Secured Communication Through the NAT-PT

Kedar Chandra Das

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montreal, Quebec, Canada

August 2005

Your file Votre référence
ISBN: 978-0-494-16255-2
Our file Notre référence
ISBN: 978-0-494-16255-2

Abstract

Secured Communication through NAT-PT

Kedar C. Das

This thesis deals with the study of Network Address Translation-Protocol Translation (NAT-PT), its limitations, and the way of avoiding the drawbacks of the protocol. NAT-PT is a transition mechanism for establishing communication between an IPv6 network and legacy systems. RFC 2766 describes the semantics of this mechanism. However, the proposed mechanism as described by RFC 2766 has a number of serious drawbacks that are of primary concern to its users. Due to these limitations, this mechanism is not widely accepted by the Internet community. Some of the most critical limitations of the proposed NAT-PT have been identified as end-to-end security, scalability, DoS attacks, etc. NAT-PT does not allow network layer and, in some cases application layer end-to-end security. As a result, the use of NAT-PT increases the threats to the existing vulnerable network security. The current study addresses the security related drawbacks of the existing NAT-PT model, and proposes a modified NAT-PT model. The modified model is able to establish secured communication between IPv6 and IPv4 as well as to correct other problems that may arise from the use of the existing NAT-PT. In addition, the current study also outlines a formal validation of the NAT-PT model with a model checker tool SPIN, which is a very powerful validation tool for distributed systems.

Acknowledgement

I would like to thank my supervisor, Professor J. William Atwood, for his valuable guidance, financial support, and advice. Besides, he recruited me to work on a research project at Ericsson Research, Montreal, Canada, and working on that project at Ericsson Lab, I got practical exposure to the various aspects of telecommunication and networking. Moreover, it was a great opportunity for me to be a coauthor with my supervisor for two publications on the project that we worked on at Ericsson. I am really grateful to him for this. I am also thankful to Ericsson Research, Montreal for allowing us to work on the project in their labs.

I wish to thank all the people of the Computer Science Department for their help and cooperation during my study at Concordia. Finally I would like to thank my friend Sulata Mojumder, who helped me doing linguistic correction of my thesis.

Table of contents

List of Figures

Chapter 1 : Overview and Objective of the research work

1.1 Introduction

In order to resolve critical limitations of the existing Internet Protocol version 4 (IPv4), The Internet Engineering Task Force (IETF) proposed a new Internet Protocol Standard--Internet Protocol version 6 (IPv6) or the Next Generation Internet Protocol (IPng). Address space limitation is one of the main drawbacks of IPv4. The Internet community realized that 32-bit IP addresses would no longer be available very soon. Since the new IPv6/IPng (128 bit address space) architecture solves the several problems including the address space problem of IPv4 in an effective way, early adoption of this new protocol is already taking place, and widespread adoption is expected over the next few years.

1.2 Transition Mechanism

The Transition Mechanism is an interim arrangement, which allows coexistence of IPv4 and IPv6 networks. Because of the vast size and coverage of the Internet, it is not possible to execute a rapid and centrally coordinated migration from IPv4 architecture to IPv6. This necessitates a gradual and smooth migration of the Internet Protocol from IPv4 to IPv6. Hence, it is essential that both IPv4 and IPv6 coexist and work

consistently and reliably with each other. Again at the same time IPv4 and IPv6 are not compatible with each other. They differ in IP Header format. The following figures depict the incompatibility of IP Header format of IPv4 and IPv6.

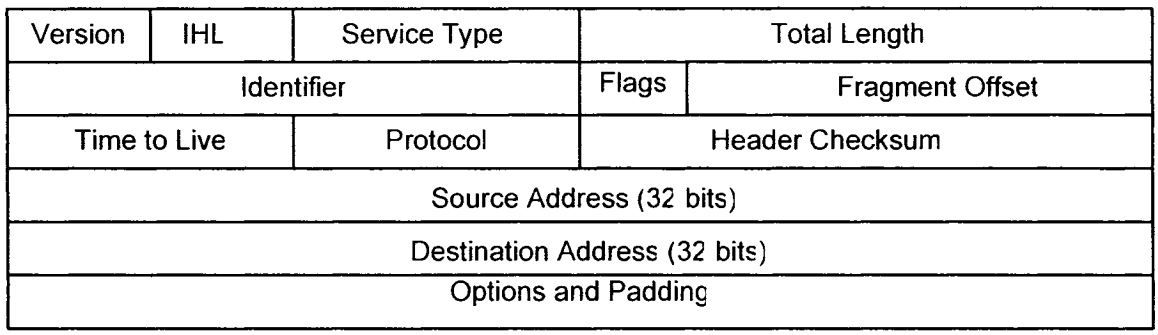

Figure 1.1 IPv4 Header

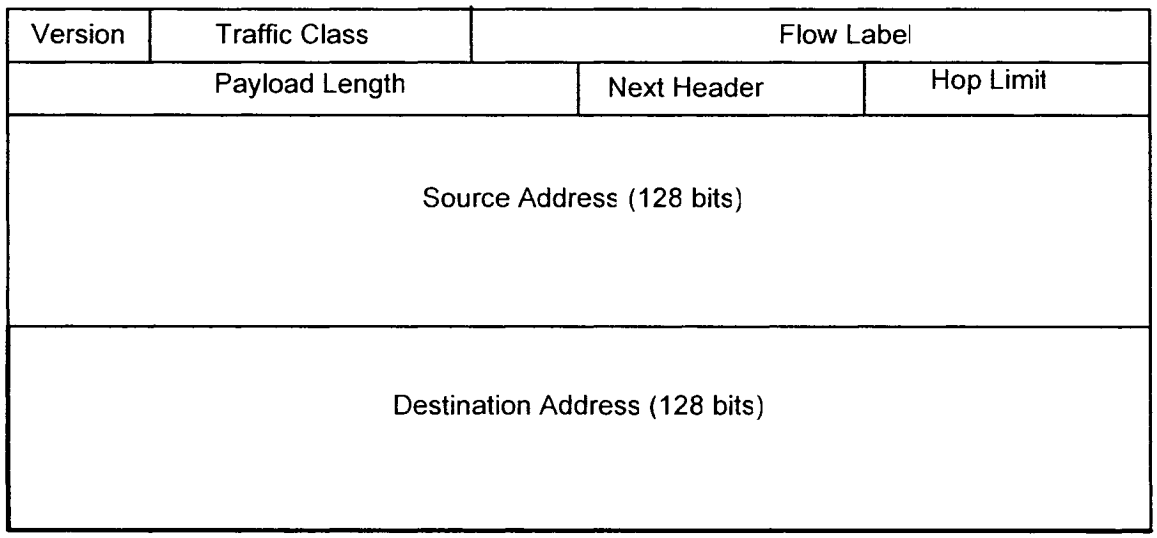

Figure 1.2 IPv6 Header

Due to the incompatibility of IP Header formats, and the coexistence of the both type of protocols, there must exist some mechanisms to interface between IPv4 and IPv6 networks. These mechanisms have been termed as Transition Mechanism for IPv6.

The IETF specifications for IPv6 contain detailed information concerning the transition issues. Most of the documents are presented in the form of RFCs, and some information is available as Internet Drafts. The RFC 2893 published by the IETF has specified a set of mechanisms for smooth, stepwise and independent transition. These mechanisms are suitable for true internetworking, coexistence, easy address mapping and name service migration, for example.

1.3 Transition Mechanisms Proposed by IETF

The IETF identified a number of transition techniques, and these techniques basically fall into three categories:

Dual-stack: This technique requires that both the protocols (IPv4 and IPv6) coexist in the same devices and networks.

Tunneling: According to this mechanism the packets from IPv6 network are shielded with IPv4 header and tunneled across IPv4 network to another IPv6 network. That means it establishes communication between two IPv6 networks across IPv4 networks. This technique helps to avoid order dependencies when upgrading hosts, routers, or regions.

Translation: This transition method allows IPv6-only devices to communicate with IPv4-only devices.

Each transition mechanism addresses a specific problem and thus applies to a specific situation. Thus, they are supplementary to each other and not complimentary. A Complex situation may require a combined application of all the above mechanisms.

1.3.1 Dual Stack or Dual IP layer

The most straightforward procedure to satisfy the requirement for full intersystem compatibility is to include a complete IPv4 implementation alongside new IPv6 systems. This is what we call an IPv6/IPv4 node. Such a node is able to transmit both IPv4 and IPv6 packets and thus able to interact with all IP systems in the network.

The dual stack approach does not necessarily imply that the system should contain two separate protocol stack implementations; rather it will contain duplicated IP layer. It is imperative that a dual stack node has both IPv4 and IPv6 addresses assigned to it. It also includes "Resolver" libraries capable of dealing with A, AAAA, or A6 records. From the application point of view, there are still two separate APIs and whether IPv4 or IPv6 API is to be used depends on the destination address.

1.3.2 Tunneling

This transition mechanism establishes communication between two IPv6 networks through IPv4 network(s). In this approach, two IPv6 networks

must be equipped with encapsulating and de-capsulating nodes. These encapsulating and de-capsulating nodes are usually the gateway of the two communicating networks, and they must be dual-stack. The source network encapsulates IPv6 packets within IPv4 packets; the encapsulated packet pass through IPv4 networks as IPv4 packets, and finally the destination network de-capsulates the encapsulated packet. As it apparently seems that the IPv6 packet passes through a tunnel, the transition approach is known as Tunneling. RFCs 2529, 3053, and 3056 provide the complete specifications and descriptions of different types of tunneling.

1.3.3 Translation

This transition mechanism establishes communication between IPv4 and IPv6 networks. IP address and IP header formats in IPv6 are different from those of IPv4. Therefore, the IPv4 node does not understand the IPv6 IP header, and vice versa. So to establish communications between two address realms, there needs to be a translator in between two types of network.

1.3.3.1 The Existing Translation Mechanism

The Internet Engineering Task Force (IETF) proposed IPv6 protocol specifications and different transition mechanism standards through Internet Drafts and "Request for Comment" (RFC). RFC 2766 outlines the

only existing framework for translation mechanism. According to the framework specified in RFC 2766, British Telecommunication (BT) developed NAT-PT for FreeBSD, and it works fine. We have set up testing environment in the networking lab at Concordia, and we tested NAT-PT for establishing communication between IPv6 and IPv4 networks. We tested NAT-PT for HTTP server and HTTP client applications.

1.3.3.2 Drawbacks of the Proposed Translation Mechanism

In the current research, we will focus on NAT-PT details, its limitations and devise mechanism to overcome these limitations. Chapter two describes the NAT-PT in detail as described in RFC 2766. Like any other system, NAT-PT as specified in RFC 2766 is not flawless. Rather, it brings several serious concerns for its user. Among them the security is the most important hitch, because NAT-PT breaks end-to-end security. Other than security, there are also several other concerns such as scalability problem, DoS attack etc. Chapter three identifies and discusses the problems of the existing NAT-PT in detail.

1.4 The Objective of the Current Research

The objective of the current research is to develop a modified NAT-PT model, which will be able to establish communication between IPv4 and IPv6 networks securely and efficiently.

Through our research work, first we will identify and analyze the reasons behind the vulnerabilities of NAT-PT as mentioned in the previous section, and then propose a new framework for NAT-PT, which will remove or minimize most of the problems of the existing framework.

We will see in Chapter 2 that the NAT-PT translates IP header, so it is not possible to apply existing end-to-end network layer security for NAT-PT. As an alternative measure to address end-to-end security, we can divide the NAT-PT environment into several regions—IPv4 end host to IPv4 trusted host, IPv4 trusted host to the NAT-PT to IPv6 trusted host, and the IPv6 trusted host to IPv6 end host, establish security among the smaller regions separately, and then deploy a mechanism so that all the individual secured regions work together seamlessly. The NAT-PT within a framework of distributed nature will implement that mechanism. In order to make NAT-PT working as a distributed system, I have taken into consideration of the frameworks described in the following subsections.

1.4.2 MIDCOM

RFC 3234 defines a MiddleBox as "any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host". The MiddleBox Communication (MIDCOM) protocol is the framework for the MiddleBox;

and the MIDCOM requires application intelligence for its operation. According to the definition of MiddleBox, the NAT-PT box is categorized as a MiddleBox, and the framework for the NAT-PT box is MIDCOM. We, therefore, implement NAT-PT as an application of the MiddleBox Communication (MIDCOM) framework. The MIDCOM framework requires a protocol to achieve its commitment while the protocol has not been chosen yet the Simple Network Management Protocol (SNMP) is the leading contender. So at this point we will develop our MIDCOM model for NAT-PT using SNMP.

1.4.2 SNMP

SNMP is the acronym for "Simple Network Management Protocol". The SNMP, an Internet standard, defines methods for governing network management and the monitoring of network devices and their functions. It is a well-matured and proven protocol for network management. Although the NAT-PT is an operational protocol, the distributed NAT-PT system requires a level of managerial services. SNMPv3 also ensures secured communication between SNMP agents and SNMP manager. This feature is essential for establishing secured communication through NAT-PT. In addition, SNMP is a highly robust protocol. The main objective of our research is to modify the existing NAT-PT framework so that it can establish secured and efficient communication between IPv4

and IPv6 networks. Hence, we have chosen SNMP as the protocol in the framework for our distributed NAT-PT.

We will develop a distributed NAT-PT model using SNMP. NAT-PT will work as SNMP agent and ALGs and other modules will work as SNMP manager. The primary objective is to develop a secured NAT-PT. So, we will obviously select SNMPv3 as the communication protocol between the NAT-PT agent and the manager. The proposed model contains three security regions—IPv4 to security proxy, security proxy to NAT-PT host to IPv6 security proxy, and IPv6 security proxy to IPv6 end host. These three security regions will work seamlessly among themselves to ensure authenticity and the integrity of data exchanged through NAT-PT.

As the distributed NAT-PT works within the SNMP framework, it requires Management Base Information (MIB). I wrote a MIB for NAT-PT using the Abstract Syntax Notation dot 1 (ASN.1).

1.4.3 Validation

Completeness and logical consistency are the two most important criteria for a model. We have verified our proposed model with SPIN model checker tool. It supports high-level language called **PRO**cess **ME**ta **LA**nguage (PROMELA), and it is very suitable for tracing the logical design errors in distributed system. We have defined the procedures and

rules of the model using PROMELA and verified that all the processes of the model can interact successfully.

1.5 Thesis Organization

RFC 2766 specified NAT-PT semantics in detail. In Chapter 2, I describe the existing NAT-PT as specified in RFC 2766. I elaborate the limitations, and drawbacks of the existing NAT-PT in Chapter 3. Chapter 4 describes the MIDCOM framework as outlined in RFC 3303. Chapter 5 describes SNMP, and the distributed NAT-PT within SNMP framework in detail. Chapter 6 explains validation of the proposed model. Then I summarize the present works and point out the direction for future work in Chapter 7. As mentioned in subsection 1.3.3.1, we tested NAT-PT, and the detailed description of the test environment setup and test procedure is given in Appendix A. The full MIB (NAT-PT MIB) is given in Appendix B. Appendix C gives the overview of the functional analysis of the existing NAT-PT.

1.6 Summary

This chapter serves as the background of the research work. It gives a brief idea about IPv6, what is transition mechanism and what are the proposed transition mechanisms. We have seen that all the transition mechanisms proposed by IETF fall into three categories—Dual Stack, Tunneling, and Translation. We are interested in translation mechanism. Then we have set out the goal of the research. This chapter opens a

section, which describes the validation procedure of the research work. Finally it outlines the organization of the thesis.

As we are interested in translation mechanism, Chapter 2 presents the ins and out of translation mechanism and NAT-PT.

Chapter 2 : Transition Mechanism and NAT-PT

This chapter gives a brief overview about transition mechanism (NAT-PT), and then elaborates it with an example.

2.1 NAT-PT

NAT-PT stands for "Network Address Translation-Protocol Translation". Here the **NAT-PT** refers to the translation of IPv4 header including IPv4 address into semantically equivalent IPv6 header, and vice versa.

2.1.1 Address Translation

The address translation means the replacing of IPv4 addresses of a network packet with IPv6 addresses, and vice versa. The source and destination address of a network packet must be either IPv4 or IPv6, not mixture of the two types. The IPv6 IP address is not backward compatible. So a packet with IPv4 as source address and destination address cannot be forwarded to an IPv6 network. Similarly a packet with IPv6 as source and destination addresses cannot get into an IPv4 network. Therefore, to establish communication between the IPv6 and IPv4 worlds, there should be a mechanism in between IPv4 and IPv6 network to change the source and destination addresses from either IPv4 to IPv6 or IPv6 to IPv4. This mechanism is the core part of NAT-PT. It uses a pool of IPv4 addresses, and when a session is initiated across the V4-V6 boundaries, the NAT-PT assigns an IPv4 address to the

communicating IPv6 node. This is called address mapping between IPv4 and IPv6 addresses. This mapping is dynamic and exists until the end of the session. The IETF proposed two flavors of NAT-PT—Traditional and Bi-directional. The traditional NAT-PT requires that the session be initiated just from one direction—from IPv6 network, however, the bi-directional NAT-PT is free from that session initiation limitation, and it is practically suitable for all situations. Hence, we are interested in Bi-directional NAT-PT.

When an IPv6 node initiates a session to communicate with an IPv4 node in an external domain, the NAT-PT assigns an IPv4 address from the address pool to the IPv6 node. This assigned address will work as the source IP address. This means the original source IPv6 address is now translated to an equivalent IPv4 address.

2.1.2 Protocol Translation

Given that all the fields of IPv6 headers are not the same as that of IPv4 header, NAT-PT translates IP/ICMP headers to make end-to-end IPv6 to IPv4, and vice versa communication possible. For any packet outbound from the IPv6 domain, other than the source IP address translation, the NAT-PT also changes other fields of IP header, and the checksums of TCP, UDP, and ICMP header in such a way that the packet becomes semantically equivalent to an IPv4 packet without loss of information. In

the same way for inbound packets to IPv6 domain, the NAT-PT translates the destination IP address, and the other fields as mentioned above, and the packet becomes an IPv6 network packet.

2.1.3 Application Layer Gateway

Several applications such as DNS server and SIP send IP addresses and other information that varies from IPv4 to IPv6, in payload of network packet. So in addition to IP header translation, it is necessary to translate payload of a network packet for establishing communication between two applications in different address realms. NAT-PT does not snoop payload, so it is application unaware. NAT-PT environment uses X-ALG to alter payload. X stands for specific application and ALG is the acronym of "Application Layer Gateway". DNS-ALG, FTP-ALG, SIP-ALG are examples of application specific ALGs.

2.1.4 NAT-PT Concept with an Example

Let

Node IPv6-A have an IPv6 address -> FEDC:BA98::7654:3210

Node IPv6-B have an IPv6 address -> FEDC:BA98::7654:3211

Node IPv4-C have an IPv4 address -> 132.146.243.30

NAT-PT have a pool of addresses of 120.130.26.1 and 120.130.26.2.

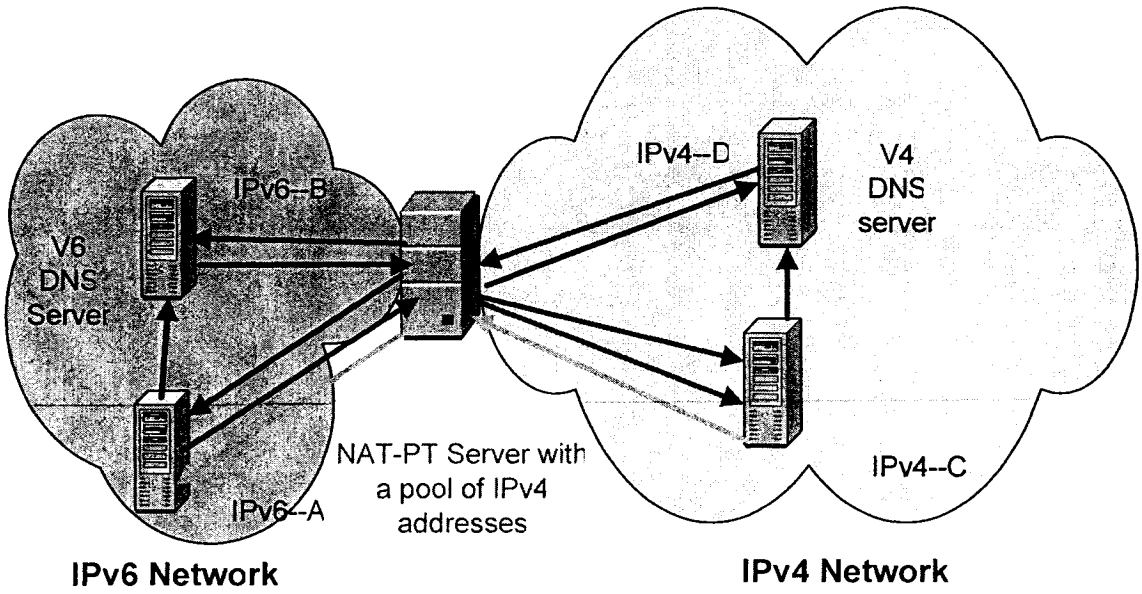

Figure 2.1 Communication between IPv4 and IPv6 through NAT-PT

2.1.4.1 Session Originating in IPv4 side

For example, let IPv4-C attempt to initialize a session with node IPv6-A by making a name lookup ("A" record) for Node-A. The name lookup that means DNS query goes to the local DNS and from there it is propagated to the DNS server of the IPv6 network through NAT-PT and DNS-ALG. NAT-PT will look for the mapping between destination address of the name lookup packet, and the IPv6 DNS server address. If the said mapping exists, NAT-PT will replace the destination address of the packet with mapped IPv6 address and the source address with PREFIX:132.146.243.30. The PREFIX is a 96 bit fixed one. Then the DNS-ALG intercepts the payload of the packet, translates the query type from "A" to "AAAA" or "A6", and then NAT-PT forwards it to the DNS server in the IPv6 network.

