國立交通大學

網路工程研究所

碩 士 論 文

一個提供快速網路位置轉換之代理行動 IP 架構

A Proxy Mobile IP Architecture with Fast Network Address Translation

研 究 生：林振全

指導教授：陳耀宗　教授

中 華 民 國 九 十 九 年 七 月

一個提供快速網路位置轉換之代理行動 IP 架構
A Proxy Mobile IP Architecture with Fast Network Address Translation

研 究 生：林振全　　Student：Chen-Chuan Lin

指導教授：陳耀宗　　Advisor：Yaw-Chung Chen

國 立 交 通 大 學
網 路 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

July 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年七月

一個提供快速網路位置轉換之代理行動IP架構

學生：林振全　　　　　　　　　指導教授：陳耀宗 博士

國立交通大學網路工程研究所

摘要

近年來隨著各種無線網路技術的發達以及行動裝置的普及，人們可在這些裝置上透過無線網路存取各種服務，例如線上影音串流。另一種需求也油然而生－不但要能在固定的位置無線上網，還要在上網的同時移動而不受影響。代理行動IP (Proxy Mobile IP, PMIP)可讓行動裝置於不同的子網路中漫遊而不造成上層的連線中斷，且不需對行動裝置多作修改。不過其標準僅支援點對點的網路類型，而目前最普遍的 IEEE 802.11網路則屬於廣播型網路。另一方面，在目前IPv4可用位址不足的情況下，另外提供IP位置給行動裝置是一大難題。NAT (Network Address Translation)是目前最常使用的解決方案，卻不與MIP (Mobile IP)或PMIP相容。本論文提出一種名爲NAToD (Network Address Translation on Demand) 的新式跨階層的NAT架構，可解決原本MIP/PMIP與NAT不相容的問題，並可在 IEEE 802.11網路下運作。本架構僅需更新網路存取點(Access Point)的軟體即可讓行動裝置擁有IP的可移動性，不需修改行動裝置，可降低建置的成本及時間。本架構亦可在IP轉送時略過部分流程而節省時間。從實驗的結果顯示，在大部分情況下NAToD可得到較佳的效能。

關鍵字：行動IP、代理行動IP、網路位置轉換、IEEE 802.11

A Proxy Mobile IP Architecture with Fast Network Address Translation

Student: Chen-Chuan Lin　　　　Advisor: Dr. Yaw-Chung Chen

Institute of Network Engineering
National Chao Tung University

ABSTRACT

A variety of wireless technologies have been popularized in recent years and enable users to connect to the Internet from almost everywhere. This brings the demand on the user mobility that allows a mobile device roams from one place to another without any application disruption. Proxy mobile IP (PMIP) provides network based mobility management to support unmodified mobile device but it cannot apply on non point-to-point networks suck as IEEE 802.11. On the other hand, the demand for IP addresses by mobile terminals can be solved by NAT which is unfortunately not compatible with MIP/PMIP. In this thesis, we propose a novel cross-layer network address translation scheme called Network Address Translation on Demand (NAToD) and integrate it with proxy mobile IPv4. The scheme allows PMIP running on the most popular IEEE 802.11 networks, and it only requires software upgrading on the AP itself. Doing this way, we can reduce the deployment cost as well as shorten the system deployment time. The experiment result shows that NAToD achieve better performance in most cases.

Keywords: Mobile IP, Proxy Mobile IP, NAT, IEEE 802.11

Acknowledgement

這篇論文能夠順利地完成，首先要感謝我的指導老師陳耀宗教授，老師在研究上的指導讓我受惠良多，並且給予極高的自由度，讓我接觸了不同的領域、增廣見聞。同時感謝詹家泰教授、留忠賢教授與蔡文能教授在口試時給予意見，使論文更加完整。

接著要感謝實驗室的阿華學長、小郭學長、文康學長及育嘉學長，平時給我研究及生活上的指點，特別要感謝文康學長，在本論文上提供了不少寶貴的意見。另外要感謝我的同學這兩年一起奮鬥打拼、互相扶持，給我不少的幫助。

此外我還要感謝在系計中的工作伙伴，除了工作上的交流外，還時常開大食團及掃雷團，度過了不少歡樂的時光。最後要感謝這兩年曾經幫助過我的人，任何的幫助都對我有深遠的影響，我由衷的感謝你們。

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, the population of mobile IP-based devices like PDA, netbook and smartphone have grown rapidly. These devices can connect to the Internet through a variety of wireless network technologies (e.g. Wi-Fi, WiMAX, 3G). Real-time applications like VoIP and video streaming together with wireless technologies make users' daily life more convenient. Without mobility, users need to stay in a fixed location to have a persistent connection to the Internet. Most Internet applications suffer service disruption while user is in moving. For example, IPTV will get stuck or have frame skipped. In the worst case, the user needs to reconnect to the service manually. These problems make a user waiting for a long service disruption time. This issue is critical to real-time applications.

To provide mobility for a terminal host, many approaches have been proposed to provide continuous service while a mobile terminal changes its point of attachment. Mobility management can be taken place on different layers. Application layer approach such as Session Initiation Protocol (SIP) [12] maintains an application session while a mobile node changes its IP address. But this approach requires modification to the application software. Network layer approach like Mobile IP [11][5] and Proxy Mobile IP [3] [9] provide a fixed global IP address to maintain upper layer sessions and without involving the application. Mobility in data-link layer is like intra-ASN mobility in cellular network. The advantage of network layer mobility is that it does not require modifying and redesigning the application software. It is not only supported on cellular network, but also available on other types of wireless networks.

Internet Engineering Task Force (IETF) has defined several network layer mobility management protocols such as Mobile IPv4, Mobile IPv6, Proxy Mobile IPv6 and recently completed Proxy Mobile IPv4. Some extension like Fast Handover [7][6], Route Optimization [1] and Hierarchical Handover [13] are also introduced to improve the performance or to reduce service disruption time during handover. Most of them require the mobile node to involve the handover process as well as to modify mobile node's network stack. These requirement might become a barrier for the deployment with only a short time.

The network based mobility management protocol like PMIP does not require, involving with a mobile node. It can be deployed by upgrading software to the infrastructure. IEEE 802.11 WLAN is the most popular Internet access technology for home, company, campus and even for the entire city. The coverage of WLAN can be widen by increasing numbers of access points. Unfortunately, the current PMIP standard is not suitable for IEEE 802.11 because PMIP can only support point-to-point access link, while IEEE 802.11 uses broadcast access scheme.

In another aspect, the demand of IP addresses is increased for more and more mobile terminals accessing the Internet. Each mobile node needs at least one IP address to access the Internet and requires more to operate with mobile IP. Currently, IPv4 address space is almost exhausted, and it faces a tremendous demand on IP address. Two solutions have been applied, one is upgrading the current IPv4 into Internet Protocol version 6 (IPv6), the other is applying the Network Address Translation (NAT) [14]. Until now, the deployment of IPv6 is still slow. Google has measured that less than 1% hosts are using IPv6 worldwide in 2008 [8].

In this thesis, we proposed a PMIPv4 scheme that can be applied on the most popular IEEE 802.11 wireless network and integrated with NAT to support mobility with merely a little demand on IP address. Our proposed scheme is one of the most feasible IP mobility solutions in the current network environment.

This thesis is organized as follows: In Chapter 2, we introduce how mobile IPv4 and proxy mobile IPv4 work, and describe the problem we face when combining MIP/PMIP with NAT. In Chapter 3, we explain the proposed NAToD scheme and how it works with PMIP and operates with different access link technologies. In Chapter 4, we implement the proposed scheme on Linux kernel and perform some experiment to verify that the proposed scheme can provide mobility for an un-modified MN. We also compare the performance of proposed NAToD and original NAT implementation. Finally we give a conclusion in Chapter 5.

Chapter 2

Background

When a node changes its point of attachment (POA) to the Internet, it needs to change its IP address as well if the new link is not in the same subnet. Once an IP address is changed, the upper layer connection will be broken. The Mobile IP and Proxy Mobile IP enable a mobile node to change its point of attachment without changing its global IP address.

2.1 Mobile IP

Mobile IP is a host-based mobility management protocol. A host needs to modify its TCP/IP stack to support Mobile IP. An MN uses two addresses to connect to the Internet, Home Address (HoA) and Care-of-Address (CoA). An MN always uses HoA as the address for L4 protocol to communicate with Corresponding Node (CN) no matter where it resides. When an MN is in the home network, it picks a Home Agent(HA) to provide mobility. At this time, the MN does not need any help from HA, it directly sends packets to CN via the access router in the home network.

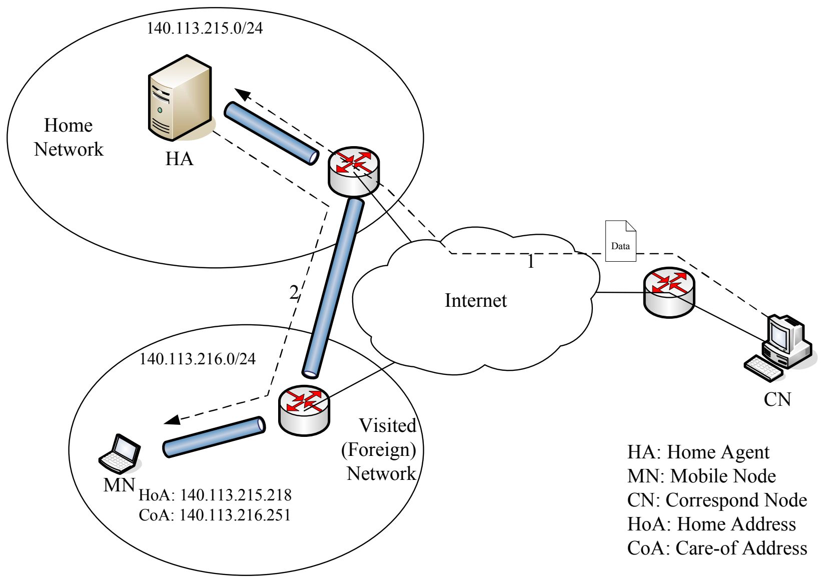


Figure 2.1 Mobile IPv4 architecture on co-located care-of-address mode

When an MN is away from home, it should get a new address to connect to the Internet. The address is called care-of-address. An MN registers its current position to its HA, which intercepts all the packets destined to MN's HoA, and forwards them to MN's CoA through IP tunnel as shown in Figure 2.1. On the opposite direction, MN sends packets to CN directly or it tunnels packets back to HA first. There are two operating mode in MIPv4: foreign-agent care-of-address mode and co-located care-of-address mode.

The latter one operates like MIPv6, it does not require any help from Foreign Agent (FA). When making registration, the MN gets a global unique CoA and communicates to HA by itself instead of by FA. The HA creates an IP tunnel to the MN which is the end point. Since IPv4 address is limited so the other mode may be applied. FA CoA mode lets many MNs share a single CoA. The MN asks FA to register to its HA which creates a tunnel to FA and forward packets to FA, and then FA forwards packets to MN based on the inner IP header. In this mode, it requires installing an FA in every visited network. For both modes of MIPv4, MN sends packets to CN directly without having to tunnel packets back to HA.

2.2 Proxy Mobile IP

Proxy Mobile IP provides a network-based mobility. The mobility requirement is done by the network itself and the MN does not need to involve in handover process, so it can support unmodified mobile node. The proxy mobile IPv4 standard was released in early 2010. The PMIP architecture is as shown in Figure 2.2.

