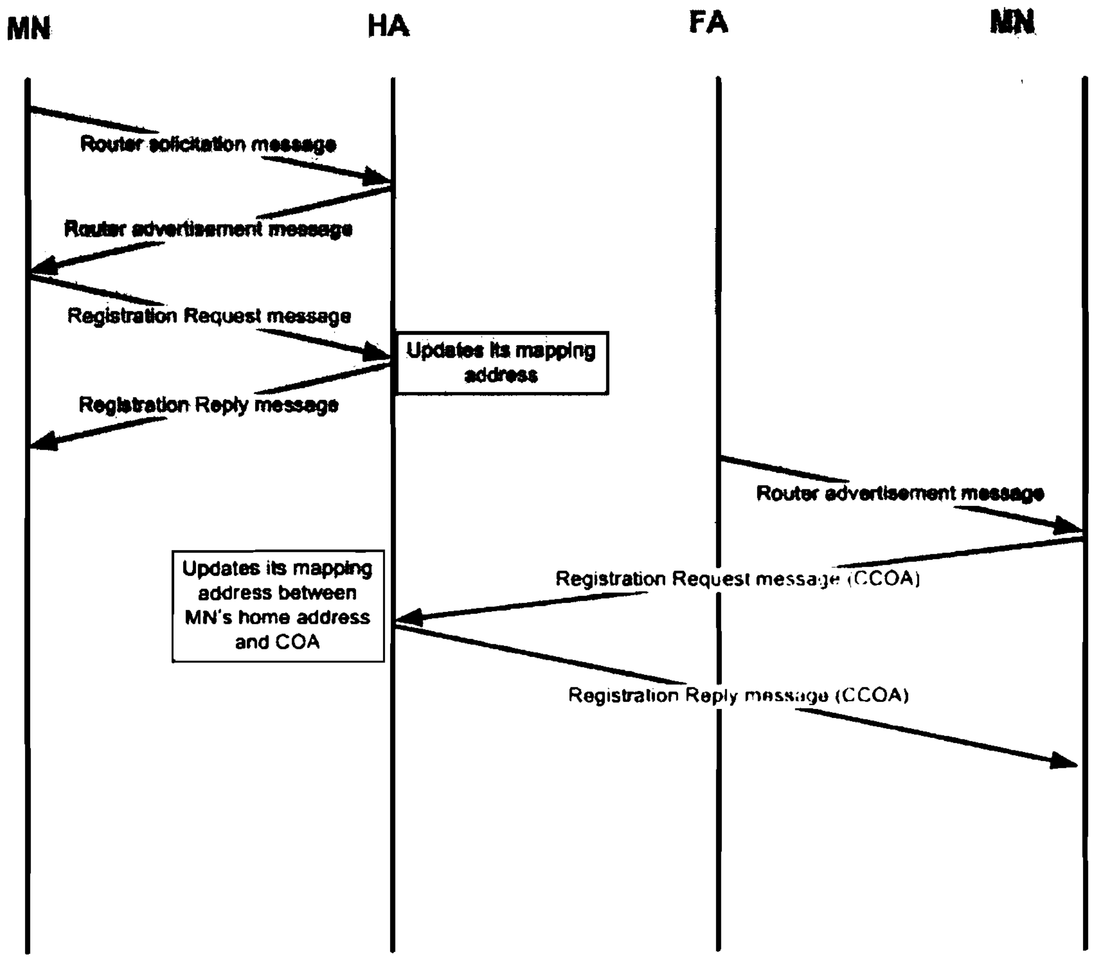

MN
HA
FA
MN
Router solicitation message
Router advertisement message
Registration Request message
Updates its mapping address
Registration Reply message
Router advertisement message
Updates its mapping address between MN's home address and COA
Registration Request message (CCOA)
Registration Reply message (CCOA)

Figure 3.7 Location Discovery and Registration Process of Mobile IPv4 when MN uses CCOA

3.4.4 Data delivery mechanism of mobile IPv4 Protocol

When the Correspond Node wants to communicate with the MN, there are two main cases for data delivery.

1. **The mobile node exists in its home network.**

In this case, the packets are routed from CN to MN's home network by using the standard IPv4 routing mechanism.

2. **The mobile node is at foreign network.**

While the MN moved in the foreign network, the procedure of data delivery mechanism may occur three cases depend on the type of a Care of Address of MN uses.

Case 1: The MN uses a Foreign Care of Address (FCOA) which is IP address of the Foreign Agent. The procedure occurs in this case can be shown in the following steps.

1. The CN sends a packet to the MN's home address in MN's home network.
2. After a packet arrived to mobile's HA, it will intercept a packet and tunnels the original packet to the mobile's Foreign Agent.
3. The Foreign Agent detunnels the received packet and delivers it to the MN by Layer 2 address.
4. Then the MN receives packet, it sends any packets to the CN by using the standard routing mechanism. Each step can be illustrated in Figure 3.8.

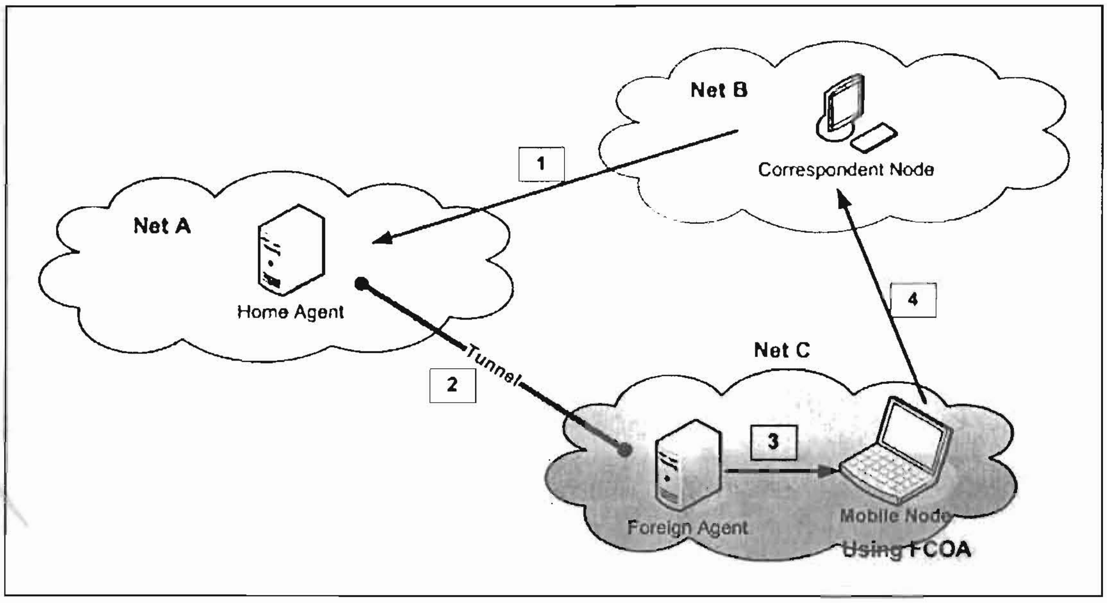


Figure 3.8 Data Delivery when mobile node using FCOA

Case 2: The MN uses a Co-located Care of Address (CCOA) which is obtained via some dynamic assignment protocol such as DHCP. This occurred procedure can be shown in the following steps.

1. The CN sends a packet to the MN's home address in MN's home network.
2. When a packet arrives to mobile's HAt, it intercepts a packet and tunnels it to the mobile's Co-located Care of Address.
3. Then the MN received a packet, it will detunnel this packet and sends any packets to CN by using the standard routing mechanism. Each step can be illustrated in Figure 3.9.

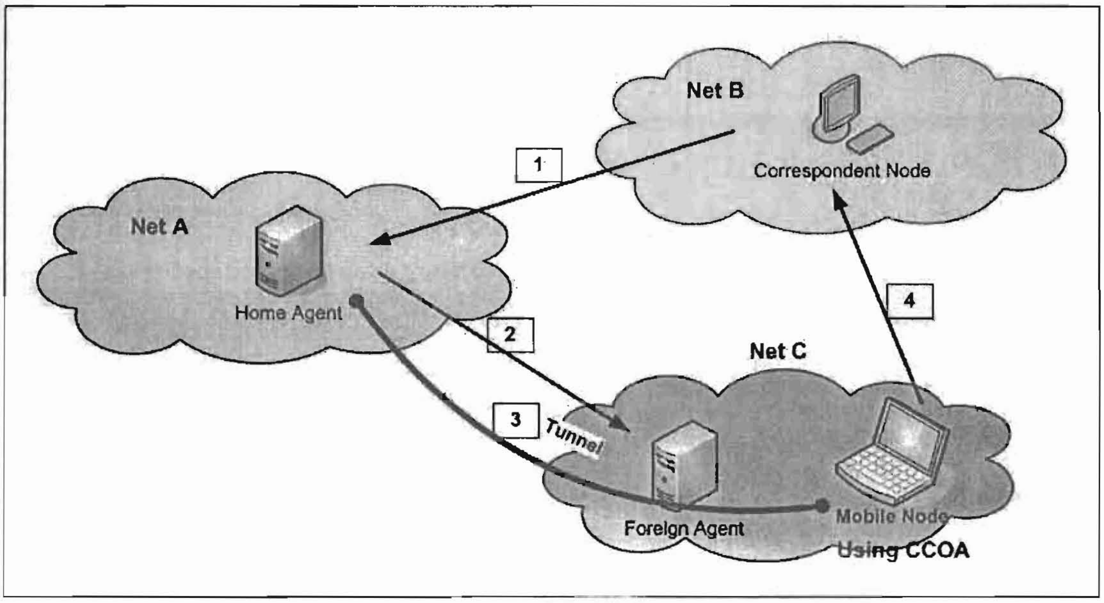


Figure 3.9 Data Delivery when mobile node using CCOA

Case 3: where the CN and MN are in the same foreign network, the HA is not required to forward packets to MN.

1. The CN can instead send the datagram directly to MN because no routing is involved.

3.4.5 Conclusion

From the previous section can conclude that if the MN uses FCOA, the detunneling informed at FA and FA sends the packets to MN by Layer 2 address. Otherwise if the MN uses CCOA, the packets are detunneled at MN. The FA acts as the default router.

3.5 Mobile IPv6

Mobile IPv6 is a mobility support protocol for IPv6 [27] at the network layer which allows an IPv6 node to transparently maintain the connections while moving from one subnet to another. Each host is identified by its home address although it may be connecting to through another network. When connecting through a foreign network, a host sends its location information to a home agent which intercepts packets intended for the host and tunnels them to the current location. If a host does not supports Mobile IPv6, all the existing connections on the host are terminated when it changes its point attachment.

Mobile IPv6 retains most of the concepts that were present in Mobile IPv4 such as Correspond Node, Home Agent and Mobile Node. Mobile IPv6 extends the IPv6 protocol by introducing new destination option to IPv6 protocol.

There are the new options as following:

1. Binding Update
2. Binding Acknowledgement
3. Binding Request
4. Home Address

The options are carried in the destination option extension header, the destination option can be piggybacked correct destination instead of having to send a separate UDP packet.

The Home Address option is used with all packet that is sent in the foreign network. Additional the Home Address option is always used when the Binding Update option is present.

3.5.1 Location Updates - Registering the COA

When a Mobile node attaches to a new foreign network, it sends a registration request to its Home Agent to register its care-of address. If the MN is using a foreign agent Care of Address, the registration request is sent via the foreign agent. The registration request includes an extension with a cryptographic authentication value. The MN calculates the authentication value based on fields in the registration request and a static key that is specified on both the HA and the MN.

The Home Agent authenticates the MN's registration request by calculating its own authentication value and comparing it with the authentication value from the MN. After the HA authenticates the registration request, it sends a registration reply message to the MN. The registration reply also includes a lifetime for the registration

3.5.2 Registration and deregistration Care of Address

When a Mobile Node moves to a new link, it sends a binding update to its Home Agent or other Correspondent Nodes in its list to inform them about its current Care of Address. As illustrates in Figure 3.10, the destination option header containing the mobile IPv6 Binding Update option consists of an A an H and an L bit and the fields: lifetime, identification, MN's home address and Care-of Address. The A bit indicates whether the receiver should reply with a Binding Acknowledgement or not. The H bit is used if the mobile node wants the receiving node to be its HA. The L bit is sent if the MN also wants to receive packets destined to its link-local home address.

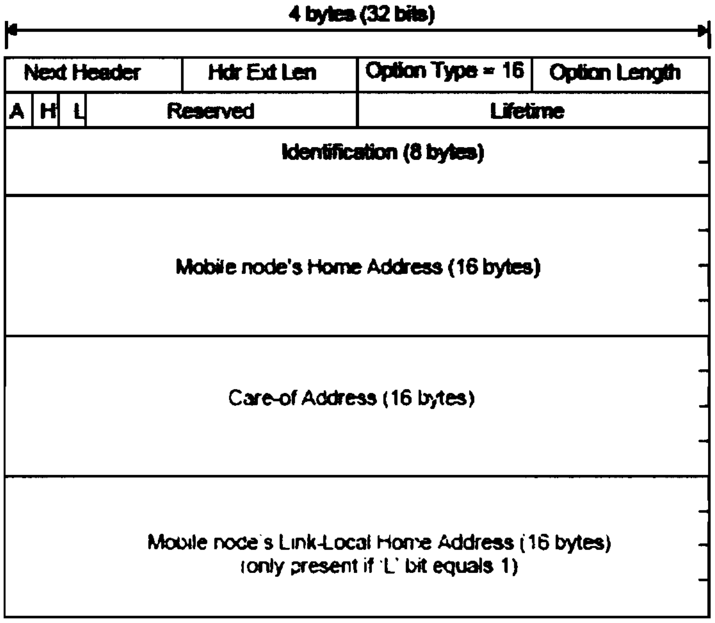

4 bytes (32 bits)
Next Header
Hdr Ext Len
Option Type = 16
Option Length
A
H
L
Reserved
Lifetime
Identification (8 bytes)
Mobile node's Home Address (16 bytes)
Care-of Address (16 bytes)
Mobile node's Link-Local Home Address (16 bytes)
(only present if 'L' bit equals 1)

Figure 3.10 Destination Option Header is Containing the Mobile IPv6 Binding Update Option

The identification field, the home address field and the Care-of Address field is the same as in the registration request field of IPv4.

A Binding Acknowledgement is sent to the Mobile Node by its Home Agent or any other Correspondent Node to indicate that the Binding Update was successfully received and whether it was accepted or not. The status field is used for this purpose. This is illustrated in Figure 3.11. The lifetime, identification and MN's home address are the other fields of this option, which are copied from the received Binding Update. An extra field is the refresh field, which indicates how long the sender of the Binding Acknowledgment will store the Care of Address of the MN. This is used as an indication for the MN how often it has to send a Binding Update to that node. A Binding Acknowledgment is required if the A bit in the Binding Update is set to 1.

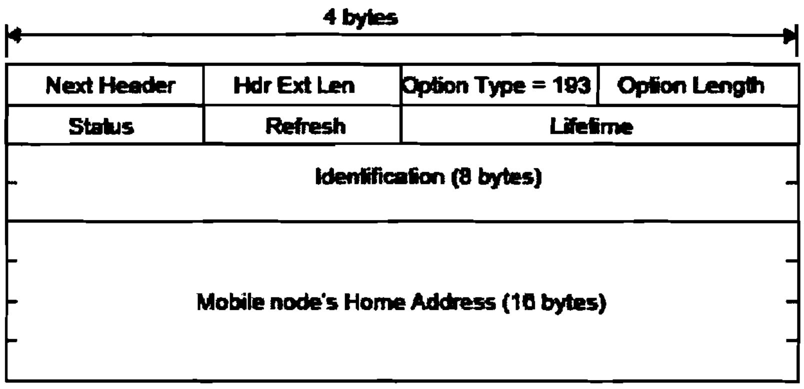


Figure 3.11 Destination Option Header Containing the Mobile IPv6 Binding Acknowledgement option

If a CN wants to know the Care of Address of a Mobile Node, it sends a Binding Request to that MN. The only information in this request is the request itself so the Binding Request only contains the option type and the option length fields, as illustrated in Figure 3.12. The MN does not necessarily have to respond to the request by sending a Binding Update. The request is also used to get new lifetime and refresh fields, when they are expired or need to be refreshed.