DNS-ALG intercepts the payload of the DNS response to find the resolved address of the Node-A, and will advise NAT-PT to create mapping between the IPv6 address of Node-A, and an address from the pool (say it is 120.130.26.1). Then DNS-ALG will replace the A6 and FEDC:BA98::7654:3210 with A and 120.130.26.1 respectively.

NAT-PT will forward the translated response to Node-C, which will initiate a session as follows:

SA=132.146.243.30, source TCP port = 1025

DA=120.130.26.1, destination TCP port = 80

As NAT-PT already holds a mapping between FEDC:BA98::7654:3210 and 120.130.26.1, receiving the packet NAT-PT can translate the packet to:

SA=PREFIX::132.146.243.30, source TCP port = 1025

DA=FEDC:BA98::7654:3210, destination TCP port = 80

And the communication can now proceed as normal.

2.1.4.2Session Originating in IPv6 side

Let say Node-A wants to set up a session with Node-C. Node-A starts it by making a name look-up ("AAAA" or "A6" record) for Node-C. Since Node-C may have IPv6 and/or IPv4 addresses, the DNS-ALG on the NAT-

PT device splits the original "AAAA/A6" query into "A" query and "AAAA/A6" query, and forwards both the queries to the external DNS. In this case the NAT-PT will receive response from external DNS for "A" record. As the return value is for "A" record, the DNS-ALG will add the fixed PREFIX to the return value (IPv4 address) to convert the IPv4 DNS response to IPv6 DNS response, and forward it to the Node-A. Hence, the Node-A receives an IPv6 DNS response, and the resolved IP address looks like PREFIX::IPv4_Address. For this example DNS response looks as below:

Node-C AAAA PREFIX::132.146.243.30 or to

Node-C A6 PREFIX::132.146.243.30

Now Node-A can use the resolved IP address like any other IPv6 address and communicate with IPv4-C, and the V6 DNS server can even cache it as long as the PREFIX does not change.

Node IPv6-A creates a packet with:

Source Address, SA=FEDC:BA98::7654:3210

Destination Address, DA = PREFIX::132.146.243.30

IPv6 network needs to be pre-configured so that all packets containing the PREFIX::/96 in the destination address must go to the NAT-PT gateway, where it is translated to IPv4 format. The NAT-PT peels off the PREFIX of the destination address to get the actual IPv4 address of the

Node-C. It also allocates an IPv4 address from address pool (say it 120.130.26.1), creates mapping between FEDC:BA98::7654:3210 and 120.130.26.1, and saves the created mapping into NAT-PT mapping table. Along with the address mapping NAT-PT stores other information related to the session. The mapping remains in the mapping table for the lifetime of the session.

The translated IPv4 packet has SA=120.130.26.1 and DA=132.146.243.30. The NAT-PT will use session information to recognize any returning traffic, and it will replace the source address by SA=PREFIX::132.146.243.30, and the destination address by DA=FEDC:BA98::7654:3210. Now it is easy to route this packet inside the IPv6-only stub network.

2.2 Summary

This chapter describes Network Address Translation and Protocol Translation (NAT-PT) in detail. NAT-PT has two parts—Network Address Translation, and Protocol Translation. With the help of Application Level Gateway (ALG), NAT-PT provides services to applications, which carry IP address as the payload of network packet. In this chapter we have looked at ALG, and why do we need it. Then this chapter describes NAT-PT with a specific example, which explains the network packet flow from IPv4 to IPv6 network through NAT-PT and vice versa.

Our objective is to develop a NAT-PT model, which will address the drawbacks of the existing NAT-PT. So we will see the drawbacks of the existing NAT-PT model in Chapter 3.

Chapter 3 : Limitations of NAT-PT and Proposed Solutions

This chapter describes the limitations of proposed NAT-PT along with a number of proposed solutions to overcome these limitations.

3.1 Limitations

The following are the major drawbacks of the NAT-PT—

3.1.1 Scalability Problem

NAT-PT will translate all the requests and responses pertaining to a session between IPv6 and IPv4. Therefore, all the requests and responses pertaining to a particular session must be routed via the same NAT-PT router. In some cases, the volume of network traffic may exceed capacity of the NAT-PT router, and consequently the single router may not be able to handle network traffic efficiently.

3.1.2 Protocol Translation Limitations

IP Header in IPv4 differs from that of IPv6, and it requires protocol translation. However, this translation is not straightforward. For example, the IPv4 option headers semantics and syntax have been changed significantly in IPv6.

3.1.3 End-to-end Security

The proposed NAT-PT as described in RFC 2766 suffers from the drawback of end-to-end network layer security. Also transport and application layer security may not be possible for applications that carry IP addresses to the application layer. This is an inherent limitation of the Network Address Translation function.

When IPv6-only node-A initiates a session to establish communication with IPv4 only node-C, the packet from node-A will contain FEDC:BA98::7654:3210 as the source address, and PREFIX::132.146.243.30 as the destination address. The computations of IPSec (ESP or AH) and TCP/UDP/ICMP checksum depend on the source and destination address of the packet. To establish communication between IPv4 and IPv6 network, NAT-PT replaces the IPv6 address of the node-A with an IPv4 address (120.130.26.1) that has no relation to the original address of the node-A. The recipient node-C will not understand the original IPv6 address and there is no way for recipient to verify the TCP/UDP/ICMP checksums.

3.1.4 Prefix Assignment

NAT-PT uses a prefix to translate an IPv4 address into an IPv6 address. However, RFC2766 does not describe how an IPv6 node learns the prefix that is used to route packets to the NAT-PT box. Here the solution is to

add the prefix to the routing table of the IPv6 nodes so that IPv6 nodes can forward any packet containing the destination address constructed from prefix to the NAT-PT box. The use of fixed prefix may cause reachability problem, which will arise if the NAT-PT box were to shut down. Again somehow providing a fake prefix to IPv6 nodes, an attacker will be able to steal all of the node's outbound packets to IPv4 nodes.

3.1.5 DNS-ALG

Bi-directional NAT-PT requires DNS-ALG, which translates A record into AAAA or A6 record and vice versa. Since DNS-ALG modifies the payload of DNS query or response, DNS-SEC will not work with NAT-PT.

3.1.6 Source Address Spoofing Attack

We consider two cases:

1. Attacker is in the same stub domain as the NAT-PT (IPv6 side of the NAT-PT)
2. Attacker is outside of the NAT-PT stub domain.

3.1.6.1 Attacker in the NAT-PT Stub Domain

An attacker in the IPv6 side of the NAT-PT sends a packet destined for an IPv4-only node-C on the other side of NAT-PT, and the attacker forges its source address to be an address that is topologically inside the stub domain. This address may belong to another node, or it may be unassigned.

3.1.6.1.1 Address Depletion Attack

RFC 2766 describes the DoS (Denial of Service) attack. If the IPv6 attacker sends many such packets, each with a different source address, then the pool of IPv4 addresses may get used up, and exhausted resulting in a Denial of Service attack.

There are also several other attacks such as reflection attacks, resource exhaustion attacks, and broadcast/multicast attacks. Reflection attack occurs when the attacker uses an IPv6 address of existing node as the source address of the packet. Then targeted IPv6 node will be the recipient of a reflection attack, as IPv4 node will send response packets to the victim node. On the other hand, if IPv6 source address set to that of a non-existent node, then the return packets will be dropped, which will cause resource exhaustion attack. If the IPv6 source address is a multicast address then the return packet from the IPv4 node will be sent to the multicast address, resulting in a multicast attack.

3.1.6.2 Attacker outside of NAT-PT Stub Domain

An attacker is on the IPv4 side of the NAT-PT sends a packet destined for an IPv6-only node-A, which is behind NAT-PT, and the attacker forges its source address to be an address that is topologically in the IPv4 side of the NAT-PT. The Attacker may use an address belonging to another node, or unassigned one as the source address of the packet. In this case all

the same attacks are possible as the case described in the previous section.

On the other hand, it is not difficult for an attacker to know the IP address of the NAT-PT. In this case, an attacker that knows the IP address of the NAT-PT box can send packets directly to the box. It can use NAT-PT resources, preventing legitimate IPv6-only nodes from accessing NAT-PT services.

3.2 Possible solutions

The following techniques may be applied to avoid the problems as described in the previous section. Some of these techniques have been extracted from Internet drafts, and the mailing lists of different IETF working groups. New techniques have also been devised and included here as part of the current research.

3.2.1 Scalability Problem

We can use mNAT-PT to avoid this problem. We explored the solution of this problem through our proposed model described in Chapter 5.

3.2.2 End-to-end Security

End-to-end security is not possible with NAT-PT. The reason is outlined in section 3.1.3. However, our proposed model provides an alternative solution to this problem.

3.2.3 Prefix Assignment

The following techniques were stated in [DNSALG] and [mNATPT], as well as in e-mail communication on the v6ops mailing list. DNS servers and DNS-ALG may be used in outgoing connections to return the prefix information to the IPv6 node. This is a way to avoid the problem of a statically pre-configured prefix. For example—

When an IPv6-only node wishes to initiate communications with an IPv4-only node, its resolver library would send an "AAAA" query. This query can be passed through the DNS-ALG, which would receive an "A" record in response. In this case, the DNS-ALG can pre-append the appropriate prefix for the NAT-PT and translate "A" record into an "AAAA" or "A6" record and return it to the IPv6 node.

The DNS-ALG can also monitor the state of a number of NAT-PT boxes and return the prefixes of those that are running. However, solutions stated above may cause other problems also. If the prefix is provided by the DNS-ALG, then IPv6 network will not be aware of that prefix. When an IPv6 node receives a packet with a source address of the format prefix:ipv4 address, it will not be able to respond to the message unless its routing table is not configured dynamically or NAT-PT box does not become a default gateway for any IPv6 node. mNATPT will not work with the default gateway solution.

3.2.4 DNS-ALG

The Internet draft mentioned in the reference described the following solutions for DNS-ALG:

DNS-ALG translates DNS record. So the end host (IPv6 node or IPv4 node) will not be able to verify the signature on a DNS record. However, if the host sets the "AD is secure" bit in the DNS header, then it is possible for the local DNS server to verify the signatures.

Another option for DNS-ALG is to verify the received records (like a DNS resolver), translate them, and sign the translated records (like a DNS server).

A third option would be for a host to have an IPSec security association with the DNS-ALG to protect DNS records.

In this case my suggestion is to implement DNS-ALG with V6 DNS server, and the existing DNS server can be modified to accommodate DNS-ALG. A DNS query may be from IPv4 network via NAT-PT or original IPv6 DNS query. PREFIX of the source address will distinguish these two types of DNS queries.

3.2.5 Source Address Spoofing Attack

This type of attack is not introduced by the NAT-PT. It also exists in the present IPv4 network. So at this point this type of attack is not our main concern. However, the Internet draft describes the following possible solutions.

3.2.5.1 Attacker in the NAT-PT Stub Domain

3.2.5.1.1 Ingress Filtering

Ingress filtering by NAT-PT will prevent an attacking node in its stub domain that forges its source address from performing a reflection attack on nodes in other stub domains. However, this does not prevent such an attacker from performing a reflection attack on other nodes in the same stub domain. This is not an attack introduced by NAT-PT.

3.2.5.1.2 Employing NAPT-PT

Instead of NAT-PT, NAPT-PT may be employed to get around the address depletion attack. NAPT-PT stands for Network Address and Port Translation—Protocol Translation. It translates TCP/UDP ports of IPv6 nodes into TCP/UDP ports of the translated IPv4 addresses. However, as the draft points out, IPv4-node-initiated NAPT-PT sessions are restricted to one server per service.

3.2.5.1.3 Access Control List

NAT-PT should be given a list of nodes to which NAT-PT will offer its translation services. IPSec security association might be another option between the NAT-PT and node to which it will offer its services.

3.2.5.2 Attacker outside of the NAT-PT Stub Domain

3.2.5.2.1 Filtration

NAT-PT should filter out packets from outside that claim to have a source address behind NAT-PT. These are the attacks that are not introduced by NAT-PT.

3.2.5.2.2 Discard Packets

NAT-PT should drop packets that are sent directly to its IP address rather than being routed to it via the PREFIX. If NAT-PT maintains a list of nodes to which it will offer its services, this type of attack will be minimized as well. Or for further security, an IPSec security association could be required between NAT-PT and nodes to which it will offer its services.

3.3 Summary

The NAT-PT proposed by IETF suffers from several drawbacks. This chapter analyzes them in detail. NAT-PT breaks End-to-End security. In addition, scalability problem, problem due to the prefix assignments, etc., are also serious concerns for NAT-PT users. This chapter also

outlines the possible ways of resolving the limitations of the existing NAT-PT.

As we are interested in developing Middle Box Communication (MIDCOM) Framework for NAT-PT, Chapter 4 describes proposed MIDCOM protocol in detail.

Chapter 4 : Middle Box Communication (MIDCOM) Framework

4.1 What is MIDCOM?

MIDCOM is a framework for a device known as middle box that requires the application intelligence for its operation. RFC 3033 defines the MIDCOM framework. NAT-PT box can be considered as a middle box that works on Network layer data; hence, NAT-PT can be an application of MIDCOM framework.

4.2 Motivation for MIDCOM

RFC 3033 mentions the following points as the motivation for MIDCOM:
Tight coupling of application intelligence (ALG) with MiddleBox (NAT-PT box) makes maintenance of MiddleBox hard with the advent of new applications.

Built-in application awareness typically requires updates of operating systems with new applications or newer versions of existing applications.

Operators requiring support for newer applications will not be able to use third party software/hardware specific to the application and will remain at the mercy of their MiddleBox vendor to make the necessary upgrade.

Embedding intelligence for a large number of application protocols within the same MiddleBox increases complexity of the MiddleBox and is likely to be error prone and degrade in performance. Hence, if ALGs and other modules are tightly coupled with NAT-PT, then with the change of application the corresponding ALG will be required be changed, and as ALG is coupled with NAT-PT, the operators will be dependent on the vendor of NAT-PT, which may be unrealistic. Again scalability problem may be solved using the MIDCOM architecture. Tightly coupled NAT-PT system will do all the translation and computation sequentially; this will slow down the network performance. On the other hand, MIDCOM architecture will enable the system to process and translate network packets concurrently.

4.3 MIDCOM Framework

RFC 3303 describes the MIDCOM architecture and framework. MIDCOM is a model that will consist of Middle Box and trusted third parties' application. Application intelligence of the Middle Box will be delegated to the trusted third parties. These third party applications will assist Middle Box in performing its operations. As a result the application intelligence will not be required to be embedded in the MiddleBox. This trusted third party is referred to as the MIDCOM Agent. That means MIDCOM framework consists of two types of entities--MiddleBox and MIDCOM

agent. The MIDCOM protocol is the means of communication between MiddleBox and MIDCOM agents.

A MiddleBox consists of several layers such as interface layer, function layer, policy layer, etc. Function layer implements one or more of the MiddleBox functions selectively on multiple interfaces of network device. Varieties of MIDCOM agents will be able to communicate with MiddleBox function using MIDCOM protocol. MiddleBox upper layer is the MIDCOM protocol interface layer between MiddleBox and MIDCOM agents. That layer facilitates the communication between MIDCOM agents and the specific MiddleBox function. MIDCOM protocol must establish selective communication between MIDCOM agent and one or more middle box functions on the interface. The following diagram identifies a possible layering of the service supported by a MiddleBox and a list of MIDCOM agents that might interact with it.

Firewalls, NAT, NAT-PT, VPN tunneling etc. are some examples of MiddleBox application or services. The MiddleBox applications are known as MiddleBox functions. Each function may have different function specific policy rules. For example, Firewall function specific rule is Access Control Lists (ACL), and NAT-PT function specific rules are address-maps and session-state, etc. Again each rule may include

MiddleBox function specific attributes, such as timeout values, NAT-PT translation parameters, etc.

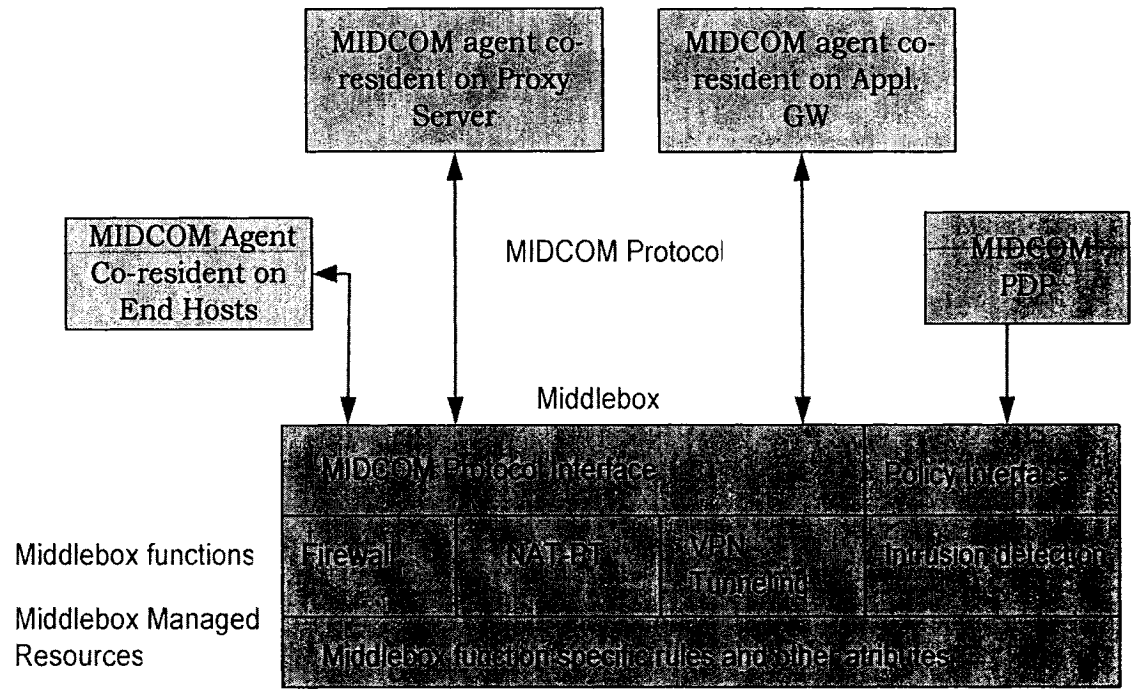

Figure 4.1 MIDCOM agents interfacing with a MiddleBox

According to MIDCOM proposal, application specific MIDCOM agents may be the co-resident on the MiddleBox or external to the MiddleBox. Examples of Application specific MIDCOM agents are SIP-ALG, DNS-ALG, etc. These agents assist MiddleBox in performing functions unique to application and MiddleBox service. For example, an application specific MIDCOM agent such as DNS-ALG, assisting a NAT-PT MiddleBox, performs payload translations, whereas a MIDCOM agent assisting a firewall MiddleBox requests the firewall to permit access to application specific, dynamically generated session traffic.

4.4 MIDCOM MIB

MIDCOM MIB provides a means for MIDCOM agents to control MiddleBox resources and for MIDCOM manager to asynchronously notify the MIDCOM agents of relevant state changes. MIDCOM functions are independent of MIDCOM MIB. These may be vendor specific. However, MIDCOM MIB will have rule-change parameters and a pointer to the application specific MIB objects. MIB for MIDCOM functionalities will actually contain the detailed objects. For instance, multiple agents might end up using the same NAT-PT BIND, yet each agent might define their own Lifetime parameter and directionality for the bind. As a result, the agent specific Bind identifier is set uniquely, independent of the NAT-PT native bind. Yet, the agent specific bind has a pointer to the NAT-PT bind.

MIDCOM MIB is designed to meet the MIDCOM requirements (RFC 3304). A set of MIB objects, one per each MiddleBox resource type, is defined to run MIDCOM transactions. The resulting resources, along with rule-changing parameters and a pointer to FW/NAT-PT MIB objects are maintained as MIB tables, one for each resource type.

4.5 Agent Registration for Notification

midcomAgentTable, a MIDCOM MIB object, is designed to keep records of all the agents that engage in a MIDCOM session with the

MiddleBox.Each active row of the table corresponds to a MIDCOM agent. The agent includes the notification parameters within this row to allow MiddleBox to send asynchronous notifications back to the agent. Also included is an agent-unique MiddleBox Identifier a MiddleBox should use to identify itself during the notifications.

