A Performance Study on Teredo IPv6 Tunneling Mechanism

Shiang-Ming Huang, Quincy Wu, Yi-Bing Lin, *Fellow, IEEE*

{smhuang,solomon,liny}@csie.nctu.edu.tw

Abstract

在現有的 Internet Protocol version 4 (IPv4)網路中引入 Internet Protocol version 6 (IPv6)時，Tunneling 是經常被使用到的機制。由於 IPv6 網路尚未普及，IPv6 網路設備往往需要經由現有的 IPv4 網路以 Tunneling 機制傳送 IPv6 封包，才能與其它 IPv6 網路互通。然而，如果 IPv6 設備位在 Network Address Translation (NAT)後端的 private 網路，傳統 Tunneling 機制完全沒有辦法幫助它連上 IPv6 網路。本篇論文整理目前各種修正 Tunneling 機制讓 NAT 後端 IPv6 設備連上 IPv6 網路的方法，並將這些方法分類爲 NAT-aware Tunneling 和 NAT-unaware Tunneling 兩大類。除此之外，我們也介紹 Teredo 這個分散式的 Tunneling 方法，並提出我們在 Linux 平台開發的 Teredo 實作：NICI-Tededo。NICI-Teredo 是 Linux 平台上的第一個 Teredo 實作，它讓 Teredo 能更輕易的被廣爲佈建，提供處於 NAT 後端廣大 IPv6 使用者連上 IPv6 網路的媒介。最後，我們透過實際量測，驗證 NICI-Teredo 有非常好的封包轉送效能。

Keywords: IPv6, Teredo, NAT, Tunneling

CHAPTER 1

Introduction

以 Internet Protocol version 4 (IPv4)爲主的網路風行世界近三十年後，IETF (Internet Engineering Task Force)推出了 Internet Protocol version 6 (IPv6)作爲下一世代 Internet 通訊協定。和 IPv4 相比，IPv6 具有更大的位址空間，而且擁有效率更好的封包 routing 機制、security 機制和對 QoS (quality of service)更好的支援，因此，IPv6 極具機會取代 IPv4 成爲未來 Internet 上的主流。由於現有的網路設備還是以支援 IPv4 爲主，只有較新穎的網路設備才會提供 IPv6 的支援，如何在現有 Internet 架構中引入 IPv6，是一個複雜且重要的問題。鑒於此，爲了使現有 IPv4 網路能平順地轉換到 IPv6，IETF Ngtrans 工作小組擬定了三個方面的解決方案[RFC2893]，分別針對不同方面的需求：

雙堆疊 (Dual-Stack)

讓網路設備支援 IPv4 及 IPv6 兩個通訊協定(protocol stack)，也就是同時擁有和 IPv4 及 IPv6 網路溝通的能力(如圖 1-1)。Dual-Stack 的目的在使 IPv4 及 IPv6 兩個通訊協定可在相同的設備上及相同的網路中共存，如此就不用在導入 IPv6 時，造成雙倍的硬體投資。目前 router 廠商如 Cisco、Juniper、Nortel、Nokia、Hitachi 等，伺服主機作業系統如 HP、IBM、Sun、Microsoft Windows、Linux、FreeBSD 等，均已支援 IPv4/IPv6 Dual-Stack。

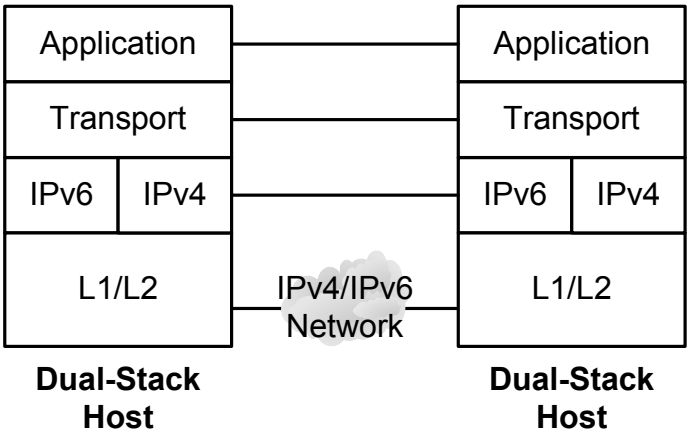

圖 1-1、Dual-Stack

隧道 (Tunneling)

Tunneling 將 IPv6 封包封裝成 IPv4 封包負載(payload)，以 IPv4 封包負載的型式在 IPv4 網路中傳送(如圖 1-2)，目的在使未直接相連的 IPv6 網路彼此間能夠溝通，如此即使所有網路轉換成 IPv6 的時程先後不一，依然能維持彼此間的互通。現有 Tunneling 做法如 Configured & Automatic Tunnel (RFC 2893)、6to4 Tunnel (RFC 3056)、Tunnel Broker (RFC 3053)等。

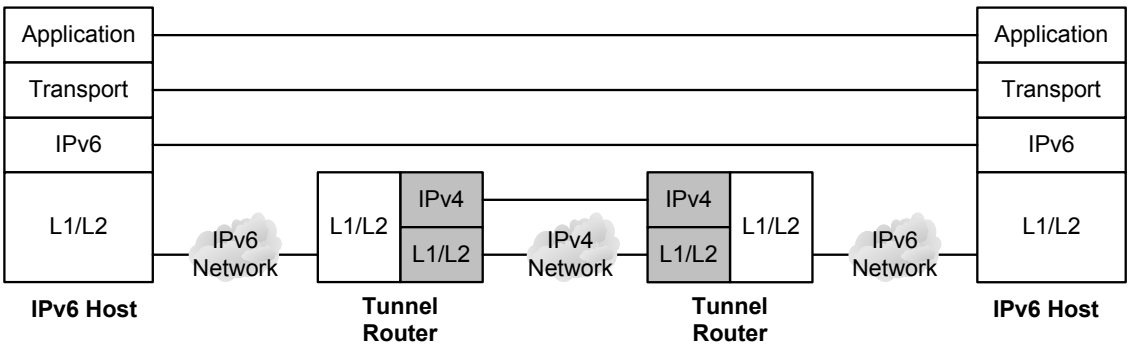

圖 1-2、Tunneling

轉換 (Translation)

Translation 將 IPv4 封包和 IPv6 封包互相轉換(如圖 1-3)，目的在使 IPv6 網路能與現存的 IPv4 網路互通，使得 IPv6 網路能夠享用現有 IPv4 網路豐富的網路資源及使用族群。目前較流行的 Translation 做法有 SIIT (RFC 2765)、NAT-PT (RFC 2766)、BIS (RFC 2767)、BIA (RFC 3338)等。