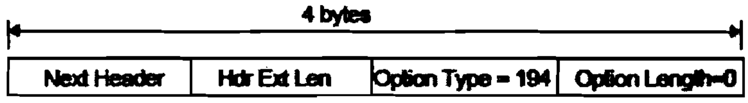


Figure 3.12 Destination Option Header Containing the Mobile IPv6 Binding Request Option.

3.5.3 Data delivery mechanism of Mobile IPv6 Protocol

When Correspond Node want to send the packets to the Mobile Node, firstly CN checks that it has a binding for destination or MN. So data delivery mechanism can occur two case depend with the CN has or does not has the binding of MN.

Case 1: The binding exist

1. The CN attaches a routing header with a single route segment to the packet and set the destination address to Care of Address indicated by the binding and set original destination address to the route segment within the routing header.

2. When the packet arrives at the MN, MN detects the routing header and send packet forward to the address indicated routing header.

This way using source routing and binding caches. Mobile IPv6 allow CN to communicate directly to the MN avoid the triangle routing.

Case 2: Correspond Node does not have a binding for mobile node. The procedure of data delivery can be explained in the following steps.

1. The CN sends the packet to the original destination address which causes the packet to be routed to MN's home network.
2. When the packets arrived to the MN's HA, HA intercepts the packet intended to MN and tunnels the packet to MN's Care of Address (COA) using IPv6 in IPv6 encapsulation.
3. After that the MN received the packet and detunneled it. The MN attaches a Home Address option to hide its Care of Address for communication between CN and it. This procedure can be illustrated in Figure 3.13.

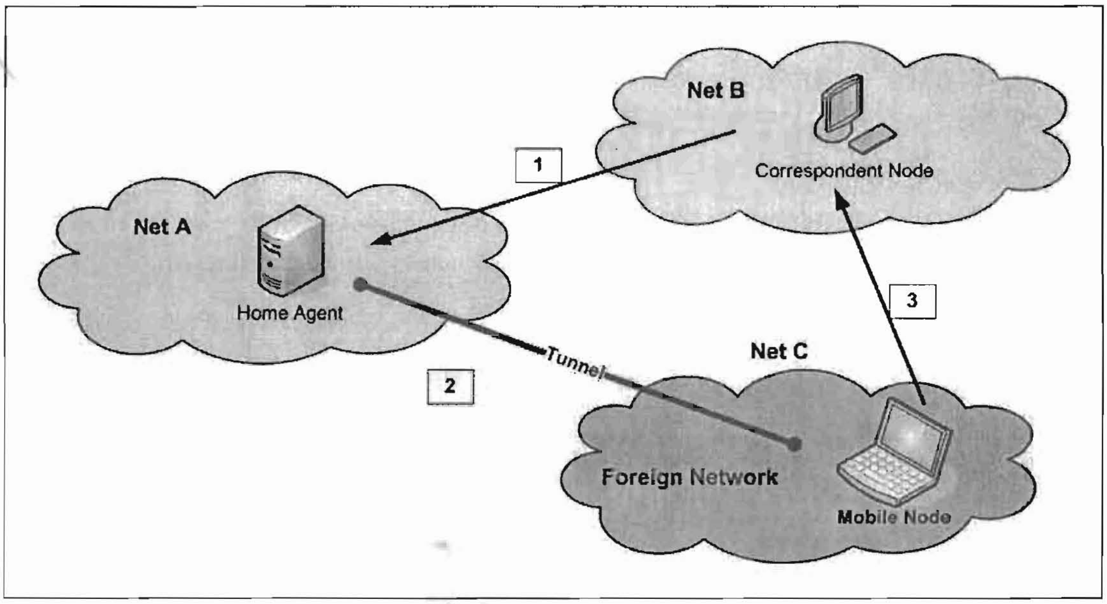


Figure 3.13 Data Delivery when does not have a binding for mobile node.

3.5.4 The signaling flow of MIPv6 procedure

In Figure 3.14 illustrates all signaling flow of MIPv6 procedure. The first step, MN boot on the home network. When MN acquired a home address, MN send BU message to its HA to registration. Then MN moved to the foreign network, MN receive new address which called Care-of Address (CoA). MN send BU message to its HA for updating new address in binding cache. After HA sent BA message back to MN and MN received BA. The tunnel between HA and MN is established.

Then Correspondent Node (CN) connects to MN that moved to foreign network. The data packets are sent to MN's network. HA will intercept these packets and encapsulate them send to MN through tunnel.

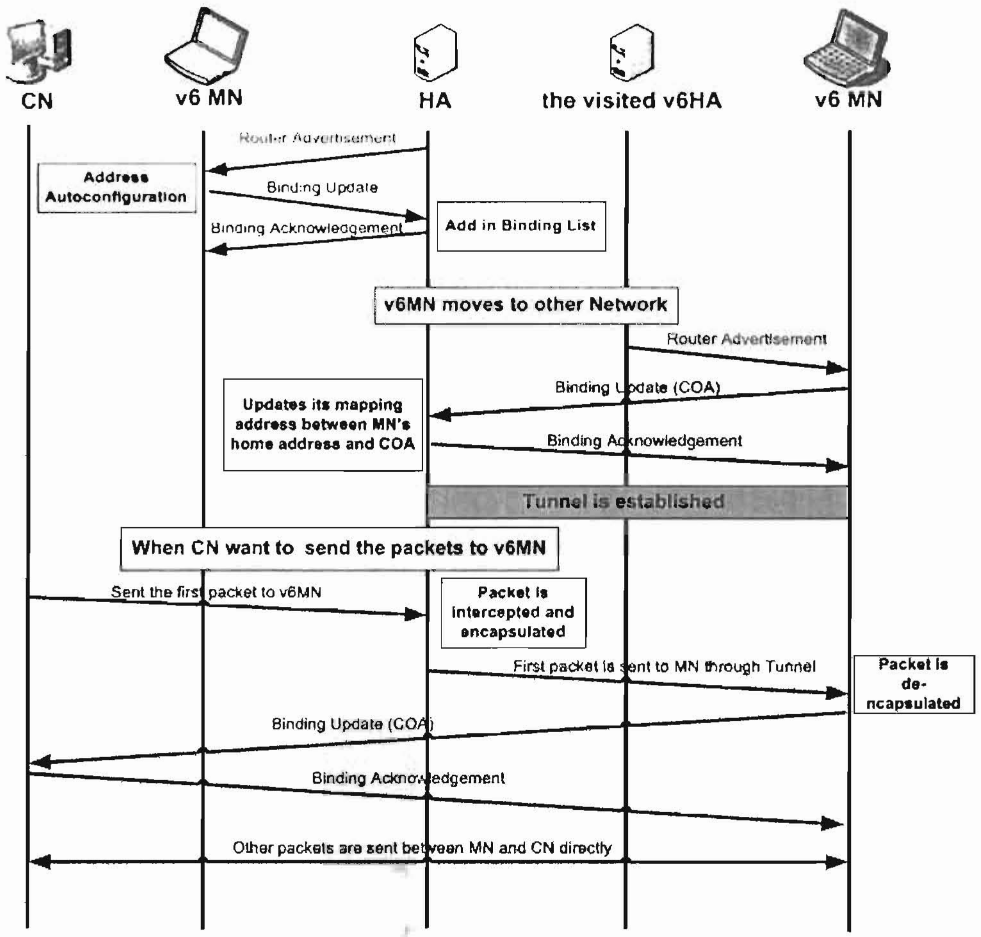

CN
v6 MN
HA
the visited v6HA
v6 MN
Router Advertisement
Address Autoconfiguration
Binding Update
Binding Acknowledgement
Add in Binding List
v6MN moves to other Network
Router Advertisement
Binding Update (COA)
Updates its mapping address between MN's home address and COA
Binding Acknowledgement
Tunnel is established
When CN want to send the packets to v6MN
Sent the first packet to v6MN
Packet is intercepted and encapsulated
First packet is sent to MN through Tunnel
Packet is de-ncapsulated
Binding Update (COA)
Binding Acknowledgement
Other packets are sent between MN and CN directly

Figure 3.14 The signaling flow of MIPv6 procedure

3.6 Security for Mobile IP

Today security is always concern in any internetworking environment, especially important with mobility network which usually uses wireless communication that risks more than wire communication. Anybody can intercept the packets, so the Mobile IP has a number of weakness.

Therefore this section discusses the security aspects of the Mobile IP. First we briefly describe the security architecture for Internet Protocol as IPSec [29] which is designed by IETF to provide interoperable, high quality, cryptographically-based security. The IPSec defines a suit of protocols which describe security mechanisms and services for the IPv4 and the IPv6 and upper layer. The protocols of IPSec suit are the IP Authentication Header (AH) [30] protocol and Encapsulation Security Payload (ESP) [31] protocol that can be illustrated IPSec Security Architecture for IP Network in Figure 3.15. AH provides authentication for as much of the IP header as possible, as well as for next level protocol data. ESP provides authentication for the payload data. ESP header is inserted after the IP header and before the next layer protocol header (transport mode) or before an encapsulated IP header (tunnel mode). Also ESP can be used combination with AH.

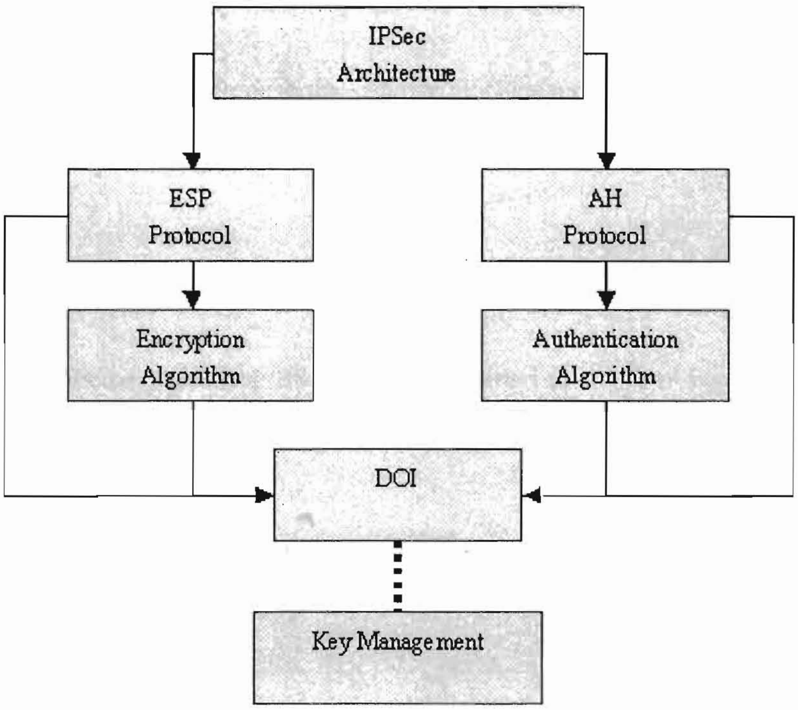


Figure 3.15 IPSec: Security Architecture for IP Network

Security issue of MIPv4

In MIPv4 majority of risk is associated with the authentication of the Mobile Node. Registration information such as the Registration Request (RReq) and the Registration Reply (RRep) must be authenticated to make sure that the received information is truth and sent from the really sender. Otherwise malicious node can disguise as Mobile Node by sending fake Care-of Address to the Home Agent that makes the actual nose unavailable. The malicious node can reach access to the Mobile Node's traffic.

Security issue of MIPv6

For MIPv6, the security features are integrated and provided as an extension to header. Information that is sent between the Mobile Node and the Home Agents is protected by IPsec AH [30] and ESP [31] protocols. IP security can be applied to the Binding Updates (BU) and Binding Acknowledgement (BA) for making sure that the Home Agent received the really information from the really sender.

3.7 Comparison of Mobile IPv4 and Mobile IPv6

Mobile IPv6 shares many features with Mobile IPv4, but the protocol provides many improvements over Mobile IPv4. The following is a list of the major differences between Mobile IPv4 and Mobile IPv6 concept that can be helpful for migrating from Mobile IPv4 to Mobile IPv6.

- IPv4 addresses are 32 bits long as for IPv6 addresses are 128 bits long.
- Mobile IPv4 uses tunnel routing to deliver data-packets to Mobile Nodes as for Mobile IPv6 uses tunnel routing and source routing with IPv6 Type 2 routing headers.
- Mobile IPv4 deploys Foreign Agents for Mobile Node movement detection and to decapsulate data-packets addressed to the Mobile Node's Care-of Address as for IPv6 Mobile Nodes decapsulate messages sent to its Care-of Address itself and uses IPv6 Router Advertisements for movement detection, thereby eliminating the need for Foreign Agents.

- Mobile IPv4 uses Agent Discovery for Movement Detection as for Mobile IPv6 uses IPv6 Router Discovery.
- Mobile IPv4 Route Optimization is an extension to the protocol, not part of the base RFC that requires pre-configured and static security associations and was difficult to operate with ingress-filtering routers as for Mobile IPv6 Route Optimization is a fundamental part included in the protocol and provides integrated return routability to dynamically secure route optimization and operates effectively with ingress-filtering routers.
- Mobile IPv4 reverse tunneling is an extension to the protocol as for Mobile IPv6 bi-directional tunneling is part of the core protocol.
- Mobile IPv4 uses one home address as for Mobile IPv6 uses a globally routable home address and a link-local home address.
- Mobile IPv4 uses ARP to determine the link layer address of neighbors as for Mobile IPv6 uses IPv6 neighbor discovery and is de-coupled from any given link layer.
- Mobile IPv4 dynamic home agent address discovery uses a directed broadcast approach and returns separate replies from each home agent to the Mobile node as for Mobile IPv6 dynamic home agent address discovery uses anycast addressing and returns a single reply to the Mobile node.
- Mobile IPv4 Mobile nodes can obtain care-of Addresses via agent discovery, DHCP and manual configuration as for Mobile IPv6 Mobile Nodes can obtain care-of addresses via stateless address auto-configuration, DHCP, and manual configuration
- Mobile IPv4 uses foreign agent care-of address and a co-located care-of Address as for Mobile IPv6 care-of addresses are all co-located.

3.8 Summary

This chapter, we give a Mobile IP concept, functional of both IPv4 and shows the necessary packet format of Mobile IP that uses for registration to Home Agent. In addition, we discussed the comparison of Mobile IPv4 and Mobile IPv6 that had the different architecture made both cannot compatible.

In next chapter, we will introduce the transition mechanisms that were propose for interoperation of IPv4 and IPv6. Also we will choose one for solving of the uncompatible of MIPv4 and MIPv6 to still connectivity when MN moves to different IP version network.

CHAPTER 4

MOBILITY MECHANISM BETWEEN MIPv4 and MIPv6

In chapter 3, we saw that Mobile IPv4 and Mobile IPv6 have different architectures. They have differences of basic operation that mean they cannot directly inter-operate during the period of transition from IPv4 to IPv6. In this chapter, we provide the problem statement first and also present the design for a solution to these problems at a high level that introduces intelligence in the Mobile Nodes and Home Agent to enable connectivity between Mobile Node and Correspondent Node while allowing Mobile Nodes to move between IP protocol versions by using the existing Transition Mechanisms.