4.6 MIDCOM Transactions and Relevant Tables

MIDCOM transactions may be divided into group transactions and resource transactions. A transaction is atomic and the results of a transaction are saved into relevant tables at the end of the transaction. The same agent may review results of a transaction conducted by an agent anytime prior to executing another transaction of the same kind. midcomTransGroupTable is defined to allow multiple agents to simultaneously add or delete Group identifiers and set group-wide parameters such as Lifetime and Max Idle time. The agent transfers results of the transaction into midcomGroupTable for later reference and further parameter modification. midcomTransBindTable, midcomTransNatSessionTable, and midcomTransFilterTable are defined to allow multiple agents to simultaneously request MiddleBox resources and set parameters such as Lifetime and Max Idle time. Results of the transactions are transferred respectively into the relevant resource table, namely midcomBindTable, midcomNatptSessionTable and

midcomFiltertable for later reference and further parameter modification by the agent.

4.7 MIDCOM Protocol

The communication protocol between MIDCOM agent and a MiddleBox is known as the MIDCOM protocol. It allows the MIDCOM agent to invoke services of the MiddleBox and allow the MiddleBox to delegate application specific processing to the MIDCOM agent.

Several protocols can be considered as strong candidate to be chosen as MIDCOM protocol. Among them SNMP, COPS, COPS-PER, RSIP, DIAMETER are important. Different Internet drafts evaluated these protocols as MIDCOM protocols. Although each protocol meets the requirements to some extent, however, SNMP has been chosen as the ultimate MIDCOM protocol. Compared to the other evaluated protocols, SNMP is very mature and well understood. It is a proven truth that SNMP can operate well in numerous different real-world scenarios. There are also a lot of mature toolsets available for quickly and reliably realizing SNMP managers and agents.

4.7.1 SNMPv3 as MIDCOM protocol

The following diagram is the MIDCOM operational model when SNMPv3 is adopted as the MIDCOM protocol.

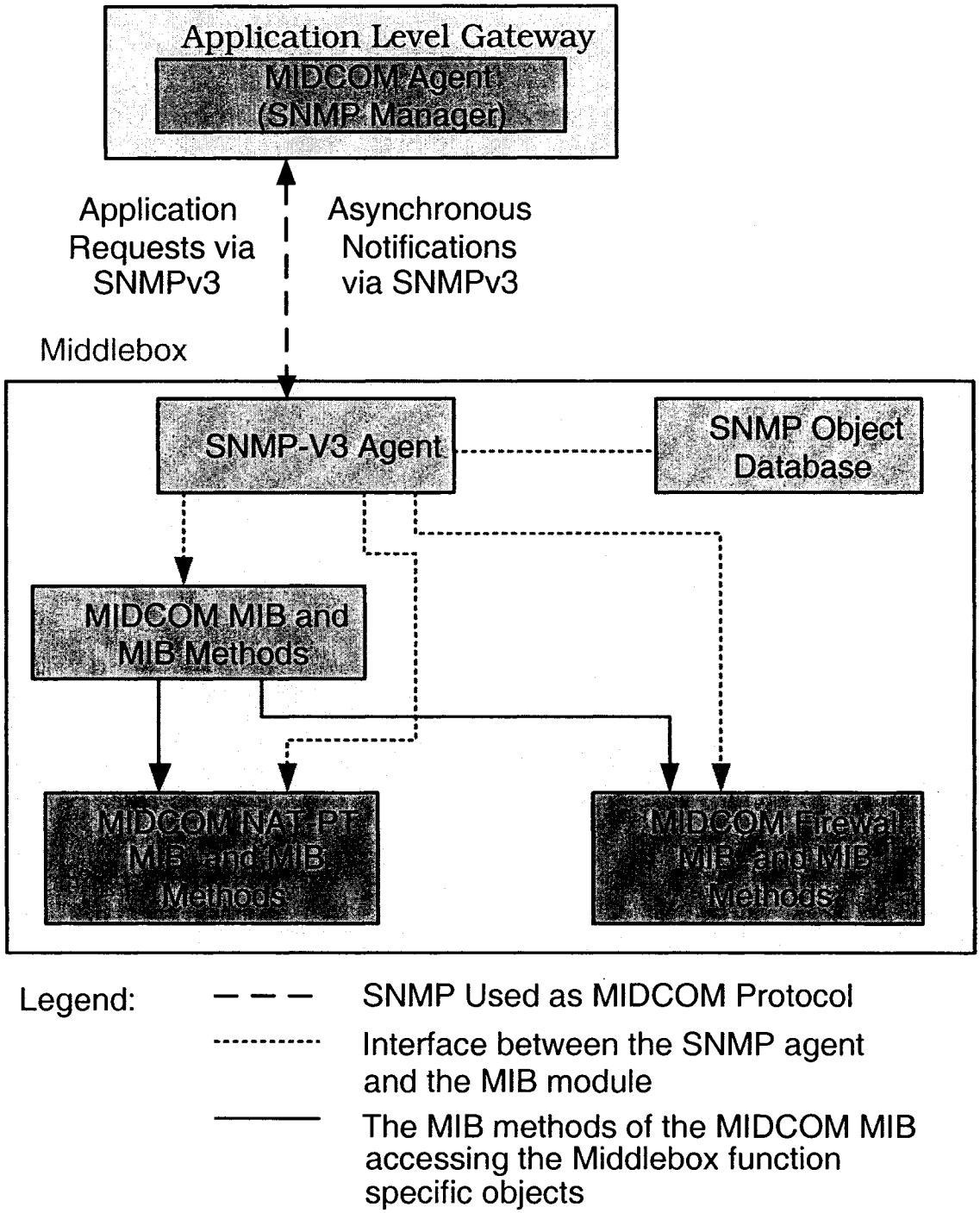

Figure 4.2 SNMPv3 Operating as MIDCOM Protocol

The above SNMP based model includes MIDCOM MIB and MiddleBox function MIB objects.

4.7.1.1 Secure Communications and MIDCOM Protocol

MIDCOM protocol must establish communication between the agents and the MiddleBox reliably. SNMPv3 provides the facilities for secured communication.

SNMPv3 is designed to provide secure communications between two end-points. It defines MIB modules to allow the monitoring and configuration of all these security features. They are defined in RFC 3411-RFC 3418, and RFC3410 provides an overview of these capabilities.

4.8 NAT-PT within MIDCOM Framework

To accommodate NAT-PT within MIDCOM framework, the functionalities of NAT-PT can be decomposed as below:

- NAT-PT main module
- Security Module/Security Proxy
- Protocol Translator
- Application Layer Gateways

4.8.1 NAT-PT

MiddleBox will implement this module. It is the main module of NAT-PT mechanism. This module will read the network packet from IPv4 and IPv6 interfaces, check mapping, create mapping if necessary, generate message to get the services of MIDCOM agents (Figure 4.3). The

communication between MIDCOM agents and MiddleBox will be performed through MIDCOM protocol. MiddleBox will communicate with out of path agents. With the help of MIDCOM agents the NAT-PT will get completely translated packet, and it will forward the packet to the destination address.

4.8.2 Security Server/Security Proxy

NAT-PT does not allow end-to-end security. So to establish secured communication between IPv4 and IPv6 networks, the most reliable method is to deploy security proxy in between the two networks of different address realms. The framework will resemble to a network of like client/server communication through relaying.

Security Server/Proxy will work as an agent of the proposed middle box. So if any node from IPv4 side wants to establish communication with IPv6 side, IPv4 node has to establish client server communication with the security Server/Proxy. Figure 4.3 describes the flow diagram through MiddleBox. Once client-sever connection has been established, subsequent packets will be allowed to pass to the MiddleBox; otherwise all packets will be discarded. In addition, the Security Proxy will check the mapping of NAT-PT, if a mapping between the destination address of the packet and an IPv6 address exists; security agent will allow the packet to pass to the NAT-PT box. This agent will be able to apply its

intelligence to resolve the scalability problem. If multiple NAT-PTs (mNAT-PT) are available, then proxy server will find one NAT-PT that is not overloaded with traffic. If the IPv6 network is behind a firewall, the proxy server will update the MiddleBox firewall access control list so that the network packet from IPv4 side can pass to IPv6 side. Considering all the points described above, the functionalities of security proxy can be summarized as below:

If communicating parties require security, secured Client/Server connection will be established between proxy server and IPv4 node or between proxy server and IPv6 node before establishing connection between IPv4 and IPv6 network. Secured connection means the authenticity of the communicating parties. The existing network security protocols can ensure secured connection between an IPv4 client and an IPv4 security server. IPSec is one of the most recommended security protocols. Security Proxy will work as MIDCOM agent.

For any subsequent packet originating in IPv4 side, the proxy server will check the mapping between the destination address of the packet and IPv6 address, and if the said mapping exists, the proxy server will allow the packet to pass to the NAT-PT. For a packet originating in IPv6 side, the proxy server will check the mapping of the source address to an IPv4 pool address.

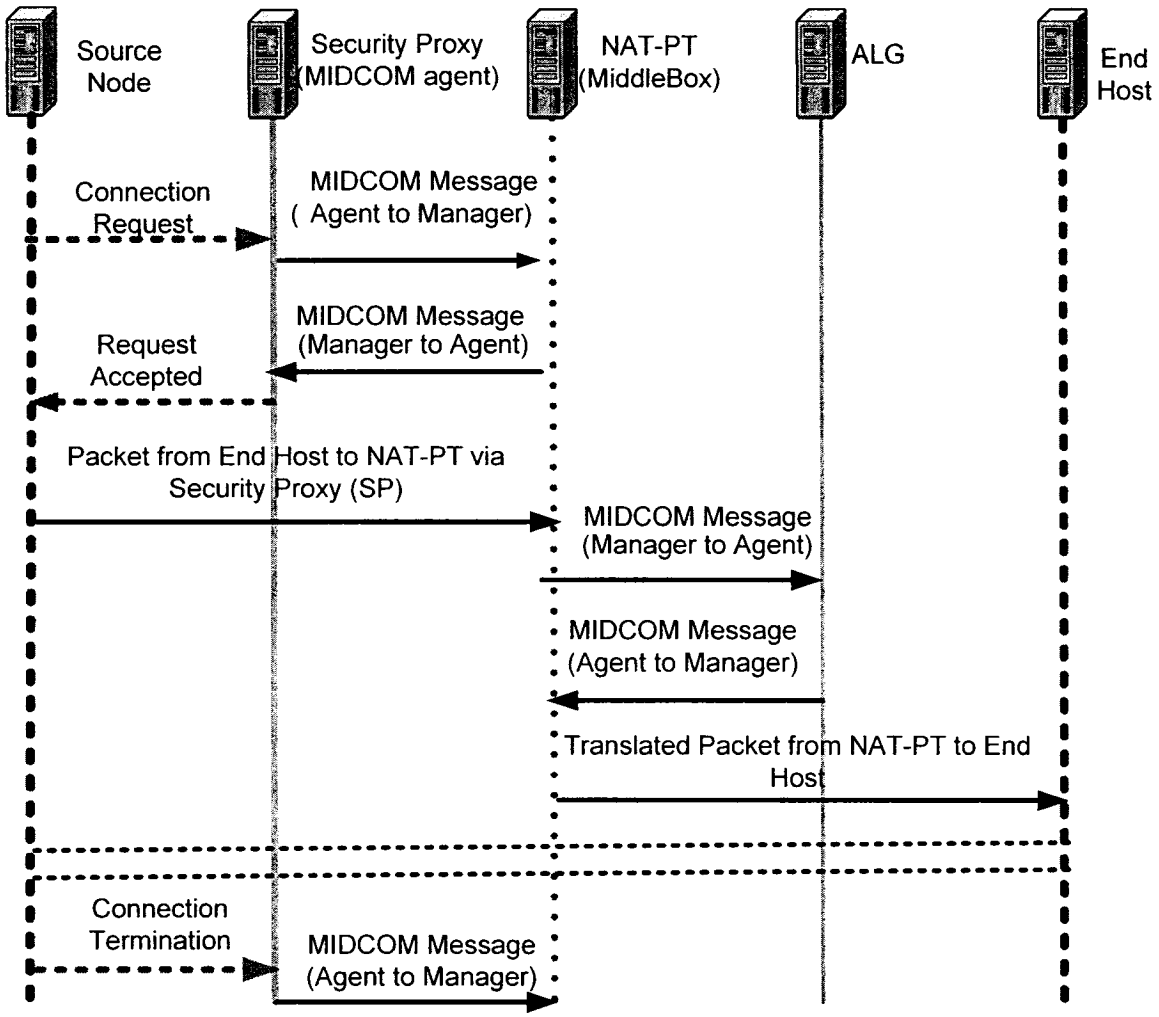

Figure 4.3 Message Flow through MiddleBox

Security proxy will manage the firewalls. Given that security proxy is a MIDCOM agent, it should be able to communicate reliably with MiddleBox. It will update the firewall access control list. It will ensure the scalability of NAT-PT in the case of mNAT-PT.

This method has the advantage that it will always work as long as both clients have connectivity to the server. For simplicity, we can use two proxy servers—one in IPv4 network and another in IPv6 network. If it is

the case of single proxy server, it must have two network protocol stacks –IPv4 and IPv6. As the one side of proxy server supports IPv4, any client B from IPv4 network will be able to establish secured connection between the client and the proxy using existing network security protocols. It is also true for any IPv6 node. Its obvious disadvantages are that it consumes the server's processing power and network bandwidth unnecessarily, and communication latency between the two clients is likely to be increased even if the server is well connected. The TURN protocol [TURN] defines an implementation method relaying in a relatively secure fashion.

4.8.3 Application Layer Gateways (ALGs)

NAT-PT requires the services of one or more ALG(s). It is the responsibility of the ALG manager of the NAT-PT to find the appropriate ALG. The ALG manager will check the source and destination port of the packet to do it.

Examples of several Application Layer Gateways (ALGs) are DNS-ALG, FTP-ALG, SIP-ALG, etc. To relieve the NAT-PT from the burden of heavy workload any ALG can be implemented as a MIDCOM agent. However, those agents will be out of path agents. So any communication between MIDCOM agents and MiddleBox should start from MiddleBox (NAT-PT). MiddleBox will use MIDCOM protocol to communicate with any ALG. The

security of communication will be ensured by MIDCOM protocol itself. MiddleBox will use PDP to find the identification of the MIDCOM agent, the residence of ALG.

4.8.4 Protocol Translator

The Protocol Translator will translate the IP header from IPv4 to IPv6 and vice versa. IP headers are not the same in IPv4 and IPv6. Any packet traversing through NAT-PT requires the services of Protocol Translator, and it is mandatory. So the Protocol Translator should be implemented as one module of NAT-PT instead of implementing it as an MIDCOM agent. Both the ALG Manger and Protocol Translator are the parts of the NAT-PT, so after reading the packet, NAT-PT will communicate with appropriate ALG, and then invoke the Protocol Translator. In that case the IP header translation and payload translation may be done concurrently.

4.9 Summary

This chapter describes MIDCOM architecture, protocol for MIDCOM. According to the networking terminology, Middle Box is a device, which requires application intelligence for its operation; and MIDCOM is a framework middle box. RFC 3033 defines the MIDCOM framework. MIDCOM framework consists of Middle Box and trusted third parties' application. These third party applications will assist Middle Box in performing its operations. NAT-PT will reside in a Middle Box and work

as MIDCOM application. This chapter also explains how to develop MIDCOM framework for NAT-PT. For adopting NAT-PT with MIDCOM framework, we have to decompose NAT-PT system into NAT-PT main module, Security Module/Security Proxy, Protocol Translator, and Application Layer Gateways. This chapter contains the detailed description of how all the modules will work within MIDCOM framework.

We have chosen SNMP as MIDCOM protocol for NAT-PT model. Hence, the next chapter will describe SNMP protocol, and then our proposed NAT-PT model.

Chapter 5 : NAT-PT with SNMP Framework

5.1SNMP Framework

The Simple Network Management Protocol (SNMP) is an application layer protocol that facilitates the exchange of management information between two types of SNMP entities—SNMP manager and SNMP agents. So, an SNMP supported network must have two types of applications—SNMP managers and SNMP agents. Figure 5.1 describes SNMP managed Network Architecture.

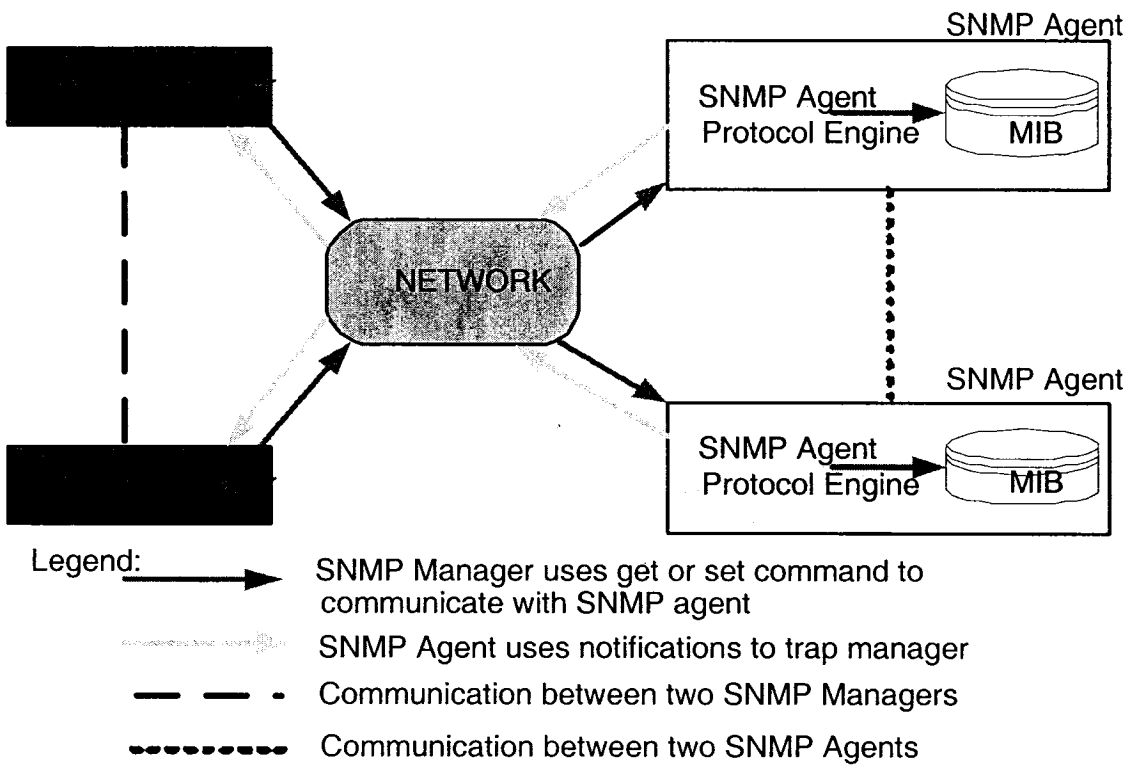

Figure 5.1 SNMP Managed Network Architecture

Agents are the entities that interface to the actual device being managed. An agent is actually software and it resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP.

Network Management System (NMS) executes applications called SNMP manager that monitor and control managed devices.

A managed device is a network node that contains an SNMP agent, and resides on a managed network. Network components such as Bridges, Hubs, Routers, printer, network workstations or network servers, etc., are examples of managed devices. These managed devices contain managed objects. These managed objects might be hardware configuration parameters, performance statistics, even network packet payload and so on, that directly relate to the current operation of the device or the system in question. These managed objects are arranged in what is known as a virtual information database, called a Management Information Base (MIB). SNMP allows managers and agents to communicate for the purpose of accessing these objects.

5.2 Functional Analysis of NAT-PT

The functional analysis of the NAT-PT will be helpful for adopting NAT-PT into SNMP framework. For the purpose of completeness and better

understanding, I added the complete description of the functional analysis of the NAT-PT (according to RFC 2766) in appendix **C**.

5.3 Distributed NAT-PT with SNMP

Our ultimate goal is to put NAT-PT in a MiddleBox; hence we will keep the same decomposition of NAT-PT as described in Chapter 4.