The mobility is accomplished by two network entities – Home Agent and Proxy Mobility Agent (PMA) [1]. HA is placed in MN's home network, it intercepts all the packets destined to MN from home network's router even if MN is at home. The PMA acts like FA in MIPv4, when MN is attached to a PMA, PMA registers the current CoA to HA. The CoA is called Proxy CoA which be shared by multiple MNs. Then HA forwards the packet to PMA through a bi-directional tunnel. Finally, the packet reaches MN. For outgoing traffic, PMA acts as MN's default router, instead of forwarding packet to CN directly, PMA forwards the packet back to HA first, and then HA forwards the packet to CN.

PMIP allows MN to only detect L2 change, but still in the same L3 network. To do so, the PMA/MAG emulates MN's home network. In IPv6 MAG "unicast" Router Advertisement with MN's home prefix; In IPv4, DHCP is the most common address configuration tool, and DHCP server always assigns the same IP for a specified MN in a PMIP domain. Internet Protocol Control Protocol (IPCP) can also be used in PMIPv4 environment. Both PMIPv4 and PMIPv6 standards can only support point-to-point access links.

[1]IPv6 uses the term Local Mobility Anchor (LMA) and Mobile Access Gateway (MAG)

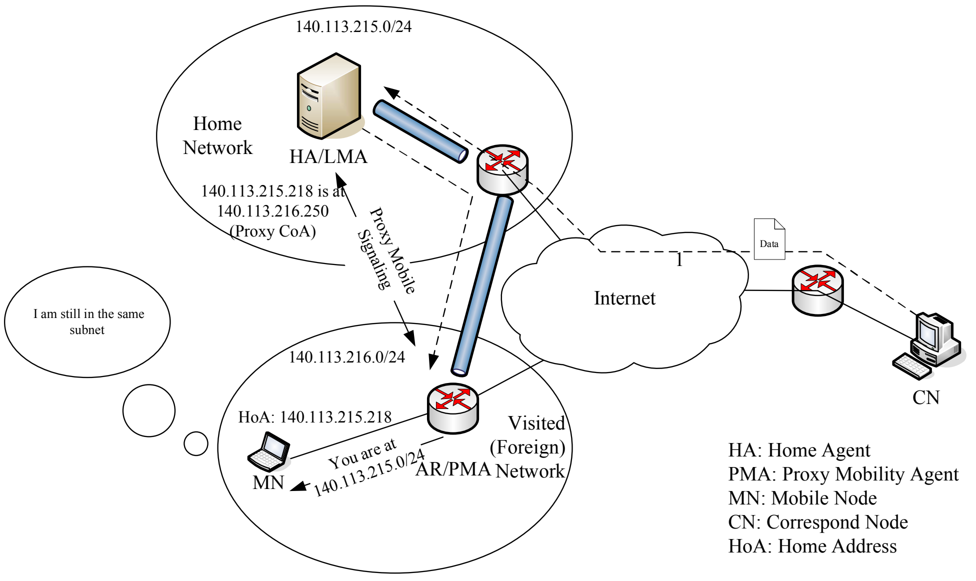


Figure 2.2 Proxy mobile IP architecture

2.3 Network Address Translation

NAT maps IP address from one domain to another, usually from private-use range to public IP. It is usually as a solution to save IPv4 address. The NAT serves N internal hosts and maintains a public IP pool which contains M IP addresses. The N hosts seldom get online concurrently so the number of M is usually less than N. When an internal node starts a connection, NAT picks an IP address from the pool and assign it to the host and all the further connection for the same host. When the internal host shutdown or does not communicate with external host for a while, the mapping will expire and the assigned public IP will be returned to the pool.

In some environment, especially Local Area Network (LAN), where only a single public IP address is allowed, mapping several private IP addresses into a single one might cause L4 port conflict and makes the upper layer connection broken. Port Address Translation (PAT) is introduced which will also translate L4 port number to avoid this problem. PAT is a kind of extension of NAT and the translation table is larger than NAT and operation is more complex. In the following sections, we use the term "NAT" to stand for PAT.

2.4 Interoperate with NAT

Mobile IP requires two IP addresses for each MN. One is a permanent address on MN's home link and the other one is a temporary address – care-of-address. In current IPv4 network, IP address is almost run out. The most common way to solve address shortage is NAT, which makes several MNs to share one single home address or care-of-address. MIP is incompatible with NAT in some environment. Table 2.1 is the comparison and in the following sections, we will explain in detail why it does not work.

Table 2.1 Comparison among MIPv4, PMIPv4 and NAT

Type	Mode	NAT on HoA	NAT on CoA
Mobile IP	Co-located CoA	X	X
	FA CoA	X	O
Proxy Mobile IP	PPP-link	O	N/A
	Broadcast-link	X	

2.4.1 Mobile IP with NAT on Home Address

One way is to do a NAT translation on home address, that is, multiple MNs share a single home address. When an incoming packet is intercepted by home agent, HA will lookup the NAT table and modify the IP header to change destination IP address, and change L4 header

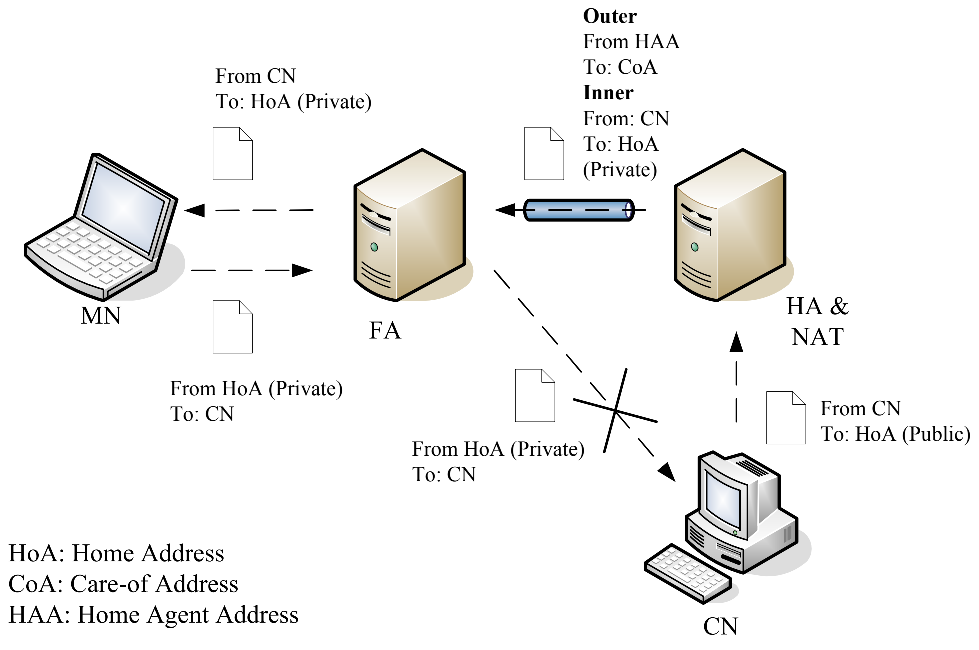


Figure 2.3 MIPv4 uses FA-CoA with NAT

if necessary, and finally tunnel the packet to MN's care-of-address.

It looks trivial, but this way is not feasible in MIPv4 environment. For the packet sent by the MN, instead of tunneling back to HA, the standard IP routing is applied. The packet will not pass the NAT device in the home network and an IP datagram with private IP address will be forwarded to the Internet. It is illegal for a packet with private IP address traversing through the Internet as shown in Figure 2.3.

When using foreign agent care-of-address mode of MIPv4, one solution is that the FA should also be responsible for translating packets sent by MN. To synchronize NAT table with HA and FA, the overhead of control messages is too large to be acceptable. When a FA wants to add a new translation entry into the table, it must communicate with the original NAT on HA to check whether the new translation is conflicted or not.

For co-located care-of-address mode, the MN is the end point of the tunnel, the outgoing packet will be forwarded to CN instead of via FA. In this mode, MN must support address translation and communicate with HA to synchronize NAT table. It is more complex than FA-CoA mode.

2.4.2 Mobile IP with NAT on Care-of-Address

The other solution in Mobile IPv4 is to use private IP address for care-of-address. MIPv4 has two type of care-of-address. For co-located care-of-address, each MN in the same foreign network obtains a private IP address, and registers that address to their own home agent.

The FA-CoA mode is designed to relief the address requirement in the limited address space. Private IP address can be applied on MN's access link. Once an FA received an incoming packet, it forwards the packet based on MN's address on the access link. Both public and private IP address can be applied on the access link because that address is only used between MN and FA. When MN sends a packet, it uses its home address as source and CN's address as destination and forwards the packet to CN via FA. The address of that link will never exist on any packet in the Internet. So, FA-CoA mode can support NAT on the Foreign Agent.

For Co-located CoA mode, if an MN is behind an NAT with an assigned private IP address and registers to HA with the private IP address. The NAT or FA must rewrite the registration request to let HA know MN's corresponding public IP. The protocol works fine since then, but once the HA creates a tunnel to MN, it will encounter failure. The IP-in-IP tunnel is simply an outer IP header combined with the other inner IP header, it does not contain any identification like port in TCP and UDP or any sequence in ICMP. Unmodified NAT router cannot translate such header format and the tunnel will be broken as shown in

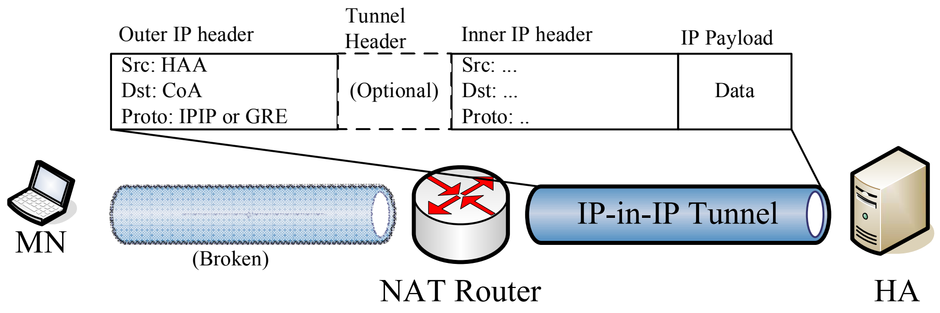


Figure 2.4 IP tunnel cannot cross NAT router

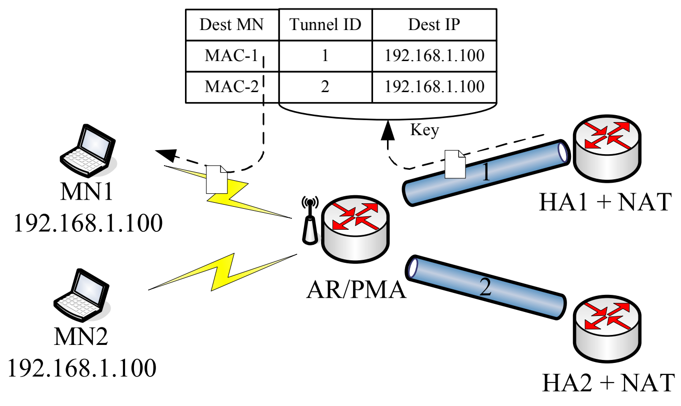
Figure 2.4.