The first section explains why a mobility mechanism between MIPv4 and MIPv6 is required, and sets out some assumptions and requirements for a solution. The planned technique to be used, and a rational for the approach, is given in the next section. In section 4.3 we give an overview of new registration messages that will be used when the MN moves to a different network version.

4.1 Introduction

When the MN moves to another network, the MN always acquires a new address which is called the Care-of Address (CoA). The MN must send new this IP address to its Home Agent to update the Binding Cache or Binding Entry, as explained in Chapter 3. These can be done with mobility management protocols that are defined for IPv4 and IPv6, though the procedures for each vary. A MIPv4 capable node can use Mobile IPv4 to maintain connectivity while moving between IPv4 subnets. Similarly, MIPv6 capable nodes can use Mobile IPv6 to maintain connectivity while moving between IPv6 subnets. However, MIPv4 cannot carry an IPv6 CoA should a with an IPv6 HA move to an IPv6 only segment, and such a MN will have no IPv4 CoA to send, and a MIPv6 message would not be understood by such a HA. This is a reason for the Mobile Node cannot maintain connectivity while moving between the different IP version subnets.

We have seen in Chapter 2 that the transition between IPv4 and IPv6 will almost certainly result in there being a period when some segments of the network are IPv4 only, and other segments are IPv6 only. A mobile node may have no option but to move directly from a segment of one of those types to a segment of the other type. Traditional Mobile IP (v4 and v6) fails in this environment. We set out to provide a solution to this problem, to allow unrestricted movement between IPv4 and IPv6 segments, with either an IPv4 or IPv6 connected Home Agent, and any set of IPv4 and IPv6 Correspondent nodes.

In addition to enabling communications in the, perhaps, rare case of a HA being on a network that runs only one IP protocol, and the MN moving to a network that runs only the other, we anticipate benefits nodes on dual stack networks. Suppose that a Mobile Node moves within a dual stack access network. Every time the mobile node moves, it currently needs to send two mobile IP messages to its Home Agent to allow its IPv4 and IPv6 connections to remain. There is no reason for such duplication. Further if local mobility optimizations are deployed, such as Hierarchical Mobile IPv6 (HMIPv6) [32], and Fast handovers for Mobile IPv4 [33], the mobile node will need to update the local agents running each protocol. It is not desirable to have to send two messages and complete two sets of transactions for the same basic optimization.

In order to realize this result, we need to make some assumptions, and set some design restrictions. First, we must assume that there is some form of communication from the foreign network in which the MN has arrived, to the entire Internet, both IPv4 and IPv6 segments. Obviously if there is no way to send any data to (say) an IPv4 only part of the Internet, and our Home Agent is in that part, then we cannot possibly communicate with it, or cause it to learn the MN's current location. We do not specify the means of this communication, anything that enables packet exchange should suffice, but we expect some form of protocol translation will probably be involved.

Second, we cannot assume any control over the network into which the MN has moved, nor the CN or its network. The foreign network may be any random access network available to the MN, the CN might be any node on the Internet, The solution must operate with no changes required to any of those nodes or networks. We note that even if we could request updates to a protocol translation engine, assuming one actually exists, no such updates would be useful, as communications between the MN and its HA are encrypted for security purposes. We cannot ask the protocol translator to modify MIP packets traveling between MN and HA between MIPv6 and MIPv6 formats, nor any similar solution, such modification is not possible.

Third, we do assume that any MN that wants to have the option of migrating between v4 only and v6 only network segments, can be upgraded to run new code, implementing our new protocol, and that its HA can similarly be upgraded to understand the new protocols. A new solution obviously needs a new implementation, and these nodes are the only reasonable candidates for this upgrade. Further, as we are dealing with upgrading a communication protocol, as neither MIPv4 nor MIPv6 as currently defined can function in this environment, we necessarily must upgrade both parties to this communication, those being the MN and HA.

Fourth, it is obvious that a MN that is able to migrate, and function, on both IPv4 only and IPv6 only networks must necessarily be a dual stack implementation. We do not assume that the HA is necessarily dual stack, but we do assume that its Mobile IP implementation at least understands both IPv4 and IPv6.

The remainder of this chapter will overview the design of a solution to the problem observed, and explain the obstacles that needed to be overcome to find a solution, and why the protocol is designed as it has been. The following chapter will then provide the details of the protocol designed.

4.2 The altered technique and literature review

In this section we will analyze and choose a suitable transition mechanism to enable continuing connection between Mobile Node and Corresponding Node when MN moves to a network running only a different IP version than it was previously connected to.

Our main aim is to enable continued communications, we are not concerned with optimizations, so we will concern ourselves only with enabling communications between the MN and its HA. We assume that communications between HA, using the MN's Home Address as the MN's agent, and CN are possible, perhaps using protocol translation, so that the CN is able to send data to the MN, provided the HA is able to find, and communicate securely with, the MN.

We also assume that the MN can connect to whatever foreign network it has settled upon, and obtain a CoA from that network. The problem then is how to enable secure communications between the MN and HA. This raises the first problem to overcome. In traditional Mobile IP, MN's are either configured with the address of their HA, or use a HA discovery mechanism to find a HA given knowledge of the Home Address - that is, of the address prefix the HA must be connected to. In our environment, knowledge of the HA's address, or address prefix, is useless to the MN, as we assume the MN is, or might be, connected to a network that only communicates using a version of IP that does not understand those addresses. We cannot send a packet to an address, either known or calculated, that is not from the appropriate IP version.

Further, for this particular problem, protocol translation cannot assist. Or more correctly, to enable assistance from a protocol translator, the MN would first need to implement a mechanism to find the translator, a task for which there is no current solution, translators are generally designed to be transparent, but would then need to modify the translator to allow it to supply a suitable local protocol address to be used to contact a specified other protocol address. Since one of our design requirements is not to require modifications to the translator, any solution along these lines is excluded.

On the other hand, other applications, e-mail, WWW, etc., also need to find a local protocol address to use when the target of their communications exists only on a segment of the network that uses the other protocol. The difference is that those applications start with a domain name, and a (transparent) DNS-ALG associated with the protocol translator is used to provide a suitable address mapping, See section 2.3.1 for the details.

Thus our first modification is to require that the HA have a domain name, with that domain name registered in the DNS with suitable address Resource Records. We then require that the MN be informed of the name of the HA.

Now, whenever the MN needs to communicate with the HA, and the HA address it has is from the other IP version, the MN instead performs a DNS lookup, asking for the address of the HA in the IP version it currently needs. With the assistance of a DNS-ALG, or any other similar mechanism that would work for other communicating (static) applications, a suitable address, in the correct protocol version, should be available.

The MN now has all it needs to be able to send a data packet to the HA. That packet should contain suitable address information to enable the HA to reply. The next issue is what this packet should contain. Clearly, the desire is to send a Binding Update, or a Registration message, to the HA. A second problem now appears. Any such message must inform the HA of our current CoA. The issue is that that CoA is in, we are assuming, a format (that is, for a protocol) that is useless to the HA, sending an IPv4 only HA (a HA connected only to an IPv4 network, whether or not it comprehends IPv6) an IPv6 CoA is useless to it, only IPv4 addresses work.

We could now attempt the solution above, in reverse. That is, we could require the MN to have a domain name, and for that name to be registered in the DNS. We would then require the MN to update its DNS registration to its DNS server, supplying its new CoA, and inform the HA of its domain name, and that it has moved. The HA would then perform a DNS lookup, and obtain a suitable address of the protocol version it requires. This might be technically possible, but is impractical. The DNS design demands slow updates - that is, old answers are cached for a period of minutes, to weeks, and updates made during the cache validity time are invisible until later. A mobile node would quite possibly move several times before any new address was visible to he HA. Further, even if the DNS Resource Record Time to Live was set to zero, to prohibit caching, DNS update coherency between the multiple servers for each DNS zone also involves unavoidable delays - an update to one server will take time to eventually reach all servers. Any server may be queried by the HA, there is no way to expect the updated address will be available. Even if it were, this mechanism seems as if it would be unduly cumbersome and error prone.

Instead, we note that when the MN sends a packet to the HA, using the address it is able to obtain from the DNS (since the HA address is stable, it suffers none of the DNS consistency problems) the HA receives with that packet an address it can use to reach the MN. The MN has no knowledge of the value of that address, it was supplied by the address translation module of the protocol translator, but we know it must exist. So, for our solution we assume that we can make use of this fact, and allow the HA to obtain the MN's CoA in the protocol version it needs, from the source address field of the update packet we send it.

This then creates a new problem. Security of the update demands that the MN prove its identity to the HA. This is done using encryption using a common shared key between HA and MN. Only those two nodes know the key, so if one of them receives a packet it did not send itself, correctly encrypted using the shared key, it can assume that the other node sent that packet. However, in our scenario, the CoA cannot be protected this way, as that is supplied by the translator, a node not privy to the shared secret. This immediately opens the possibility of spoofing attacks, where any attacking node could send one of our new update packets to the HA, and impersonate the MN.

Fortunately, though the CoA cannot be protected, the rest of the update can be, and we can use encryption, as is done with traditional MIP to assure the HA that the update was sent by the MN. Further, the MN knows that it is unable to send a CoA inside the update packet, so it can inform the HA of that, and this information is protected. Only when the HA is told (securely) that it should extract the CoA from the outer packet headers will it do that. With this, the only attack left is for the attacking node to intercept a valid update message from MN to CN, alter the (translated) packet source address, and substitute some other value, and forward that to the HA. This allows the attacker to disrupt communications between MN and HA (a Denial of Service attack), and possibly to direct them to a victim (a different Denial of Service attack), but no more than that. The communications remain encrypted, and so private. Even then this assumes that the attacker it able to select an update message from the (encrypted) communication stream between HA and MN, which while not impossible if the attacker can obtain knowledge of when the MN has moved and so is likely to be sending an update, it is not easy.

Further, after altering the source address of the update, the HA will send its acknowledgment of the update to that address. If the altered address is such that this reply is not correctly received by the MN, the MN will simply assume that the update request was lost, and retransmit it, after some small random interval. If received without being attacked, that update would appear to the HA as if the MN had simply moved again, and the correct CoA would be entered in its Binding tables. To prevent this, the attacker would need to be continually intercepting all traffic from the MN to the HA, and rewriting all the packet source addresses. Again, this is not impossible, but is not easy, it requires the attacker to remain active, and so liable to detection, and the gain from the attack is quite small.

We have no full solution to this problem, but we consider it of small enough importance that perhaps it can be overlooked for now. Once the transition to IPv6 is complete, and this protocol will no longer be relevant, and the problem will vanish.

Once we adopt these methods, we encounter another, smaller, problem. That is, can we necessarily assume that a protocol translator will correctly translate and forwarded encrypted MIP packets in an IP in IP tunnel. Experience with current available translators suggests that would be risky. On the other hand, to be useful, a protocol translator must be able to translate routine TCP and UDP packets containing arbitrary data, where no translation of the data is required or expected. This suggests that our solution should adopt a UDP or TCP packet wrapper for the tunnel between MN and HA. Between those two possibilities we have adopted UDP. We do not need TCP's error recovery, and do not desire its flow control. UDP has less (additional) overhead, and is a closer approximation to the original Mobile IP encapsulation methods.

Having made that choice, we obtain an additional, desirable but not really planned, side benefit. The original issue that caused all the problems that we are attempting to solve is the unavailability of IPv4 addresses. Any solution that requires every mobile node to (even temporarily) be allocated an IPv4 address of its own does not seem suitable to this environment. Any IP in IP encapsulation demands the IP address be unique to the end node, there is no more addressing available to identify the recipient. Any MN in an IPv6 only network would need to be allocated an IPv4 address (a global IPv4 address) to allow it to communicate with its HA. Selecting UDP (or TCP) adds an additional 16 address bits, the port number, which can be used to permit many nodes to share one external IPv4 address, with the translator selecting the correct IPv6 destination for each packet based upon the combination of the incoming IPv4 destination address and the UDP (or TCP) destination port number. This is the same technique used by traditional (IPv4) Network Address Translation (NAT) to allow one global IPv4 address to be shared amongst many nodes using only unique local addresses.

We now have the basis of a solution. We use the DNS to obtain the HA address to use, when required. We send encrypted data in UDP packets, with the binding request containing an explicit notification for the HA to obtain the CoA from the source address of the packet it received. This allows the MN and HA to communicate, sending binding updates, and then exchanging data, which the HA relays to and from the various CN. Aside from additional delays, and a larger encapsulation overhead on the tunnel, the one real cost is the possibility of an unlikely DoS attack.

4.2.1 Other Approaches

Our work is not the only attempt to solve this problem. Other approaches have different steps and the different signaling. An example is as in "Global Mobile IPv6 Addressing Using Transition Mechanisms" [34], where Edgard Jamhour and Simone Storoz proposed an approach for implementing mobile networks with global Internet connectivity using dynamic tunneling with the 6to4 transition mechanism. However tunneling mechanisms aren't the most suitable mechanisms for our work because it operates only one way that is when the MN moves from an IPv6 to IPv4 network. Also it cannot solve inter-operation between MIPv4 and MIPv6 that is MN still uses the original MIPv6. Nevertheless, this approach modified MN to establish 6to4 tunnel to 6to4 relay to allow obtaining an IPv6 address for sending the registration message to its HA. The limitation of this approach is the IPv4 foreign network must advertise a public IP address for the MN.

Another proposal, "Seamless Mobility Between Current and Future IP Network" [43] describes and proposes a solution for Mobile IPv4 that allows mobile users to roam seamlessly between access networks that using different versions of the Internet Protocol, while maintaining a secured IPv4 connection to its home network. This work is implemented as a prototype in ipUnplugged's Roaming Gateway and Roaming Client products to demonstrate how the solution can be used. This solution has been named Mobile IPv4 over IPv6 solution that uses "Dual Stack Mobile IPv6" where the MN registers over IPv6 with the HA and has a new extension that mates it possible to transfer IPv4 traffic in the IPv6 tunnel as well as IPv6 traffic. This solution makes the assumption that the HA must be globally reachable through an IPv6 address and MN needs to know the IPv6 address of the HA. A HA on an IPv4 only network, which necessarily has no IPv6 address, is not supported. The operation of solution is that the MN is attached to an IPv6 access network. The MN can configure itself with an IPv6 address using stateless or stateful address autoconfiguration, or possibly DHCPv6. When the MN notices that it is attached to an IPv4 network, it sends a Mobile IPv4 Registration in an IPv6 UDP datagram to the IPv6 address of HA. The HA sets up IPv4 in IPv6 tunneling towards the Mobile Node and sends a Registration Reply back to the MN. For this solution, a new Mobile IPv4 extension needs to be defined that can store the IPv6 CoA of MN as the original CoA field in the Registration Request is too small. This extension is attached to Registration Request that is sent to the HA. However this solution has the problem that is MIPv4 over IPv6 handles the situation when the home network of a mobile user is running IPv4 and the mobile user wants to run IPv4 applications, while being away from the home network. If the home network is running IPv6, Mobile IPv6 can be used to allow the mobile user to run IPv6 applications, but Mobile IPv6 can only be used when attached to an IPv6 network. If the user is attached to an IPv4 access network, Mobile IPv6 services won't be available.

4.3 The designed registration packet format

In the previous section we explained the choice of UDP as the tunnel encapsulation method. We assume that whatever protocol translation mechanism is in use will correctly translate and forward arbitrary UDP packets.