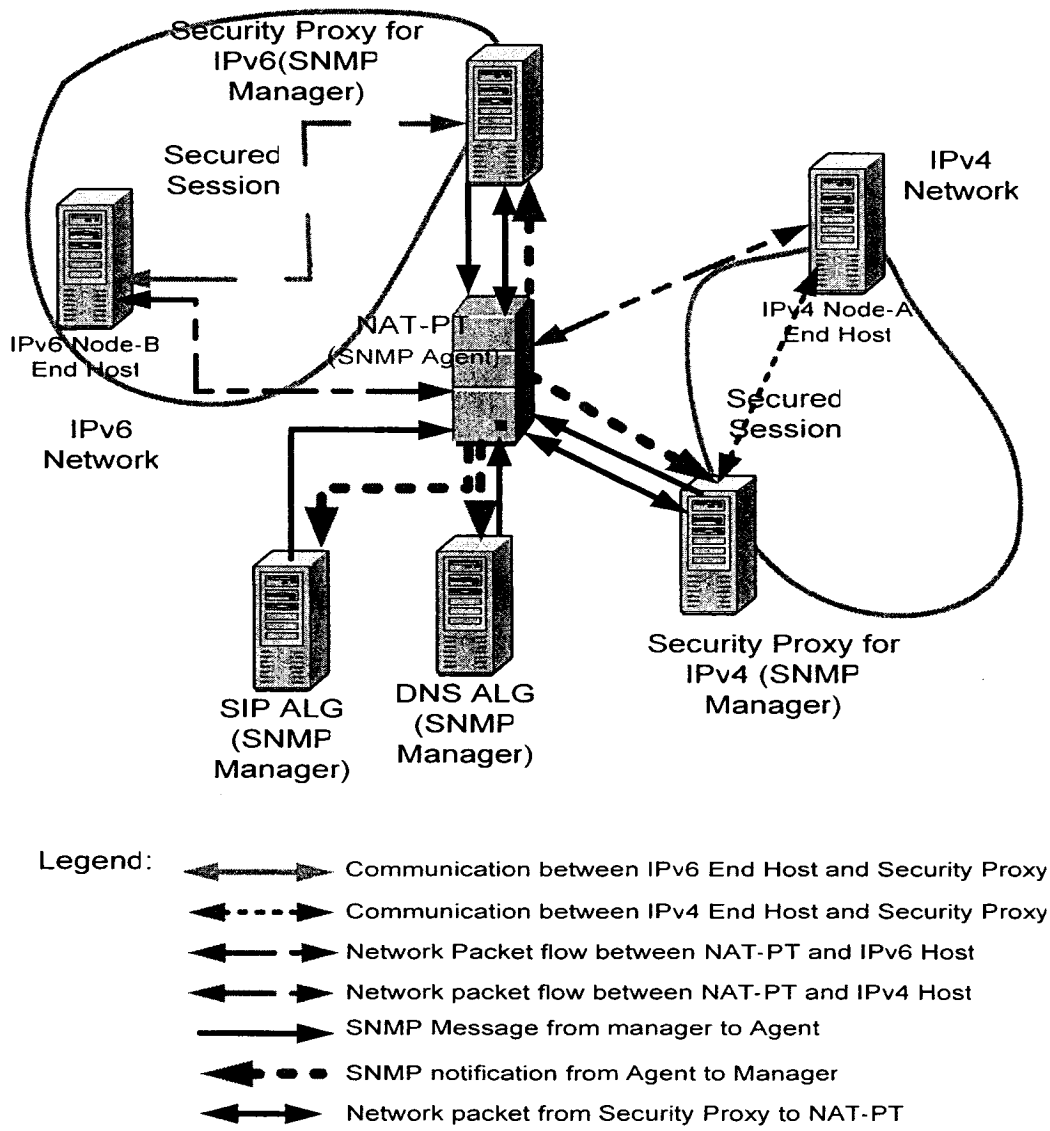

Figure 5.2 Distributed NAT-PT with SNMP

That means the decomposition will be as below:

- NAT-PT main module
- Security Module/Security Proxy
- Protocol Translator
- Application Layer Gateways

5.4 How does the Model work?

Figure 5.2 describes the proposed model of distributed NAT-PT with SNMP. All the network packets, that need to be translated, must traverse through NAT-PT, which will read the packets and maintain information required for other modules. Therefore, according to the proposed model only the NAT-PT will work as SNMP agent; all other modules are SNMP managers. It is not a contradiction to our ultimate goal--MIDCOM protocol. Although NAT-PT is an SNMP agent, however any packet traversing through NAT-PT must pass through security proxy, which is an SNMP manager. Security Proxy will read just IP header and security header of a packet. The following sections describe the scenario of packet flow. End hosts will establish secured communication with Security Proxy that will provide NAT-PT a clue to generate decryption key to decrypt the encrypted message for a particular session. Security Proxy will use SNMP messaging system to send the clue information to NAT-PT. This clue information depends upon the algorithm used for encryption and decryption of the message and the key management policy. The

following subsections 5.4.1 and 5.4.2 describe the functionalities of NAT-PT and the NAT-PT with SNMP framework.

5.4.1 Functional Analysis of NAT-PT with SNMP Framework

We will define several objects in NAT-PT MIB, which will make NAT-PT working under the SNMP framework. The following are the essential objects for our NAT-PT framework—

NAT-PT Address Pool Table (natptIPv4AddressPool), NAT-PT Static mapping Table (natptStatMapping), NAT-PT Mapping Table (natptMapping), NAT-PT Session Info Table (natptSessionInfo), NAT-PT Received Packet Info table (natptRecievedPacketInfo), NAT-PT Packet for Sending (natptPacketForSending), NAT-PT Notification Object (natptNotification). We will describe these objects in detail in NAT-PT MIB chapter.

Network packet may be classified depending on various criteria. However, NAT-PT functionalities will not be changed based on each criterion.

5.4.1.1 Session Originating in IPv4 Network

5.4.1.1.1 Security Proxy for IPv4 Network (SPv4)

The figure 5.3 describes the data flow for a session originated in IPv4 network. SPv4 will have a list of IPv4 addresses, which are the pool addresses of NAT-PT. The establishment of any new session will require

one free IPv4 address in the address pool. SPv4 will always maintain an updated address list. If mNAT-PTs are deployed, SPv4 will maintain another list for load balancing.

The end host of IPv4 network will establish secured session with Security Proxy before starting communication through NAT-PT. Then the Security Proxy/Server will do the following—

After receiving a session initiating packet from any host of IPv4 network, SPv4 will analyze the packet to get the IP header, security header, and the address of the target host of the packet. Then it will authenticate the IPv4 host. If the packet passes security screening, it will check its address list to find either there is a free address in the list or not. If a free address is available in the list, SPV4 will send an SNMP message to the NAT-PT to verify whether the address of the target host is in the mapping table or not, and to block one IPv4 pool address for the forthcoming session. If the target host address is in the mapping table then it will store session information for the session, and send an acknowledgement to the sender of the session initiating node. For each session there will be a session ID, and the SPv4 will use pre-defined algorithm to generate session ID using source address, source port, and the security info for the session. The session initiating packet should contain the security parameters, and the address of the target host address. After sending acknowledgement, it will construct an SNMP message with the Session

ID, decryption key for the session, the source address, and the address of the target host. It will send the SNMP message to the NAT-PT using setRequest command to set values for the parameters of newly created session. At the same time the SPV4 will send a request to SPV6 to forward security info for the IPv6 end host to the NAT-PT. The request message is an SNMP message containing the session ID, target IPv6 host, and other relevant session parameters.

If there no is free pool address, then the session cannot continue. If multiple NAT-PTs (mNAT-PT) exist, then security proxy can locate another NAT-PT with a free pool address.

For any subsequent packet from IPv4 host for which the session has been created, it will check the destination address of the packet. If the destination address is in either NAT-PT Static Mapping table or NAT-PT Mapping table then proceed, otherwise drop the packet.

At the end of the session SPv4 will receive a session ending information and it will construct an SNMP message with the Session ID to terminate the session and return the IPv4 pool address to the address pool used by NAT-PT. It will also update its own address list. SPv4 will use timer for making timeout of idle sessions.

ICMP packet does not require be checked whether it is session initiating or session ending or it is in the middle of the session. SPv4 will just forward the ICMP packet to the NAT-PT.

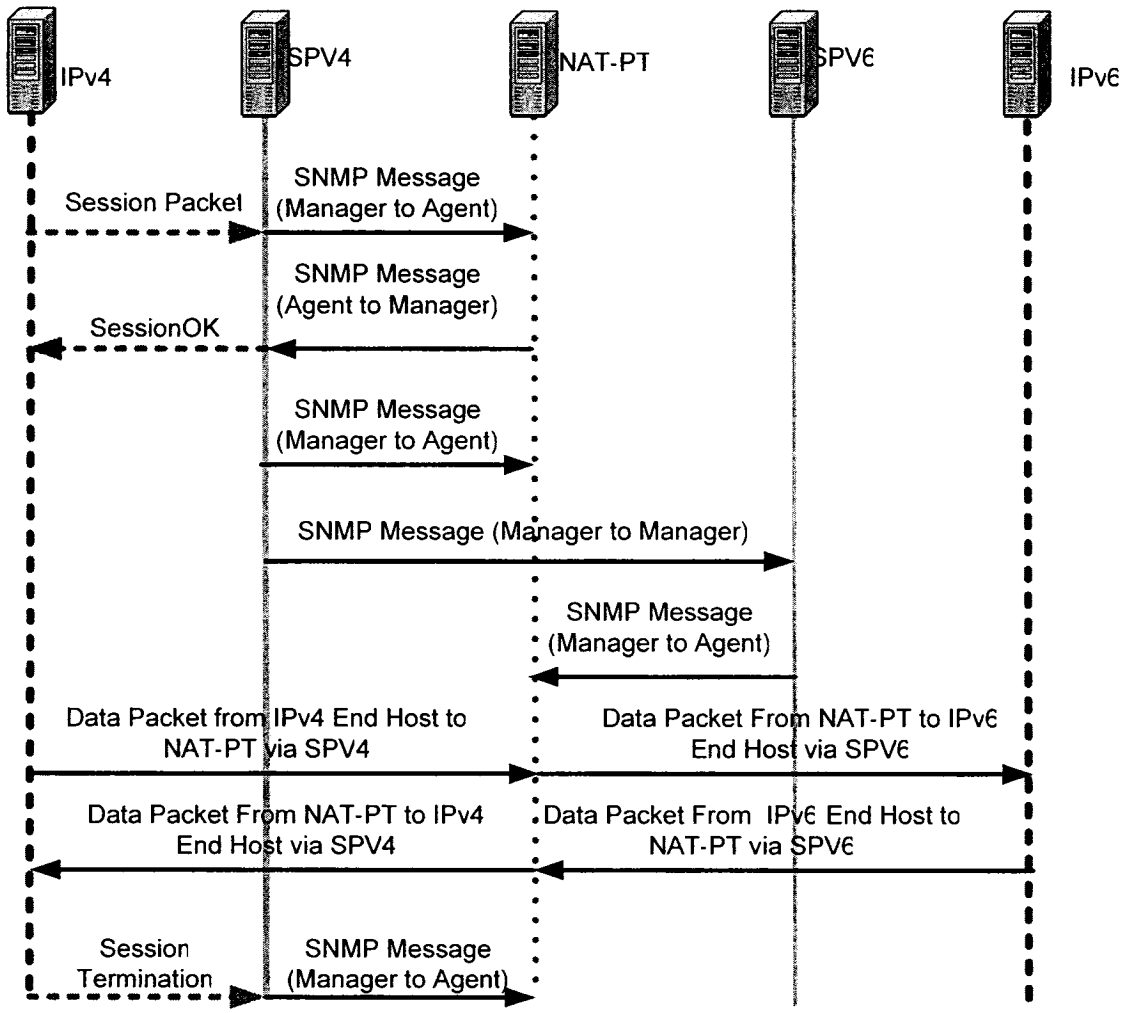

Figure 5.3 Message Flow Diagram of a session originated in IPv4

The source and destination port will identify the ALG required for the packet. If there is any ALG for that packet, security proxy will construct SNMP message (used between two SNMP managers) to inform the corresponding ALG to translate the payload of the packet.

5.4.1.1.2 NAT-PT

NAT-PT module will work as SNMP agent. So it will send notification to SNMP managers, take necessary actions according to the message from the SNMP managers. NAT-PT will also read the packet and analyze the packet as described below—

The NAT-PT will read IP header to get the source address, destination address, source port, and destination port of the packet. Then it will create a session ID with the source address, source port and the security info of the packet. It will use the same algorithm as used by the Security Proxy.

The NAT-PT will get the security information from NAT-PT Session info table using Session ID received in the previous step. It will also store the checksums value of the transport layer protocol.

Then ALG manager of the NAT-PT will check whether there is any ALG for the packet or not. The source port or destination port of the packet will identify an ALG. If the packet requires the services of ALG, the NAT-PT will read, decrypt the payload of the packet, and update the [NAT-PT Received Packet Info] table. It will overwrite the previous information stored in the MIB table against the same session ID.

If the NAT-PT does not receive setRequest or getRequest message from SNMP managers regarding the current packet, it will construct an SNMP notification message and send the notification to SNMP managers (ALG and other modules). The notification message will include the packet ID and time stamp, and Transport layer protocol, SYN Flag, etc. The contents of the message will depend on the receiver to whom the notification is being sent.

Afterward the NAT-PT will invoke the Protocol Translator module, which will translate the IP header of the packet. The Protocol Translator may be collocated with the NAT-PT or may be in a different host directly connected to the NAT-PT host. If the Protocol Translator is not collocated with the NAT-PT, then an Inter Process Communication method may be used for communication between the NAT-PT and the Protocol Translator.

Protocol Translator will translate the IP header from IPv4 to IPv6 and vice versa. The NAT-PT will pass IPv4 header to the Protocol Translator that will get the mapping address from the mapping table and replace the IPv4 destination address with the IPv6 address. It will then create IPv6 source address concatenating PREFIX with IPv4 source address. Then it translates the IPv4 header into IPv6 header according to the protocol specifications. For TCP packet Protocol Translator recalculates TCP

checksums and for UDP it recalculates UDP checksums. After the translation of IP Header, the Protocol Translator will return the translated header to the NAT-PT module.

5.4.1.1.3 ALG

Figure 5.4 describes the flow diagram for packets requiring the service of an ALG. NAT-PT will determine whether a particular packet will require the service of an ALG or not. If the received packet requires the services of ALG(s), NAT-PT will update the payload field of the [NAT-PT Received

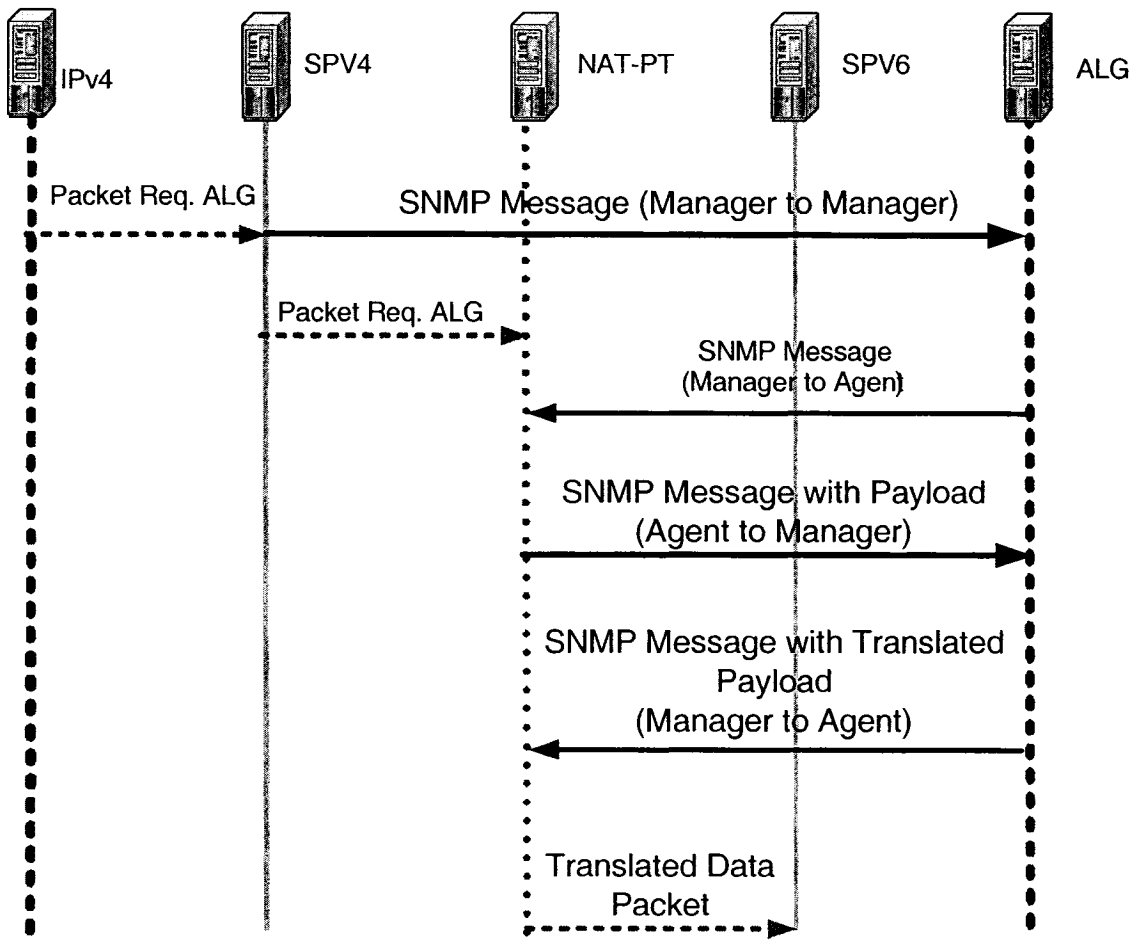

Figure 5.4 Flow Diagram of Packets required the Service of an ALG

Packet table] and send notification to the concerned ALG. After receiving the notification ALG will save the Packet ID and do the following—

First ALG will construct an SNMP Message, which will consist of Packet ID received from NAT-PT, and payload object ID of MIB. It will issue getRequest command to get the payload of the IPv4 packet. Then it will translate the payload according to the protocol specification.

ALG will convert IPv4 Address (if any) of the payload into IPv6 address just concatenating PREFIX before the IPv4 address as PREFIX::IPv4_Address. After the translation of the payload, the ALG will issue setRequest command to update the payload field of the [NAT-PT Packet For Sending] table.

5.4.1.1.4 Packet Forwarding

Thereafter, the NAT-PT will construct the IPv6 message, apply IPSec to the packet and then forward the packet to the Security Proxy for IPv6 that will route the packet to the destination address. The source address of the packet will be PREFIX::IPv4_Address and the destination address will be an IPv6 address obtained from the mapping table. In addition, NAT-PT can terminate a session, and return the pool address to the address pool. The termination of a session will depend on the idle time of a session, or the instruction from Security Proxy.

5.4.1.2 Session originating in IPv6 network

5.4.1.2.1 Security Proxy for IPv6 Network (SPV6)

The figure 5.5 describes data flow datagram for a session originating in IPv6 network. The end host of IPv6 network will establish a secured session before starting any communication through NAT-PT. Thereafter, the security proxy will do the following:

Like SPv4, SPv6 will also have a list of IPv4 addresses, which are the pool addresses of NAT-PT. The establishment of any new session will require one free IPv4 address in the address pool. SPv6 will always maintain an updated address list. If mNAT-PTs are deployed, SPv6 will maintain session list for load balancing.

After receiving a session initiating packet from any host of IPv6 network, SPv6 will analyze the packet to get the IP header, security header, and the address of the target host of the packet. The destination address of the packet may be in the form of PREFIX::IPv4_Address or IPv6 Address. SPv6 will authenticate the IPv6 host. If the source of the packet is genuine, it will check its address list to find either there is a free address in the list or not. If a free address in the address list is available, then SPV6 will send an SNMP message to the NAT-PT to block one IPv4 pool address for the forthcoming session. The next step is to create a Session ID. Using a defined algorithm, SPV6 will create a session ID with the

source address, source port, and security info of the packet. SPV6 will then store session information for the created session, and send an acknowledgement to the sender of the session initiating node. The session initiating packet should contain the security parameters, and the address of the target host address. After sending acknowledgement, the SPV6 will construct an SNMP message with the Session ID, decryption key for the session, the source address, and the address of the target host. It will now send the constructed SNMP message to the NAT-PT using setRequest command to set values for the parameters of newly created session. At the same time the SPV6 will send SNMP messages to SPV4 and other SNMP managers to inform about the newly created session. The request message is an SNMP message (Message format Manager to Manager) containing the session ID, target IPv6 host, and other relevant session parameters.

If there is no free pool address available, then the session cannot continue. Alternatively, if multiple NAT-PTs (mNAT-PT) exist, then security proxy can locate another NAT-PT with a free pool address.

For any subsequent packet from IPv6 host for which the session has been created, SPV6 will check the source address of the packet. The source address may or may not exist in the mapping table. If the source address does not exist in the mapping table, it will create a mapping

between the IPv6 source address and the previously blocked IPv4 pool address. It will issue an SNMP message to set the values for the created mapping. After the creation of the mapping, SPv6 will reset the flag of the blocked address as "In Use".

The source and destination ports of a network packet will identify the ALG required for the packet. If there is ALG for that packet, the security proxy SPV6 will construct a message (used between two SNMP managers) to inform the corresponding ALG manager to translate the payload of the packet.