One solution is using Generic Routing Encapsulation (GRE) tunnel (IP protocol type 47), it contains an optional key field to identify different tunnels. But most NAT only support TCP, UDP, and ICMP packets, they cannot translate GRE tunnel. Strictly speaking, MIPv4 cannot work with NAT on care-of-address in co-located CoA mode.

2.4.3 Proxy Mobile IP with NAT (on Home Address)

For Proxy Mobile IP, an MN only uses one IP address – Home Address and every outgoing packet will be tunneled back to HA before being forwarded to CN. Every packet can be translated correctly without the problem from NAT with MIP. So, PMIP can operate well with NAT on MN's home address.

Let's consider the following scenario. Two mobile nodes come from different home network, each of them has been assigned a private IP that happens to be the same, but they share different public IP with other MNs. IP address conflict occurs in this situation. Although addresses conflict, the AR/PMA can still distinguish each MN by their L2 addresses. AR/PMA creates a map between tunnel ID and MN's L2 address (and which interface the



Dest MN	Tunnel ID	Dest IP
MAC-1	1	192.168.1.100
MAC-2	2	192.168.1.100

Figure 2.5 PMIP with NAT

MN connects in if needed) as shown in Figure 2.5. By looking up that table, the packet can be forwarded correctly for both incoming and outgoing traffic.

For point-to-point access link like WiMAX IP convergence sublayer, private address conflict does not affect MN because their traffic through base station to access router is separated, and Address Resolution Protocol (ARP) will not apply in this link because it is not necessary. As a result, mobile nodes will not detect address conflict by broadcasting ARP packets.

On the other hand, broadcast link like IEEE 802.11 and Ethernet cannot apply to this scenario. Each MN will detect address conflict and most OSs will show the warning message. Although PMIP can still work but it will result in with very poor quality of experience to users.

2.5 Related Works

Zhen et al. [15] proposed a scheme to share home address by multiple MNs. The MN translates packets for both incoming and outgoing traffic and does not have to maintain a global translation table. The basic idea is that each MN is assigned the same HoA with different port ranges. This allows MN to translate packet itself without conflict with other MNs. When MN starts a session, the Mobile IP layer maps port into assigned port range and sends the packet. For the incoming packet, the HA intercepts the packet and forwards it to MN based on the port range.

The main drawback is that each MN is only allowed a limited number of concurrent sessions, depending on the pre-allocated port range. If an MN wants to maintain more sessions, it needs an extension to allow MN to ask more port ranges after initialization. When an MN is at home, it might detect address confliction because all of the MNs are set to the same address.

The mobile IP NAT traversal draft[10] uses UDP datagram to traverse through NAT when an MN is in a foreign network using private IP (MIP + NAT on co-located CoA mode). When an MN sends a registration request, it creates an IP/port mapping on the NAT, and HA can send a packet with the same port to MN to pass though the NAT. In this extension, HA makes the IP tunnel over UDP packet, and sends to MN by the previously created NAT hole. The keep alive feature is very important, otherwise the mapping on the NAT might expire and will break the IP-in-UDP tunnel.

MobileNAT[2] is an architecture which combines NAT with Mobile IP and supports micro and macro-mobility. This scheme allows HoA and CoA to be any combination of public/private address and also allows an MN moving from NAT domain to non-NAT domain. This requires both MN and FA modification to support NAT because translation might occur on MN, FA, or both.

Although the above proposed schemes are trying to save IP usage in the MIP environment, the additional modification has to be taken on MN, HA, NAT router or all above. This makes much difficulty for the deployment to every MN. So, in the next chapter, we propose a network based mobility architecture that can be applied to NAT to relieve the demand of the IPv4 address for mobile terminals. This scheme was originally proposed in [4], in this thesis, we have implemented this scheme on Linux with a variety of experiments.

Chapter 3

The Proposed Scheme

In this chapter, we propose a novel NAT scheme called NAToD which can be installed on every HA. MNs are behind the NAToD without knowing the existence of the NAToD. Unlike PMIP, in which every MN has its own unique home address, HA assigns all MNs with the same IP address, HA also uses this address on its external interface. As mentioned previously, our scheme is based on PMIPv4, all signaling flows are almost the same. In most cases, address translation will not occur and packets pass the NAToD directly.

Just like PMIP, the incoming traffic from CN to MN is received by HA first, a NAToD translation will be performed if necessary and then the packets will be forwarded to PMA through a bi-direction tunnel. Finally, PMA forwards the packet to MN on the access link. On the other hand, outgoing traffic will be forwarded to HA first, and performed a translation before being forwarded to CN. The proposed NAToD does not require any modification to the MN.

3.1 Network Address Translation on Demand

The proposed NAToD uses the same IP address for the NAToD router and all the hosts under it. Although these hosts use the same IP address, the router can still distinguish between them by MAC address. In other words, NAToD maps multiple MAC addresses into a single IP address. Figure 3.1 is an NAToD example in Ethernet environment. The NAToD router does not occupy any additional IP address and does not require configuring any IP address on its internal interface.

When a host connects to the NAToD router and requests for an IP address, DHCP protocol will be applied. This DHCP server is installed on the NAToD router and will assign the same address for every client instead of picking up an unused IP in a pre-configured pool. Although the default router of internal hosts should be the NAToD router, DHCP cannot configure this way because the NAToD router does not have any IP address on the internal interface. The default router must be in the same network of configured IP address and netmask, so we cannot set a private IP address in the internal interface as client's default router's IP.

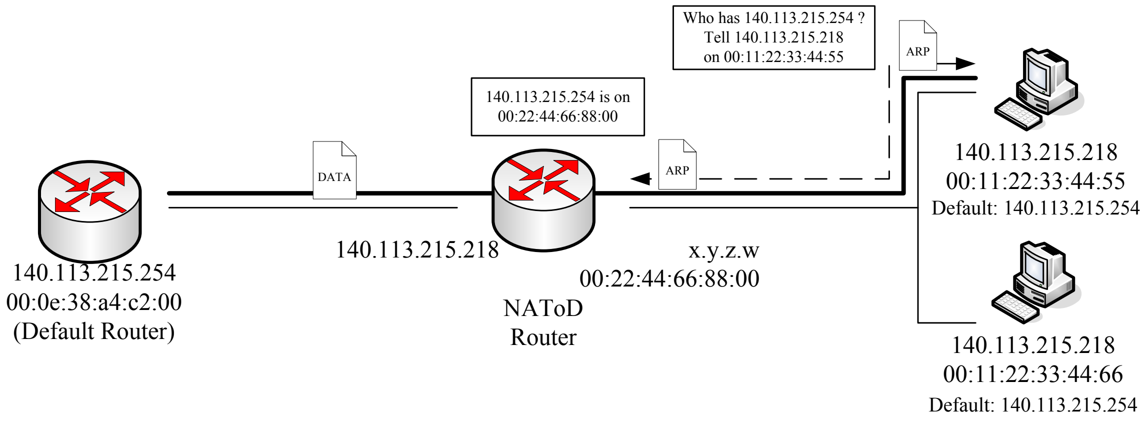


Figure 3.1 NAToD network architecture

Instead, all hosts will be told that their default router is the external router (NAToD's default router) even the external router, e.g. 140.113.215.254 in Figure 3.1 is not on the internal link, the NAToD router can still capture internal hosts' outgoing traffic by Proxy ARP. This means that the NAToD router is transparent to internal hosts. Also, the NAToD scheme has to apply some technique so that each internal host will not detect address conflict and this will be discussed in Section 3.5.

To fit NAToD, the translation table has to be extended with more fields to contain link layer information as shown in Figure 3.2. The new field we introduced is the source MAC address and source interface. Source MAC is used to identify each host. Source interface-id is also necessary, and it becomes part of binding cache when combined with PMIPv4 and we will discuss it in Section 3.2.1. This can also support hosts attached from different internal interfaces. The external IP field has become unnecessary because NAToD router keeps the source IP address unmodified and does not need to pick an external IP for any session.

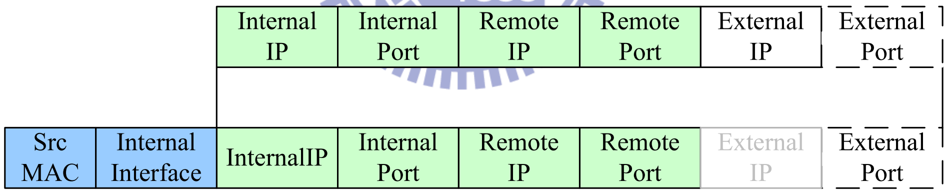


Figure 3.2 Comparison of NAT and NAToD translation table structure

When an internal host starts a new session, it transmits a packet to NAToD router. The router gets a {SrcMac, SrcIP, SrcPort, DstIP, DstPort} tuple from packet's header. And then it checks whether this tuple has already existed in the NAToD table. If not, NAToD inserts this tuple into NAToD table with received interface ID, and forwards the packet to the external interface. In this case, all headers above the network layer are unchanged.

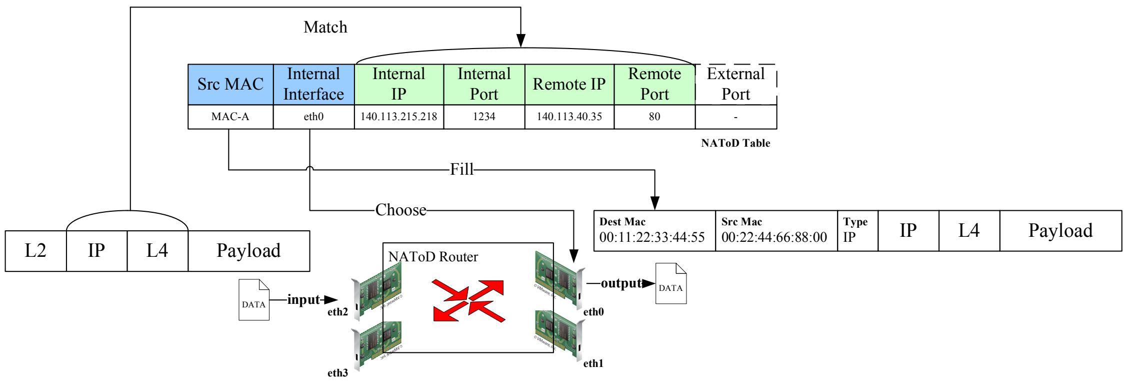


Figure 3.3 NAToD incoming packet flow

If the tuple has already existed, the collision will happen. That is, more than one internal hosts use the same port to connect to the same external host and port. The NAToD router then needs to search the table, chooses an unused port to translate the packet, and then recomputes the checksums of IP and L4 header. Finally, the packet is passed to routing routine and forwarded as a normal packet.

For incoming traffic, NAToD router checks the NAToD table to find out whether the corresponding record exists. If the record existed, the packet needs to be passed to internal host and packet's header has to be translated if necessary, otherwise the packet has to be dropped. We do not handle this packet in a traditional way. Since the destination address equals to NAToD's external IP address, the routing function will pass the packet to the upper layer instead of forwarding it.

The NAToD will cover both routing and address resolution phase for the incoming packet. All the information needed by these two functions has already been stored in the NAToD table – interface and link layer (MAC) address. Once we get the interface information, the routing decision can be made. Address resolution is not necessary, because it is already presented in the NAToD table. In short, NAToD function made both the routing decision and filling the L2 header, the packet is ready to be transmitted on the internal interface as shown in Figure 3.3.