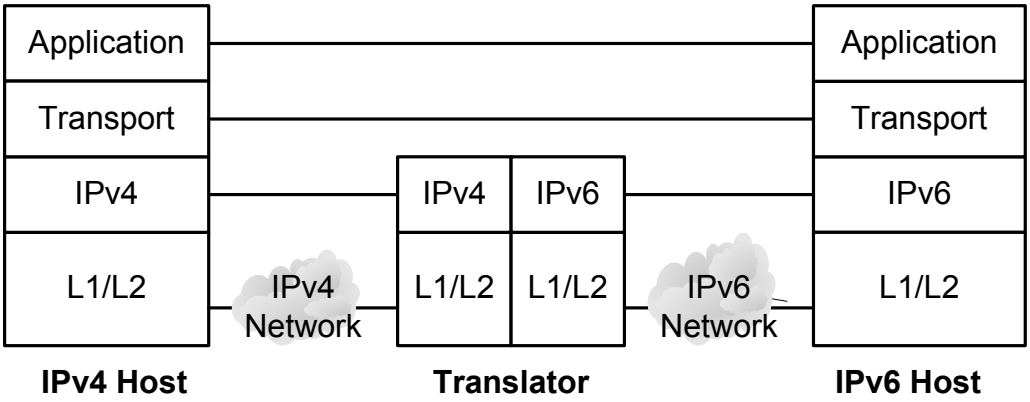

圖 1-3、Translation

上面三種解決方法中，現有的 Tunneling 方式，不論是 Configured Tunnel [RFC 2893]或是 6to4 Automatic Tunnel [RFC 3056]、Tunnel Broker [RFC 3053]等做法，均要求建立 Tunnel 的兩端必需具備一個 public routable IPv4 位址。這對目前許多有興趣嘗試 IPv6 的玩家而言，他們所處的網路若在 NAT [RFC3022] (Network Address Translation，或稱爲 NAPT、NAT/PAT)的 private 網路中，就無法以 Tunneling 的方式連到世界上的 IPv6 網路，成爲現存環境下的「孤島」。接下來我們分別介紹 IPv6 Tunnel 的運作方式以及 NAT 的運作方式，接著以 Configured Tunnel 爲例，說明 NAT 對 Tunneling 造成的影響。

1.1 Example 1: Configured Tunnel for Dual-Stack

以圖 1-4 爲例，IPv4 網路中有兩台 Dual-Stack Hosts A 和 B，A 的 IPv4 位址爲

140.113.131.80，IPv6 位址爲 2001:238:F88:210::8，B 的 IPv4 位址爲 61.218.105.10，IPv6 位址爲 2001:238:F88:210::9。在 A 和 B 之間設定 Configured Tunnel 如下：A 設定 Tunnel 的目的地爲 B 的 IPv4 位址，B 設定 Tunnel 的目的地爲 A 的 IPv4 位址。

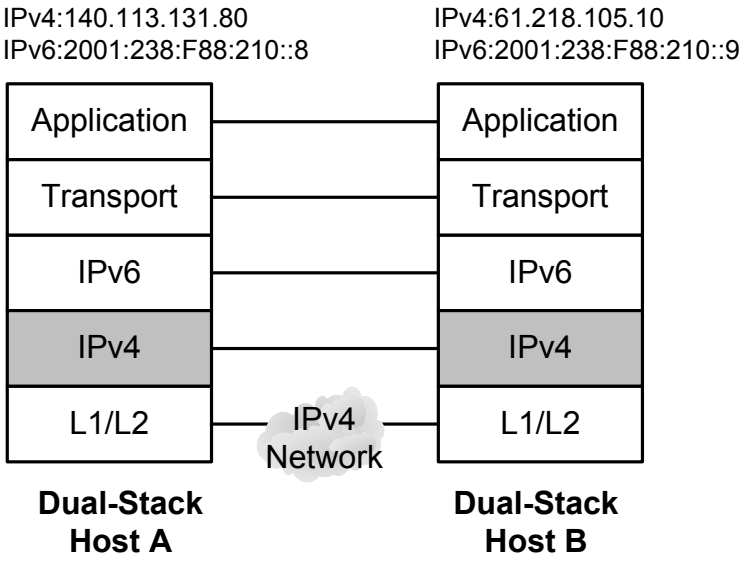

圖 1-4、Protocol Stack of Configured Tunnel

A、B 根據 RFC 2473 (Generic Packet Tunneling in IPv6 Specification) [RFC2473]以 IPv4 表頭封裝(encapsulate)IPv6 封包。A 和 B 之間傳送 IPv6 封包的過程如圖 1-5，步驟如下：

Step A.1. A 把要送給 B 的 IPv6 封包加上來源位址爲 140.113.131.80，目的位址爲 61.218.105.10，next level protocol 爲「IPv6」的 IPv4 表頭(header)，形成 Tunnel 封包。此封包被送至 IPv4 網路，並經若干路由器轉送後抵達目的位址 61.218.105.10。

Step A.2. 當 B 收到 A 送過來的 Tunnel 封包，B 將 Tunnel 封包的 IPv4 表頭拆掉，即可得到 IPv6 封包。當 B 要送 IPv6 封包給 A，B 將 IPv6 封包以來源位址、目的位址和 next level protocol 各爲 61.218.105.10、140.113.131.80 和「IPv6」的 IPv4 表頭包住，再將此封包送至網路。

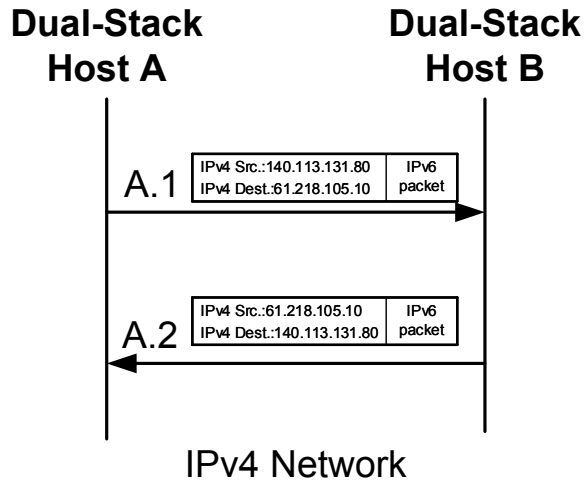

圖 1-5、IPv6 Packet Transmission using Configured Tunnel

1.2 Example 2: NAT

NAT [RFC 3022]是 IETF 制定的標準，它讓處於 private 網路(使用 private IPv4 位址)的多個網路設備共用一個 global routable IPv4 位址，以解決 IPv4 位址不足的問題。NAT 依其運作方式，分爲 full cone、restricted cone、port-restricted cone 和 symmetric 四種類型[RFC3489]。以下我們均以 port-restricted NAT 爲例說明運作方式及相關的解決方案。