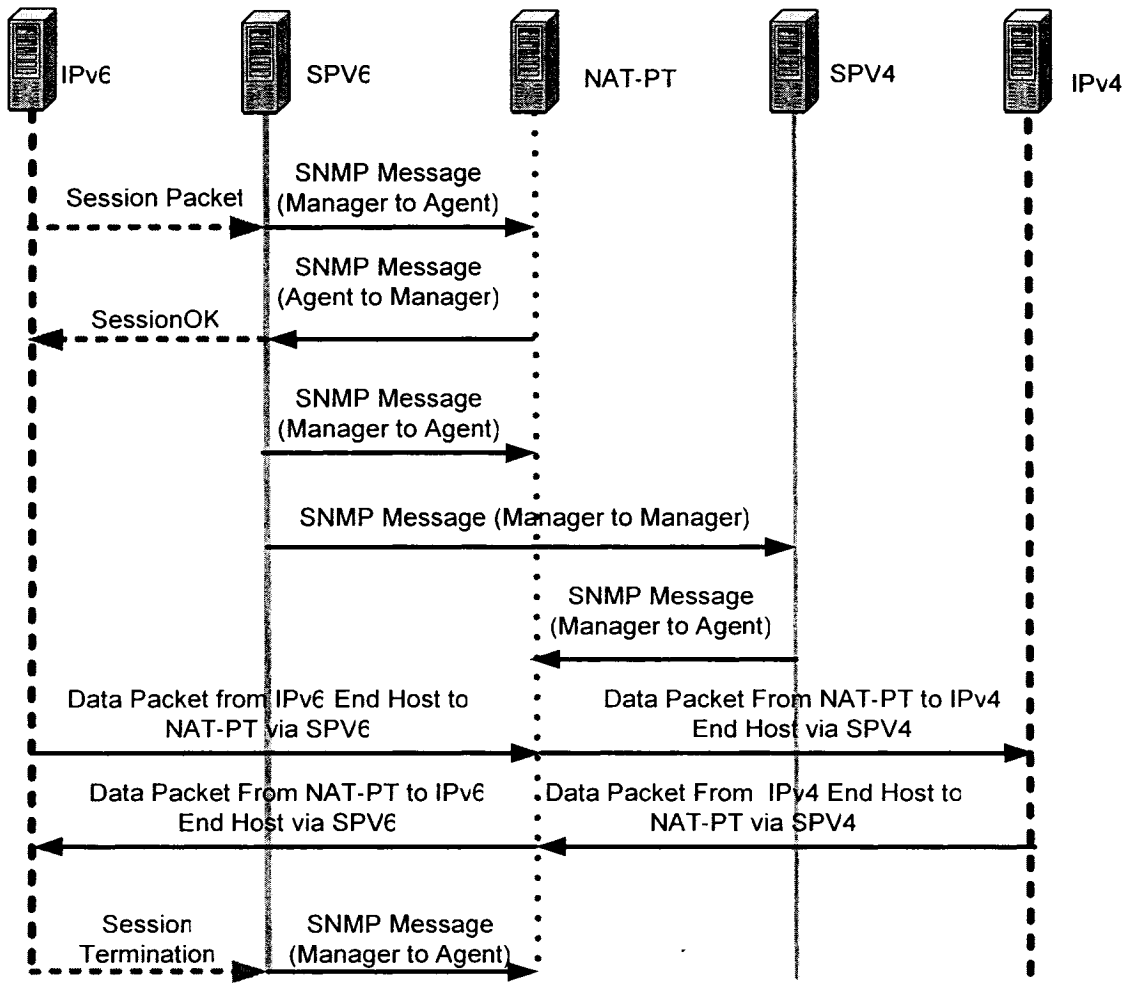

Figure 5.5 Message Flow Diagram of a session originating in IPv6 host

At the end of the session, SPv6 will receive session ending information and construct an SNMP message with the Session ID to terminate the session. It will then return the IPv4 pool address to the address pool used by NAT-PT. It will also update its own address list. SPv6 will use timer for making timeout of idle sessions.

ICMP packet does not need to be checked whether it is session initiating or session ending or the packet is in the middle of the session. The SPV6 will just forward the ICMP packets to NAT-PT.

5.4.1.2.2 NAT-PT

After receiving a message from SPv6, NAT-PT will read the packet IP Header, and create session ID using the same method as used by SPv6. It will get the security information from [NAT-PT Session info] table using the created session ID.

The ALG Manager will check whether there is any ALG for the packet or not. The source port or destination port of the packet will give the information about ALG. If the packet requires the services of an ALG, NAT-PT will read, decrypt the payload of the packet, and update the [NAT-PT Received Packet Info] table.

NAT-PT will check the [NAT-PT Received Packet Info] table to find information for an existing packet with the same session ID. If there is any, the NAT-PT will overwrite the previous one and update the values of different fields of that table.

IPv6 source address may or may not exist in the mapping table. If it exists, then the source address may be in the static mapping table or in

the dynamic mapping table. If the IPv6 source address exists in the [NAT-PT Static Mapping] table, the packet is either a DNS response or a DNS query.

In the case of a DNS response, DNS ALG will require an IPv4 address from the address pool to create mapping of the resolved IPv6 address and an IPv4 pool address. The NAT-PT will then issue an SNMP notification for the DNS-ALG. The notification message will include an IPv4 address from the IPv4 address pool. The IPv4 pool address will be one that has been blocked before by the SPv4 at the time of establishing secured session.

In the case of DNS query, there should be a mapping between the source address and an IPv4 pool address in the mapping table. If the IPv6 source address exists in the [NAT-PT Mapping] table, the NAT-PT will not require creating a new mapping. It will just translate the packet as usual. Otherwise the NAT-PT will create a mapping between an IPv4 address, which has been blocked by Security Proxy, from the pool of IPv4 addresses, and the source IPv6 address; and write that entry to the [NAT-PT Address Mapping] table. By this if the NAT-PT does not receive any getRequest message from DNS ALG, it will send notification to the DNS ALG.

Then NAT-PT will store mapping information into the mapping table and wait for the response from the SNMP manager, i.e., from the DNS-ALG. After receiving the response from the DNS-ALG, it will send the payload to the DNS-ALG using SNMP message format.

The NAT-PT will invoke the Protocol Translator module, which will replace the IPv6 source address with an IPv4 address. It creates IPv4 destination address peeling off the PREFIX from the IPv6 destination address of the packet, and replacing the IPv6 destination address with the newly created IPv4 address. It will recalculate transport protocol checksums. The Protocol Translator converts the IPv6 Header to IPv4 Header according to the protocol specifications, and it returns the translated IP Header to the NAT-PT module.

5.4.1.2.3 ALG

Figure 5.6 describes the flow diagram for packets requiring the service of an ALG. ALG will receive an SNMP notification message from either SPV6 or NAT-PT. After receiving the notification, the ALG will save Session ID and Packet ID, IPv4 address (if any) and other information. Figure 5.6 describes the data flow diagram for packets that require the service of an ALG.

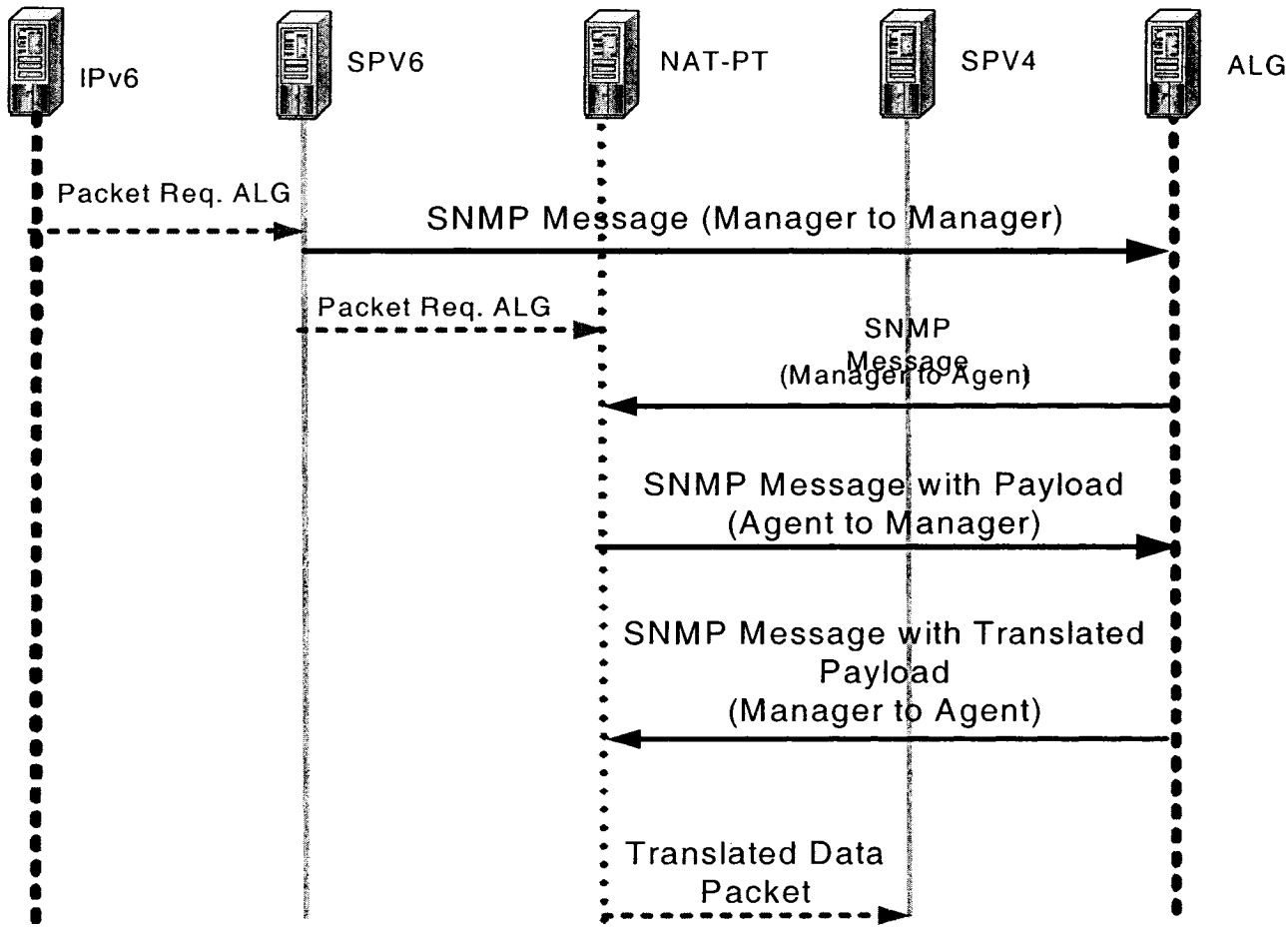

Figure 5.6 Flow Diagram of Packets required the Service of an ALG

Then the ALG will issue getRequest command to get the payload of the IPv6 packet. It will translate the payload according to the protocol specification, and replace IPv6 Address (if any) of the payload with IPv4 pool address. It will issue setRequest command to update the payload field of the [NAT-PT Packet For Sending] table.

5.4.1.2.4 Packet Forwarding

At this stage, the NAT-PT will construct IPv4 message, and apply IPSec to the packet and forward the packet to the security proxy, which will send the packet to the destination address. The source address of the packet will be IPv4 pool address and the destination address will be an IPv4 address of the end host. SPv4 or SPv6 or NAT-PT will be able to terminate any session depending on the value of SYN flag or idle time of the session. After terminating a session, NAT-PT will return the IPv4 address to the address pool.

5.5 Why two Proxies?

The model with Security Proxy will resolve scalability problem of NAT-PT in two ways. SNMP architecture allows many-to-many communications between SNMP managers and agents. Given the fact, to handle large volume of network traffic there may be situations where multiple NAT-PT may be required to be deployed. If it is the case, the Security Proxy will control incoming data traffic and balance the load among NAT-PT servers by directing sessions to the most suitable server available. As a result,

traffic will be distributed among NAT-PT servers uniformly. On the other hand, Security Proxy is an SNMP manager, and the NAT-PT is an SNMP agent; so Security Proxy will be equipped with facilities to monitor the status of all NAT-PT servers. Hence, if a NAT-PT fails during a session, the session initiating Security Proxy will be able to transfer the same session to another NAT-PT, and the session can continue without interruption. Figure 5.7 will explain the facts clearly.

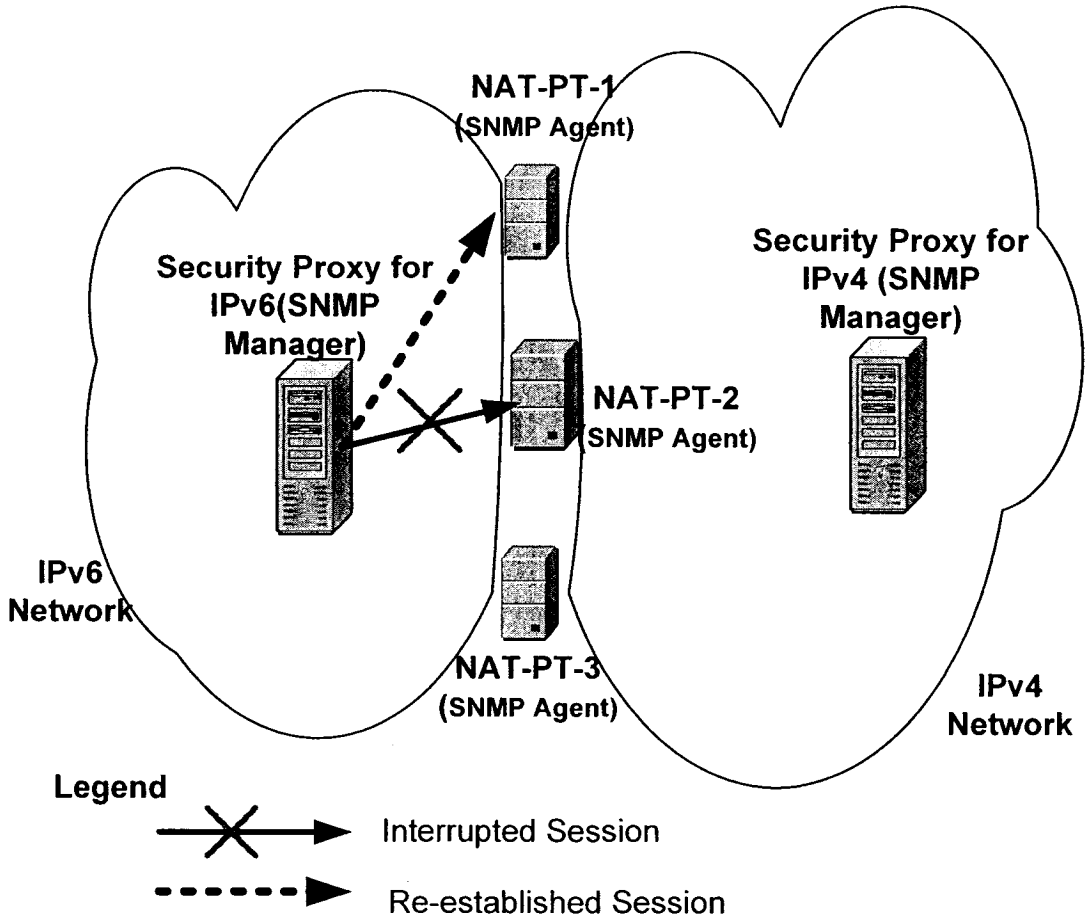

Figure 5.7 Environment with Multiple NAT-PTs

Suppose a session has been created with NAT-PT-2, and at some point of the session and for some reasons, the NAT-PT-2 is no longer able to translate network packets. Hence, the communication between IPv4 and IPv6 end hosts must stop. The deployment of multiple NAT-PTs will avoid the problem. The Security Proxy has all session information, so it can divert the same session through another NAT-PT (say NAT-PT-1) and communication can continue. It is obvious that the system with two proxies will be more flexible and fault tolerant than the system with a single proxy is.

5.6 Summary

We used SNMP as the means of communication protocol between MIDCOM agent and MIDCOM manager. So this chapter first describes the architecture of the SNMP comprehensively. Then the several sections of this chapter illustrated our proposed NAT-PT model in detail. After the sections of the proposed model, in some sections of this chapter we looked at how the proposed model would work. With the help of packet flow diagrams, those subsections explain network packet flow through the proposed model. The last section of this chapter gives the reasoning behind the using of two security proxies in the proposed model.

We proposed a modified NAT-PT model, and the next step is to validate the model. So Chapter 6 outlines the validation procedure of the model.

Chapter 6: PROMELA Model for NAT-PT and Validation

6.1 Introduction

NAT-PT translates the IP Header of network packets; therefore, it provides services to the network layer. To realize the service of NAT-PT we developed a model, which was described in chapter 5. In fact the model we developed describes a set of consistent procedures and rules. Those procedures and rules specify the behavior of the NAT-PT system. From that model it is possible to develop a finite state machine, which will be sufficient for description, but it is not convenient to work with that model. Here we used the formal language PROMELA to describe those procedural rules.

This chapter will present the formalization of NAT-PT specification to build a PROMELA model and to verify the correctness. Our goal of this verification is to prove the completeness and logical consistency of the model.

To avoid the complexity we deliberately abstract our validation process from other issues of protocol design, such as message format, message encoding, decoding, etc. The validation model defines the interactions of

processes that are required for a complete NAT-PT working environment. Like any standard validation model it does not resolve implementation details.

6.2 PROMELA

PROMELA defines a validation model in terms of three specific types of objects—*Processes, Channels, and Variables*. Processes are by definition global objects whereas variables and channels can be either global or local to a process. Channels are the communication mechanism between PROMELA processes. Like any other programming language, variables store system information either locally to a process or globally to all the processes for a system.

6.3 PROMELA for NAT-PT Working Environment

6.3.1 Processes

We have described the NAT-PT working environment with six PROMELA processes. These processes are IPv4 host (process v4), IPv6 host (process v6), security proxy for IPv4 network (process SPV4), security proxy for IPv6 network (process SPV6), NAT-PT (process natpt), Application Layer Gateway (process alg) processes. These six processes represent the formal program of procedure rules for the abstract model of NAT-PT.

6.3.2 Channels

Channels are the communication mechanism among PROMELA processes. We have not classified channels depending on the type of messages; rather we have defined two types of channels-one for receiving and another for sending messages. For example the process natpt will receive any message from process SPV4 through a fixed channel from_SPV4, and it sends messages to SPV4 using channel to_SPV4. Channels are passed as parameters of the processes. The following are the channel we have used in our program:

To_SPV4: V4 node uses this channel to send messages to SPV4, and SPV4 receives messages from this channel.

From_SPV4: V4 receives messages from SPV4, and SPV4 sends messages to V4 node through this channel.

To_natptV4: SPV4 sends messages to NAT-PT and NAT-PT receives messages from SPV4 through this channel.

From_natptV4: SPV4 receives messages from NAT-PT and NAT-PT sends messages to SPV4 through this channel.

To_alg: NAT-PT sends messages to ALG, and ALG receives messages from NAT-PT using this channel.

To_natptV6: SPV6 uses this channel to send messages to NAT-PT and NAT-PT receives messages from SPV6 through this channel.

From_natptV6: SPV6 receives messages from NAT-PT and NAT-PT sends messages to SPV6 through this channel.

To_SPV6: V6 node sends messages to SPV6 and SPV6 receives messages from V6 node through this channel.

From_SPV6: V6 node receives messages from SPV6 and SPV6 sends messages to V6 node using this channel.

6.3.3 Variables

PROMELA variables represent data that can be global, and local. For our validation purpose we defined several global variables and many more local variables. Our validation contains two types of global variables—constant and structure. Structure represents the message formats and most of the constants are flag values. As we want this model to be as simple as possible we used IP address as integer, record type of DNS message just as byte. For the DNS type, 'A' means AAAA or A6 and 'a' means A record.

Although, we want this model to be as simple as possible, yet it should be sufficiently powerful to represent all types of coordination problems that can occur in the NAT-PT working environment. For this reason I tried to simulate the exact situation, and I defined the following global structure for IP Header and Message format:

IP_Header: It is a global data structure, which represents the IP Header of network packet. Protocol Translator of NAT-PT translates the IP

Header from IPv4 to IPv6 and vice versa. So IP Header is important for the validation.

Message: It is a global data structure, which represents the payload part of network packet. We made it generic so that it can represent any type of payload—such as payload for DNS packet, and payload for ICMP message.

Session: It is a data structure to represent the payload of a session info exchange packet.

SNMP_Message: It represents the SNMP message format. For commands like setting request, getting request, or trap, SNMP entities exchange message with message format.

SNMP_ALG: It is also global data structure used as SNMP message format. We separated it from the other just for simplicity. Other than the global data structure we define several data structures local to NAT-PT, SPV4 and SPV6 processes. NAT-PT works as an SNMP agent, so it will maintain MIB. The data structures actually simulate Management Information Base (MIB). These are as described below—

ipheader_natpt: This data structure stores IP Header information, and NAT-PT uses that information for the purpose of translation.

ipv4_ipv6_mapping: It holds the mapping between an IPv6 address and an IPv4 pool address. Each entry in this structure is just for one session.

session_info: It is also structure type data structure used for storing session info of a session.

payload_natpt: Like ipheader_natpt, its purpose is to store payload, which requires the service of an ALG.

pool_addr: It represents the address pool table of NAT-PT.