We propose to use the UDP protocol for the designed packet in only two special scenarios. The first is where the MN registered with an IPv6 Home Agent and moved to an IPv4 foreign network, and the second where the MN registered with an IPv4 Home Agent and moved to an IPv6 foreign network.

For the general scenarios where the MN registered with IPv6 Home Agent, including a dual stack Home Agent, and moved to an IPv6 foreign network or where the MN registered with IPv4 Home Agent, including a dual stack Home Agent, and moved to IPv4 foreign network, the original registration messages, from MIPv6 or MIPv4 respectively, can be used unchanged.

We propose encrypting the payload of the UDP packets to provide assurance of identity of the sender, and protection from snooping, as is done with traditional MIP. For our purposes there is no particular need to use the Encapsulating Security Payload (ESP) [31] protocol as the encapsulation method, but to avoid needlessly altering more of the implementation that is required, even at the cost of some additional overhead, our current proposal assumes ESP inside the UDP packet. Since ESP also protects packet headers it assumes exist, we need to also provide some headers for it to protect, which are otherwise useless for our protocol.

Finally, we note again that the first step when a MN moves to any other network is movement detection. This is however not a trivial problem, however it is a problem that must be solved for any Mobile IP solution to function. Because of that, we simply assume that movement detection exists, and ignore this part of the overall problem.

4.4 Conclusion

This chapter has provided the motivation and an outline of the solution we are proposing, along with the assumptions we have made to allow this solution to be adopted. The following chapter will give detailed information about the design.

CHAPTER 5

DESIGN

This chapter gives the detailed design of the solution proposed in the previous chapter. Section 5.1 summarizes requirements on the design of the protocol, and gives details of its operation. Packet formats to handle the cases of a Mobile Node moving from IPv6 to IPv4, and IPv4 to IPv6 are given in sections 5.2 and 5.3 respectively.

5.1 Requirements and Operation of the Protocol Designed

We propose a method to permit mobility support based on Mobile IP that allows migration between IPv4 and IPv6 networks. We assume the existence of a packet translation service, for example NAT-PT, between IPv4 only and IPv6 only sections of the network, and that there is a DNS ALG type service to ensure that appropriate address information is available to all clients in their required protocol version, regardless of the actual address used by the target of the query.

We demand that the solution require no modifications to any CN, nor to the network infrastructure, including the protocol translator, other than the MN itself, and its HA. The solution must allow the MN to roam between IPv4 only and IPv6 network segments, and of course, dual stack segments of the network, and must consider the security of the overall system.

5.1.1 Simple mobility network topology

To allow an explanation of the protocol and its operation, we utilize the simple network topology shown in Figure 5.1.

The HA and CN exist in the IPv4 network and the foreign HA is located in the IPv6 network. Both HA have registered domain names with DNS-Server. When the MN is located in the IPv4 network it has only an IPv4 address or A type Resource Record in the DNS, and when located in the IPv6 network it has only an IPv6 address or AAAA type DNS Resource Record. For this work, we assume that the MN supports both IPv4 and IPv6 mobility and moves between the IPv4 and IPv6 networks. The protocol translation here is shown as NAT-PT. This is a router between the IPv4 and IPv6 networks and contains the DNS-ALG and connected dual-stack DNS server. When the MN has moved to a different network, the registration procedures of both MIPv4 and MIPv6 can be performed by using the services provided by the NAT-PT, or other translation service, and the DNS-ALG procedure.

This section describes the operations of the proposed mobility method, which is comprised of three steps, which are the locating of mobility agents, registration with the HA, and the data delivery mechanism between MIPv4 and v4CN, or between MIPv6 and v6CN when the MN moves to the other network. The operation procedure of each step will be explained according to the IP version of network and the position of the MN, HA and CN.

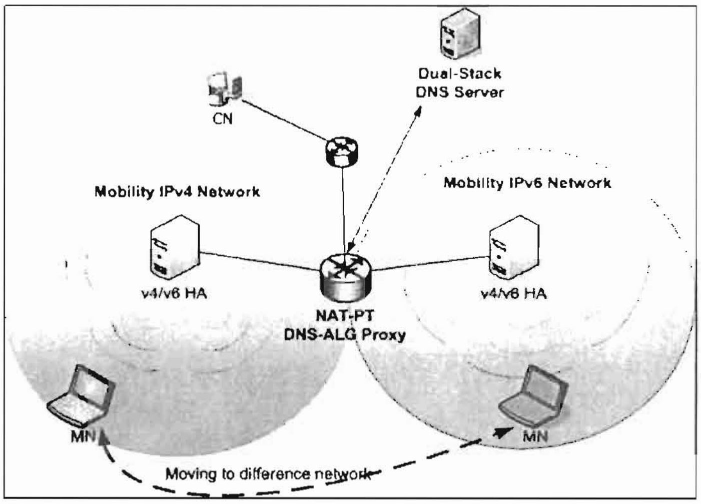


Figure 5.1 Simple mobile network topology

5.1.2 Locating of Mobility Agents

When the MN moves into a new IPv4 or IPv6 network, it obtains a new CoA. The version of the assigned CoA depends on the current network, and so this CoA can be used as a source IP address for the MN in the foreign network. The MN procedure to obtain the CoA in IPv4 and IPv6 networks differs as follows: the MN moves into an IPv6 network, it uses Router Solicitation to request a Router Advertisement which allows it to auto-configure an IPv6 address as its CoA. Otherwise the MN moved into an IPv4 network, MN uses Agent

Solicitation to request Agent Advertisement from FA by sending a multicast or broadcast request to acquire the CoAv4.

5.1.3 Registration to the HA

When the MN has moved from IPv4 to IPv6 and obtained its CoAv6, it then starts to register the CoAv6 with its HA on the IPv4 network. The registration procedure from the MN in the IPv6 network to its HA of IPv4 network can be explained as follows. The signaling for this procedure is shown in Figure 5.2.

1) The MN requests the NAT-PT and DNS-ALG to get an IPv6 address for its HA by using the domain name of the HA.
2) The NAT-PT looks for the IP address of the HA (HAv4) using the DNS-ALG and assigns an IPv6 address for the HA from its address pool. This v6HA address is coupled with v4HA address and this relationship is recorded in the mapping table. The v6HA address is returned to the MN.
3) The MN sends the designed registration request message to the NAT-PT through the IPv6 network. The IPv6 header is the first and MIPv6 header information are included inside UDP datagram.
4) When the NAT-PT receives the registration request message from the MN, the IPv6 header is translated to IPv4 and the message forwarded to the HA. After the HA receives the registration request message, also checks the value of CoAv4. If CoAv4 is 0.0.0.0, HA know that the MN moved to different IP version network. HA will use a source address of registration message to update CoA in binding entry.
5) Then the HA replies the registration reply message to the MN.
6) When the NAT-PT sees the registration reply message, it checks the source and destination IP addresses, translates the header, and sends the message to the MN in the IPv6 network.

In the other case, the v6MN moves to the IPv4 network, the registration procedure from the MN located in the IPv4 network to its HA in the IPv6 network is the same as when a v4MN moves to v6MN, and can be shown in Figure 5.3. But the version of IP address and the mobile IP message format are different.

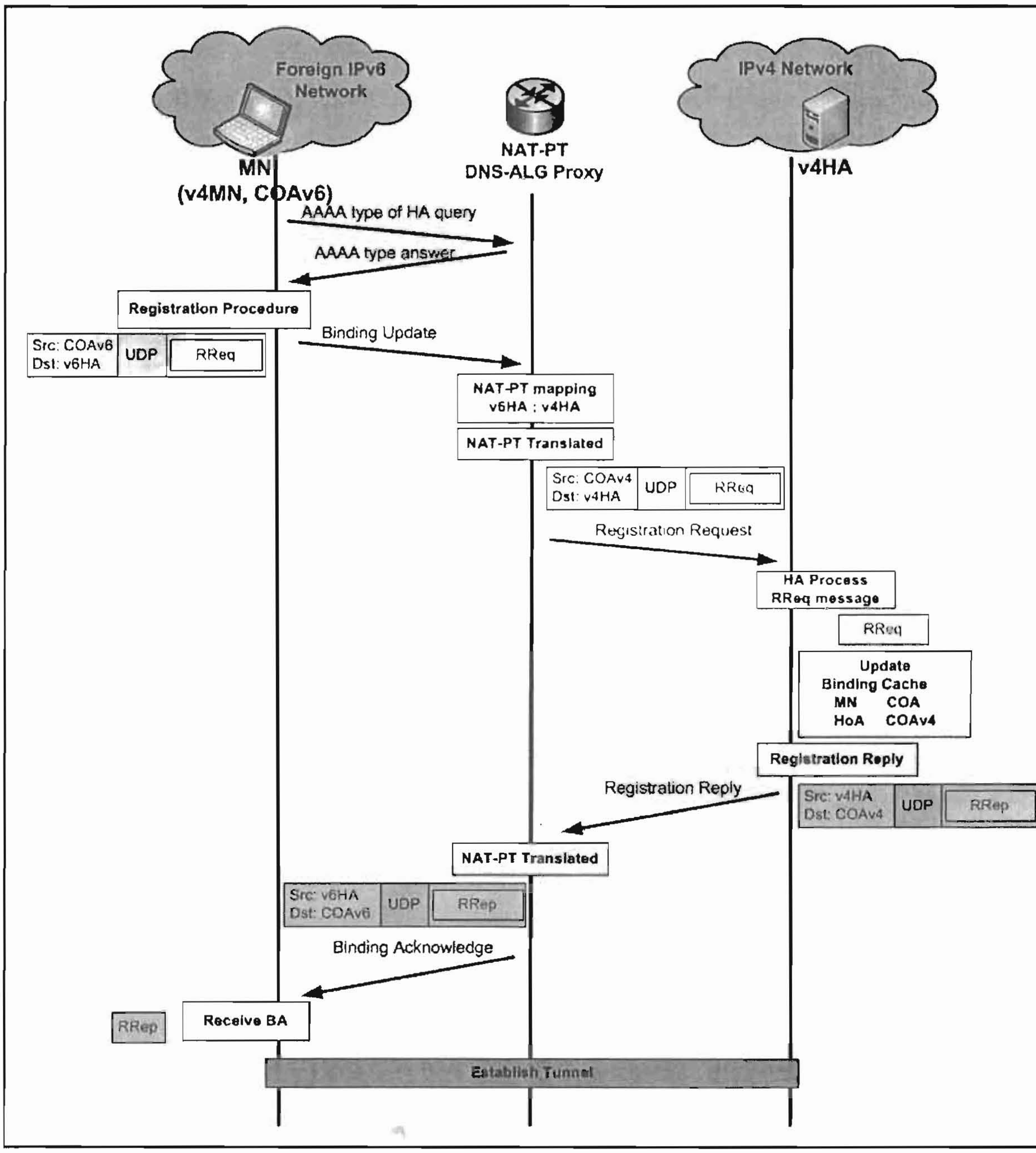


Figure 5.2 The registration procedure of the MN in the IPv4 network to its HA in an IPv6 network.

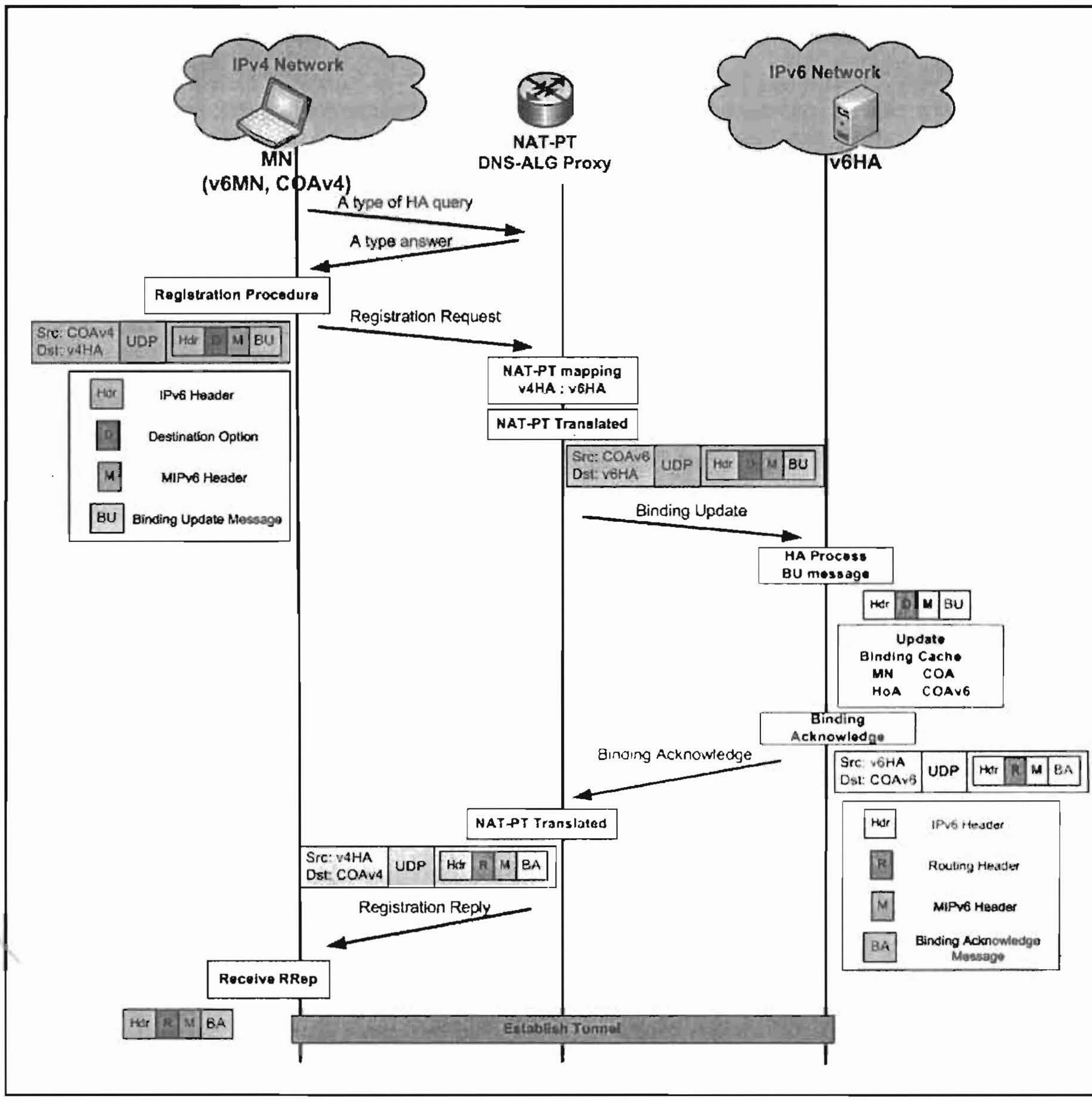


Figure 5.3 The registration procedure of the MN in the IPv6 network to its HA in an IPv4 network.

5.1.4 The way to acquire CoA of MN

The previous section explains the registration procedure that occurred after the MN move to the different network by the operation of the NAT-PT with DNS-ALG.

Before the HA update the binding cache with the CoAv4 or CoAv6 from the source address. The HA will check CoA in the binding message that can separated the operation to four cases. The first case and second case are MN moved to other network which is same IP version. These will update the binding cache by normal procedure. The third case is v4MN moved to the IPv6 Network, the value of CoA in the registration message is fixed to "0.0.0.0" that indicate HA to know that v4MN moved to the IPv6 network and a truth CoA address should get from a source address of the registration message. The last case v6MN move to the IPv4 network, the binding update message must be sent through UDP datagram. Also the value of CoA is fixed to "::"which is unspecified IPv6 address. HA will use a source address of the binding update message which new IP address of MN or CoA to update binding cache. The operation of third case and fourth case can be shown in the following flowchart in Figure 5.4 and Figure 5.5.