在 private 網路的設備將 NAT 設定爲 default gateway，NAT 即可藉由攔截穿過 NAT 的封包，配合 NAT 上紀錄的位址對應表轉換 IPv4 封包的欄位內容，以達成 public IPv4 位址和 private IPv4 位址的轉換。至於 NAT 上的位址對應表，則是藉由 NAT 攔截 private 網路往 public 網路傳送的 IPv4 封包而建立。位址對應表的格式如表 1-1，表中共分八個欄位。位址對應表中 **Transport protocol** 欄位紀錄 IPv4 標頭中 next level protocol 欄位的值(此欄位紀錄 IPv4 封包上層使用的通訊協定類型)。**Internal IP 位址**、**Internal port**、**Remote IP 位址**和 **Remote port** 這四個欄位分別紀錄 IPv4 封包(private 網路往 public 網路傳送)的來源 IP 位址、來源 port、目的 IP 位址和目的 port。External IP 位址欄位紀錄 NAT 提供給 private 網路共用的 global routable IPv4 位址，External port 欄位紀錄 NAT 派

定給此位址對應的 port。Lifetime 欄位紀錄此位址對應的使用時效，此位址對應持續一段時間沒有使用將會失效(Lifetime 的值依 NAT 廠牌及通訊協定的不同而有所不同)。當位址對應失效，NAT 就可把派定給此位址對應的 port 回收，留給另一組位址對應使用。

表 1-1、Address Mapping Table on NAT

Internal IP 位址	External IP 位址	Remote IP 位址	Transport protocol
192.168.100.2	140.113.131.80	61.218.105.10	TCP
Internal port	External port	Remote port	Lifetime
9876	5432	80	

NAT 封包轉換: private 網路→public 網路

NAT 收到從 private 網路送向 public 網路的 IPv4 封包後，將此封包的 next level protocol、來源 IP 位址、來源 port、目的 IP 位址和目的 port 和對應表中 Transport protocol、Internal IP 位址、Internal port、Remote IP 位址和 Remote port 欄位相比。若欄位皆相符則把 IPv4 封包的來源 IP 位址、來源 port 分別以位址對應表的 External IP 位址、External port 取代。若位址對應表中沒有符合的紀錄，NAT 會派定一個未被佔用的 port 給這個連線，並根據此封包的欄位內容建立新的位址對應。

NAT 封包轉換: public 網路→private 網路

NAT 收下從 public 網路送來的 IPv4 封包，將此封包的 next level protocol、來源 IP 位址、來源 port、目的 IP 位址和目的 port 和對應表的 Transport protocol、Remote IP 位址、Remote port、External IP 位址和 External port 欄位相比。若欄位皆相符則把 IPv4 封包的目的 IP 位址、目的 port 分別以位址對應表中 Internal IP 位址、Internal port 取代。若位址對應表中沒有符合的紀錄，此封包會被 NAT 丟棄。

以一個實例說明 NAT 運作方式。在 private 網路的 client (IP 位址爲 192.168.100.2)經過 NAT (IP 位址爲 140.113.131.80)向在 public 網路的 server (IP 位址爲 61.218.105.10，port 爲 80)以 TCP 傳送資料，其過程如圖 1-6，步驟如下：

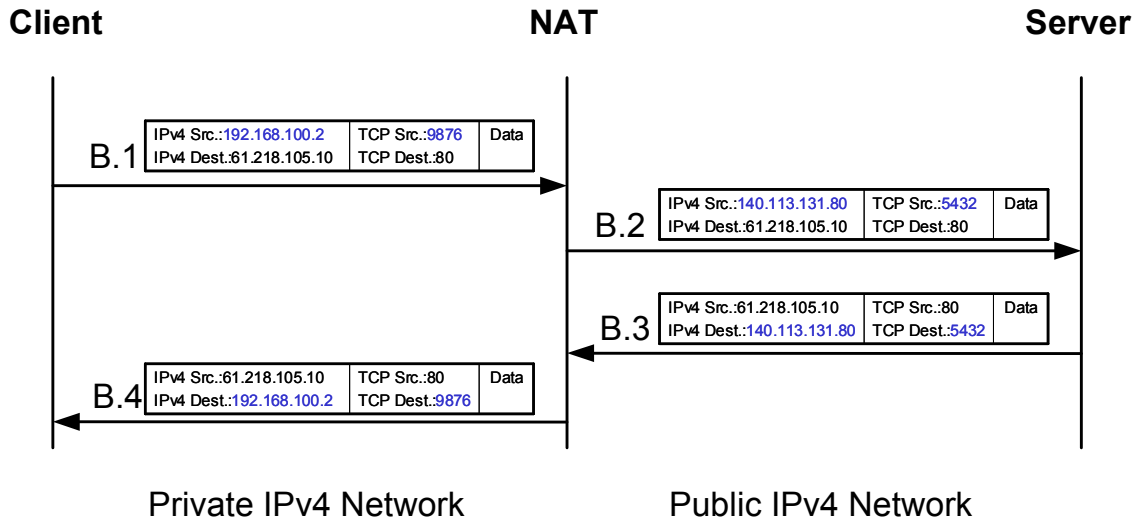

圖 1-6、NAT Translation Procedure

Step B.1. 當 client 端的程式要和 server 交談，client 端程式開啓一個 socket。Client 端程式用此 socket 向 server 發送封包，此封包的 next level protocol、來源 IP 位址、來源 port、目的 IP 位址和目的 port 會分別被設定爲 TCP、192.168.100.2、9876、61.218.105.10、80。此封包被送往 NAT (default gateway)。

Step B.2. NAT 攔截 client 往 server 送的封包。NAT 根據此封包的 next level protocol、來源 IP 位址、來源 port、目的 IP 位址和目的 port 在位址對應表建立一個新的紀錄，如表 1-1 所示。NAT 派定位址對應表的 External 欄位，接著根據位址對應表 Internal 和 External 欄位的對應，將封包的來源 IP 位址和來源 port 由 192.168.100.2、9876 替換爲 140.113.131.80、5432，目的 IP 位址和目的 port 不作變動。