According to the proposed design, the security proxy works as an SNMP manager, so it will have control over SNMP agent, and it can issue "set_Request" command to get information from NAT-PT. However, the issuing of commands frequently might be expensive for a system. To make the system efficient, security proxy uses its own record (database) and updates time to time. To present the record of the security proxy I defined several structured type variables, which should be local to the security proxy. As the structure type object cannot be defined locally in PROMELA I placed them globally. These are given below:

Session_Object: It keeps the session information for a particular session.

pool_addr: It contains the same information as the NAT-PT pool_addr does. Security Proxy is not supposed to initiate a session if there is no free pool address in the pool address table.

6.4 Behavior Model

Figures 6.1 and 6.2 describe the state diagram for NAT-PT. The flow of control levels for all running processes, the specification of all values of for local and global variables, and the contents of message channels

define a state of a model. Again the ordered set of states represents the behavior of a validation model. The PROMELA model for NAT-PT comprises of several states. These are the essential part of the model.

6.4.1 NAT-PT

6.4.1.1 Waiting State

At the very beginning it will be waiting for a network packet. The network packet may arrive at the NAT-PT from any direction. The NAT-PT starts its activities upon receiving any packet from SPV4, SPV6 or ALG. If NAT-PT receives a message from SPV4, the message may be categorically of two types—one SNMP message originated in SPV4 and the other network packet originated in IPv4 side. The next flow of control will depend on the type of received message. If it is an SNMP message, the control will move to the SNMP_Process state, and so on. NAT-PT is symmetric with its two security proxies. So when the NAT-PT receives messages from SPV6, the flow of control will be exactly same as when it receives messages from SPV4. The communication between NAT-PT and ALG will be only through an SNMP message. So when NAT-PT receives any message from ALG, the control will move to the ALG_SNMP_Process state. However, at the very beginning NAT-PT is not supposed to receive any packet from ALG. If it does not receive any message it will not do anything.

6.4.1.2 SPV4 SNMP Message Process State

Upon receipt of an SNMP message from SPV4, the control will jump to this state.

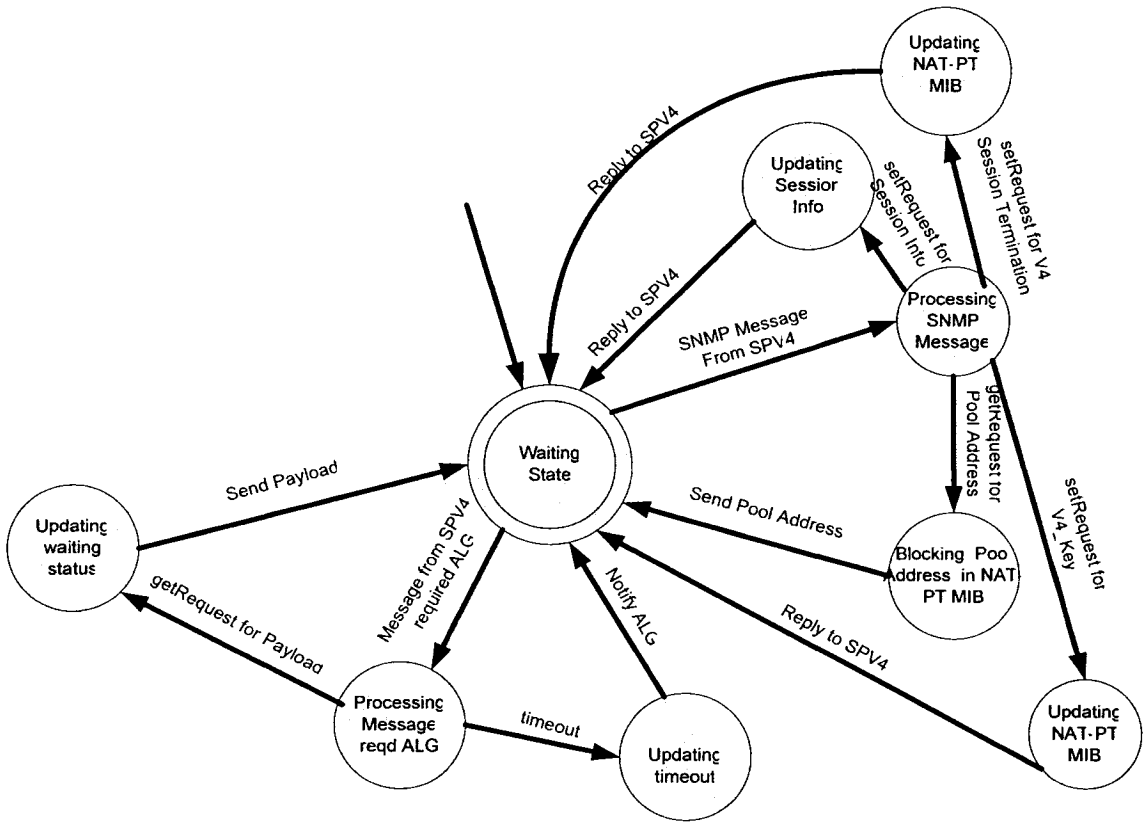

Figure 6.1 State Diagram for NAT-PT (Part 1)[1]

The processing of the SNMP message will depend upon the objective of the message. The objective may be either to initiate a session, terminate a session, or to send V4 key for a session initiated by IPv6 side. If the objective is to initiate a session, the control will jump to V4_Session_Initiate state. Similarly for the other cases the next states may be either V4_Session_Terminate or V4_Key_Update. In any case the

[1] *Figure 6.1 describes the state diagram for NAT-PT when it receives message from SPV4 and the message is either a network packet requiring the service of an ALG, or an SNMP message.*

NAT-PT will update its MIB tables. The NAT-PT will inform the SNMP manager (SPV4) of the processing result, and the control will return to the initial state.

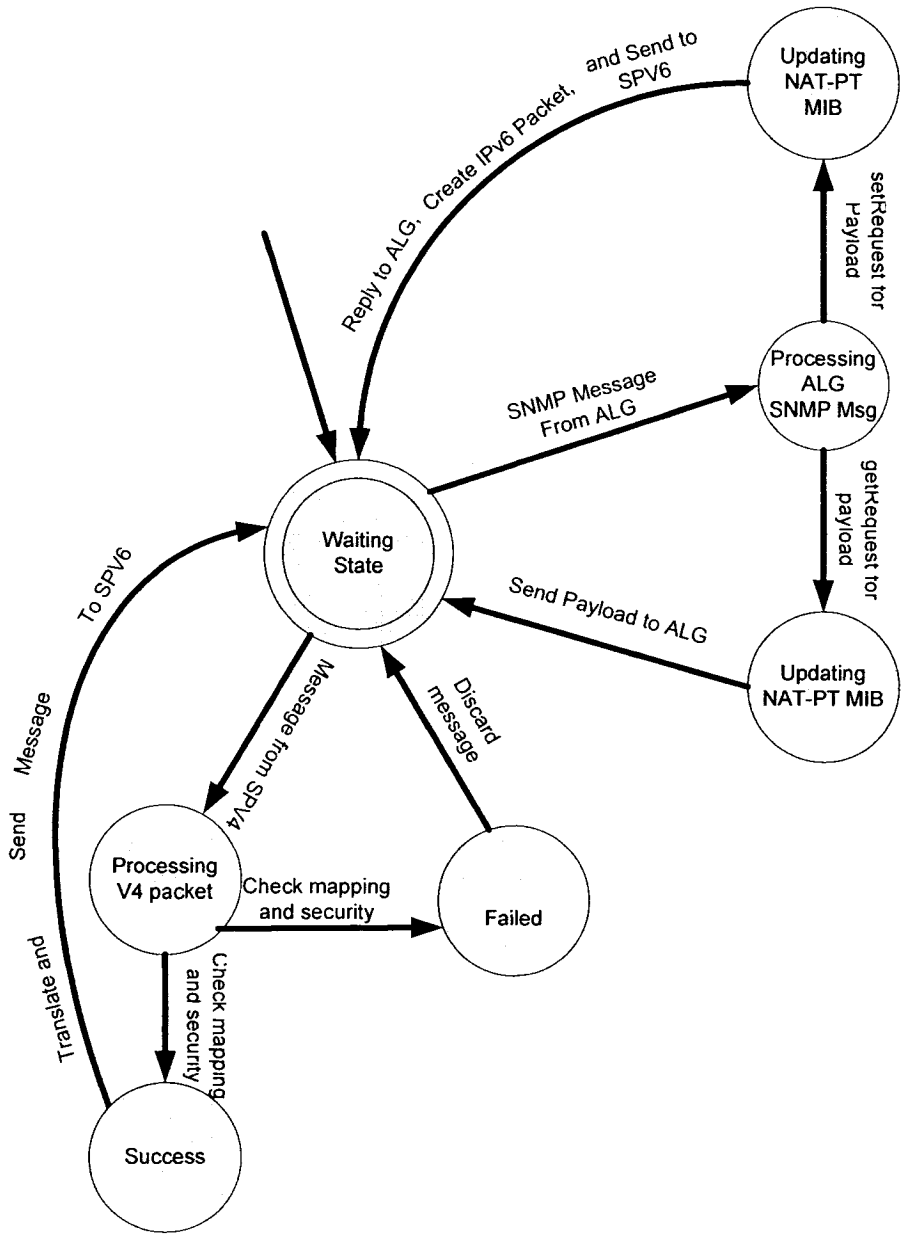

Figure 6. 2 State Diagram for NAT-PT (Part 2)[2]

[2] *Figure 6.2 describes the state diagram for NAT-PT when it receives message from SPV4 and the message is a network packet not requiring the service of an ALG, or when the NAT-PT receives an SNMP message from an ALG*

6.4.1.3 V4 Network Packet Process State

When NAT-PT receives any network packet from V4 network via SPV4, and the packet does not require service of an ALG, the control will move to this state. NAT-PT will replace IPv4 header with IPv6 header, encrypt the message, and forward it SPV6, and control returns to the initial state.

6.4.1.4 V4 Network Packet requiring service of an ALG

When NAT-PT receives any network packet from V4 network via SPV4, and the packet requires the service of an ALG, the control will move to this state. The NAT-PT will save the IP Header and Payload, send notification to the ALG, wait for response from the ALG. The control will move to the initial state.

6.4.1.5 SPV6 SNMP Message Process state

The NAT-PT will be in this state when it receives an SNMP message from SPV6. Like the SPV4_SNMP_Message_Process state, the objective of the SNMP message may be to initiate a session, to terminate a session or to send V6 key to the NAT-PT. The processing of the SNMP message will depend upon the objective of the message. Like an SNMP message from SPV4, the objective of the received SNMP message determines the next state where the control will flow. Depending upon the objective of the

message, the control will move to the V6_Session_Initation, V6_Session_termination or V6_Key_Update. The processing may be successful or unsuccessful. In any case, the NAT-PT will inform the SNMP manager (SPV4) and the control will return to the initial state.

6.4.1.6 V6 Network Packet Process

The NAT-PT with security proxies makes a symmetrical network. So the processing states for network packet received via SPV6 will be as same as for the packets received from SPV4, the name of the states will be bit different—such as Route_To_SPV4.

6.4.1.7 ALG SNMP Process

If the NAT-PT receives any SNMP message from ALG, the control jumps to this state. ALG may request information from the NAT-PT, or it can send the translated payload for NAT-PT. If required, NAT-PT will update MIB tables and control will return to the initial state.

6.4.2 Security Proxy

The following are the states of the Security Proxy—

6.4.2.1 Initial State

This is the waiting state. In this state the Security Proxy (either SPV4 or SPV6) waits for the incoming message. It may receive message from IPv4 host (if SPV4) or IPv6 host (if the security proxy is SPV6), or from NAT-

PT. Depending on the message the security proxy receives, it moves to the next state.

6.4.2.2 Initiate Session

The Security Proxy will receive a session initiation request from an end host. At this state the security proxy will authenticate the requesting node, verify resources for the forthcoming session, initiate a session, and inform NAT-PT. It will confirm the requesting node.

6.4.2.3 Terminate Session

Like the session initiation, the Security Proxy will receive a session termination request from an end host. At this state the security proxy will authenticate the requesting node, release the resources for the session, and inform NAT-PT. The control will return to the waiting state.

6.4.2.4 Route Message to NAT-PT

If the Security Proxy receives any message other than a session initiation request, it will forward the message to NAT-PT. However, before forwarding the message, the Security Proxy will ensure the authenticity and integrity of the message. As the end host and the Security Proxy use same network protocol either it is IPv4 or IPv6, we assume that any end host can communicate with Security Proxy using the existing network security measures reliably. After forwarding the message to the NAT-PT, the control will return to the waiting state.

6.4.2.5 Send SNMP Message to NAT-PT

The Security Proxy sends SNMP message when it receives any session request from an end host, and it may be for session initiation and session termination. Other than this the Security Proxy will send V4 Key for the session initiated by SPV6 or send V6 Key for the session initiated by the SPV4.

6.5 Validation Result

The interactions between processes prove the absence of dead lock, live lock of the model. The model also verified assertion violation, invalid end state and in all cases the test results are positive.

6.6 Summary

This chapter formalized NAT-PT specification to build PROMELA model and to verify the correctness, completeness, and logical consistency of the model. At the beginning, this chapter introduced **PROMELA.** And then we have seen that a PROMELA model consists of three types of objects—Processes, Channels, and Variables. After the introduction, the chapter describes what processes, channels, and variables are used for NAT-PT model. The section 6.4 explains the behavior model of the NAT-PT system. The section 6.5 concluded the chapter with the validation result.

In the Chapter 7 we will summarize what we achieved, and what will be our direction for future work.

Chapter 7 : Conclusion and Suggested Future Work

7.1 What we achieved

In this section we will see what we achieved through this research work. We can summarize our achievement as described below:

- Analyzed the existing NAT-PT model and its drawbacks.
- Explored standard framework for adopting NAT-PT into the framework.
- Proposed a modified NAT-PT model.
- Specified how it would work.
- Validated the model.
- Wrote a NAT-PT MIB for the model.

7.2 What is New?

We developed a new NAT-PT model. In this section we will explore the difference between the existing NAT-PT and our proposed model. We can summarize the differences between the proposed model and the existing one as below:

- Our proposed NAT-PT model works as a distributed system, but the existing one is not a distributed system.
- For the integration of a new application to the NAT-PT environment, it will not be required to recompile the system,

whereas in case of the existing NAT-PT, if we want to integrate any new application we have to recompile the whole NAT-PT system.

- The proposed model addresses the vulnerability to the security threats of the existing NAT-PT system.
- The proposed model will not suffer from scalability, and load balancing problem, where as these drawbacks are the serious concerns of the existing NAT-PT.

In the next section we will see how the proposed model will serve better than the existing one.

7.3 How does Proposed Model remove drawbacks of NAT-PT?

The advantages of distributed NAT-PT have already been described in Chapter 4 and 5. This chapter describes how the distributed NAT-PT avoids the limitations mentioned in Chapter 3.

7.3.1 Scalability Problem

All the requests and responses of a particular session must traverse through the NAT-PT for which the session has been created. If the volume of traffic is huge, the single NAT-PT router may not be sufficient to handle huge network traffic efficiently. Multiple NAT-PTs (mNAT-PT) may be deployed to get around to that problem. The proxy server will do

load balancing among different NAT-PT boxes to avoid scalability problem. Again, our proposed proxy server will work as an SNMP manager; it will be able to monitor any SNMP agent under its jurisdiction. If a particular session is interrupted due to the problem of the NAT-PT box for which the session has been created earlier, the security proxy will monitor it and create the same session with one of the other operating NAT-PT boxes. So scalability problem may be avoided with the deployment of security proxy.

7.3.2 End-to-end Security

Security is an important concern for Internet community. The existing Internet community is vulnerable to different security threats. Moreover, the use of NAT-PT architecture will add the security treats described in Chapter 3 on top of the existing network threats. Chapter 3 describes that the most important drawback of the NAT-PT proposal is the lack of end-to-end network layer security. Moreover, transport and application layer security may not be possible for applications that carry IP addresses to the application layer. This inherent limitation of the Network Address Translation function can be avoided by deploying a proxy server. According to the proposed model, any node from either IPv6 or IPv4 network will have to establish a secure session with the proxy server. The proxy server will ensure the authenticity of the communicating node. Only after the establishment of the secured

session, the network packets from either side will be forwarded through the NAT-PT. If a node fails to establish secured session with the proxy server, it will not be able to get the translation services of the NAT-PT.

IPSec may be used to establish secured session between the communicating node and the proxy server. IPSec provides security services for either IPv4 or IPv6, but not for the both at the same time. We proposed two proxies—one for IPv4 and another for IPv6. So it is possible to use existing network layer security system such as IPSec to establish secured communication between IPv4 host and IPv4 Proxy Server, and the same for the IPv6 part of the network.

Proxy server is also a NAT-PT module. It will work as an SNMP manager. Therefore, proxy server will communicate with NAT-PT through SNMP messages. We have mentioned that SNMPv3 will ensure the security of the messages between proxy server and NAT-PT.

NAT-PT verifies the data integrity of network packets traversing through it. It requires a cryptographic key to decrypt the data. Because the IP security Architecture, which provides various security services for network traffic at the IP layer, is based on cryptography. So the key management is an important aspect of the IPSec. There exist two important key management specifications associated with the IPSec—the

Internet Security Association and Key Management Protocol (ISAKMP) and the Internet Key Exchange (IKE). Which key management protocol will be used and how it will be used is an implementation choice. The cryptographic key may be exchanged between the security proxy and a communicating node or between the NAT-PT and a communicating node. In the first case the key should be shared with NAT-PT, otherwise the NAT-PT may not be able to read any encrypted packet traversing through it. Security proxy will share the key with NAT-PT via SNMP messages. As the secured session will be established between the proxy server, and a communicating node, the second choice may not be practical. Although distributed NAT-PT does not ensure the end-to-end security, however, the above analysis clearly indicates that the threats due to lack of end-to-end network security can be avoided through the use of a distributed system.

7.3.3 Source address spoofing attack

Chapter 3 describes that vulnerability of any network to this type of attack is not introduced by NAT-PT. The use of NAT-PT architecture might just worsen the situation; however, the use of IPSec between the proxy server and a communicating node will mitigate these possible additional threats.

7.3.4 Performance

The proposed NAT-PT as described in RFC 2766 works sequentially; that means NAT-PT first translates IP header of the packet, then checks for ALG and invokes ALG, if any, for its services. However, the protocol translator, and ALG(s) of the distributed system can work concurrently. Exchange of SNMP messages among SNMP components will mount the overhead of the distributed system. It is expected that the concurrent computation and processing of the messages will remove that overhead. Here we can expect performance improvement by the use of distributed NAT-PT.

7.4 Future Work

Developing a good Implementation framework for the proposed model is the future work of the proposed model. We can start implementation from the scratch, or with the prior permission we can use NAT-PT developed by British Telecom (BT). BT developed NAT-PT for FreeBSD. Since we are interested in Linux, we can make it portable for Linux. For the development of a complete system, it is necessary to systematically go through the steps mentioned below:

A. Requirement analysis

B. Designing a protocol model

C. Validating the model with appropriate validation tool

D. Designing the software

E. Implementation of the software

F. Performance test

G. Writing MIB for the distributed system

H. Getting SNMP Engine

I. If necessary change the SNMP

J. Register the MIB and SNMP manager and agents with the SNMP engine.

As a part of the present research, we completed the steps A, B, C, and G. The step D can be started from C. We used SPIN model checker to validate the protocol model. Steps H, I, and J are trivial and not complicated at all.