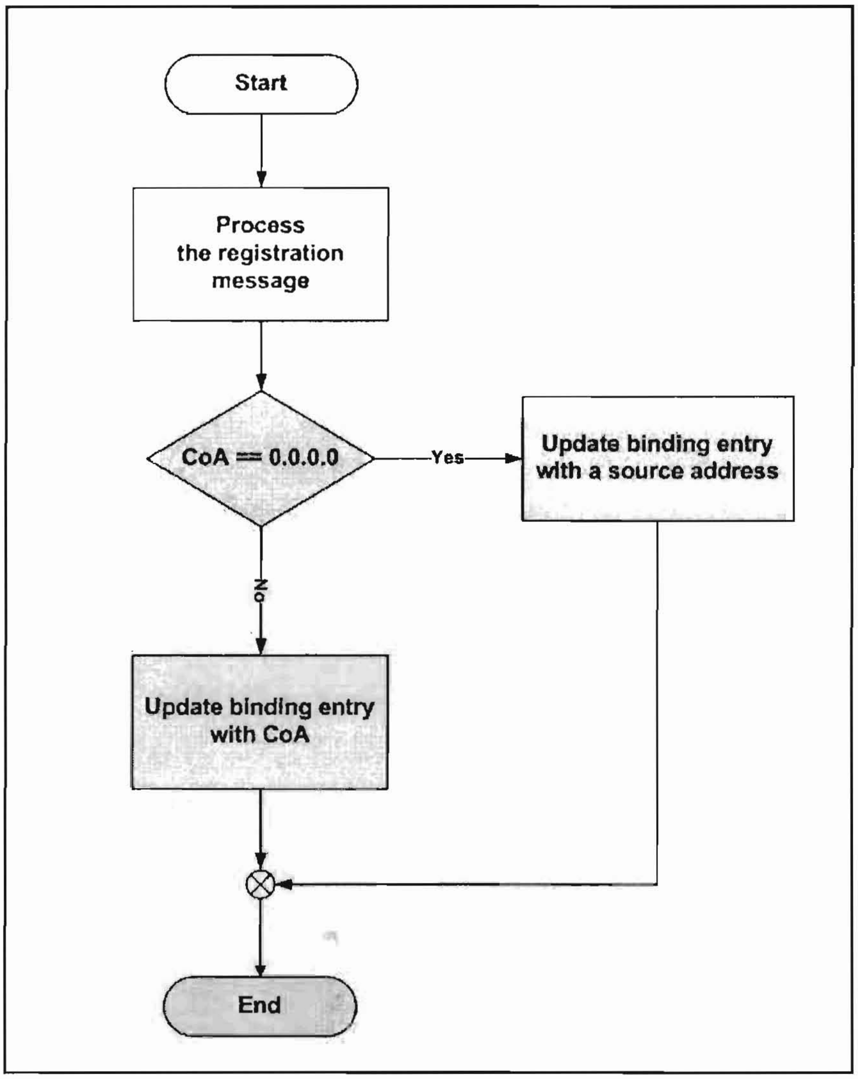

Start
Process the registration message
CoA == 0.0.0.0
Yes
Update binding entry with a source address
No
Update binding entry with CoA
End

Figure 5.4 Update Binding entry operation of HA when v4MN move to the IPv6 network.

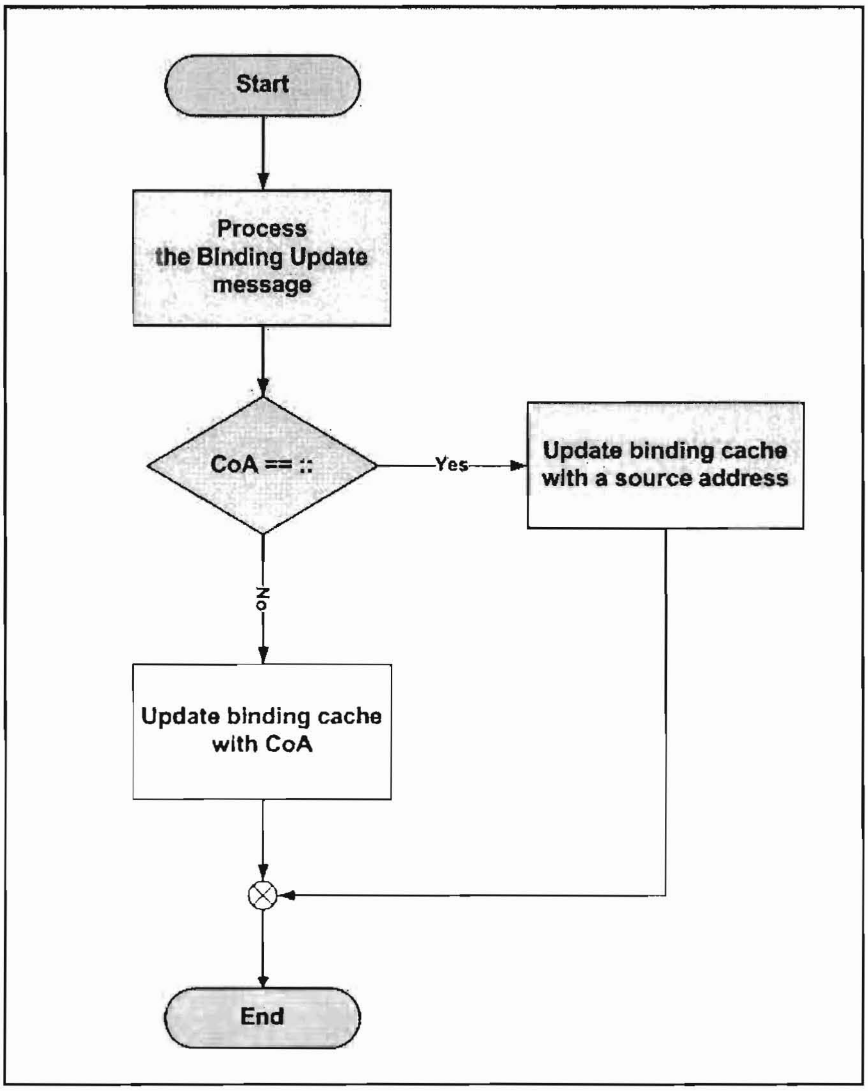

Start
Process the Binding Update message
CoA == ::
Yes
Update binding cache with a source address
No
Update binding cache with CoA
End

Figure 5.5 Update Binding Cache operation of HA when v6MN move to the IPv4 network

5.1.5 Data Delivery Mechanism

After the MN completes the registration procedure to its HA, the MN can receive the packets from the CN, even if the MN is located in a different network because the HA intercepts the packets and tunnels the packets toward the MN in the foreign network. Therefore, the mobility support is possible in IPv4 and IPv6 networks. For example, when the HA is located in the IPv4 network and the MN is moved from IPv4 to IPv6 network, the desire is that communications with the CN can continue. The operations can be explained as follows.

1) In case where the CN sends the first packet to the MN, the HA intercepts and encapsulates it by using the registered COAv4. Then the HA delivers the packet to the MN through the NAT-PT for header translation.
2) NAT-PT determines the IPv6 address for the HA and the COAv4 is mapped to the COAv6 using the mapping table. Then this packet is sent to the MN in the IPv6 network.
3) When the MN receives the packet it decapsulates and processes this packet. When the MN wants to send a response packet, it encapsulates this packet and sends to its HA through the NAT-PT.
4) The HA receives and decapsulates the response packet, then it sends it to the CN.
5) If the CN doesn't support Route Optimization, all next packets are sent to the MN must route to the HA for tunneling to the MN. We believe that with NAT-PT support. Route Optimization will be possible, however the procedures for this one yet to be determined.

In the other case, the CN and MN are located in the IPv6 network and the MN moves to an IPv4 network. The data delivery operates similarly to the previous case.

5.2 The designed packets in case: The MN registered with IPv6 HA and moved to IPv4 foreign Network.

5.2.1 The designed Binding Update packet

The designed Binding Update packet can be illustrated in Figure 5.6 which contains IPv4 header which identifies the protocol field with 17 in decimal number or 0x11 in hexadecimal which mean UDP protocol. Inside UDP datagram contain the designed Binding Update message. The designed Binding Update message likes the general BU packet of MIPv6. But an alternate care-of address is specified to "Unspecified IPv6 address" or "::" or "0:0:0:0:0:0:0:0" and contain in a datagram of IPv4 UDP packet. The detail of the designed Binding Update packet can be explained as following:

IPv4 Header

- An IPv4 source address is an IPv4 Care-of Address.
- An IPv4 destination is an IPv4 of Home Agent which acquired from NAT-PT and DNS-ALG.
- Protocol Field is set to 17 in decimal or 0x11 in hexadecimal which mean UDP protocol.

UDP Datagram

IPv6 Header

- An IPv6 source address is an IPv6 home address of Mobile Node.
- An IPv6 destination address is an IPv6 address of its Home Agent.
- The first next header is IPv6 destination option (0x3c).

Destination Option

- The next header of destination option is Mobile IPv6 (0x87).
- Home address is an IPv6 home address of Mobile Node.

Mobile IPv6 /Network Mobility

- Payload protocol is IPv6 no next header (0x3b).
- Mobility header type is Binding Update (5).

Binding Update

- Sequence number field is a 16 bits unsigned integer which used by the receiving node to sequence BU and by the sending node to match a returned BA with this BU.
- Lifetime is the number of time units which remain before the binding must be considered expired. One time unit is 4 seconds.

Mobility Options

- PadN option is used to insert two or more octets of padding in mobility option area of mobility message.
- Alternate Care-of Address is UNSPECIFIED IPv6 Address or "::" or "0:0:0:0:0:0:0:0".

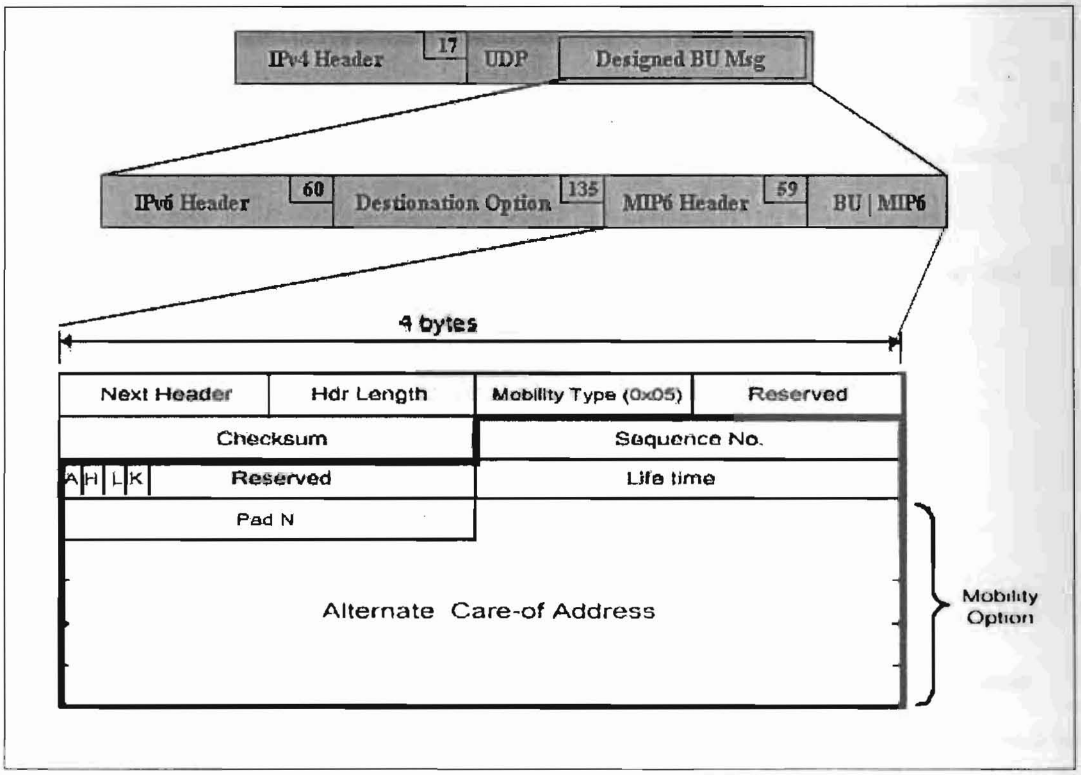


Figure 5.6 The designed BU packet for MN moves from v6 to v4 network.

5.2.2 The designed Binding Acknowledgement packet

The designed Binding Acknowledge packet can be illustrated in Figure 5.7 which contains IPv6 header which identifies the next header field with 17 in decimal number or 0x11 in hexadecimal which mean UDP protocol. Inside UDP datagram contain the designed Binding Acknowledge message. The designed Binding Acknowledge message likes the general BA packet of MIPv6. But contain in a datagram of IPv4 UDP packet. The detail of the designed Binding Acknowledgement packet can be explained as following:

IPv6 Header

- An IPv6 source address is an IPv6 address of Home Agent.
- An IPv6 destination is an IPv6 Care-of Address which got from source address of the received BU packet.
- Next header is set to 17 in decimal or 0x11 in hexadecimal which mean UDP protocol.

UDP Datagram

IPv6 Header

- An IPv6 source address is an IPv6 address of Home Agent.
- An IPv6 destination address is an IPv6 home address of Mobile Node.
- The next header is set to IPv6 routing type 2 or value 0x2b in hexadecimal.

Routing Header

- The next header is Mobile IPv6 equals 135 in decimal or 0x87 in hexadecimal.
- Home Address field contains an IPv6 home address of MN.

Mobile IPv6 /Network Mobility

- Payload protocol is IPv6 no next header (0x3b).
- Mobility header type is Binding Acknowledgement (6).

Binding Acknowledgement

- Status field is set to zero which mean Binding Update is accepted.
- Sequence number field is a 16 bits unsigned integer which used by the receiving node to sequence BU and by the sending node to match a returned BA with this BU.
- Lifetime is the number of time units which remain before the binding must be considered expired. One time unit is 4 seconds.

Mobility Options

- PadN option is used to insert two or more octets of padding in mobility option area of mobility message.