Src MAC	Internal Interface	Internal IP	Internal Port	Remote IP	Remote Port	External Port
MAC-A	eth0	140.113.215.218	1234	140.113.40.35	80	-
MAC-B	eth0	140.113.215.218	1234	140.113.40.36	80	-
MAC-C	eth0	140.113.215.218	1234	140.113.40.35	80	5678

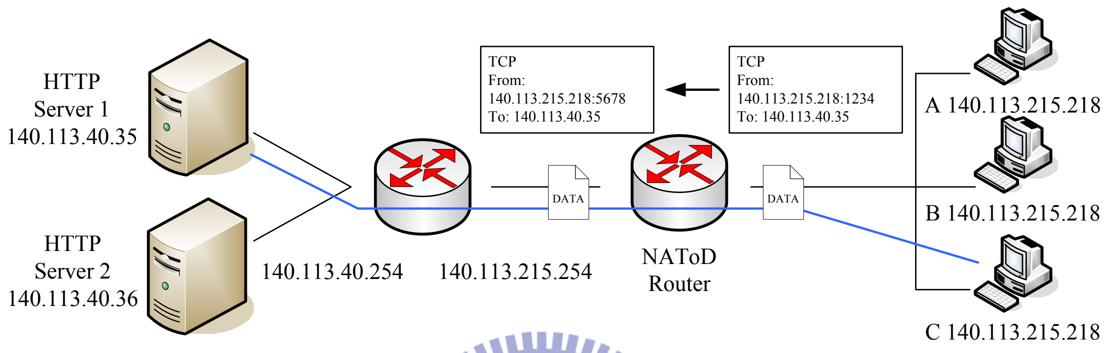

HTTP Server 1 140.113.40.35
HTTP Server 2 140.113.40.36
140.113.40.254
140.113.215.254
DATA
NAToD Router
DATA
TCP From: 140.113.215.218:5678 To: 140.113.40.35
TCP From: 140.113.215.218:1234 To: 140.113.40.35
A 140.113.215.218
B 140.113.215.218
C 140.113.215.218

Figure 3.4 Example of session conflict

Figure 3.4 is an example scenario of session conflict. There are three hosts under a NAToD router. Host A starts a HTTP session with an outer HTTP server (destination port 80) by a randomly selected source port 1234, and default destination port 80. This tuple does not exist in the NAToD table, the router adds the tuple and forwards the packet to HTTP server without modifying any IP or TCP header. Host B connects to another HTTP server and the randomly selected source port is the same. The two tuples are not conflicted because the destination IPs are different. The packet for the second HTTP connection can also pass the NAToD router without modification. Host C starts a HTTP session to the first HTTP server, the randomly selected source port is still the same – by chance. In this case, NAToD router picks a random port, e.g. 5678, to replace the original source port, and add the tuple together with the information of port replacement into the translation table.

3.2 Proxy Mobile IPv4 with NAToD

The proposed scheme combines PMIPv4 with NAToD. A single home address is shared by a number of mobile nodes in NAToD scheme. Every packet from MN to CN is forwarded to HA first and checked whether that session needs to be translated or not. We use DHCP as address configure mechanism and every PMA is equipped with our modified DHCP server (instead of DHCP Relay mode which can also be applied in PMIPv4).

3.2.1 Binding Cache and NAToD Table

Binding Cache/List keeps tracking the association between MN's home address and care-of-address. HA receives registration request from PMA and updates the corresponding entry with the new care-of-address. When MN sends a packet and forwards it to HA, HA look up the binding cache to validate the packet. For the incoming packet, HA forwards packet to PMA based on the care-of-address provided by binding cache. HA checks whether the packet is forwarded from care-of-address for outgoing traffic.

The proposed scheme only allows sessions initiated from the MN. For a valid incoming packet, there must be an existing translation entry which contains the forwarding information. If the entry does not exist, the packet should be illegal and dropped by HA. It is not necessary to look up binding list. The only time it needs to look up Binding list is when a session starts. The NAToD router inserts a new tuple into the translation table, it must check whether the SrcMac and the source of tunnel (care-of-address) is matched in the binding list or not. Once a tuple is created, every packet of that session will match this translation entry and will be valid.

When a binding entry is expired or de-registered by PMA, the HA iterates through the translation table and deletes all the entries with that MN's MAC address. From now on, packet involved with that MN will be dropped.

3.2.2 Access Router/Proxy Mobility Anchor

In PMIP environment, PMA is responsible for emulating MN's home network. The DHCP server on the PMA gets MN's home address through AAA message and it is assigned to MN no matter where the MN is. DHCP server and PMA are usually combined with AR. MN always gets the same IP address from different DHCP server and feels that it is still stayed in the home network. For broadcast link like Ethernet or IEEE 802.11, when an MN sends ARP request for its gateway's IP, the PMA replies proxy ARP message for MN, so the MN will forward all its outgoing packets to PMA without knowing that L3 has changed. For point-to-point link, ARP is not necessary and MN will always forward packets to PMA.

To prevent MNs from detecting each other using their own IP address, the Base Station (BS) should block all broadcast ARP packets between each MN, and only forwards broadcast ARP packets to AR.

3.2.3 Ethernet frame over GRE Tunnel

MIP or PMIP uses IP-in-IP or IP over GRE as the bi-directional tunnel between HA and AR. When a packet is received from the tunnel, AR removes the outer header and finds out which MN is the destination through the inner IP header. The NAToD uses the same IP so that if two MNs belong to the same HA under the same AR, the AR is unable to distinguish which MN is the correct destination.

One solution is that the AR also keeps track with the MAC/IP/Port tuple. But it may require an extra effort, and the occurrence of port collision may occur in the AR, and cause the AR to do NAToD translation too. The additional translation brings extra overhead, and in the worst case, collision happens both on AR and on HA side.

To solve this problem, we use Ethernet frame as the payload of GRE tunnel instead of IP (Transparency Ethernet Bridge, ethertype 0x6558). The AR can get MN's MAC address from the Ethernet header and forwards the packet to MN's access link. If the access link is IEEE 802 type link, the frame can be forwarded without any modification. If the link is not in IEEE 802 family, like PPP, AR assigns a pseudo MAC address to identify that MN, and adds a table to map Ethernet frame to that access link's frame header.

3.2.4 Stateless Forwarding

Before forwarding packet from MN to HA, the AR must look up binding list to fetch MN's home agent address. In our proposed scheme, MN's IP address is also home agent's IP address, AR can directly copies MN's IP address from the source field of inner IP header to the destination of the outer IP header without having to query any other table.

In the opposite direction, if the access link is also an IEEE 802.3 link, the AR can forward the payload of GRE packet – the original IP packet with Ethernet header to internal interface, without doing any additional table lookup or modify packet's linker layer header. This means PMA does not require keeping any state about MNs as shown in Figure 3.5.

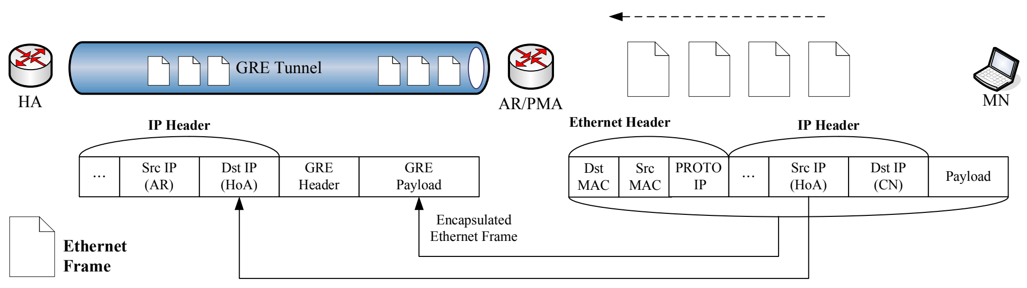


Figure 3.5 Ethernet frame over GRE tunnel & stateless forwarding

3.3 Signaling Flow

The initial attachment signaling flow is described as follows.

1. MN establishes a L2 link with the BS and performs L2 authentication/authorization. At the same time, the AAA client exchanges AAA messages with AAA server and downloads the profile of MN including MN's home address.

2. After L2 link is established, the MN acquires L3 address by DHCP and sends a DHCP discover message.

3. When DHCP server receives DHCP discover, it sends a DHCP offer to MN with MN's home address from the profile which was obtained previously. Note that normal DHCP server will perform a Duplicated Address Detection (DAD) before sending DHCP offer. But in NAToD, we do not need it because IP address is duplicated for some MNs.

4. The MN makes sure to use this address, it sends a DHCP request to confirm the offered IP address.

5. Instead of sending a DHCP ACK immediately, DHCP server sends a Proxy Registra-

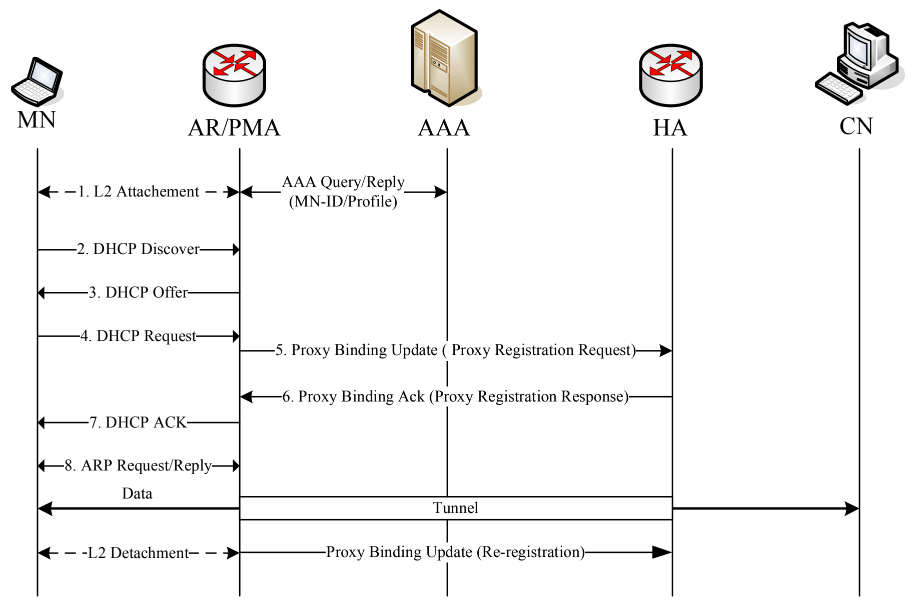

MN
AR/PMA
AAA
HA
CN
1. L2 Attachement
AAA Query/Reply (MN-ID/Profile)
2. DHCP Discover
3. DHCP Offer
4. DHCP Request
5. Proxy Binding Update (Proxy Registration Request)
6. Proxy Binding Ack (Proxy Registration Response)
7. DHCP ACK
8. ARP Request/Reply
Data
Tunnel
L2 Detachment
Proxy Binding Update (Re-registration)

Figure 3.6 PMIP4 with NAToD signaling when MN is attached first time

tion Request (PRRQ) message to HA.

6. The HA receives and confirms the PRRQ, updates binding entry for the MN, and sends Proxy Registration Response (PRRP) message.

7. After received a PRRP, DHCP Server sends DHCP ACK to MN now. At the same time, both HA and AR creates a bi-directional tunnel to each other, and is ready to forward packet for that MN.