BIBLOGRAPHY

[1] RFC 2460 Internet Protocol, Version 6 (IPv6) Specification www.ietf.org/rfc/rfc2460.txt

[2] RFC 2893 Transition Mechanisms for IPv6 Hosts and Routers www.ietf.org/rfc/rfc2893.txt

[3] RFC 3053 IPv6 Tunnel Broker, www.ietf.org/rfc/rfc3053.txt

[4] RFC 3056 Connection of IPv6 Domains via IPv4 Clouds, www.ietf.org/rfc/rfc3056.txt

[5] RFC 2766 Network Address Translation - Protocol Translation (NAT-PT) www.ietf.org/rfc/rfc2766.txt

[6] RFC 2185 Routing Aspects Of IPv6 Transition www.ietf.org/rfc/rfc2185.txt

[7] RFC 2401 Security Architecture for the Internet Protocol www.ietf.org/rfc/rfc2401.txt

[8] RFC 2402 IP Authentication Header www.ietf.org/rfc/rfc2402.txt

[9] Internet Draft: NAT-PT Applicability http://www.join.uni-muenster.de/Dokumente/drafts/draft-satapati-v6ops-natpt-applicability-00.txt

[10] RFC 3303 MiddleBox communication architecture and framework, www.ietf.org/rfc/rfc3303.txt

[11] RFC 3304 MiddleBox Communications (MIDCOM) Protocol Requirements, www.ietf.org/rfc/rfc3303.txt

[12]Internet Draft: MiddleBox Communications (MIDCOM) Protocol Evaluation, http://www.ietf.org/proceedings/I-D/draft-ietf-midcom-protocol-eval-06.txt

[13]Internet Draft: MIDCOM Protocol Semantics, http://www.ietf.org/proceedings/I-D/draft-ietf-midcom-semantics-04.txt

[14] Internet Draft: Evaluation Of DIAMETER Against MIDCOM Requirements, http://mirrors.isc.org/pub/www.watersprings.org/pub/id/draft-taylor-midcom-diameter-eval-01.txt

[15] Internet Draft: Using SNMP as MIDCOM Protocol, http://mirrors.isc.org/pub/www.watersprings.org/pub/id/draft-quittek-midcom-snmp-eval-00.txt

[16]Internet Draft, MiddleBox Communications (MIDCOM) Protocol Managed Objects Analysis, http://www.ietf.org/proceedings/I-D/draft-ietf-midcom-mib-analysis-00.txt

[17] Internet Draft: MiddleBox Communications (MIDCOM) Protocol Managed Objects, http://bgp.potaroo.net/ietf/idref/draft-barnes-midcom-mib/

[18] Internet Draft: Peer-to-Peer communication across MiddleBoxes, http://ietfreport.isoc.org/all-ids/draft-ford-midcom-p2p-00.txt

[19]RFC 3410 Introduction and Applicability Statements for Internet Standard Management Framework, www.ietf.org/rfc/rfc3410.txt

[20]RFC 3411 An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, www.ietf.org/rfc/rfc3411.txt

[21]RFC 3413 Simple Network Management Protocol (SNMP) Applications, www.ietf.org/rfc/rfc3413.txt

[22]RFC 3418 Management Information Base (MIB) for the Simple Network Management Protocol (SNMP), http://www.ietf.org/rfc/rfc3418.txt

[23]Internet Draft-Definitions of Managed Objects for Network Address Translators (NAT), http://www.ietf.org/internet-drafts/draft-ietf-nat-natmib-09.txt

[24]Linux-based User space **NAT-PT** (Network Address Translation - Protocol Translation) ETRI/PEC - IPv4/IPv6 Translation Technology - **NAT-PT** - Overview

[25]net-SNMP http://www.net-snmp.org/

[26]SNMP Tutorial: The Management Information Base (MIB), http://www.dpstele.com/protocol/2000/nov_dec/snmp_mib.html

[27]The SPIN MODEL CHECKER Primer and Reference Manual by Gerard J. Holzmann

[28]MODEL CHECKING with SPIN, http://spinroot.com/

[29]ASN.1 Information Site http://asn1.elibel.tm.fr/en/index.htm

APPENDIX A: NAT-PT Environment Setup and Testing

We set up testing environment for NAT-PT at the HPLS network lab of the computer science department. NAT-PT works between IPv4 and IPv6 networks. For it's testing we need one IPv6 LAN, one IPv4 LAN and border router in between these two networks. We use the same OS—Linux for all hosts. Our testing environment consists of the following network segments—

IPv6 LAN

It consists of just one IPv6 node—plum, and the fully qualified domain name is plum.ipv6.con. As it is an IPv6 node we disabled IPv4 address on this machine. Its IPv6 address: 3ffe:: 2b0:doff:fedd:cea4, the scope is global, and the network interface card: eth0. We installed Apache http server to turn it into http server host.

IPv4 LAN

It also consists of just one IPv4 node—forest, we have not changed its domain name. Its fully qualified domain name is forest.cs.concordia.ca. Its IP address is 132.205.45.24. In the same way we installed Apache http server on it.

Border Router

It is NAT-PT host. Although we use the terminology Border router, but actually it does not use protocol stack router software from operating system to route a packet. Rather NAT-PT itself forwards a packet to the destination host. So the disabling of ip_forwarding of the Border Router is must for NAT-PT. We set up the "cherry" as the host for the NAT-PT. Its network interface card eth0 and eth1 are connected to the IPv4 and IPv6 LAN respectively. So we configured eth1 so that it can support IPv6 protocol, and we also disabled its IPv4 address. The IP address assigned to eth0 is 132.205.45.162, and that to eth1 is 3ffe::2e0:2ff:fe24:4472

IPv6 DNS Server

We installed IPv6 supporting DNS server (BIND 9.2) on the IPv6 node. We configured the DNS server for IPv6 protocol.

IPv4 DNS server

We did not make any change in the IPv4 DNS server; as a result we could use the real DNS server assigned for IPv4 network (for the node "forest").

Configuration

We have two IPv4 addresses for IPv4 address pool. These are– 142.133.71.16 and 142.133.71.17

We used 142.133.71.17 for the static mapping between IPv6 DNS server address and IPv4 pool address. We included that IPv4 pool address in the resolv.conf file as one of the DNS server of the IPv4 node. Pool addresses are not connected to any interfaces; so the arp request for any pool IP address will not resolve to MAC address. We modified the routing table entry of "forest" so that any packet with any pool address as the destination address is forwarded to "cherry". When any packet arrives at cherry with a pool address as the destination address, the NAT-PT intercepts the packet, and forwards to the destination address according to the mapping. As in our case, plum hosts all IPv6 server applications, NAT-PT forwards any packet originated in the IPv4 side directed to IPv6 node, to plum.

The NAT-PT converts any IPv4 address to IPv6 address by prefixing a PREFIX (96 bits) before IPv4 address. Our simple prefix is aaaa:bbbb:cccc:dddd:eeee:ffff. We edited the /etc/resolv.conf file of plum as described below:

Search ipv6.con cs.concordia.ca

Nameserver 3ffe::2b0:doff:fedd:cea4

Nameserver aaaa:bbbb:cccc:dddd:eeee:ffff::132.205.44.61

132.205.44.61 is the IP address of the real IPv4 DNS server.

We modified the routing table of plum, because any packet of which destination contains the PREFIX, should be forwarded to the NAT-PT. From this destination address the NAT-PT peels off the PREFIX and gets the IPv4 address to forward the packet to the destination address.

With the setup and configuration described above, we tested the NAT-PT for http server and client applications.

APPENDIX B: Management Information Base or MIB

Introduction

Managed objects are accessed via a virtual information store, termed as the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI).

The section describes the NAT-PT MIB, and I have written it following the information provided in Internet draft "draft-ietf-natpt-natptmib-09.txt". In this document I followed SMIv2 standard to write Management Information Base (MIB) for device implementing NAT-PT function.

MIB Objects

We know that the SNMP managers and agents exchange information using MIB objects. For the NAT-PT model we defined several MIB objects. The following are the key objects used in NAT-PT MIB—

natptIPv4AddressPoolTable

It will consist of topple (IPv4_Addresses, flag).

natptStatAddrMapTable

Address mapping is an important factor for NAT-PT configuration. NAT-PT looks up address map entries in order to determine the translation parameters for each packet traversing the NAT-PT box. The table will contain the mapping between IPv4 and IPv6 address, and the destination port of the packet and a port number assigned by NAT-PT. Address map entries may be defined in this MIB using natptStatAddrMapTable.

Default timeouts, Protocol table and other scalars

Protocol specific idle NAT-PT session timeouts are defined in DefTimeouts object in the NAT-PT MIB. The scalars, natptAddrMapNumberOfEntries and hold the number of entries that currently exist in the Address Map table.

natptAddrMapTable

Entry to this table will indicate each NAT-PT session. natptAddrMapTable is indexed by natptSessionIndex. Statistics for NAT-PT sessions are also maintained in the same table.

natptIPHeaderTable

The formats of IP Header in IPv4 and IPv6 are not same. So IP Header of each packet traversing through NAT-PT device should be translated. This table will contain information about IP Header, the origin network, checksums, and session ID, and the Protocol Translator will use that session ID.

natptPayLoadTable

This table will be used by ALGs if the payload required to be translated. It will consist of topple (natptSessionIndex, natptpayLoad).

natptReceivedPacketTable

The fields of this table are—

natptReceivedPacketID, natptSessionID, natptIPv4Header, natptIPv6Header, natptIPv4Payload, natptIPv6Payload

NAT-PT will read network packet received from either SPv4 or SPv6 and update the values for the different fields.

natptPacketForSendingTable

The fields of this table are—

natptPacketForSendingID, natptSessionID, natptIPv4Header, natptIPv6Header, natptIPv4Payload, natptIPv6Payload, natptPacketStatus.

The SNMP managers and the different modules required for NAT-PT will update the values of the fields of this table. This table will contain the translated IP Header and the Payload (if require).

Notifications

NAT-PT will use this object to notify SNMP managers in case of necessity. natptPacketDiscard notifies the end user/manager of packets being discarded due to lack of address mappings.

A Management station may use the following steps to configure entries in the NAT-PT-MIB—

1. Create an entry in the natptInterfaceTable specifying the value of ifIndex as the interface index of the interface, which NAT-PT is being configured. Specify appropriate values, as applicable, for the other objects e.g. natptInterfaceRealm, natptInterfaceServiceType, in the table.

2. Create one or more address map entries sequentially in reduced order of priority in the natptStatAddrMapTable.

3. To configure NAT-PT for TCP, UDP and ICMP protocols, the management station can set the protocol specific scalars.

4. The Address Mapping and Address-Port Mapping Table will have the entries created due to this NAT-PT configuration. A Management Station may also, if deemed necessary, create Address Mapping or an Address-Port Mapping entry and link those entries to the appropriate address map configured.