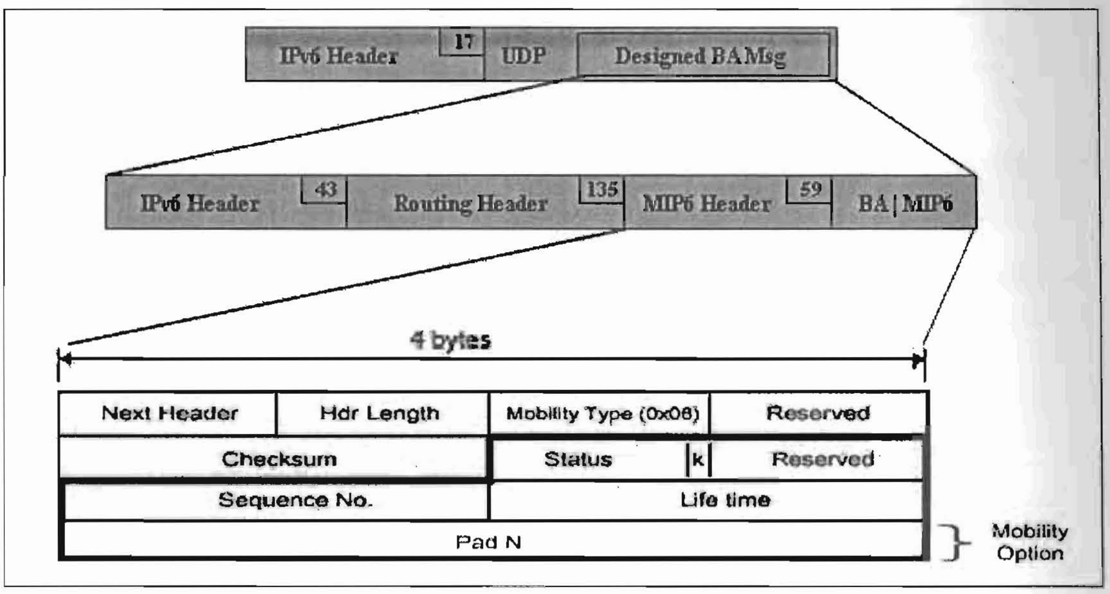


Figure 5.7 The designed BA packet for MN moves from v6 to v4 network

5.3 The designed packets in case: The MN registered with IPv4 Home Agent and moved to IPv6 mobility Network.

5.3.1 The designed Registration Request packet

The designed Registration Request packet can be illustrated in Figure 5.8 which contains IPv6 header which identifies the next header field with 17 in decimal number or 0x11 in hexadecimal which mean UDP protocol. Inside UDP datagram contain the designed Registration Request message. This RReq message likes the general Registration Request packet of MIPv4 and also contain in a datagram of IPv4 UDP packet. But a Care-of Address is specified to "Unspecified IPv4 address" or "0.0.0.0". The detail of the designed Registration Request packet can be explained as following:

IPv6 Header

- An IPv6 source address is an IPv6 Care-of Address.
- An IPv6 destination is an IPv6 of Home Agent which assigned from NAT-PT pool address.
- Next header Field is set to 17 in decimal or 0x11 in hexadecimal which mean UDP protocol.

UDP Datagram

Mobile IP

- Message type is set to the Registration Request or 1 in decimal.

Flags

  - Flags are set to 0x20 in hexadecimal to enable D bit for using Co-located Care-of Address.

- Lifetime field is preferred lifetime of binding which has two special values are 0x0000 and 0xffff used for deregistration and infinite lifetime in order.
- Home Address is an IPv4 home address of Mobile Node.
- Home Agent is an IPv4 address of MN's Home Agent.
- Care-of Address is specified to UNSPECIFIED IPv4 Address or 0.0.0.0.
- Identification field is a 64 bit number which constructed by MN. This used for matching Registration Request with Registration Reply and used for protecting against replay attacks of registration message.
- Extension Field is used for the most common of them being the authentication extension. This portion can be fixed one or more extension.

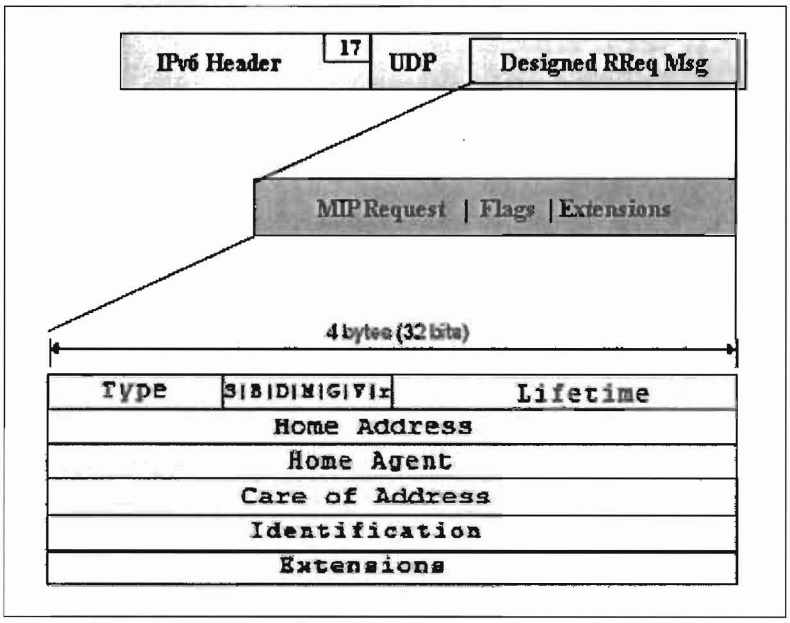


Figure 5.8 The designed RReq packet for MN moves from v4 to v6 network

5.3.2 The designed Registration Reply packet

The designed Registration Reply packet can be illustrated in Figure 5.9 which contains IPv4 header which identifies the protocol field with 17 in decimal number or 0x11 in hexadecimal which mean UDP protocol. Inside UDP datagram contain the designed Registration Reply message. This RRep message likes the general Registration Request packet of MIPv4 and also contain in a datagram of IPv4 UDP packet. The detail of the designed Registration Reply packet can be explained as following:

IPv4 Header

- An IPv4 source address is an IPv4 address of Home Agent.
- An IPv4 destination is an IPv4 address which got from source address of RReq packet.
- Protocol field is set to 17 in decimal or 0x11 in hexadecimal which mean UDP protocol.

UDP Datagram

Mobile IP

- Message type is set to the Registration Reply or 3 in decimal.

Flags/Code Field

  - Code field is set to 0 in decimal to inform MN that the registration message is accepted.

- Lifetime field is preferred lifetime of binding which has two special values are 0x0000 and 0xffff used for deregistration and infinite lifetime in order.
- Home Address is an IPv4 home address of Mobile Node.
- Home Agent is an IPv4 address of MN's Home Agent.
- Identification field is a 64 bit number used for matching Registration Request with Registration Reply and used for protecting against replay attacks of registration message. The value is based on the identification field from the Registration Request message from MN.
- Extension Field is used for the most common of them being the authentication extension. This portion can be fixed one or more extension.

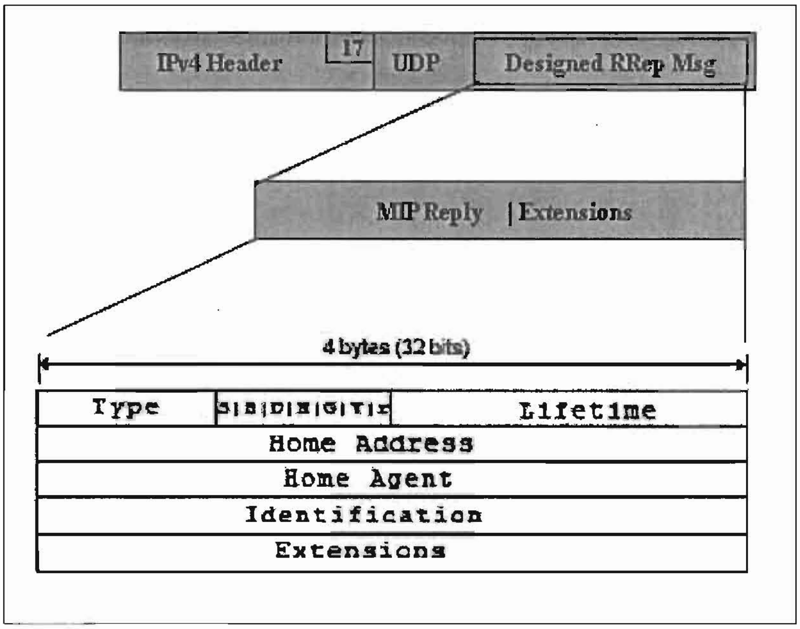


Figure 5.9 The designed RRep packet for MN moves from v4 to v6 network

CHAPTER 6

IMPLEMENTATION AND TESTING

This chapter describes the implementation of mobility mechanism between MIPv4 and MIPv6 that keep on the registration CoA procedure. The key implementation of this work only implement on the HA and MN, but not modify something on NAT-PT and DNS-ALG.

An overview of implementation is presented in section 6.1. Section 6.2 describes the A mechanism mobility between MIPv4 and MIPv6 implementation. The topology for testing and the result are described in subsection of Section 6.3.

6.1 Overview of Implementation

This work, we proposed a mobility mechanism between MIPv4 and MIPv6 that concentrated on the registration CoA procedure. Both MIPv4 and MIPv6 must send a new CoA to their HA when it moved to the foreign network. For implementation of this mechanism, we look for the software architecture of Mobile IPv4 and Mobile IPv6 to modify this solution. The MIPv4 software was developed by the Portland State University. It supports FreeBSD and Linux Operating System [29]. There are several MIPv6 softwares that support on the many Operation Systems such as Windows, Linux, NetBSD, and FreeBSD [30]. However for this work, we prefer to implement on UNIX Operating System that the functions required by the implementation are modifications to the existing SHISA mobility stack [36] on FreeBSD 5.4 [37].

SHISA is implemented on top of the KAME IPv6 stack [38]. Our solution needs to implement the SHISA to provide the MIPv6 for mobility networks. The SHISA implementation provides the Mobile IPv6 functions for MN, HA and CN.

SHISA consists of several user space programs and a modified kernel. Table 6.1 shows the programs of the SHISA stack [35]. In the MIPv6 protocol, the mnd program manages the MN functions. The binding information on the node that are stored in the binding update database via mnd program. The had program provides the HA functions and manages the signaling processing on the HA side.

Table 6.1 SHISA programs

Program	Function
mnd	The mobile host functions
had	The HA functions (for MIPv6 and NEMO BS)
cnd	The CN functions and to provide route optimization of CN
babymdd	A simple movement detector of MN and MR

6.2 A mechanism between MIPv4 and MIPv6 Implementation

The goal of this work is propose a solution which still connectivity when MN move to the other network. We focused about the registration procedure.

When MN moved to the other network, MN acquires the new address from the foreign network and aware that it already moved from home network. This state is movement detection. But in the original of SHISA program, babymdd daemon cannot detect when MN moved to the different IP version network. To do the registration procedure, we modify and use the easy movement detection program run on MN when MN moved to the different IP version network. This program will send the registration information to "mnd" daemon for sending the registration message to HA. This interrupt or shortcut by call baby_md_reg() function and send Mobility message via Mobility Socket that illustrated in Figure 6.1.

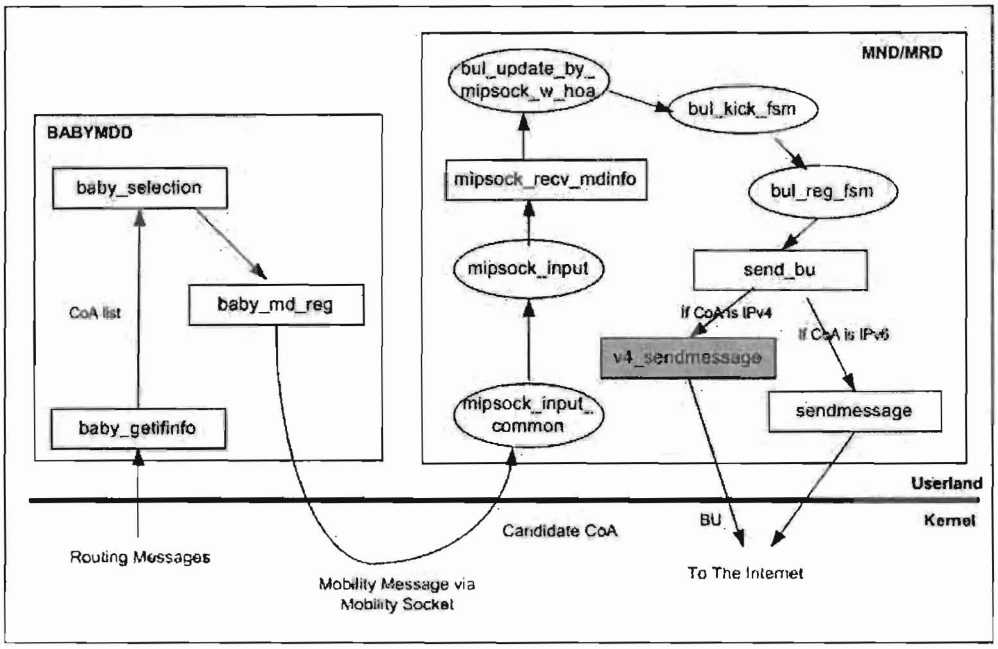


Figure 6.1 Detecting IPv4 care-of address and Sending a binding update message

After MN aware that it moved to the other network. MN must to send the registration message in our format that depend on scenario of moving. The designed registration packet can be generated in the v4_sendmessage() function and wrapper with UDP before send to HA through Internet. The process of detecting IPv4 Care-of Address and sending a designed BU message can be illustrated in Figure 6.1.

In Figure 6.2 illustrated the process of HA, receiving a designed BU message, establish tunnel and sending a BA back to MN. The HA can receive the designed BU message from udp_input_common() function. After check the BU packet and update binding cache. HA will send binding acknowledgement back to MN. This BA message is wrapper with UDP packet.

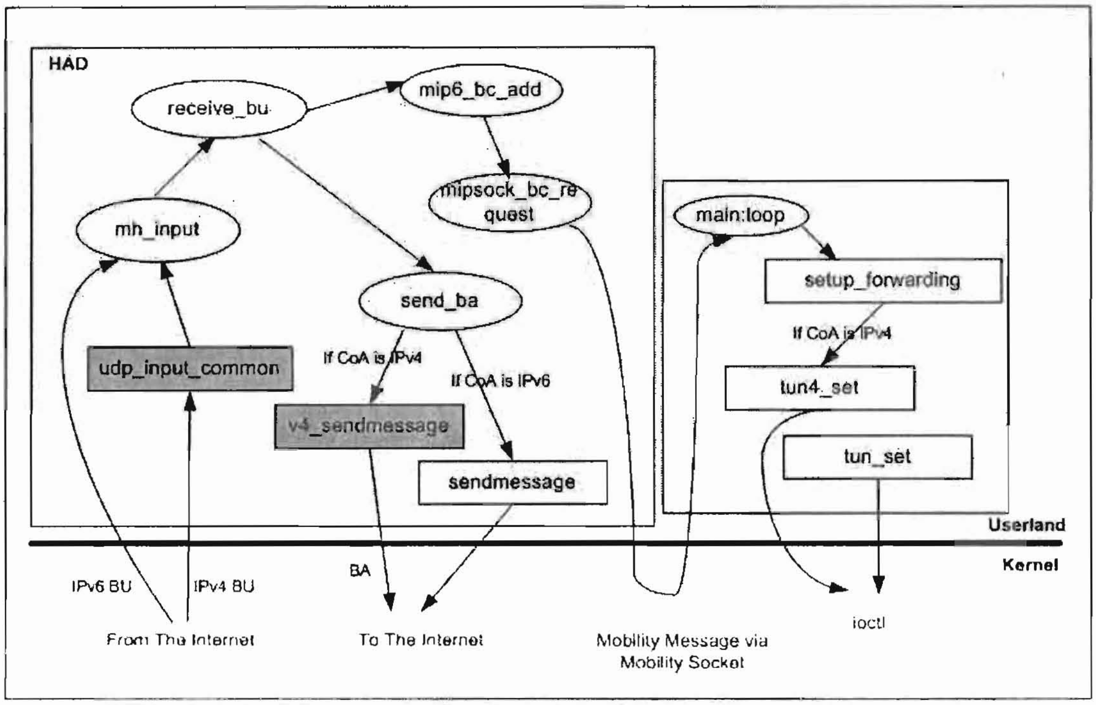


Figure 6.2 Receiving a BU message, Establishing a bi-directional tunnel, and Sending a BA message

Then MN receive BA from it HA through udp_input_common() function. Also the MN process BA and establish IPv4 tunnel to its HA. The process of MN while MN is receiving a BA and establishing tunnel can be illustrated in Figure 6.3.