Step B.3. Server 收到 client 送過來的訊息，回送資料給 client。Server 送出 next level protocol、來源 IP 位址、來源 port、目的 IP 位址、目的 port 分別爲 TCP、

61.218.105.10、80、140.113.131.80、5432 的封包。

Step B.4. Server 回送給 client 的封包被 NAT 收下。NAT 配合位址對應表檢查此封包的 next level protocol、來源 IP 位址、來源 port、目的 IP 位址、目的 port，並根據位址對應表將此封包的目的 IP 位址、目的 port 由 140.113.131.80、5432 替換爲 192.168.100.2、9876。此封包被送至 private 網路，由 client 端在 TCP port 9876 的 socket 收下。

經由上述程序，NAT 後端的 client 可以成功的開啓和 public 網路 server 的交談。

1.3 Example 3: NAT and Configured Tunnel

當我們結合 NAT 和 Tunneling 技術時，上述的 private 位址替換會發生問題，使得 IPv6 Tunnel 無法正確運作。在以圖 1-7 爲例的網路環境中，有 Dual-Stack Hosts A 和 B。A 在 NAT 的 private 網路，而 B 在 public 網路。A、B 使用 Configured Tunnel 連接 IPv6。

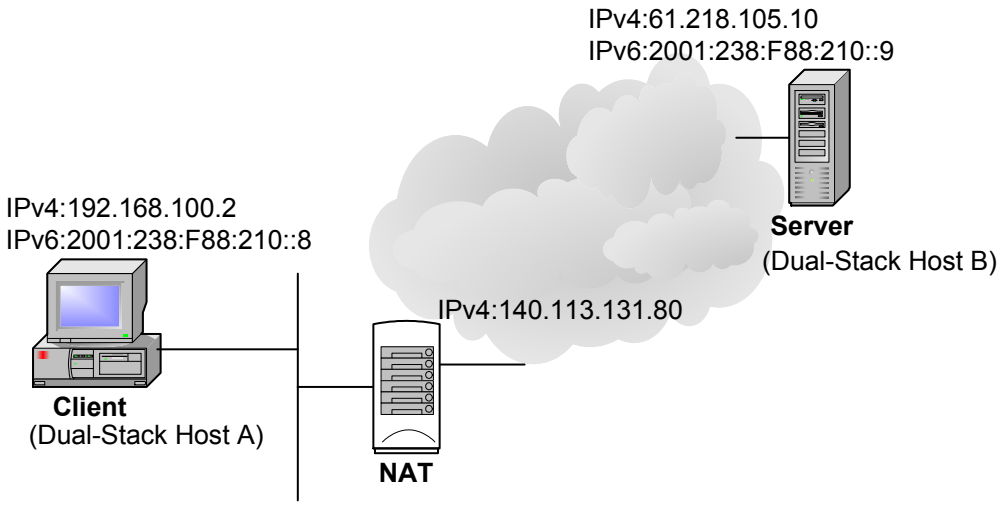

圖 1-7、Configured Tunnel with NAT

依 1.1 節所描述的 Configured Tunnel 設定方式，A 設定的 Tunnel 目的地爲 B 的 IPv4 位

址，而 B 設定 Tunnel 目的地爲 A 的 IPv4 位址。若 A 和 B 之間傳送 IPv6 封包，其傳送過程如圖 1-8，步驟如下。

Step C.1. A 把要送給 B 的 IPv6 封包加上來源 IP 位址爲 192.168.100.2，目的 IP 位址爲 61.218.105.10，next level protocol 爲 IPv6 的 IPv4 表頭，形成 Tunnel 封包。此封包被送往 NAT。

Step C.2. Tunnel 封包經過 NAT，其 IPv4 來源位址被改爲對應在 NAT 上的 IPv4 位址。NAT 把 Tunnel 封包送至 public 網路。

Step C.3. 當 B 收到 A 送來的 IPv6 封包，要回送 IPv6 封包給 A。B 將要送給 A 的 IPv6 封包加上來源 IP 位址爲 61.218.105.10，目的 IP 位址爲 192.168.100.2 (根據 B 的 Tunnel 設定)，next level protocol 爲 IPv6 的 IPv4 表頭，再將此 IPv4 封包送至網路。由於 IPv4 表頭的目的 IP 位址爲 private 位址，此封包會在網路中被丟棄，無法送達 A。

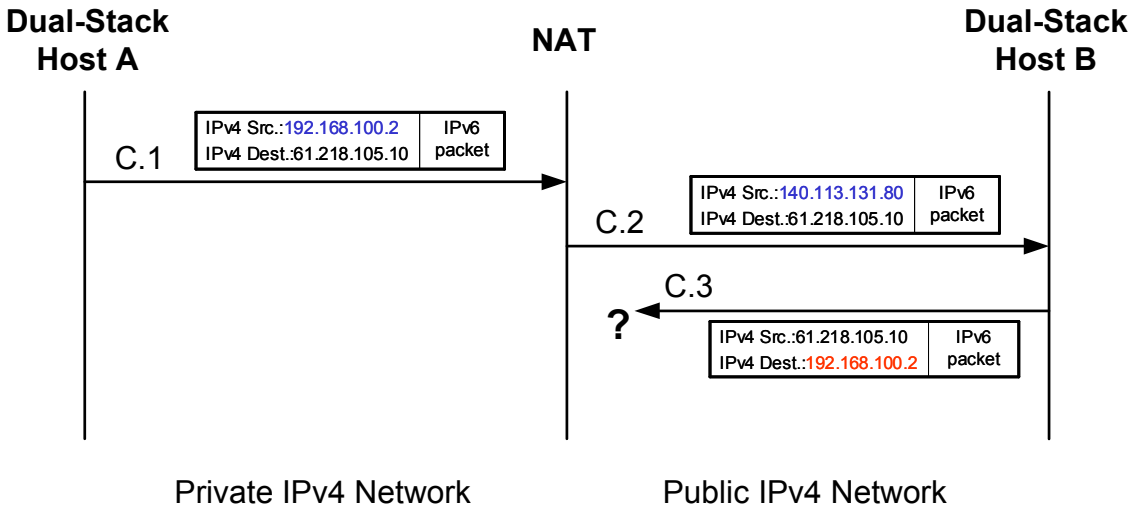

圖 1-8、Packet Transmission in Configured Tunnel with NAT

從這個例子中可以看出，雖然 A 可以將資料傳送到 B，但由於 B 的回應無法回到 A，造成雙方網路通訊無法建立。以原有 NAT 的運作方式，目前 IPv6 Tunneling 沒辦法讓 NAT