8. The MN can send and receive packets now if the access link is a PPP link, or send a ARP request for the default router's MAC address if the link is a multiple access link like that in IEEE 802 family.

The MN might move its position after attached to the serving BS. When the MN detects that the current BS's received signal strength decreased to a threshold, it starts handover procedure. How does an MN choose target AP/BS is out of the scope in our proposed scheme, instead, we focus on the L3 handover process. After MN decided the target BS, MN disconnects L2 connection with old-BS and p-PMA detects the MN detached event, and sends a De-Reg Request to HA.

1. MN establishes a L2 link with BS like step one in the previous signaling steps.
2. MN sends a DHCP Request to detect whether the L3 condition has been changed or not, and confirms whether the MN can use the original address (home address).
3. DHCP server sends PRRQ to ask HA to update its binding cache to new-AR.
4. If HA confirms that PRRQ. It looks up the NAToD table, updates all the entry corresponding to MN with the interface field to the new tunnel interface, and then HA replies PRRP to DHCP Server. From now on, all of the incoming packets will be delivered to the new AR through the new tunnel.
5. After receiving PRRP, the mobility function has been done by the AR. DHCP server sends DHCP ACK to MN, allows MN to use its original IP, and then MN thoughts it's still in the same subnet. At the same time, new AR and HA create a bi-directional GRE tunnel to each other.
6. Although MN feels that it did not move and the default router is still the same, but AR's MAC address has already been changed. AR sends a gratuitous ARP message to ask MN to update its ARP cache if the L2 is a non-PPP link, so that the outgoing packets can be delivered to AR correctly.

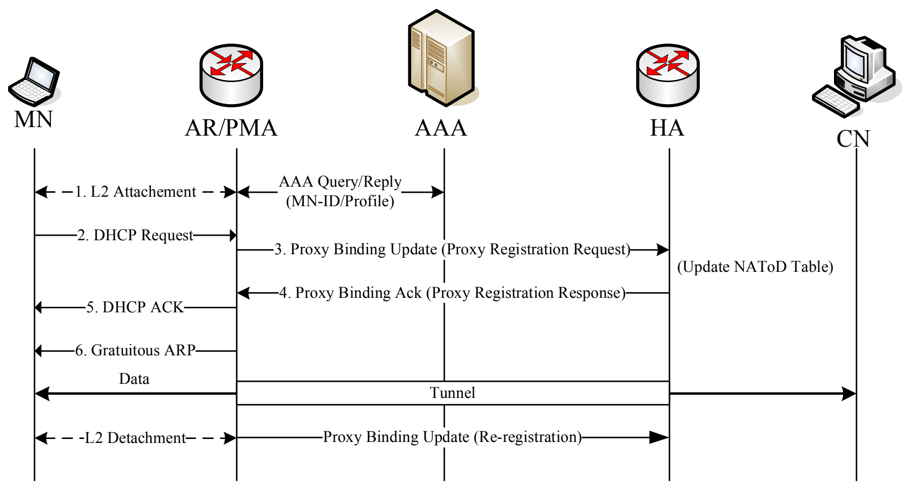

MN
AR/PMA
AAA
HA
CN
1. L2 Attachement
AAA Query/Reply (MN-ID/Profile)
2. DHCP Request
3. Proxy Binding Update (Proxy Registration Request)
(Update NAToD Table)
4. Proxy Binding Ack (Proxy Registration Response)
5. DHCP ACK
6. Gratuitous ARP
Data
Tunnel
-L2 Detachment
Proxy Binding Update (Re-registration)

Figure 3.7 PMIP4 with NAToD signalling when MN is re-attached

3.4 Message Format

PMIPv4 uses the same message format with MIPv4 which is described in Section 3.3 of RFC 3344. The NAToD uses MAC address as an identifier for each MN. We need to modify the original registration request message to meet our environment. The request is carried by an UDP packet with destination port 434. The payload of UDP datagram is the MIPv4 packet with variable optional extensions. The most important parts are Type, Home Address, Home Agent, and Care-of-Address. The additional field we add is the red one in Table 3.1 – MAC address.

Table 3.1 Packet format of registration request message

0	8	16	24
Type	Flag	Life time	
Home Address			
Home Agent			
Care-of-Address			
MAC address (bit 0-31)			
MAC address (bit 31-47)		Padding	
Identifier			
extension . . .			

3.5 Supported Access Link Technology

The proposed scheme can be applied on both broadcast and point-to-point access links.

3.5.1 IEEE 802.11

The proposed scheme is primarily designed for IEEE 802.11 environment. To apply the scheme with IEEE 802.11, the most important part is to adjust access point's forwarding behavior – blocking the broadcast ARP traffic as shown in Figure 3.8. Most APs act like a bridge between wired Ethernet link and the air link. If an AP received a broadcast ARP message, it must only forward the message to the wired port (the AR side). Also, AR must not send broadcast ARP on that link, instead, it sends ARP through unicast.

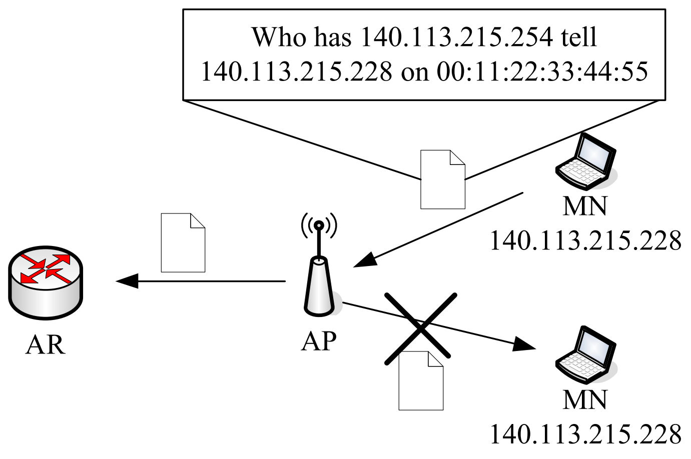

Who has 140.113.215.254 tell 140.113.215.228 on 00:11:22:33:44:55
MN
140.113.215.228
AR
AP
MN
140.113.215.228

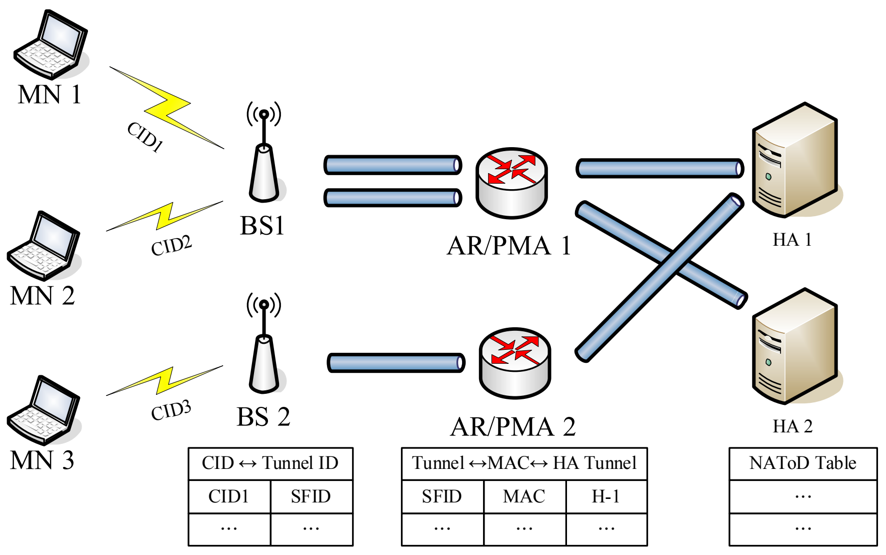
Figure 3.8 AP blocks broadcast ARP traffic

In most OS implementation, before passing a frame to upper layer the IEEE 802.11 layer will convert IEEE 802.11 frame header to Ethernet frame header (strip IEEE 802.11 and LLC header). Once the bridge layer receives a Ethernet frame with IP, it can directly forward the frame through the tunnel to HA.

3.5.2 WiMAX IP Convergence Sublayer

WiMAX uses a connection orientated link. Although its air link is a point to multi-point media, the traffic between each Subscriber Station (SS) and Base Station (BS) is separated by connection ID (CID). The most common configuration is that several BSs connected to a single Access Router (AR). The BS maps each CID to Service Flow ID (SFID) and bridges traffic to a GRE tunnel assigned with different keys (usually same as SFID). The other endpoint of tunnel is the access router. So, the link between each SS and AR can be treated as a point-to-point link.

The traffic is already isolated so it can be applied with NAToD directly or with PMA on the AR. With our proposed scheme, the access router just bridges one tunnel to another as



CID ↔ Tunnel ID	
CID1	SFID
…	…

Tunnel ↔MAC↔ HA Tunnel		
SFID	MAC	H-1
…	…	…

NAToD Table
…
…

Figure 3.9 PMIPv4 with NAToD on WiMAX

shown in Figure 3.9. One tunnel is connected to BS and SS, and the other tunnel is connected to HA for that MN. Because the payload of SS-tunnel is IP packet, the AR must add a pseudo Ethernet header so that the NAToD can work correctly.

3.5.3 Ethernet

Ethernet is the most common wired access link. Although it is a broadcast architecture, the NAToD can still be applied on this kind of link by using port-based VLAN. Each port is configured to different VLAN-ID and the port connected to PMA or NAToD is set to trunk type – with IEEE 802.1Q VLAN tag. With this configuration, every port is isolated to each other and all traffic will be forwarded to the trunk port and separated by VLAN-ID as shown in Figure 3.10. When receiving a packet, PMA or NAToD router can identify each host by the VLAN id and insert that id when packet is forwarded back to them. PMA creates a map between VLAN id and tunnel id to HA, and can simply operate as a bridge and forward

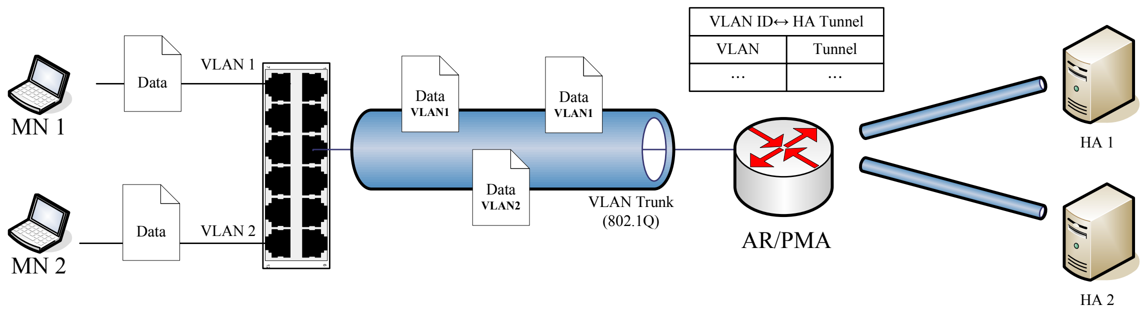


Figure 3.10 PMIPv4 with NAToD on Ethernet

packets to each other between the two interfaces.

3.5.4 PPP

The proposed NAToD can also be applied on dial-up PPP(PPTP/VPN). The PPTP server creates a virtual MAC address based on the authentication username. When a client connects to PPTP server, a pseudo interface is created, and can be used in NAToD table to distinguish each client. The PPTP server will also need to be modified to change the way it assigns IP (PPP/IPCP), so that the server can assign all clients with the same IP instead of picking one free IP from the pre-configured pool.