Definitions

```
NAT-PT-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32,
    Gauge32,
    Counter64,
    TimeTicks,
    mib-2,
    IpAddress,
    Integer32,
    NOTIFICATION-TYPE
        FROM SNMPv2-SMI
    TimeInterval,
    TEXTUAL-CONVENTION
        FROM SNMPv2-TC
```

```
    MODULE-COMPLIANCE,
    NOTIFICATION-GROUP,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    StorageType,
    RowStatus
        FROM SNMPv2-TC
    ifIndex
        FROM IF-MIB
    Ipv6Address, Ipv6IfIndexOrZero
        FROM IPV6-TC;

    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    InetAddressType,
    InetAddress,
    InetPortNumber
        FROM INET-ADDRESS-MIB;

natptMIB MODULE-IDENTITY
    LAST-UPDATED "20040206"
    ORGANIZATION "Concordia University"
    CONTACT-INFO
          "Kedar Das
           Concordia University
           kc_das@cs.concordia.ca"
    DESCRIPTION
        "This MIB module defines the generic managed objects for NAT-PT"

natptMIBObjects OBJECT IDENTIFIER ::= { natptMIB 1 }

   NatptProtocolType ::= TEXTUAL-CONVENTION
   STATUS       current
    DESCRIPTION
        "It is a list of protocols that are supported by the NAT-PT."

SYNTAX   INTEGER {
                none (1),  -- not specified
                other (2), -- none of the following
                icmp (3),
                udp (4),
                tcp (5)
              }

NatptSessionStateInfo ::= TEXTUAL-CONVENTION
     STATUS       current
```

```
        DESCRIPTION
          "It is a list of session state associated with NatptProtocolType. Any
          change in this TEXTUAL-CONVENTION should also be reflected in
          the definition of NatptProtocolType."
        SYNTAX   INTEGER {
                    other (1),
                    icmp (2),
                    udpStart (3),
                    udpMiddle (4),
                    udpEnd(5),
                    tcpStart(6),
                    tcpMiddle(6)
                    tcpEnd(6)
                 }

NatptSessionId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
          "A unique id that is assigned to each session by a NAT-PT."
          SYNTAX   Unsigned32 (1..4294967295)

NatptPacketID ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
          "It is a unique ID assigned to each packet by a NAT-PT."
          SYNTAX   Unsigned32 (1..4294967295)


NatptMapMode ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
          "It is an indication whether the mapping is an address mapping or
          an address-port mapping."
        SYNTAX   INTEGER {
                    addressBind (1),
                    addressPortBind (2)
                 }
--
-- Default Values for the NAT-PT Protocol Timers
--

natptDefTimeouts OBJECT IDENTIFIER ::= { natptMIBObjects 1 }
--
-- UDP related NAT-PT configuration
```

```
--

natptUdpDefIdleTimeout OBJECT-TYPE
   SYNTAX     Unsigned32  (1..4294967295)
   UNITS      "seconds"
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
         "The default UDP idle timeout parameter."
   DEFVAL { 300 }
   ::= { natptDefTimeouts 1 }

--
-- ICMP related NAT-PT configuration
--

natptIcmpDefIdleTimeout OBJECT-TYPE
   SYNTAX     Unsigned32  (1..4294967295)
   UNITS      "seconds"
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
         "The default ICMP idle timeout parameter."
   DEFVAL { 300 }
   ::= { natptDefTimeouts 2 }

--
-- Other protocol parameters
--

natptOtherDefIdleTimeout OBJECT-TYPE
   SYNTAX     Unsigned32  (1..4294967295)
   UNITS      "seconds"
   MAX-ACCESS read-write
   STATUS     current
   DESCRIPTION
      "The default idle timeout parameter for protocols represented by
      the value other (2) in NatptProtocolType."
   DEFVAL {60}
   ::= { natptDefTimeouts 3 }

--
-- TCP related NAT-PT Timers
--
natptTcpDefIdleTimeout OBJECT-TYPE
   SYNTAX     Unsigned32  (1..4294967295)
```

```
    UNITS       "seconds"
    MAX-ACCESS read-write
    STATUS      current
    DESCRIPTION
        "The default time interval, a NAT-PT session for an established TCP
        connection is allowed to remain valid without any activity on the
        TCP connection."
    DEFVAL { 86400 }
    ::= { natptDefTimeouts 4 }

natptTcpDefNegTimeout OBJECT-TYPE
    SYNTAX      Unsigned32  (1..4294967295)
    UNITS       "seconds"
    MAX-ACCESS read-write
    STATUS      current
    DESCRIPTION
        "The default time interval, a NAT-PT session for a TCP connection
        which is not in the established state is allowed to remain valid
        without any activity on the TCP connection."
    DEFVAL { 60 }
    ::= { natptDefTimeouts 5 }

--
-- The Static Address Mapping Table
--
natptStatAddrMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatptAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
          "This table lists static address map required for NAT-PT."
    ::= { natptMIBObjects 3 }

natptStatAddrMapEntry OBJECT-TYPE
    SYNTAX      NatptAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This entry represents an address mapping between an IPv4
        address and IPv6 address. The IPv4 address will be a pool address
        and the IPv6 address is the address of the IPv6 real DNS server."
      ::= { natptAddrMapTable 1 }

NatptStatAddrMapEntry ::= SEQUENCE {
    natptStatAddrMapIndex                NatptAddrMapId,
    natptStatAddrMapIPv4Addr             InetAddress,
```

```
    natptStatAddrMapIPv6Addr        InetAddress,
    natptStatAddrMapRowStatus       RowStatus
}


natptStatAddrMapIndex  OBJECT-TYPE
    SYNTAX      NatptAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Along with ifIndex, this object uniquely identifies an entry in the
        natptStatAddrMapTable. Address map entries are applied in the
        order specified by natptStatAddrMapIndex."
    ::= { natptStatAddrMapEntry 1 }


natptStatAddrMapIPv4Addr OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object specifies the first IPv4 address used in the static
        mapping."
    ::= { natptStatAddrMapEntry 2 }

natptStatAddrMapIPv6Addr OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object specifies the IPv6 address of the IPv6 DNS server."

    ::= { natptStatAddrMapEntry 3 }

natptStatAddrMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The status of this conceptual row. Until instances of all
        corresponding columns are appropriately configured, the value of
        the corresponding instance of the natptAddrMapRowStatus
        column is 'notReady'. None of the objects in this row may be
        modified while the value of this object is active(1)."

    REFERENCE
```

```
            "Textual Conventions for SMIv2, Section 2."
    ::= { natptStatAddrMapEntry 4 }

--
-- Address Map section
--
natptAddrMapNumberOfEntries OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
        "This object maintains a count of the number of entries that
        currently exist in the natptAddrMapTable."
    ::= { natptMIBObjects 4 }

--
-- The NAT-PT Address MAP Table
--

natptAddrMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatptAddrMapEntry
    MAX-ACCESS not-accessible
    STATUS      current
    DESCRIPTION
        "This table holds information about the currently active NAT-PT
        Mappings."
    ::= { natptMIBObjects 5 }

natptAddrMapEntry OBJECT-TYPE
    SYNTAX      NatptAddrMapEntry
    MAX-ACCESS not-accessible
    STATUS      current
    DESCRIPTION
        "Each entry in this table holds information about an active address
        BIND. These entries are lost upon agent restart."
    ::= { natptAddrMapTable 1 }

NatptAddrMapEntry ::= SEQUENCE {
    natptAddrMapIPv6Addr                Ipv6Address,
    natptAddrMapIPv4Addr                InetAddress,
    natptAddrMapId                      NatptBindId,
    natptAddrMapTranslationEntity       NatptTranslationEntity,
    natptPoolAddrIndex                  NatptAddrMapId,
    natptAddrMapSessions                Gauge32,
    natptAddrMapMaxIdleTime             TimeInterval,
    natptAddrMapCurrentIdleTime         TimeTicks,
```

```
    natptAddrMapIPv4Translates          Counter64,
    natptAddrMapIPv6Translates          Counter64,
    natptAddrMapRowStatus               RowStatus
}

natptAddrMapIPv6Addr OBJECT-TYPE
    SYNTAX     Inet6Address
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "This object represents the IPv6 address of an IPv6 node, which
        maps to the IPv4 pool address represented by
        natptAddrMapIPv4Addr."
    ::= { natptAddrMapEntry 1 }

natptAddrMapIPv4Addr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
        "This object represents an IPv4 pool address."
    ::= { natptAddrMapEntry 2 }

natptAddrMapId OBJECT-TYPE
    SYNTAX     NatptBindId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "This object represents a mapping id that is dynamically assigned
        to each mapping by a NAT-PT.
    ::= { natptAddrMapEntry 3 }

natptAddrMapTranslationEntity OBJECT-TYPE
    SYNTAX     NatptTranslationEntity
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
        "This object represents the direction of sessions for which this
        mapping is applicable and the endpoint entity (source or
        destination) within the sessions that is subject to translation using
        the MAP."
    ::= { natptAddrMapEntry 4 }

natptPoolAddrIndex OBJECT-TYPE
    SYNTAX     NatptAddrMapId
    MAX-ACCESS read-create
```

```
    STATUS      current
    DESCRIPTION
        "This object is a pointer to the natptPoolAddrTable entry (and the
        parameters of that entry) which was used in creating this MAP."
    ::= { natptAddrMapEntry 5 }

natptAddrMapSessions OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
        "This object represents the number of sessions currently using this
        MAP."
    ::= { natptAddrMapEntry 6}

natptAddrMapMaxIdleTime OBJECT-TYPE
    SYNTAX      TimeInterval
    MAX-ACCESS read-create
    STATUS      current
    DESCRIPTION
        "This object indicates the maximum time for which this bind can
        be idle with no sessions attached to it."
    ::= { natptAddrMapEntry 7 }


natptAddrMapCurrentIdleTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
        "At any given instance of time, this object indicates the time that
        this bind has been idle with no sessions attached to it."
    ::= { natptAddrMapEntry 8 }

natptAddrMapIPv4Translates OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
        "The number of packets that were successfully translated using
        this bind entry."
    ::= { natptAddrMapEntry 9 }

natptAddrMapIPv6Translates OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS read-only
```

```
    STATUS      current
    DESCRIPTION
        "The number of IPv6 packets that were successfully
        translated using this map entry."
    ::= { natptAddrMapEntry 10 }

natptAddrMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Until instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        natptAddrMapRowStatus column is 'notReady'. None of the
        writable objects except natptAddrMapMaxIdleTime in this row may
        be modified while the value of this object is active(1)."
    REFERENCE
          "Textual Conventions for SMIv2, Section 2."
    ::= { natptAddrMapEntry 11 }

--
-- IPv4 Pool Address Table
--

natptIPv4AddrPoolTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatptIPv4AddrPoolEntry
    MAX-ACCESS not-accessible
    STATUS      current
    DESCRIPTION
        "This table holds information about Pool addresses."
    ::= { natptMIBObjects 6 }

natptIPv4AddrPoolEntry OBJECT-TYPE
    SYNTAX      NatptIPv4AddrPoolEntry
    MAX-ACCESS not-accessible
    STATUS      current
    DESCRIPTION
        "Each entry in this table holds information about each pool
        address and its status."
    ::= { natptIPv4AddrPoolTable 1 }

NatptIPv4AddrPoolEntry ::= SEQUENCE {
    natptIPv4PoolAddr                   InetAddress,
    natptPoolAddrInUseTime              TimeTicks,
    natptIPv4PoolAddrStatus             BIT
    natptIPv4PoolAddrBlockedStatus      BIT
```

```
}


natptIPv4PoolAddr OBJECT-TYPE
   SYNTAX      InetAddress
   MAX-ACCESS  read-create
   STATUS      current
   DESCRIPTION
       "This object specifies the IPv4 addresses used for NAT-PT pool."
   ::= { natptIPv4AddrPoolEntry 1 }


natptPoolAddrInUseTime OBJECT-TYPE
   SYNTAX     TimeTicks
   MAX-ACCESS read-only
   STATUS     current
   DESCRIPTION
       "At any given instance of time, this object indicates the time that
       how long a pool address has been in use."
   ::= { natptAddrMapEntry 2 }

natptIPv4PoolAddrStatus   OBJECT-TYPE
   SYNTAX     INTEGER {
                  InUse (0),
                  Free (1)
              }
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
       "This object identifies whether the pool address is free or in use."
   DEFVAL     { Free }
   ::= { natptIPv4AddrPoolEntry 3 }

natptIPv4PoolAddrBlockedStatus   OBJECT-TYPE
   SYNTAX     INTEGER {
                  Blocked (0),
                  Not-Blocked(1)
              }
   MAX-ACCESS read-create
   STATUS     current
   DESCRIPTION
       "This object identifies whether the pool address is blocked or not."
   DEFVAL     { Not-Blocked }
   ::= { natptIPv4AddrPoolEntry 4 }
```

```
--
-- The NAT-PT Session Table
--

natptSessionInfoTable OBJECT-TYPE
   SYNTAX     SEQUENCE OF NatptSessionEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
       "This table holds information about the currently active NAT-PT
       Session"
   ::= { natptMIBObjects 5 }

natptSessionEntry OBJECT-TYPE
   SYNTAX     NatptSessionEntry
   MAX-ACCESS not-accessible
   STATUS     current
   DESCRIPTION
       "Each entry in this table holds information about each active
       session. These entries are lost upon agent restart."
   ::= { natptSessionTable 1 }

NatptSessionEntry ::= SEQUENCE {
   natptSessionID                    NatptSessionId,
   natptSessionPacketCounter         Counter64,
   natptSessionPacketOriginNetwork   INTEGER,
   natptSessionTime                  TimeTicks,
   natptAddrMapId                    NatptBindId,
   natptSecurityParameter            BIT STRING
   natptSessionMapRowStatus          RowStatus
}

natptSessionID OBJECT-TYPE
   SYNTAX     NatptSessionId
   MAX-ACCESS  read-create
   STATUS     current
   DESCRIPTION
       "This object uniquely identifies each session."
   ::= { natptSessionEntry 1 }

natptSessionPacketCounter OBJECT-TYPE
   SYNTAX     Counter64
   MAX-ACCESS  read-create
   STATUS     current
   DESCRIPTION
       "This object gives the number of the active sessions."
```

```
    ::= { natptSessionEntry 2 }

natptSessionPacketOriginNetwork OBJECT-TYPE
        SYNTAX     INTEGER {
                IPv4 (0),
                IPv6 (1)
            }
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object identifies the origin of the packet, that means
                either it is IPv4 or IPv6."
    ::= { natptSessionEntry 3 }


natptSessionTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "At any given instance of time, this object indicates the session
        time."
    ::= { natptSessionEntry 4 }

natptAddrMapID  OBJECT-TYPE
    SYNTAX     NatptBindId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "This is a pointer to the natptAddrMap table."
    ::= { natptSessionEntry 5}

natptSessionSecurityParameter OBJECT-TYPE
SYNTAX     BIT STRING
        MAX-ACCESS read-create
        STATUS      current
        DESCRIPTION
                "This object identifies the security parameter of the packet."
    ::= { natptSessionEntry 6 }

natptSessionRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
```

```
        "The object indicates the status of this conceptual row. Until
        instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        natptSessionRowStatus column is 'notReady'. None of the write
        able objects except natptSessionTime in this row may be modified
        while the value of this object is active(1)."
    REFERENCE
          "Textual Conventions for SMIv2, Section 2."
    ::= { natptSessionEntry  7}

--
-- The NAT-PT Received Packet Info table
--
natptRecievedPacketInfoTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatptReceivedPacketInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "This table holds the payload of each packet for each NAT-PT
        Session"
    ::= { natptMIBObjects 6}

natptReceivedPacketInfoEntry  OBJECT-TYPE
    SYNTAX     NatptReceivedPacketInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "Each entry in this table holds the packet payload of each session.
        These entries are lost upon agent restart."
    ::= { natptReceivedPacketInfoTable 1 }

NatptReceivedPacketInfoEntry ::= SEQUENCE {
    natptReceivedPacketID                  NatptPacketID,
    natptReceivedPacketOrigin              INTEGER,
    natptReceivedPacketIPv4Header          BIT STRING,
    natptReceivedPacketIPv4HeaderLenght    INTEGER,
    natptReceivedPacketIPv6Header          BIT STRING,
    natptReceivedPacketIPv6Header Length   INTEGER,
    natptReceivedPacketProtocol            NatptProtocolType,
    natptReceivedPacketIPv4Checksums       BIT STRING,
    natptReceivedPacketIPv6Checksums       BIT STRING,
    natptReceivedPacketIPv4PacketPayload   BIT STRING,
    natptReceivedPacketIPv4PayloadLenght   INTEGER,
    natptReceivedPacketIPv6Payload         BIT STRING,
    natptReceivedPacketIPv6PayloadLenght   INTEGER,
    natptReceivedPacketProtocol            NatptProtocolType,
```

```
    natptSessionID                      NatptSessionId,
    natptReceivedPacketMapRowStatus     RowStatus
}


natptReceivedPacketID OBJECT-TYPE
    SYNTAX      NatptPacketID
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object uniquely identifies the packet of the payload."
    ::= { natptReceivedPacketInfoEntry 1 }

natptReceivedPacketOrigin OBJECT-TYPE
        SYNTAX     INTEGER {
                IPv4 (0),
                IPv6 (1)
            }
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object identifies the origin of the packet, that means
                either it is IPv4 or IPv6."
    ::= { natptReceivedPacketInfoEntry 2 }

natptReceivedPacketIPv4Header OBJECT-TYPE
        SYNTAX     BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 1 the value of this object
                will be null."
    ::= { natptReceivedPacketInfoEntry 3 }


natptReceivedPacketIPv4HeaderLength OBJECT-TYPE
        SYNTAX     INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "As IPv4 packet header length is variable, it will give the
                exact size of the packet header."
    ::= { natptReceivedPacketInfoEntry 4 }

natptReceivedPacketIPv6Header OBJECT-TYPE
        SYNTAX     BIT STRING
```

```
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 the value of this object
                will be null."
    ::= { natptReceivedPacketInfoEntry 5}

natptReceivedPacketIPv6HeaderLength OBJECT-TYPE
        SYNTAX      INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 the value of this object
                will be zero."
    ::= { natptReceivedPacketInfoEntry 6 }


    natptReceivedPacketProtocol OBJECT-TYPE
        SYNTAX    NatptProtocolType
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will identify the Transport Layer Protocol of the
                packet."
    ::= { natptReceivedPacketInfoEntry 7}

     natptReceivedPacketIPv4Checksums OBJECT-TYPE
        SYNTAX    BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 1 then the value of this
                object will be null."
    ::= { natptReceivedPacketInfoEntry 8}

     natptReceivedPacketIPv6Checksums OBJECT-TYPE
        SYNTAX    BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 then the value of this
                object will be null."
    ::= { natptReceivedPacketInfoEntry 9}

    natptReceivedPacketIPv4PacketPayload OBJECT-TYPE
        SYNTAX      BIT STRING
```

```
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 1 then the value of this
                object will be null."
    ::= { natptReceivedPacketInfoEntry 10}

    natptReceivedPacketIPv4PayloadLenght OBJECT-TYPE
        SYNTAX      INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will give the exact size of IPv4 packet payload."
    ::= { natptReceivedPacketInfoEntry 11}

    natptReceivedPacketIPv6PacketPayload OBJECT-TYPE
        SYNTAX      BIT STRING

        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 then the value of this
                object will be null."
    ::= { natptReceivedPacketInfoEntry 12}

natptReceivedPacketIPv6PayloadLenght OBJECT-TYPE
        SYNTAX      INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will give the exact size of IPv6 packet payload."
    ::= { natptReceivedPacketInfoEntry 13}

natptReceivedPacketProtocol OBJECT-TYPE
        SYNTAX      NatptProtocolType
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will identify the Transport Layer Protocol of the
                packet."
    ::= { natptReceivedPacketInfoEntry 14}

natptSessionID OBJECT-TYPE
    SYNTAX      NatptSessionId
    MAX-ACCESS  read-create
```

```
    STATUS      current
    DESCRIPTION
        "This object identifies the session of the packet."
    ::= { natptReceivedPacketInfoEntry 15 }


natptSessionPayloadRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The object indicates the status of this conceptual row. Until
        instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        natptSessionPayloadRowStatus column is 'notReady'. None of the
        writable objects except natptSessionTime in this row may be
        modified while the value of this object is active(1)."
    REFERENCE
          "Textual Conventions for SMIv2, Section 2."
    ::= { natptReceivedPacketInfoEntry 16 }

--
-- The NAT-PT Translated Packet Info table
--
natptTranslatedPacketInfoTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatptTranslatedPacketInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "This table holds the payload of each packet for each NAT-PT
        Session"
    ::= { natptMIBObjects 6}

natptTranslatedPacketInfoEntry  OBJECT-TYPE
    SYNTAX     NatptTranslatedPacketInfoEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "Each entry in this table holds the packet payload of each session.
        These entries are lost upon agent restart."
    ::= { natptTranslatedPacketInfoTable 1 }

NatptTranslatedPacketInfoEntry ::= SEQUENCE {
    natptTranslatedPacketID                 NatptPacketID,
    natptTranslatedPacketOrigin             INTEGER,
    natptTranslatedPacketIPv4Header         BIT STRING,
```

```
    natptTranslatedPacketIPv4HeaderLenght      INTEGER,
    natptTranslatedPacketIPv6Header            BIT STRING,
    natptTranslatedPacketIPv6Header Length     INTEGER,
    natptTranslatedPacketProtocol              NatptProtocolType,
    natptTranslatedPacketIPv4Checksums         BIT STRING,
    natptTranslatedPacketIPv6Checksums         BIT STRING,
    natptTranslatedPacketIPv4PacketPayload     BIT STRING,
    natptTranslatedPacketIPv4PayloadLenght     INTEGER,
    natptTranslatedPacketIPv6Payload           BIT STRING,
    natptTranslatedPacketIPv6PayloadLenght     INTEGER,
    natptTranslatedPacketProtocol              NatptProtocolType,
    natptSessionID                             NatptSessionId,
    natptTranslatedPacketMapRowStatus          RowStatus
}


natptTranslatedPacketID OBJECT-TYPE
    SYNTAX      NatptPacketID
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object uniquely identifies the packet of the payload."
    ::= { natptTranslatedPacketInfoEntry 1 }

natptTranslatedPacketOrigin OBJECT-TYPE
        SYNTAX      INTEGER {
                IPv4 (0),
                IPv6 (1)
            }
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object identifies the origin of the packet, that means
                either it is IPv4 or IPv6."
    ::= { natptTranslatedPacketInfoEntry 2 }

natptTranslatedPacketIPv4Header OBJECT-TYPE
        SYNTAX      BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 1 the value of this object
                will be null."
    ::= { natptTranslatedPacketInfoEntry 3 }
```

```
natptTranslatedPacketIPv4HeaderLength OBJECT-TYPE
        SYNTAX     INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "As IPv4 packet header length is variable, it will give the
                exact size of the packet header."
    ::= { natptTranslatedPacketInfoEntry 4 }

natptTranslatedPacketIPv6Header OBJECT-TYPE
        SYNTAX     BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 the value of this object
                will be null."
    ::= { natptTranslatedPacketInfoEntry 5}

natptTranslatedPacketIPv6HeaderLength OBJECT-TYPE
        SYNTAX     INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 the value of this object
                will be zero."
    ::= { natptTranslatedPacketInfoEntry 6 }


    natptTranslatedPacketProtocol OBJECT-TYPE
        SYNTAX     NatptProtocolType
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will identify the Transport Layer Protocol of the
                packet."
    ::= { natptTranslatedPacketInfoEntry 7}

     natptTranslatedPacketIPv4Checksums OBJECT-TYPE
        SYNTAX     BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 1 then the value of this
                object will be null."
    ::= { natptTranslatedPacketInfoEntry 8}
```

```
    natptTranslatedPacketIPv6Checksums OBJECT-TYPE
        SYNTAX    BIT STRING
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 then the value of this
                object will be null."
    ::= { natptTranslatedPacketInfoEntry 9}

    natptTranslatedPacketIPv4PacketPayload OBJECT-TYPE
        SYNTAX     BIT STRING

        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 1 then the value of this
                object will be null."
    ::= { natptTranslatedPacketInfoEntry 10}

    natptTranslatedPacketIPv4PayloadLenght OBJECT-TYPE
        SYNTAX     INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will give the exact size of IPv4 packet payload."
    ::= { natptTranslatedPacketInfoEntry 11}

    natptTranslatedPacketIPv6PacketPayload OBJECT-TYPE
        SYNTAX     BIT STRING

        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "If the value of the packet origin is 0 then the value of this
                object will be null."
    ::= { natptTranslatedPacketInfoEntry 12}

natptTranslatedPacketIPv6PayloadLenght OBJECT-TYPE
        SYNTAX     INTEGER
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will give the exact size of IPv6 packet payload."
    ::= { natptTranslatedPacketInfoEntry 13}

natptTranslatedPacketProtocol OBJECT-TYPE
```

```
        SYNTAX    NatptProtocolType
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
                "This object will identify the Transport Layer Protocol of the
                packet."
    ::= { natptTranslatedPacketInfoEntry 14}

natptSessionID OBJECT-TYPE
    SYNTAX      NatptSessionId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object identifies the session of the packet."
    ::= { natptTranslatedPacketInfoEntry 15 }


natptSessionPayloadRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The object indicates the status of this conceptual row. Until
        instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        natptSessionPayloadRowStatus column is 'notReady'. None of the
        writable objects except natptSessionTime in this row may be
        modified while the value of this object is active(1)."
    REFERENCE
          "Textual Conventions for SMIv2, Section 2."
    ::= { natptTranslatedPacketInfoEntry 16 }

--
-- Notifications section
--

natptMIBNotifications OBJECT IDENTIFIER ::= { natptMIB 0 }

--
-- Notifications
--


natptMIBNotificationGroup NOTIFICATION-GROUP
    NOTIFICATIONS {
        natptPacketID
```

```
        natptIPv4PoolAddr
        natptPacketIPv4ToIPv6
        natptPacketIPv4ToIPv6SrcPort
        natptPacketIPv4ToIPv6DestPort
        natptPacketIPv6ToIPv4
        natptPacketIPv6ToIPv4SrcPort
        natptPacketIPv6ToIPv4DestPort
        }
    STATUS      current
    DESCRIPTION
        "Agent will send notifications to managers using the above group of
        notification objects."
    ::= { natptMIBGroups 6 }

END
```

Security Considerations

The managed objects in this MIB may contain confidential information. The Security Proxy and the NAT-PT will exchange security information among them using MIB objects. Secured Environment is very important consideration for secured NAT-PT. SNMPv1 and SNMPv2 do not provide features for such a secure environment.

We recommend here that the implementers consider the security features as provided by the SNMPv3 framework ([RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

APPENDIX C: Functional Analysis of the existing NAT-PT

Packet originating in IPv4 network destined to IPv6

Transport Layer Protocol/the packet type: TCP, UDP, ICMP etc.

Session state info of the packet: Session initiating, Session Ending and the Packet is in the middle of the session.

TCP

Case Session Initiating

- Drop the packet.

Case Middle of the session

- Check the address-mapping table to find the destination address in the table. If destination address exists, proceed.
- Get the IPv6 address corresponding the destination address from the mapping table.
- Replace the IPv4 destination address with the IPv6 address.
- Create IPv4 source address concatenating PREFIX with IPv4 source address.
- Translate IPv4 header into IPv6 header.

- Check the destination port number to find either is there any ALG or not. If there is any ALG call it to translate the payload according to the protocol specification.
- Write the packet to the destination address

Case Session Ending

Follow all the steps mentioned for the case of Middle of the session, and remove the destination address from the mapping.

UDP

Case Session Initiating

Check the static mapping table and if the destination address exists in the table—

- Get the IPv6 addresses corresponding to the IPv4 destination address from the table
- Replace the IPv4 destination address with the IPv6 address.
- Create IPv4 source address concatenating PREFIX with IPv4 source address. The address format looks PREFIX::IPv4_Address.
- Translate IPv4 header into IPv6 header.
- Check the destination port number to find either is there any ALG or not. If there is any ALG, call that ALG to translate the payload according to the protocol specification.
- Write the packet to the destination address

Case Middle of the session

- Check the address-mapping table to find the destination address in the table. If the destination address exists, proceed.
- Get the IPv6 address corresponding the destination address from the mapping table.
- Replace the IPv4 destination address with the IPv6 address.
- Create IPv4 source address concatenating PREFIX with IPv4 source address.
- Translate IPv4 header into IPv6 header.
- Check the destination port number to find either is there any ALG or not. If there is any ALG call it to translate the payload according to the protocol specification.
- Write the packet to the destination address

Case Session Ending

Follow all the steps mentioned for case Middle of the session, and remove the destination address from the mapping.

ICMP

ICMP packet does not require be checked either it is session initiating or session ending or it is in the middle of the session.

- Check the address-mapping table to find the destination address in the table. If destination address exists, proceed

- Get the IPv6 address corresponding the destination address from the mapping table.
- Replace the IPv4 destination address with the IPv6 address.
- Create IPv4 source address concatenating PREFIX with IPv4 source address.
- Translate IPv4 header into IPv6 header.
- Write the packet to the destination address

Packet originating in IPv6 network destined to IPv4

The transport layer protocol/the packet type: TCP, UDP, ICMP etc. Session state of the packet: Session initiating, Session Ending and the Packet is in the middle of the session.

TCP

Case Session Initiating

Check the mapping table to find either the source IPv6 address of the packet exists in the mapping table or not. There may be two cases—
IPv6 source address may exist in the mapping table. In that case IPv6 source address may exist in the static mapping table or in the dynamic mapping table.

Case Static mapping table

In that case it will be DNS response from IPv6 DNS server.

- Get the IPv4 address corresponding to the IPv6 address from the static mapping table and replace the source IPv6 address with the IPv4 address taken from mapping table.
- As it is DNS response to a DNS query, the destination address must be the form of PREFIX::IPv4_address. So peel off the PREFIX to get the IPv4 destination address, and replace the destination address with that address.

Case Dynamic Mapping table

In that case the packet will be a response for IPv4 packet originated in IPv4 network.

- Get the IPv4 address corresponding to the IPv6 address from the dynamic mapping table and replace the source IPv6 address with the IPv4 address taken from mapping table.
- IPv6 source address may not exist in the mapping table—
- Get an IPv4 address from the pool of IPv4 addresses, and create a mapping between the source IPv6 address and the address taken from the pool.
- Replace the IPv6 source address with the IPv4 pool address.
- Get the IPv4 DNS server address from the static mapping, and replace the destination address with the IPv4 DNS server address.

- Translate IPv6 header into IPv4 header.
- Check the destination port number to find either is there any ALG or not. If there is any ALG call it to translate the payload according to the protocol specification. If the packet is a DNS response then get IPv6 address of the queried node from the answer section of the DNS payload.
- Get an IPv4 address from the address pool and create a mapping between the IPv6 address of the answer section and IPv4 pool address. Replace that IPv6 address with IPv4 address.
- Write the packet to the destination address

Case Middle of the session

- Check the dynamic address-mapping table to find the source address in the table. As the packet is in the middle of a session, the source address should be in the address-mapping table. If the source address exists, proceed as below otherwise drop the packet.
- Get the mapped IPv4 address corresponding to the source address from the mapping table.
- Replace the IPv6 source address with the IPv4 address.
- Create IPv4 destination address peeling off the PREFIX from the IPv6 destination address of the packet.
- Translate IPv6 header into IPv4 header.

- Check the destination port number to find either is there any ALG or not. If there is any ALG call it to translate the payload according to the protocol specification.
- Write the packet to the destination address

Case End of the session

- Check the dynamic address-mapping table to find the source address in the table. As the packet is in the end of a session, the source address should be in the address-mapping table. If the source address exists, proceed as below otherwise drop the packet.
- Get the IPv4 addresses corresponding to the source address from the mapping table.
- Replace the IPv6 source address with the IPv4 address.
- Create IPv4 destination address peeling off the PREFIX from the IPv6 destination address of the packet.
- Translate IPv6 header into IPv4 header.
- Check the destination port number to find either is there any ALG or not. If there is any ALG call it to translate the payload according to the protocol specification.
- Write the packet to the destination address
- Remove the mapping entry from the table and return the pool address to the IPv4 pool address.

ICMP

- ICMP packet does not require to check either it is session initiating or session ending or it is in the middle of the session.
- Check the address-mapping table to find the source address in the table. If source address exists, proceed
- Get the IPv4 address corresponding to the source address from the mapping table.
- Replace the IPv4 destination address with the IPv6 address.
- Create IPv4 destination address peeling off the PREFIX from IPv4 destination address.
- Translate IPv6 header into IPv4 header.
- Write the packet to the destination address.