後方設備連上 IPv6 網路，是當前 IPv6 網路佈建上的一大障礙。由於現有的許多連線服務，像是 WLAN (Wireless Local Area Network)或 GPRS (General Packet Radio Service) [Lin]，大多透過 NAT 以 private IPv4 位址的方式提供服務，因此在未來所有網路設備全面支援 IPv6 的那一天到來之前，有必要爲這些位於 NAT 環境的 IPv6 使用者提供一個解決方案，以使他們能順利與世界上所有的 IPv6 網路相連。

1.4 Motivations and Contribution

爲了解決 IPv6 Tunneling 因爲 NAT 產生的問題，IETF 的 v6ops Working Group 提出一個新規格[Teredo]，能讓 NAT 內部設備以 Tunneling 連上 IPv6 網路，稱爲 Teredo。在 Teredo 剛出現的初期，世界上並沒有公開的 Teredo 環境提供一般 IPv6 使用者使用。我們在 Linux 平台上發展了一套 Teredo 實作，稱爲 NICI-Teredo [NICI-Teredo]。它是 Linux 平台上第一個 Teredo 實作。以 Linux 系統普及和架設簡易的特性，跑在 Linux 平台的 NICI-Teredo 讓 Teredo 這個 Tunneling 技術更容易被推廣開來，NICI-Teredo 的出現，也提供 NAT 後端廣大 IPv6 使用者連上世界上其他 IPv6 網路的媒介。

在本文接下來的敘述中，第二章將介紹數種幫助 IPv6 Tunnel 穿越 NAT 的解決方式，第三章介紹 Teredo 運作機制，接著提出我們的 NICI-Teredo 實作，並比較 NICI-Teredo 和其它兩個公開 Teredo 實作軟體架構，最後是這三個不同 Teredo 實作的效能分析以及結論。

CHAPTER 2

Related Work

本章分析幾種修正方法，讓 NAT 後端設備能使用 Tunneling 連上 IPv6 網路。本文將這些方法分爲 NAT-aware Tunneling 和 NAT-unaware Tunneling 兩類。NAT-aware Tunneling 必須修改 NAT 的設定，此方法適合由一般 IPv6 玩家自行設定。NAT-unaware Tunneling 不必逐一修改 NAT 的設定，此方法適合由 ISP (Internet service provider)大量佈建 IPv6 網路時使用。

2.1 NAT-aware Tunneling

這個方法需要在 Tunnel 的設定上作一些變更，並在 NAT 設定靜態位址對應表(static mapping table)。有別於 1.2 節中由 NAT 攔截封包後才建立的動態位址對應表，靜態位址對應表是在一開始就建立於 NAT 上的。我們以圖 1-7 的網路環境爲例說明如何修改 A、B 及 NAT 的設定，以達到 NAT-aware Tunneling。

- 在傳統的 Configured Tunnel 設定中，若 Host A 與 Host B 要建立 Tunnel，則 A 與 B 的 Tunnel 終點分別設定爲 B 的 IPv4 位址與 A 的 IPv4 位址。今若 Host A 位於 NAT 的 private 網路，那麼在設定 Tunnel 時，我們設定 B 的 Tunnel 終點爲 A 所在 NAT 的 public IPv4 位址，A 的 Tunnel 終點還是設爲 B 的 IPv4 位址。

•在 NAT 的位址對應表加上一個靜態對應[PF41]。當 NAT 從 public 網路收到 next level protocol 爲 IPv6 的 IPv4 封包(表示此封包爲 IPv6-in-IPv4 Tunnel 封包)，就修改此封包的目的位址爲 A 的 private IPv4 位址。此規則在 NAT 上的位址對應表如表 2-1。

表 2-1、Address Mapping Table for NAT-aware Tunneling

Internal IP 位址	External IP 位址	Remote IP 位址	Transport protocol
192.168.100.2	140.113.131.80	any	IPv6
Internal port	External port	Remote port	Lifetime
any	any	any	

(any 代表 NAT 不會檢查此欄位)

有了前述對 A、B 及 NAT 的修正，A、B 之間傳送 IPv6 封包的步驟如下。(參考圖 2-1)

Steps D.1, **D.2** 和 **Steps C.1, C.2** 相同。

Step D.3. 當 B 收到 A 經由 Tunnel 送來的 IPv6 封包，B 把要送給 A 的 IPv6 封包加上來源位址爲 61.218.105.10，目的位址爲 140.113.131.80，next level protocol 爲 IPv6 的 IPv4 表頭，形成 Tunnel 封包。此封包透過 IPv4 routing protocol 送往 NAT。

Step D.4. NAT 收下 Tunnel 封包，經 NAT 檢查發現 IPv4 表頭中 next level protocol 欄位爲 IPv6。NAT 將其目的 IPv4 位址以位址對應表中 Internal IP 位址替換 (192.168.100.2)，再將 Tunnel 封包送至 private 網路。A 收下此 Tunnel 封包後拆掉這個封包的 IPv4 表頭，得到由 B 送出的 IPv6 封包。