Chapter 4

Implementation and Experiment

We have implemented the NAToD scheme on Linux 2.4 kernel and AR on both Linux 2.4 and 2.6 kernel. HA is installed on a K8 level PC with NAToD and PMA is installed on a PC and embedded environment (Andes Core) with AP function. In the experiment, we've tested both handover latency in different OS as well as the performance of the proposed NAToD.

4.1 Implementation

On Linux we implement NAT function based on netfilter framework. The netfilter uses *conntrack* to keep tracking every connection through the host. The *conntrack* entry contains information including the direction of traffic, the source/destination IP and port, the connection state, and the protocol specified information like TCP's last ACK sequence number. NAT also stores information in the *conntrack*, including when and where to change source or destination header. To accelerate *conntrack* lookup, netfilter uses hash table to improve performance. Each tuple of the *conntrack* will be hashed and the tuple with the same hash

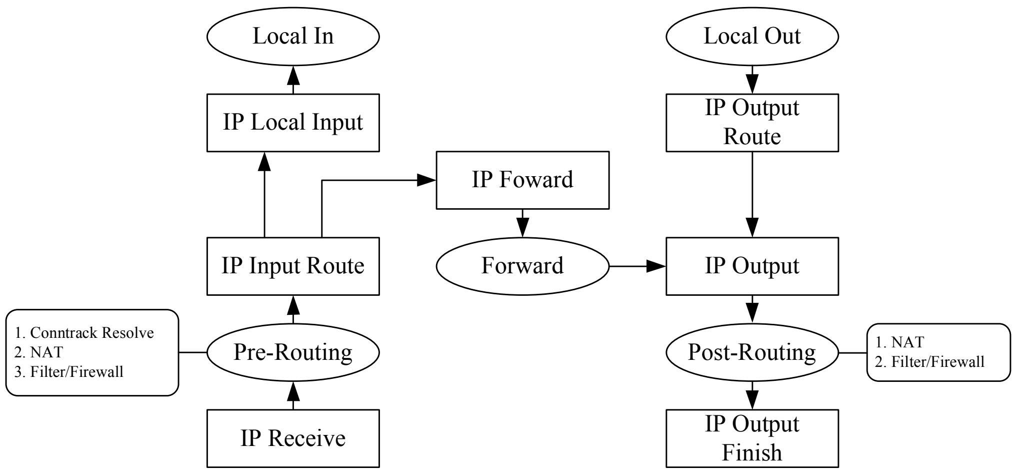


Figure 4.1 Packet flow in Linux's IP layer

value will be chained together in a linked list.

Figure 4.1 shows the packet processing flow in Linux kernel. The IP layer provides multiple hook points to allow other modules to hook and touch the packet. A packet gets into IP layer from the lower layer will first reach NF_PRE_ROUTING hook. The netfilter hooks a routine with a higher priority to analyze the packet header, get tuple and look up *conntrack* tables here. After resolving the *conntrack*, the packet data structure (struct sk_buff) will associate with a *conntrack* entry. This provides other modules to use this information like NAT and Firewall.

Figure 4.2 is an example of NAT information and the following is how the NAT module cooperates with *conntrack*. The mainip_place tells the NAT whether to change source or destination IP/port. If the type is SRC, the NAT will change the packet matching this *conntrack* after routing, and replace the source IP by Replace_IP and source port by Replace_Port and re-calculate IP and TCP checksum. When a packet travels through IP layer, the NAT will modify the incoming packet at pre-routing hook and the outgoing packet at post-routing

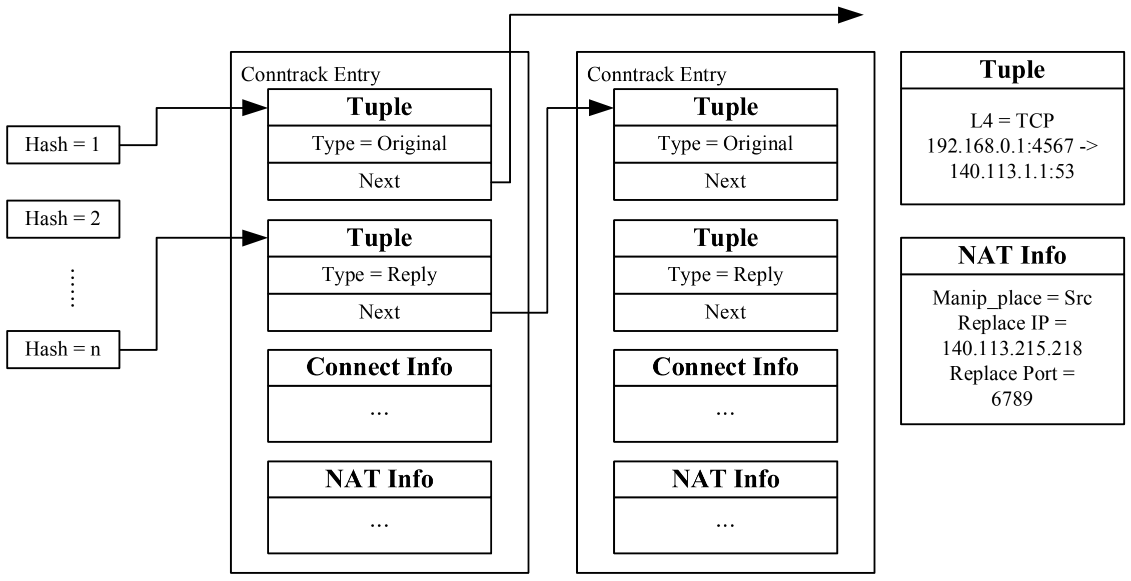


Figure 4.2 Data structure of conntrack entry

hook.

When a *conntrack* entry is initialized, it checks the NAT rule to see if this connection needs to be translated or not. If so, netfilter picks an IP from the pool to replace the internal IP address. At first the source port is remained the same, and lookup the conntrack table to examine whether the new source IP port is conflicted or not. If the confliction happened, it increases source port number by 1 until the new combination is unique. Finally NAT keeps this binding in the NAT information of *conntrack* entry.

We extend the tuple structure with MAC address, device-id and a flag to contain L2 information. The flag is used to indicate whether the MAC address &dev-id existed and needed to compare or not when performing *conntrack* lookup. (The packet generate from the upper layer of local host will not contain any L2 information). We add a hook function at pre-routing hook. When a packet with new *conntrack* gets into the hook function, NAToD checks the rule table to see whether this connection track needs to be translated or not. The initial phase is almost the same with original NAT. Then an outgoing packet is passed into the hook function, the packet will then be translated if necessary, otherwise NAToD will keep this packet unmodified and passes it to the next function. And the packet will be forwarded to next hop through normal IP routing.

When an incoming packet comes in, the NAToD will generate a new Ethernet header and decide which interface the packet should be forwarded to based on the L2 information stored in the *conntrack*. Also, it translates the packet if necessary. Finally, NAToD passes the packet to the device driver (dev_queue_xmit) directly without going through the original routing routine. NAToD will report a "stolen" state to the hook function as well as tell other hook and the following function not to do anything to this packet.

4.2 Experiment Setup

4.2.1 Handover Latency

In this test, we setup an experiment environment as shown in Figure 4.3. There are two access points running in IEEE 802.11g mode and connected to different access routers. The two ARs belong to different subnets and both are connected to an upper tier router by Ethernet. HA is located in the same subnet with AR/AP 1. MN moves and handover from AP 1 to AP 2. The CN is connected to the upper tier router with two hop distance and is also connected by Ethernet.

The CN sends Constant Bit Rate (CBR) UDP traffic to MN at about 1Mbps (1280 bytes UDP packet for every 10-10.1 ms) to MN. We observe packets on both MN side and the sniffer, which also fetches IEEE 802.11 management frames to analyze the delay on L2 and

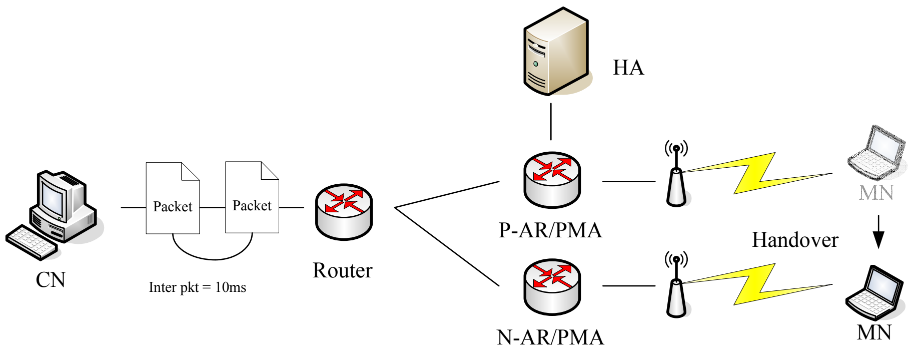


Figure 4.3 Network topology for handover test

L3 handover. The MN is running on different operating systems including Windows Vista, FreeBSD 8.1 and Linux kernel 2.6.32 to measure how different OSs effect the total handover latency.

4.2.2 NAToD Performance

Compare to normal IP forwarding procedure, an NAT router has to check the payload of IP packet, lookup NAT table, rewrite packet header and recalculate checksum. These tasks take time and decrease router's forwarding performance. In this experiment, we compare normal

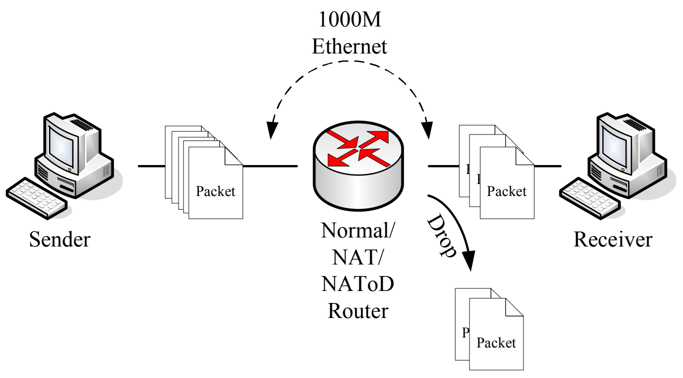


Figure 4.4 Network topology for performance test

routing, NAT and proposed NAToD by measuring their processing capacity.

To measure the performance of NAT router, we refer to the maximum number of packet the router can process per second (pps). There are two hosts connecting to different interfaces of the router. One host is the source that sends packet as fast as possible. The router forward packets to the destination host. On the destination, we measured how many packets can arrive correctly. Each host is equipped with Intel Pro/1000 Gigabit Ethernet adapter and the router has two network interface cards (NIC), and connected with 1000Mbps Ethernet as shown in Figure 4.4. In the second setup, we replace the NIC connected to the sender with Intel Pro 100 to measure how does the NIC/driver affect the result.

To make sure that the bottleneck is on the router instead of physical link, we send small packets (payload of UDP is 1 byte). The packet sending rate is more than 350,000 pps and is larger than the router's capacity. The generated traffic consists of different sessions from 1 to 65536. We tested on normal routing mode (forward packet only), NAT mode (forward + table lookup + manipulate packet) and NAToD mode (forward + table lookup) for both outgoing and incoming directions.