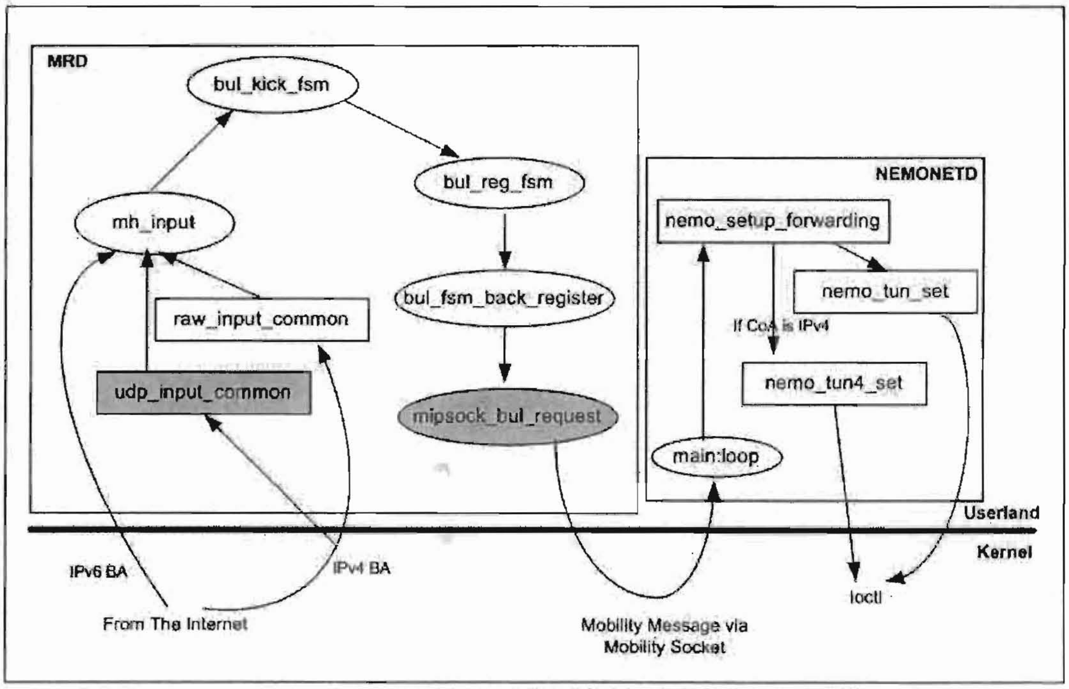


Figure 6.3 Receiving a BA and Establishing a Bi-directional Tunnel

6.3 Testing

The solution is designed for still communication when MN moved to the different IP version network. To ensure the validity of the designed solution, the implementation is used to verify this designed solution. The results are captured by Wireshark program [40].

6.3.1 The experimental network topology

The experimental network topology is used for testing mobility between MIPv4 and MIPv6 as shown in the Figure 6.4. It is separated 4 parts that following.

1. Home Agent of IPv4 mobility network
2. Home Agent of IPv6 mobility network
3. Mobile node
4. NAT-PT and DNS-ALG

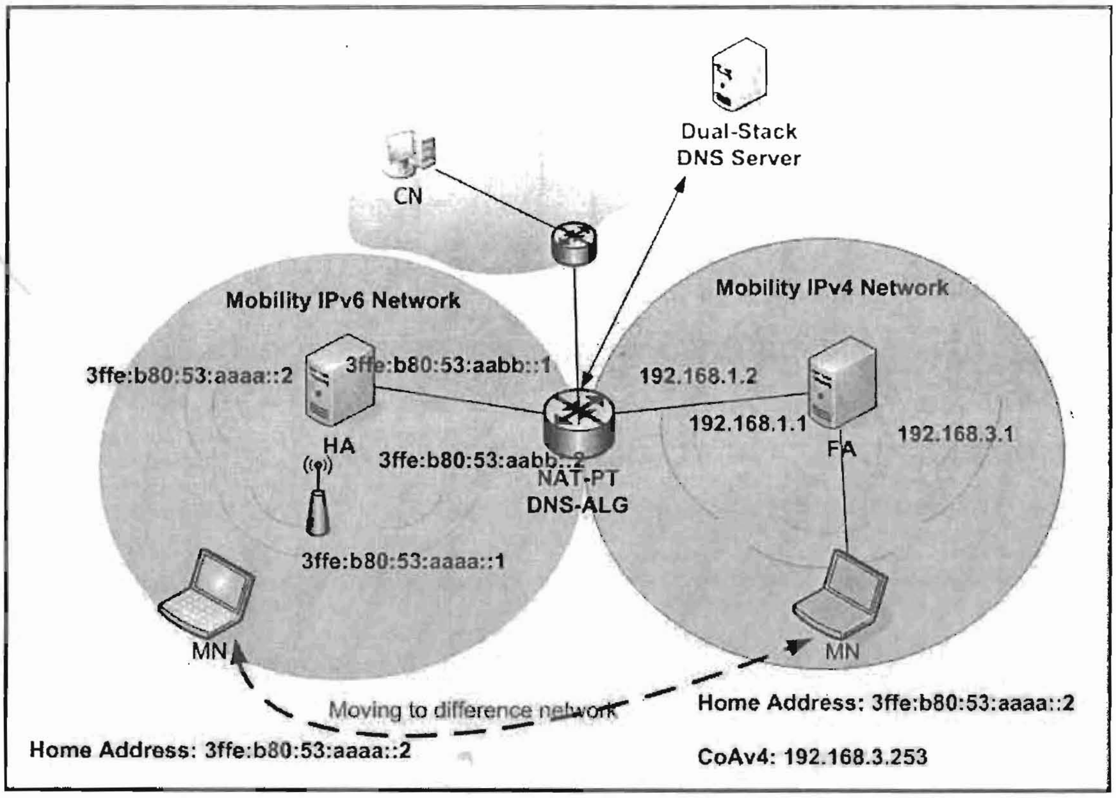


Figure 6.4 The experimental network topology

The Home Agent of both IPv4 and IPv6 mobility network is installed with FreeBSD 5.4 and recompiled with kame patch. It uses the home agent daemon or "had" that handles the request of mobile nodes registrations and configures IP stack to intercept and tunnel the packets destined to the mobile nodes.

The Mobile Node is installed with FreeBSD 5.4 and recompiled with kame patch. It uses the mobile node daemon or "mnd" that implements the state machine of MIPv6 for mobile node to request the registration with home agent while away from home.

6.3.2 Testbed components

The Test Bed components can be illustrated in Table 6.2.

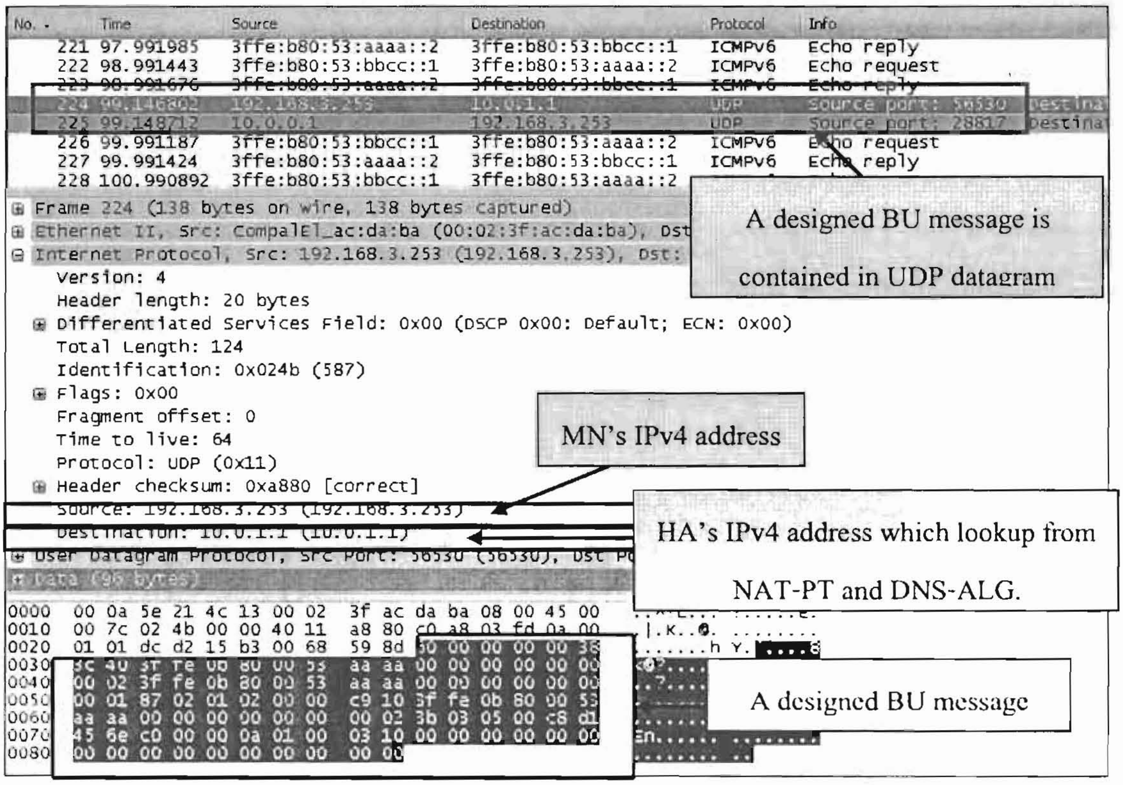
Table 6.2 Test Bed Components

Node Type	MIPv4/MIPv6
1. Mobile Node (MN)	➢ FreeBSD 5.4 ➢ Mobile IPv6 Patch (kame-20070601-freebsd54-snap)
2. Home Agent (HA	➢ FreeBSD 5.4 ➢ Mobile IPv6 Patch (kame-20070601-freebsd54-snap) ➢ For MIPv6 supports router advertisement daemon (rtadvd) ➢ For MIPv4 supports DHCPv4
3. Border Gateway	➢ FreeBSD 4.9 ➢ Supports NAT-PT and DNS-ALG (kame-20040726-freebsd49-snap)

6.4 Scenarios for testing

6.4.1 The MN registered with IPv6 HA and moved to IPv4 foreign Network.

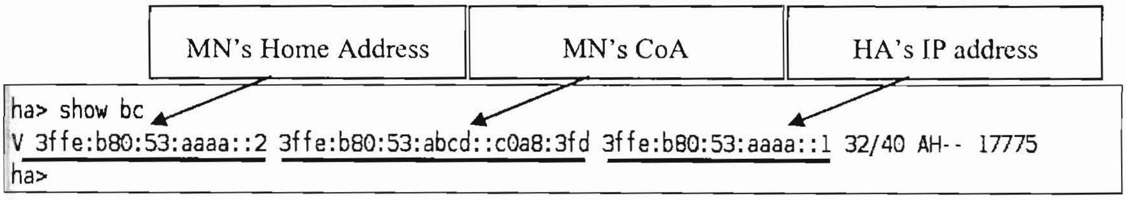
When MN moved from IPv6 to IPv4, MN send the designed BU message to its HA through UDP that shown the captured packet in Figure 6.5. The binding cache on HA can be updated that illustrated in Figure 6.6 . As HA establish tunnel to MN and send the designed BA back to MN that can be illustrated in Figure 6.7.



```
No. -   Time        Source                  Destination             Protocol  Info
    221 97.991985   3ffe:b80:53:aaaa::2     3ffe:b80:53:bbcc::1     ICMPv6    Echo reply
    222 98.991443   3ffe:b80:53:bbcc::1     3ffe:b80:53:aaaa::2     ICMPv6    Echo request
    223 98.991676   3ffe:b80:53:aaaa::2     3ffe:b80:53:bbcc::1     ICMPv6    Echo reply
    224 99.146802   192.168.3.253           10.0.1.1                UDP       Source port: 56530  Destina
    225 99.148712   10.0.0.1                192.168.3.253           UDP       Source port: 28817  Destina
    226 99.991187   3ffe:b80:53:bbcc::1     3ffe:b80:53:aaaa::2     ICMPv6    Echo request
    227 99.991424   3ffe:b80:53:aaaa::2     3ffe:b80:53:bbcc::1     ICMPv6    Echo reply
    228 100.990892  3ffe:b80:53:bbcc::1     3ffe:b80:53:aaaa::2
⊞ Frame 224 (138 bytes on wire, 138 bytes captured)
⊞ Ethernet II, Src: CompalEl_ac:da:ba (00:02:3f:ac:da:ba), Dst
⊟ Internet Protocol, Src: 192.168.3.253 (192.168.3.253), Dst:
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 124
    Identification: 0x024b (587)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
  ⊞ Header checksum: 0xa880 [correct]
    Source: 192.168.3.253 (192.168.3.253)
    Destination: 10.0.1.1 (10.0.1.1)
⊞ User Datagram Protocol, Src Port: 56530 (56530), Dst Po
⊞ Data (96 bytes)

0000  00 0a 5e 21 4c 13 00 02  3f ac da ba 08 00 45 00   ..^!L... ......E.
0010  00 7c 02 4b 00 00 40 11  a8 80 c0 a8 03 fd 0a 00   .|.K..@. ........
0020  01 01 dc d2 15 b3 00 68  59 8d 30 00 00 00 00 3c   .......h Y.....<
0030  3c 40 3f fe 0b 80 00 53  aa aa 00 00 00 00 00 00   <@?....
0040  00 02 3f fe 0b 80 00 53  aa aa 00 00 00 00 00 00   ..?....
0050  00 01 87 02 01 02 00 00  c9 10 3f fe 0b 80 00 53   .......
0060  aa aa 00 00 00 00 00 00  00 02 3b 03 05 00 c8 d1   .......
0070  45 6e c0 00 00 0a 01 00  03 10 00 00 00 00 00 00   En..... .......
0080  00 00 00 00 00 00 00 00  00 00                     ........ ..
```

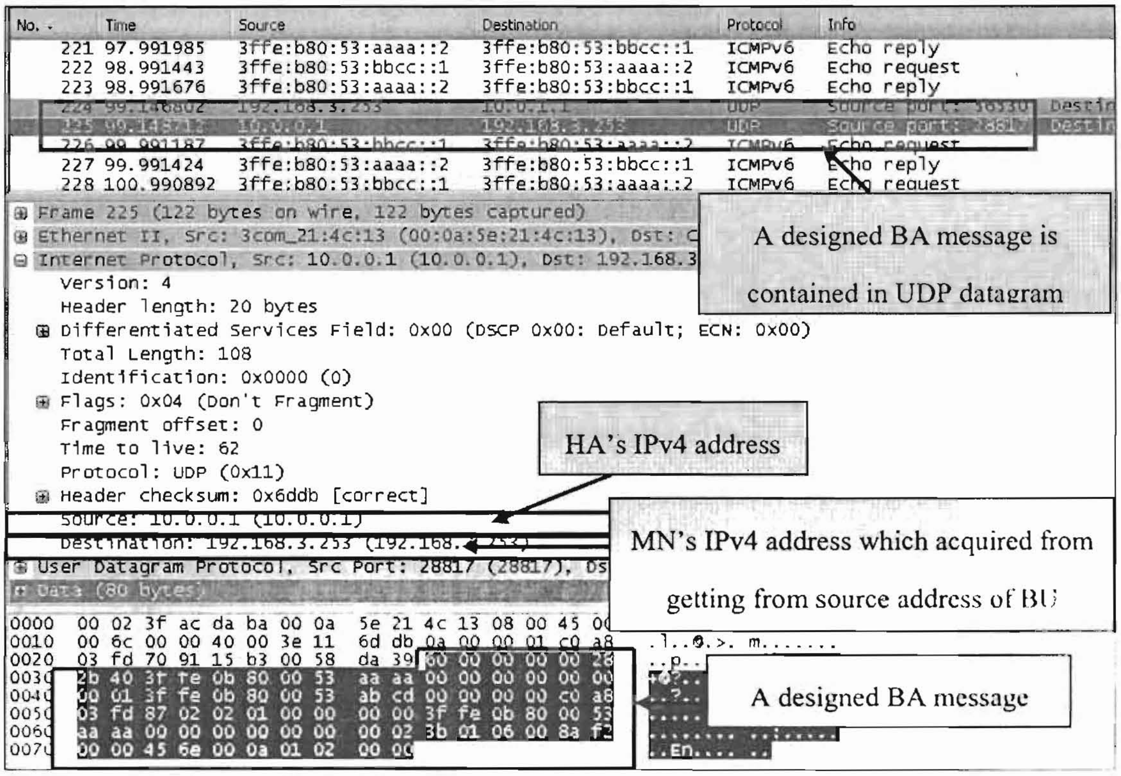
A designed BU message is contained in UDP datagram
MN's IPv4 address
HA's IPv4 address which lookup from NAT-PT and DNS-ALG.
A designed BU message

Figure 6.5 The captured Binding Update signaling



```
ha> show bc
V 3ffe:b80:53:aaaa::2 3ffe:b80:53:abcd::c0a8:3fd 3ffe:b80:53:aaaa::1 32/40 AH-- 17775
ha>
```

Figure 6.6 Binding Update database of HA.