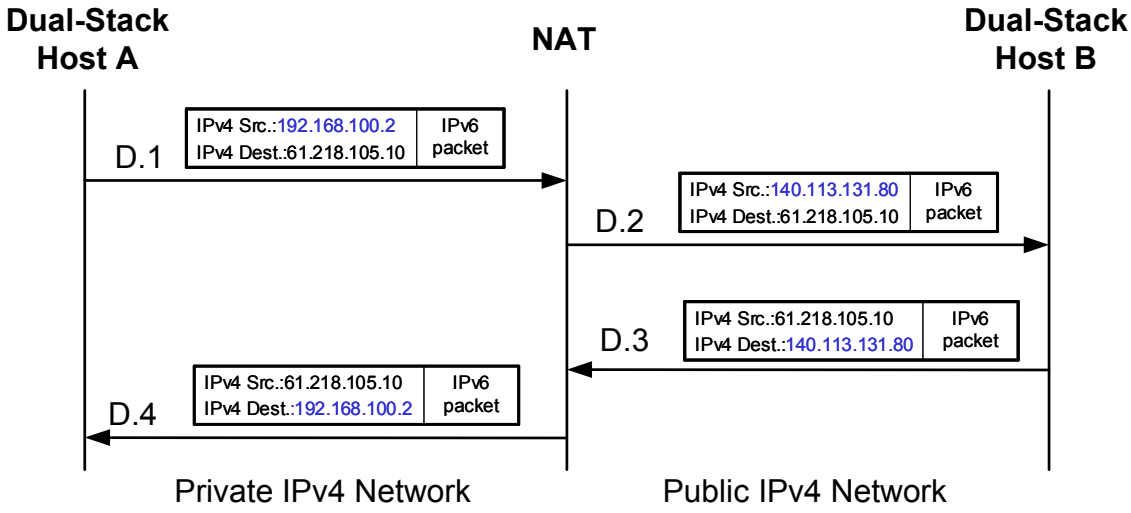

圖 2-1、Packet Transmission using NAT-aware Tunneling

圖 2-2 顯示 NAT-aware Tunneling 的 protocol stack，這個方法把 NAT 當作 Tunnel 封包的轉送點，利用 NAT 的位址對應，及 B 的 Tunnel 設定，讓 NAT 後方的設備可以正常連上 IPv6 網路。這個方法的缺點是：只能提供 NAT 後方一個 IPv6 設備建立 Tunnel，而且 NAT 必須懂得分辨封包的 next level protocol 爲 IPv6 才行。

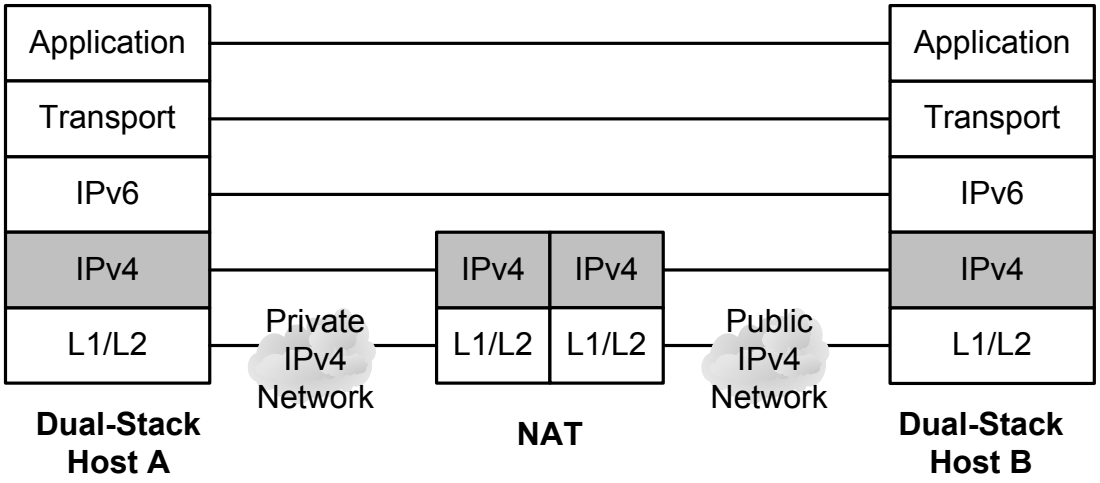

圖 2-2、Protocol Stack of NAT-aware Tunneling

2.2 NAT-unaware Tunneling

由於許多市面上出售的 NAT 設備並不見得提供使用者做 NAT-aware Tunneling 的設定，同時也不見得每位使用者都具備有足夠的網路知識進行正確的設定。因此上述方式並不適用於 ISP 等大量佈建的解決方案。在不更動 NAT 設定的前提下，以 Tunnel 讓 NAT 後方的設備連上 IPv6 網路，較著名的方法有三種：VPN [VPN]、UDP Tunnel [RFC3519] 和 Silkroad [Silkroad]。

2.2.1 VPN (Virtual Private Network)

使用 VPN 也可以讓 NAT 後方的設備連上 IPv6 網路。以圖 2-3 的網路架構爲例，網路上有 NAT 和 Dual-Stack Hosts A、B，在 public 網路另有一台 VPN server，VPN server 擁有多個可分配給 VPN client 的 public IPv4 位址。

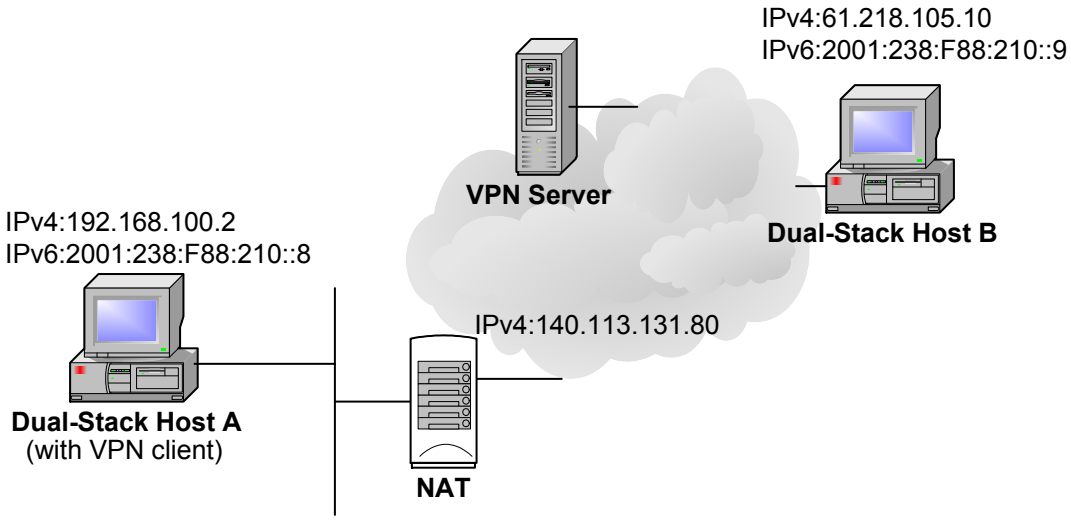

圖 2-3、VPN Tunnel with NAT

Host A 的 VPN client 利用建立 VPN 連線的通訊協定(如 L2TP、PPTP 等)向 VPN server 取得一個 global routable IPv4 位址，並在 A 和 VPN server 之間建立 VPN Tunnel。接著 A

就可以利用VPN client向VPN server取得的global routable IPv4位址和B建立Configured Tunnel。作法如下：A設定Configured Tunnel的目的地爲B的IPv4位址，B設定Configured Tunnel的目的地爲 A 向 VPN server 拿到的 IPv4 位址。