4.3 Experiment Results

4.3.1 Handover Latency

The handover latencies are different for each OS. For the following test, handover is triggered manually at 1 sec. The experiment result shows that service disruption is about 1100 ms in Windows Vista, 2000ms in FreeBSD with full scan, 60 ms in FreeBSD without scan and 130ms in Linux.

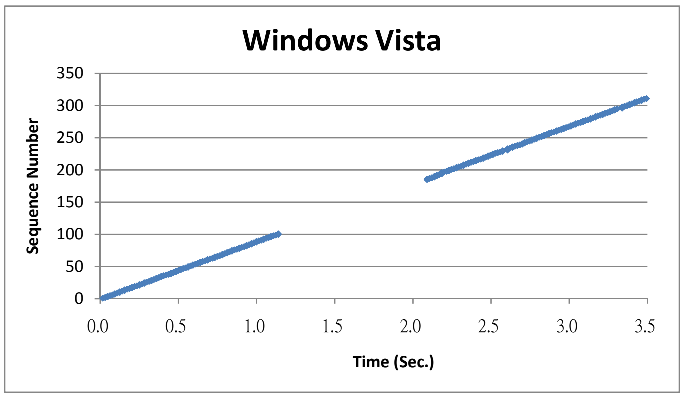


Figure 4.5 Service disruption time in Windows Vista

Figure 4.5 is the result in Windows Vista. From the information provided by sniffer, Windows Vista cost 700 ms in L2 handover and 180 ms in L3 handoff. First, Windows Vista sends disassociation and de-authentication when a mobile host leaves its serving AP. Second, it turns the wireless NIC to target-AP's channel based on previous scanning result and scan for that AP. Before it starts to scan, it is idled about 600 ms. If the target AP responded for the probe, Windows Vista stops scanning immediately and starts to connect to the target AP. According to IEEE 802.11, when station connects to an access point, it exchanges authentication message first and then sends association message. We found that Vista does not send association request right after got authentication reply from AP. Actually, it waits about 100 ms.

After L2 connection is completed, it takes about 138 ms to trigger DHCP client to send DHCP request. The RTT of DHCP request and ACK is about 8 ms (including processing time, transmission time and propagation time). If it got DHCP ACK, which means that the client can continue to use the original IP. The Vista sends ARP request for the default router again, and from now on, it can receive packet from the new network. After receiving ARP reply, it can start to send packet.

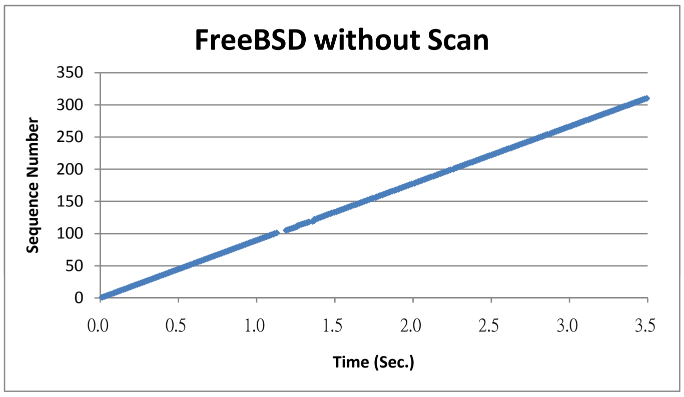


Figure 4.6 Service disruption time in FreeBSD without Scan

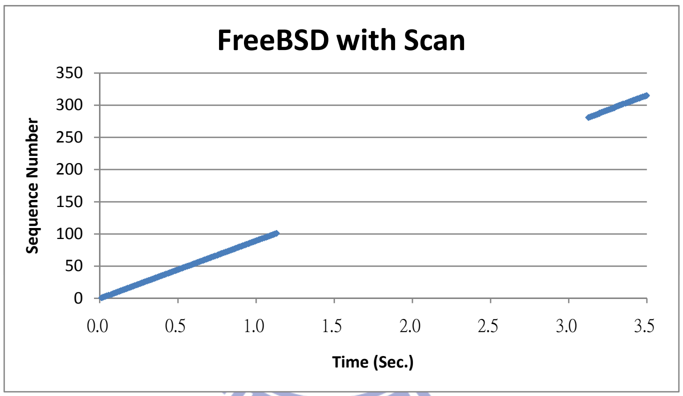


Figure 4.7 Service disruption time in FreeBSD with full Scan

In FreeBSD, there are two modes to switch from one AP to another. If the last scan was taken no earlier than a threshold, the FreeBSD IEEE 802.11 layer will try to direct connection to target AP based on the last scanning result. In this mode, the total service disruption time can be reduced to 60 ms as shown in Figure 4.6. Otherwise, it does a full scan over all available channel and it takes about 1900 ms as show in Figure 4.7. For each channel, the scan takes place for 200 ms on each channel. After finishing scanning the last channel, it starts passive scanning on channel 12 to 14.

Unlink Windows Vista, once a station finished exchanging authentication message, it starts to send association message immediately. After L2 link is connected, the DHCP client on FreeBSD takes 15 to 25 ms to detect link state and start to send DHCP request. The delay on DHCP is about 10 ms and is close to the result from Windows Vista.

Figure 4.8 is the result on Linux. It's behavior is like Windows Vista but without the idle period. After disconnected from serving AP, it probe for the target AP based on the scan cache and start to connect immediately after the target AP responded the probe. Compare to Windows, the gap between every IEEE 802.11 message is much lower (1-2 ms vs. 100 and 600 ms). In L3 handoff, the DHCP client in the Linux cannot listen to IEEE 802.11 event, the client will not send DHCP request after it associates to the target AP, instead the user has to launch DHCP client manually or execute by other program or script.

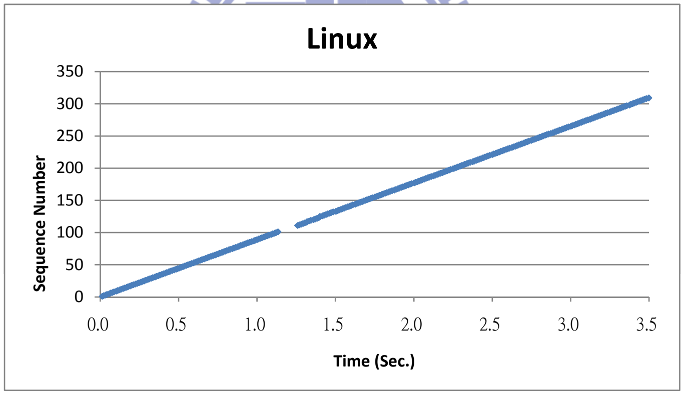


Figure 4.8 Service disruption time in Linux

There are two common DHCP client in Linux, ISC-DHCP client and *udhcpc*. The former one is a standard DHCP client and will record the current lease (including IP address, subnet mask, default router etc.), the next time it starts up, it will read the lease file and request for the previous IP first instead of asking for a new one. But this client has a longer startup delay

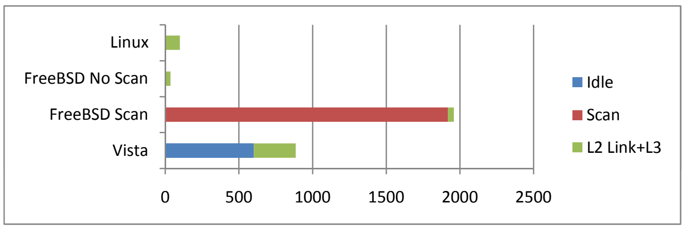


Figure 4.9 Comparison of handover process in different OS

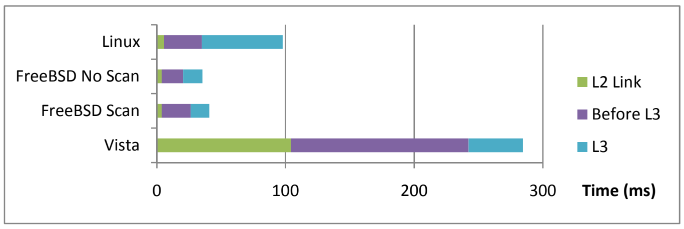


Figure 4.10 Comparison of handover process excluding the scanning

about several hundred ms. The later one is a light-weight client, it has lower start up delay but it does not support lease file. Every time it starts, it sends DHCP Discover first and need twice round-trip time between MN and DHCP server. We choose *udhcpc* as DHCP client in Linux because the delay is less than the other one even if it consumes 2 RTT.

From the experiment result, L2 handover latency is much longer than that in L3 especially when L2 needs to perform scan process. Figure 4.9 is the comparison between L2 & L3 handover latency from different OSs. We can find that full scanning cost the most time during the handover process. Figure 4.10 is the same comparison excluding scanning. Vista costs more time in L2 link connection and its DHCP client takes much more time to detect link change before starting the DHCP process.

The L3 handover latency is related to the RTT of target AP and HA. The experiment environment is in the LAN connected by Ethernet so the latency is low. From the above experiment, the delay of DHCP request is about 10 ms (about 2 ms of which is link RTT including IEEE 802.11 frame processing time and the remaining is the processing time). Practically, when an MN is away from home, the RTT from foreign network to home is usually less than 100 ms and is still lower than L2 handover latency.

4.3.2 NAToD Performance

Figure 4.11 is the average number of packets that the router can process per second. The curve forward is the number of packets that the router can process when it operates as router mode. IP routing is based on IP header and is not related to the payload of IP, so its performance is the same for different sessions. In our experiment, the performance can reach nearly 300K pps. When the router with NAT is enabled, besides normal routing, it has to lookup *conntrack* from the hash table and other routine procedure including getting tuple and checking whether the *conntrack* is valid. This makes the forwarding performance to drop even when the number of session is only 1. The two NAToD curves 'forward' and 'reverse' are the performance for outgoing and incoming traffic. Both incoming and outgoing performance are very close for NAT so we just put one result here.

From the result, we can find that outgoing traffic experienced better performance with NAToD when the number of sessions is less than 32768. NAToD needs to compare more fields than NAT when looking up translation table, but it does not have to translate for every packet. That is why the NAToD can process packets faster. When the number of sessions grows, the collision in the hash table also increases. It has to compare every *conntrack* entry with the same hash value. As a result, NAT performs better than NAToD when number of sessions is more than 32768.

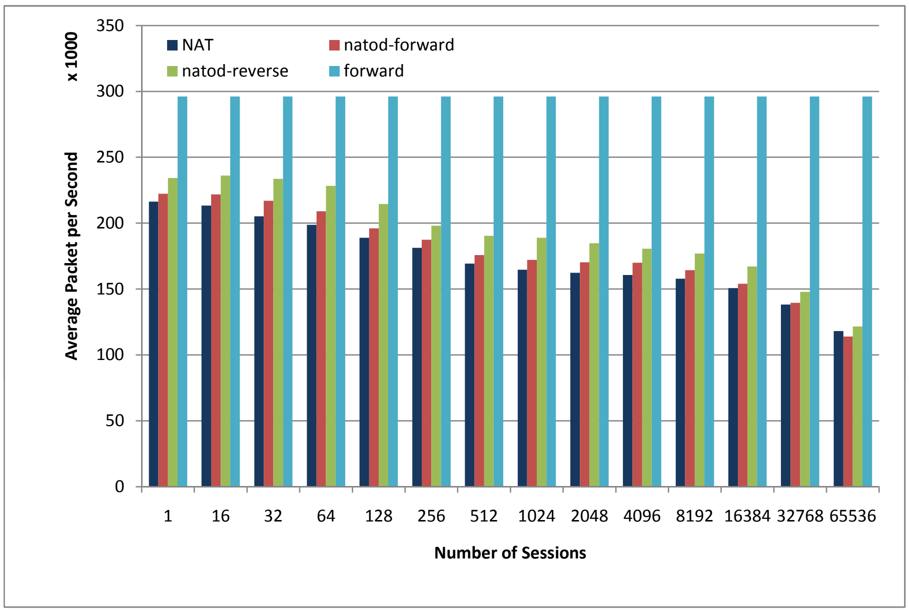


Figure 4.11 PPS vs. number of sessions

Figure 4.12 is the average packet processing time obtained from the reciprocal of the above result. The 'get tuple' curve is the time netfilter gets tuple from packet header and other routine job. The 'conntrack' curve includes get tuple as well as look up translation table but without NAT. The difference between 'conntrack' and 'get tuple' is close to the time cost by NAT table (hash table lookup). The other 3 curves NAT, NAToD forward and NAToD reverse are the time cost by traditional NAT, NAToD with outgoing traffic and NAToD with incoming traffic, respectively.