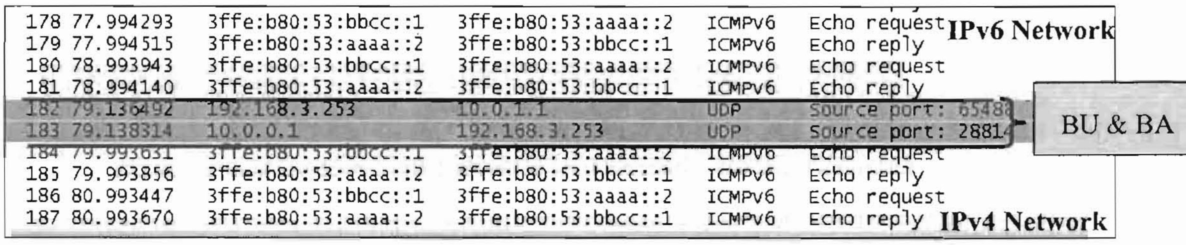
```
No. -  Time        Source                  Destination             Protocol  Info
   221 97.991985   3ffe:b80:53:aaaa::2     3ffe:b80:53:bbcc::1     ICMPv6    Echo reply
   222 98.991443   3ffe:b80:53:bbcc::1     3ffe:b80:53:aaaa::2     ICMPv6    Echo request
   223 98.991676   3ffe:b80:53:aaaa::2     3ffe:b80:53:bbcc::1     ICMPv6    Echo reply
   224 99.146802   192.168.3.253           10.0.1.1                UDP       Source port: 56530  Destin
   225 99.148712   10.0.0.1                192.168.3.253           UDP       Source port: 28817  Destin
   226 99.991187   3ffe:b80:53:bbcc::1     3ffe:b80:53:aaaa::2     ICMPv6    Echo request
   227 99.991424   3ffe:b80:53:aaaa::2     3ffe:b80:53:bbcc::1     ICMPv6    Echo reply
   228 100.990892  3ffe:b80:53:bbcc::1     3ffe:b80:53:aaaa::2     ICMPv6    Echo request
⊞ Frame 225 (122 bytes on wire, 122 bytes captured)
⊞ Ethernet II, Src: 3com_21:4c:13 (00:0a:5e:21:4c:13), Dst: C
⊟ Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 192.168.3
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 108
    Identification: 0x0000 (0)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 62
    Protocol: UDP (0x11)
  ⊞ Header checksum: 0x6ddb [correct]
    Source: 10.0.0.1 (10.0.0.1)
    Destination: 192.168.3.253 (192.168.3.253)
⊞ User Datagram Protocol, Src Port: 28817 (28817), Ds
⊞ Data (80 bytes)

0000  00 02 3f ac da ba 00 0a  5e 21 4c 13 08 00 45 00
0010  00 6c 00 00 40 00 3e 11  6d db 0a 00 00 01 c0 a8   .l..@.>. m.......
0020  03 fd 70 91 15 b3 00 58  da 39 60 00 00 00 00 28   ..p..
0030  2b 40 3f fe 0b 80 00 53  aa aa 00 00 00 00 00 00   +@?..
0040  00 01 3f fe 0b 80 00 53  ab cd 00 00 00 00 c0 a8   ..?..
0050  03 fd 87 02 02 01 00 00  00 00 3f fe 0b 80 00 53   .....
0060  aa aa 00 00 00 00 00 00  00 02 3b 01 06 00 8a f2   ........ ..;.....
0070  00 00 45 6e 00 0a 01 02  00 00                     ..En.... ..
```

A designed BA message is contained in UDP datagram
HA's IPv4 address
MN's IPv4 address which acquired from getting from source address of BU
A designed BA message

Figure 6.7 The captured Binding Acknowledge signaling

After HA send the BA back to MN in the designed format. The HA establish tunnel to MN likewise the MN will establish tunnel to HA after received BA. Then CN connect to MN, the data packet routed to the MN's home network. The HA will intercept the data packets and route them to MN through NAT-PT.

This we try to test by a "ping" application which is send a small packet from CN to MN that illustrated in Figure 6.8. The connection between CN and MN can be stilled when MN move from the IPv6 home network to the IPv4 foreign network and go back to the home network.



```
178 77.994293  3ffe:b80:53:bbcc::1  3ffe:b80:53:aaaa::2  ICMPv6  Echo request
179 77.994515  3ffe:b80:53:aaaa::2  3ffe:b80:53:bbcc::1  ICMPv6  Echo reply
180 78.993943  3ffe:b80:53:bbcc::1  3ffe:b80:53:aaaa::2  ICMPv6  Echo request
181 78.994140  3ffe:b80:53:aaaa::2  3ffe:b80:53:bbcc::1  ICMPv6  Echo reply
182 79.136492  192.168.3.253        10.0.1.1             UDP     Source port: 65488
183 79.138314  10.0.0.1             192.168.3.253        UDP     Source port: 28817
184 79.993631  3ffe:b80:53:bbcc::1  3ffe:b80:53:aaaa::2  ICMPv6  Echo request
185 79.993856  3ffe:b80:53:aaaa::2  3ffe:b80:53:bbcc::1  ICMPv6  Echo reply
186 80.993447  3ffe:b80:53:bbcc::1  3ffe:b80:53:aaaa::2  ICMPv6  Echo request
187 80.993670  3ffe:b80:53:aaaa::2  3ffe:b80:53:bbcc::1  ICMPv6  Echo reply
```

Figure 6.8 The result of ping from CN to MN.

CHAPTER 7

DISCUSSION AND CONCLUSION

In this chapter, we conclude our work to propose the solution to enable connectivity when a MN moves to a different network. We present the goals and the requirements of the solution and summarize the advantages and the limitations. Additionally, this chapter includes a discussion of the work and suggests some future work.

7.1 Conclusion

Mobile nodes are becoming more common in the Internet. They must use the mobile IP protocol for mobility support. Mobile IPv4 was designed for Internet Protocol version (IPv4) and mobile IPv6 has been designed for IPv6. Mobile IPv6 shares many ideas with mobile IPv4, and inherits some new features from IPv6. In the period of transition from IPv4 to IPv6, it's possible that the Mobile Node (MN) might move from a network supporting only IPv4 to a network supporting only IPv6, or vice-versa. However mobile IPv6 is not backward compatible with Mobile IPv4. Solutions to the mobility issues in IPv4/IPv6interconnected networks are rare in the present RFCs and Internet Drafts.

Our solution is a method for mobility support in both IPv4 and IPv6 mobile networks when they are communicating to allow handoff to the other network by using the interoperation of an existing translation mechanism which we assume to exist, one is Network Address Translation-Protocol Translation (NAT-PT), and interoperating with a Domain Name System Application Layer Gateway (DNS-ALG). NAT-PT is powerful, flexible and needs very little end user configuration. NAT-PT is a transition mechanism located at the boundary between an IPv6 and an IPv4 network. It translates IPv6 packets into IPv4 packets and vice-versa.

7.2 Discussion

The result of this work produced several benefits. However there are also many limitations which require improvements. This part discusses the procedure and result of this work.

7.2.1 Advantages

1. It is useful to discover the available IPv4/IPv6 transition strategies.
2. The result can be proposed as an alternative solution, to be used in the real world.
3. It could be a case study of mobility mechanism between MIPv4 and MIPv6 by applying the interoperation of NAT-PT with DNS-ALG.
4. It may be lead to new ideas for developing a mobility mechanism between MIPv4 and MIPv6 by using existing translation techniques.

7.2.2 Limitations

1. To obtain mobility between MIPv4 and MIPv6, the MN and its HA need to be upgraded to support the new protocols developed here.
2. This solution is designed to provide mobility between MIPv4 and MIPv6 only focuses upon the registration procedure. To be practical this solution needs to implement the other procedures such as the intelligent movement detection and data delivery.
3. This solution as designed by doesn't use Route Optimization.
4. This solution has not tested been tested for performance, nor has any comparison of overhead with other methods been made.
5. A better analysis of the security implications, and search for a solution to the DoS attack described in Chapter 4 is needed before deployment of this proposal.
6. The implementation constructed is of prototype quality only, and is currently unsuitable for practical use. It exists as a proof of concept only, demonstrating that the idea is feasible.

7.2.3 Future work

There are a few suggestions for mobility mechanisms between MIPv4 and MIPv6. A better analysis of the costs and benefits of each is needed. This work suggests one solution that can enable connectivity between MIPv4 and MIPv6. However, this solution only focus upon the registration procedure, we have not tested performance, and haven't tested this solutions security issues. We suggest the following topics as being worthy of future work.

1. To re-implement and test this method considering particularly security
2. To enable route optimization with this solution to allow IPv6 nodes to avoid indirection via the HA. This is likely to require the protocol translation mechanism be extended with MIP knowledge. For this task this is possible, as communications between MN and CN are not protected by security protocols - the packets are not encrypted, unlike those between the MN and HA.
3. To perform a comparative analysis with the other methods.
4. To adapt this solution with mobile networks (NEMO) and the nested NEMO.

References

[1] *Jon Postel:* "Internet Protocol", RFC 791, September 1981.

[2] *P. Srisuresh:* "IP Network Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[3] *Y. Rekhter:* "CIDR and Classful Routing", RFC 1817, Augest 1995.

[4] *R. Hinden:* "IP version 6 Addressing Architecture", RFC 2373, July 1998.

[5] *S. Deering:* "Internet Protocol version 6 Specification", RFC 2460, December 1998.

[6] *R. Droms:* "Dynamic Host Configuration Protocol", RFC 2131. March 1997.

[7] IETF IPv6 Transition Working Group (ngtrans). http://www.6bone.net/ngtrans/

[8] *R. Gilligan:* "Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, August 2000.

[9] *S. Deering:* "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[10] *E. Nordmark:* "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[11] *G. Tsirtsis:* "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[12] *H. Kitamura, A. Jinzaki and S. Kobayashi:* "A SOCKS-based IPv6/IPv4 Gateway Mechanism", RFC 3089, April 2001.

[13] *J. Bound, L. Toutain and H. Affifi:* "Dual Stack Transition Mechanism (DSTM)", draft-ietf-ngtrans-dstm-07, February 2002..

[14] *A. Durand :* "IPv6 Tunnel Broker", RFC 3053, January 2001.

[15] *F. Templin, T. Gleeson, M. Talwar and D. Thaler:* "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", draft-ietf-ngtrans-isatap-08, December 2002.

[16] *K. Tsuchiya and H. Higuchi:* "Dual Stack Hosts using the "Bump-In-the-Stack" technique(BIS)", RFC 2762, February 2000.

[17] *S. Lee et al:* "Dual Stack Hosts using "Bump-in-the-API" (BIA)", draft-ietf-ngtrans-bia-05, February 2001.

[18] *B. Carpenter:* "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529,March 1999.

[19] *S.Thomson:* "IPv6 Stateless Address Autoconfiguration" , RFC 2462, December 1998.

[20] *B. Carpenter and K. Moore:* "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[21] *C.Huitema*: "An Anycast Prefix for 6to4 Relay Routers", RFC3068, June 2001.

[22] *C. Huitema*: "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC4380, Febuary 2006.

[23] *P. Srisuresh, G. Tsirtsis, P. Akkiraju and A. Heffernan*: "DNS extensions to Network Address Translators (DNS_ALG)", RFC 2694, September. 1999.

[24] *J.Hagino and K.Yamamoto:* "An IPv6-to-IPv4 Transport Relay Translator (TRT)", IETF draft, draf-ietf-ngtrans-tcpudp-relay-03, April 2001.

[25] *S. Bradner and A. Mankin:* "The Reccommecdation for the IP Next Generation Protocol", RFC1752, January 1995.

[26] *C. Perkins and Ed* : "IP Mobility Support for IPv4", RFC3344, August 2002.

[27] *D. Johnson, C .Perkins and J. Arkko*: "Mobility Support in IPv6", RFC3775, June 2004.

[28] *L. Joo-Chu, S. Myung-Ki and K. Hyong-Jun*: "Implementationof NAT-PT/SIIT, ALGs and Consideration to the Mobility Support in NAT-PT Environment", IEEE, 2004.

[29] *S.Kent, BBN Corp and R. Atkinson*: "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[30] *S. Kent and R. Atkinson*: "IP Authentication Header (AH)", RFC 2402, November 1998.

[31] *S. Kent and R. Atkinson*: "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[32] *H. Soliman, C. Castelluccia, K. El Malki and L. Bellier*: "Hierarchical Mobile IPv6 Mobility Management (HMIPv6)", RFC 4140, August 2005.

[33] *R. Koodli and Ed*: "Fast Handovers for Mobile IPv6", RFC 4068, July 2005.

[34] *J. Edgrad, S. Simone and M. Carlos*: "Global mobile IPv6 addressing using transition mechanisms", Journal of the Brazilian Computer Society, April 2003.

[35] *K. Shima, K. Mitsuya and R. Wakikawa:* "SHISA: The MIPv6/NEMO BS Stack Implementation Current Status", Asia BSD Conference, March 2007.

[36] *K. Shima, K.Mitsuya, T. Momose, S. Karino, R. Wakikawa, K. Sugyou and K. Uehara:* "Designing and Implementing IPv6 Mobility stack on BSD Operating Systems", SHISA mobility stack [Online], http://www.mobileip.jp

[37] FreeBSD [Online], http://www.freebsd.org

[38] KAME project [Online], http://www.kame.net

[39] O'Reilly, IPv6 Essentials 0-596-10058-2. Chapter 11: Mobile IPv6.

[40] Wireshark [Online], http://wireshark.org

[41] Mobile-IP Implementation [Online], http://www.mip4.org/2004/implementations

[42] *S. Waldbusser, K. Frisa:* "AppleTalk Management Information Base II", RFC 1742, January 1995.

[43] *L. Anders:* "Seamless mobility between current and future IP networks", MSc Programmes in Engineering [Online], http://epubl.luth.se/1402-1617/2004/032/index-en.html

VITAE

Name	Miss Kuljaree Tantayakul
Student ID	4712004

Educational Attainment

Degree	**Name of Institution**	**Year of Graduation**
B.Eng. (Computer Engineering)	Prince of Songkla University	2004

List of Publication and Proceedings

T. Kuljareel, Robert Elz, K. Sinchai, A. Touchai: "Mobility Mechanism between MIPv4 and MIPv6", Proceeding of Communication Systems and Networks 2007 (AsiaCSN 2007), April 2007 Page(s):196-201.