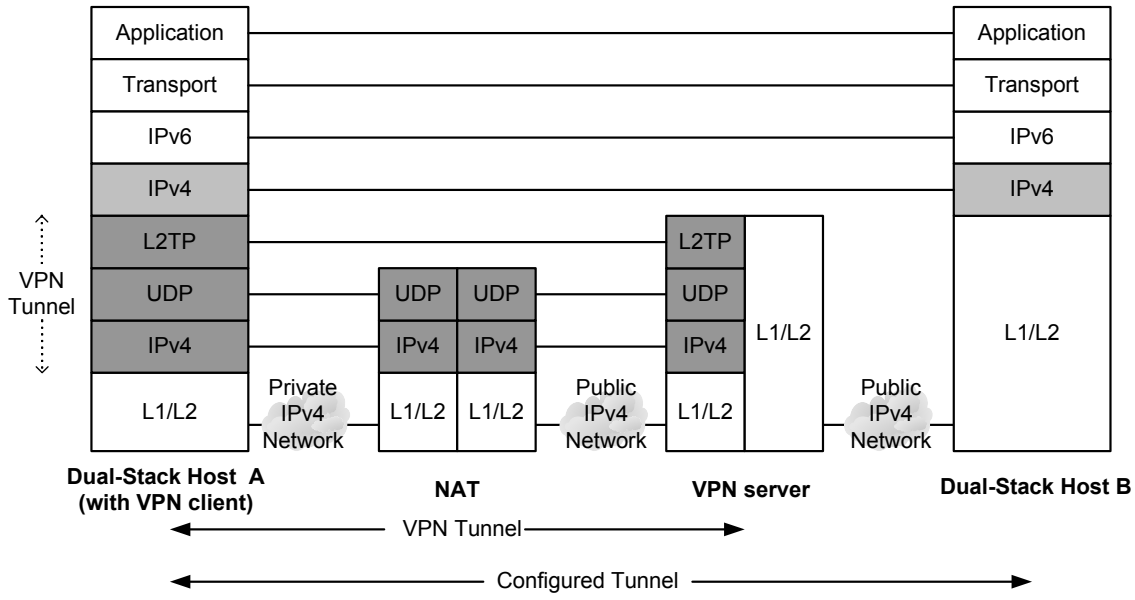

圖 2-4、Protocol Stack of VPN Tunnel

如圖 2-4 所示，在這裡使用到兩種 Tunnel 技術，A 和 VPN server 之間使用 VPN Tunnel(這裡以 L2TP 爲例，L2TP 包含 IPv4 表頭、UDP 表頭及 L2TP 表頭)，在 A 和 B 之間使用 Configured Tunnel 傳送IPv6封包。A和B之間建立Tunnel及傳送IPv6封包的流程(圖 2-5)及步驟如下：

Step E.1. A 透過 VPN client 向 VPN server 索取 global routable IPv4 位址，並透過 VPN client 和 VPN server 建立 VPN Tunnel。在 VPN client 和 VPN server 建立 VPN Tunnel 時，NAT 替 VPN Tunnel 用到的 IPv4 位址和 UDP port 建立位址對應表。

Step E.2. VPN server 派定 140.113.13.50 這個 global routable IPv4 位址給 A 的 VPN client。A 設定 140.113.13.50 爲本機的 IPv4 位址。A、B 之間設定 Configured Tunnel：設定 B 的 Tunnel 終點爲 A 向 VPN server 取得的 IPv4 位址(140.113.13.50)，A

的 Tunnel 終點爲 B 的 IPv4 位址(61.218.105.10)。

Step E.3. A 送出 IPv6 封包時，把封包用來源位址爲 140.113.13.50、目的位址爲 61.218.105.10、next level protocol 爲 IPv6 的 IPv4 表頭包住，再將此 IPv4 封包用 VPN 表頭包住，用 VPN Tunnel 將此封包送向 VPN server。

Step E.4. NAT 攔截 VPN Tunnel 封包。NAT 對 VPN Tunnel 表頭進行位址轉換，將表頭內容由 VPN header 改成 VPN header1(來源 IP 位址及來源 port 改爲 NAT 配置給這個連線的 External IP 位址及 External port)，接著將 VPN Tunnel 封包送至 public 網路。

Step E.5. VPN server 收下 VPN Tunnel 封包。VPN server 拆掉封包的 VPN 表頭，將得到的 IPv4 封包轉送給 B。B 收到 IPv4 封包後，將 IPv4 表頭拆掉即得到 IPv6 封包。

Step E.6. 欲回送 IPv6 封包給 A，B 將 IPv6 封包以來源位址爲 61.218.105.10、目的位址爲 140.113.13.50 的 IPv4 表頭包住，將此 IPv4 封包送上網路。

Step E.7. 此 IPv4 封包被 VPN server 收下，VPN server 將此封包用 VPN Tunnel 送向 NAT。

Step E.8. NAT 收下 VPN Tunnel 封包。NAT 轉換此封包的 VPN 表頭，將此封包轉送給 A。A 收到此封包，拆掉 VPN 表頭和 IPv4 表頭，得到 B 傳過來的 IPv6 封包。