The reciprocal of pps is the processing time for a single packet, but we cannot compare these numbers and calculate overhead based on dividing NAToD's processing time by NAT. The processing time includes the time cost in driver, MAC layer, IP layer and NAT. We calculate the relative time by the difference between the NAT/NAToD curve with the forward curve.

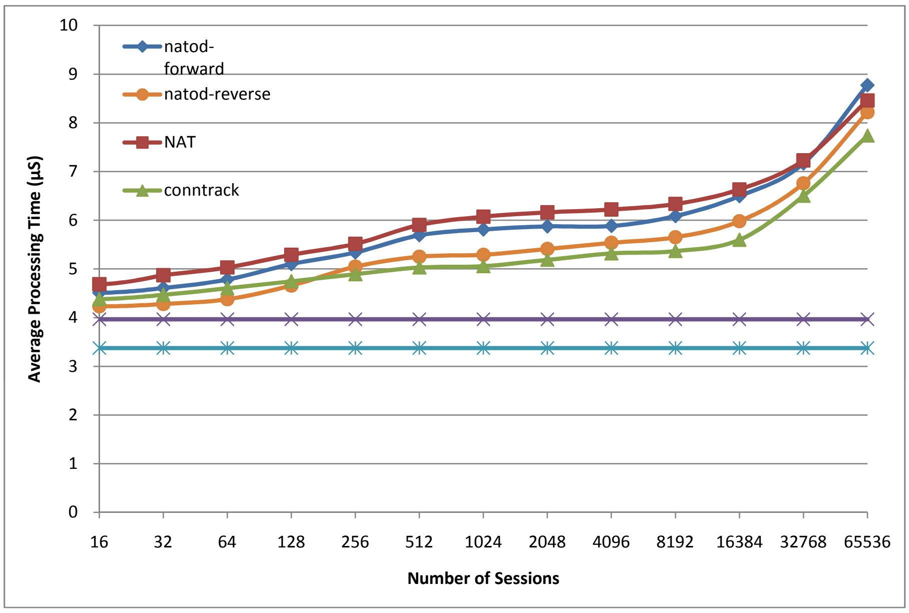


Figure 4.12 Packet processing time vs. number of sessions

For incoming traffic, NAToD scheme does not require looking up routing and ARP table, it gets better performance than NAT even when the number of session reaches 65536. When number of sessions is less than 128, it is faster than normal routing with *conntrack*. That is, the time save in routing and ARP table lookup is more than the time consumed by NAToD table.

Figure 4.13 is also the average packet processing time with different network adapters (setup 2). We can find that although packet processing time is greater than the previous result, the difference between NAT and forwarding is close to the previous result. The processing time in NAT and NAToD layer grows when the number of sessions gets larger. It shows that we should measure the relative processing time instead of simply measuring the pps.

Table 4.1 is the comparison of time consumption between NAT and NAToD. When the number of sessions is less than 4096, NAToD spends about 90% processing time of NAT.

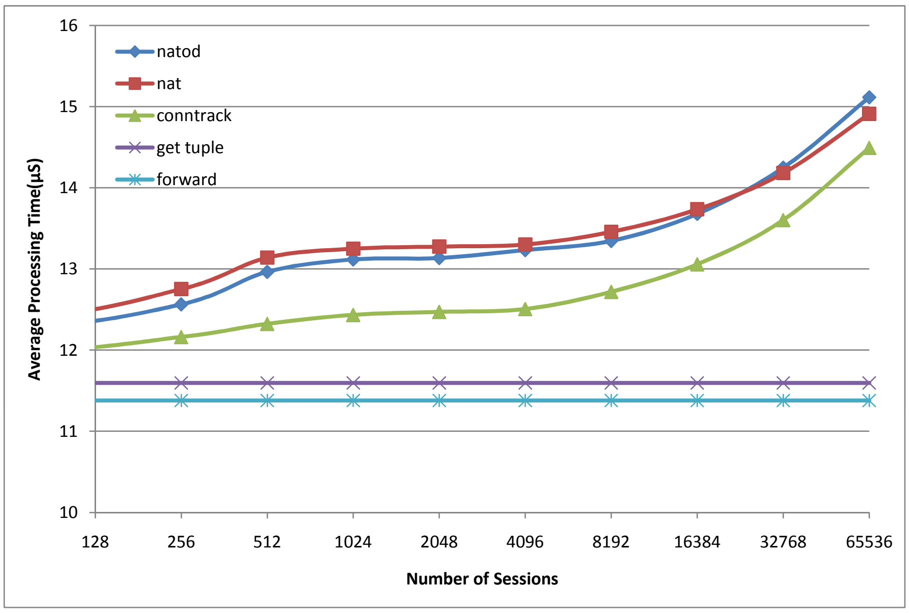


Figure 4.13 Packet processing time vs. number of session on configuration 2

NAToD spends more time when the number of sessions is more than 32768 but the overhead is still less than 10% which is acceptable.

From the above result, when the router is equipped with better NIC, it can process more packets. With poor NIC, it takes much more CPU time on interrupt service routine and cause the pps drop tremendously. Comparing the packet processing capacity between normal forwarding and NAT, with poor NIC, the number of packet drops about 24%, and packet drops 60% when using a high performance NIC. Although the later one drops more, but the number of packets it can process is almost double of the former one. The overhead of NAT (the difference between forward curve and NAT curve) is much more than the difference between NAT and NAToD.

Table 4.1 Packet processing time and overhead of NAT and NAToD

No. of sessions		≤ 128	256	512	1024	2048	4096	8192	16384	32768	65536
Setup 1	NAT	-	2.139	2.527	2.696	2.785	2.844	2.957	3.255	3.852	5.083
	NAToD	-	1.960	2.313	2.434	2.495	2.504	2.707	3.117	3.788	5.394
	overhead	-10	-8.38	-8.44	-9.72	-10.40	-11.97	-8.46	-4.25	-1.67	6.13
Setup 2	NAT	-	1.372	1.760	1.870	1.895	1.921	2.078	2.355	2.801	3.530
	NAToD	-	1.184	1.585	1.737	1.755	1.852	1.966	2.298	2.867	3.734
	overhead	-12.95	-13.69	-9.95	-7.08	-7.42	-3.60	-5.38	-2.41	2.35	5.77

Unit: μSec.

Chapter 5

Conclusion & Future Work

In this thesis, we proposed a novel NAToD scheme which can be integrated with Proxy Mobile IP. We implemented this scheme on Linux OS, and one of the mobility entity – AR/PMA & AP is also ported into an embedded environment. We provided network based mobility on the most popular IEEE 802.11 wireless networks and we verified that this scheme is workable by the connectivity experiment. This shows that we can easily deploy this scheme into a mobility domain like campus.

From the experiment, our PMIP handover latency is low that can provide almost seamless handover when MN's NIC and OS support fast roaming from one AP to another. The proposed NAToD is more complex doing translation table look up but we showed that the overhead is acceptable and our scheme performed even better than the original NAT when the number of sessions is less than 32768. On the incoming direction, the experiment result shows that we get better performance even if the number of sessions is more than 32768. Nowadays the price of RAM is not expensive, we can reduce the packet processing time by increasing the hash size of the translation table with larger size of RAM.

The NAToD still faces the problems caused by NAT e.g. the session can only be initiated by the internal host, and the host under the same NAToD router can barely communicate to each other because they do not know the existence of each other. In the future, we can solve this problem by using FQDN to identify each host. The DNS server integrated with NAToD router creates a pseudo IP address and a temporary translation entry to forward packet between two internal hosts.

Bibliography

[1] J. Arkko, C. Vogt, and W. Haddad, “Enhanced Route Optimization for Mobile IPv6,” Internet Engineering Task Force, RFC 4866, May 2007.

[2] M. Buddhikot, A. Hari, K. Singh, and S. Miller, “MobileNAT: A New Technique for Mobility Across Heterogenous Address Spaces,” *Mobile Networks and Applications*, no. 3, pp. 289–302, 2005.

[3] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, “Proxy Mobile IPv6,” Internet Engineering Task Force, RFC 5213, Aug. 2008.

[4] W.-K. Jia and Y.-C. Chen, “A NATed Mobility Management Scheme for PMIPv4 on Wireless LANs,” in *Proceedings of 11th International Conference on Mobile Data Management*, 2010, pp. 125–134.

[5] D. Johnson, C. Perkins, and J. Arkko, “Mobility Support in IPv6,” Internet Engineering Task Force, RFC 3775, Jun. 2004.

[6] R. Koodli, “Fast Handovers for Mobile IPv6,” Internet Engineering Task Force, RFC 5568, Jul. 2009.

[7] R. Koodli and C. Perkins, “Mobile IPv4 Fast Handovers,” Internet Engineering Task Force, RFC 4988, Oct. 2007.

[8] A. Lees and S. H. Gundrson, “IPv6 at Google.” Available: https://sites.google.com/site/ipv6implementors/2009/agenda/10_Lees_Google_IPv6_User_Measurement.pdf

[9] K. Leung, G.Dommety, P.Yegani, and K. Chowdhury, “WiMAX forum / 3GPP2 Proxy Mobile IPv4,” Internet Engineering Task Force, RFC 5563, Feb. 2010.

[10] O. Levkowetz, J. Forslow, and H. Sjostrand, “NAT Traversal for Mobile IP using UDP Tunneling,” Internet Engineering Task Force, Internet-Draft, Jul. 2001. Available: http://tools.ietf.org/id/draft-levkowetz-mobileip-nat-tunnel-00.txt

[11] C. Perkins, “IP Mobility Support for IPv4,” Internet Engineering Task Force, RFC 3344, Aug. 2002.

[12] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol,” Internet Engineering Task Force, RFC 3261, Jun. 2002.

[13] H. Soliman, C. Castelluccia, K. ElMalki, and L.Bellier, “Hierarchical Mobile IPv6 (HMIPv6) Mobility Management,” Internet Engineering Task Force, RFC 5380, Oct. 2008.

[14] P. Srisuresh and K. Egevang, “Traditional IP Network Address Translator (Traditional NAT),” Internet Engineering Task Force, RFC 3022, Jan. 2001.

[15] Z. Zhen and S. Sampalli, “Saving Public Addresses in Mobile IP,” in *Proceedings of International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies(ICN/ICONS/MCL)*, 2006, pp. 127–127.