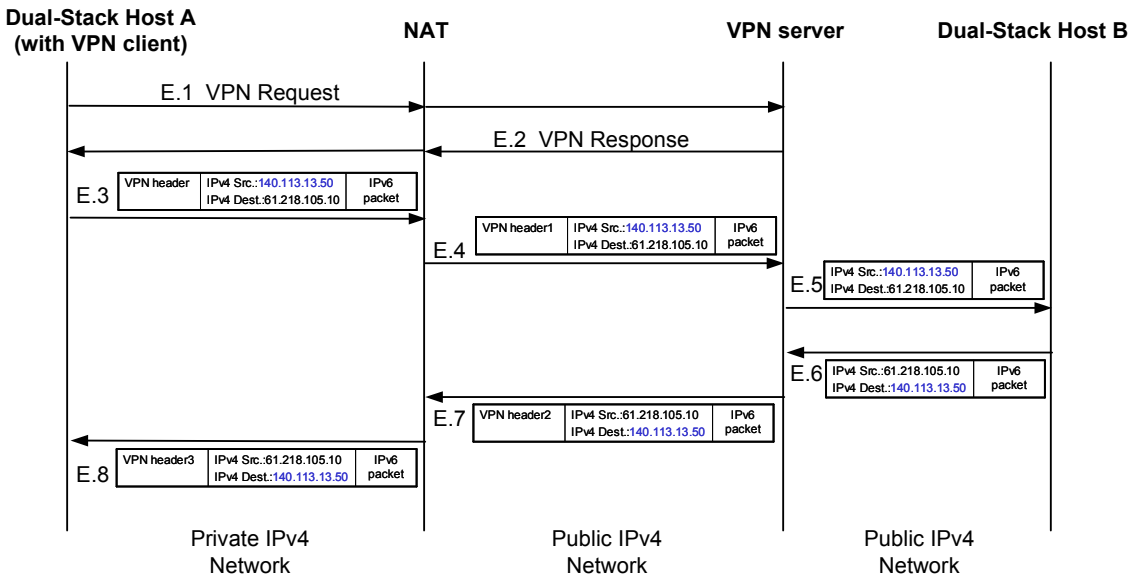

圖 2-5、Packet Transmission using VPN Tunnel

由於 VPN Tunnel 本身就具備穿越 NAT 的能力[RFC 2764]，加上 A 能向 VPN server 取得 global routable IPv4 位址，使得 NAT 對 IPv6 Tunneling 的限制不再存在，利用 VPN 可以輕鬆的建立 Configured Tunnel。然而，使用 VPN Tunnel 的缺點是每個連上 VPN server 的 VPN client 必須取得一個 global routable IPv4 位址，這破壞了原本企圖藉由 NAT 解決 IPv4 位址不足的問題，使得 NAT 的功能大打折扣。另外，VPN server 在使用上還有 SPOF(Single Point of Failure)，以及容易成爲封包傳送效能瓶頸的問題[RFC 2764]。

2.2.2 UDP Tunnel

UDP Tunnel 這個方法是第 2.1 節 NAT-aware Tunneling 的修正，其優點爲可以解決 NAT-aware Tunneling 需要更動 NAT，以及不能讓 NAT 後方多個網路設備同時連上 IPv6 網路的問題。

UDP Tunnel 和一般 IPv6-in-IPv4 Tunnel 的不同之處，是 UDP Tunnel 使用 IPv4 和 UDP 表頭建立 Tunnel，並把 IPv6 封包當作 UDP 的負載傳送。如此一來 NAT 不須判斷 IPv4 封包的內容爲 IPv6 (next level protocol 欄位爲 IPv6)，UDP 表頭帶的 port 資訊，就足以讓 NAT 正確地把封包轉送給 NAT 後方多個 Tunnel 設備。

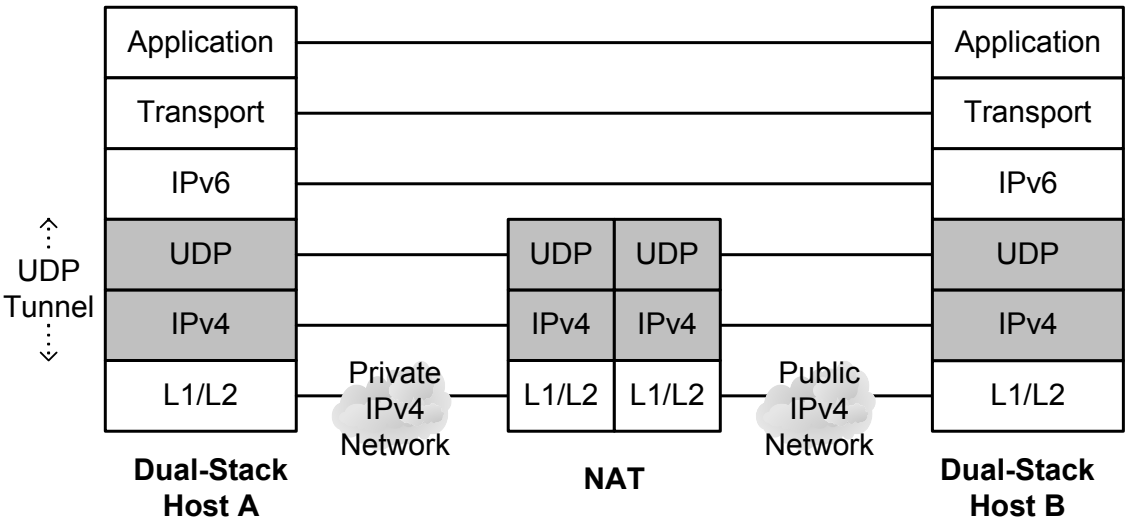

圖 2-6、Protocol Stack of UDP Tunnel with NAT

UDP Tunnel 的 protocol stack 如圖 2-6 所示。延續圖 1-7 的網路環境，同樣有 A、B 兩個 Dual-Stack Hosts 及 NAT。A 和 B 用來接收 Tunnel 封包的 UDP port 皆爲 9876，A 設定 Tunnel 的目的地爲 B 的 IPv4 位址及 port 9876，B 一開始不設定 Tunnel 的目的地。A、B 之間 IPv6 封包傳送過程如圖 2-7 所示(和圖 2-1 最大不同在於圖 2-1 使用 IPv6-in-IPv4 Tunnel，這裡使用 IPv6-in-IPv4/UDP Tunnel)，步驟如下：

Step F.1. A 把要送給 B 的 IPv6 封包用 IPv4 及 UDP 表頭包住，送給 NAT。UDP Tunnel 的 IPv4 來源位址爲 192.168.100.2、目的位址爲 61.218.105.10，next level protocol 爲 UDP，UDP 來源 port 及目的 port 皆爲 9876。

Step F.2. NAT 攔截 Tunnel 封包，根據封包內容建立新的對應表。UDP 封包的 IPv4 位址(192.168.100.2)、port 9876 被對應到 140.113.131.80、port 5432。NAT 根據對應表替換 Tunnel 封包的來源 IPv4 位址及來源 port，接著將此封包送至 public 網路。

Step F.3. B 從 UDP port 9876 收到從 A 送來的 UDP 封包，B 從此封包的來源位址得到可由 NAT 轉送封包給 A 的位址(亦即 IPv4 位址 140.113.131.80、UDP port 5432)。B 動態設定和 A 之間的 Tunnel，終點爲 140.113.131.80、UDP port 5432。當 B

回送 IPv6 封包給 A 時，B 利用 UDP Tunnel，並設 UDP 封包的 IPv4 來源位址爲 61.218.105.10、目的位址爲 140.113.131.80，next level protocol 爲 UDP，UDP 來源 port 爲 9876、目的 port 爲 5432。B 將此封包送至網路。

Step F.4. NAT 收下 Tunnel 封包，根據位址對應表修改封包的目的 IPv4 位址及目的 port 爲 192.168.100.2 及 9876，最後將此轉送封包給 A。A 將 IPv4 表頭及 UDP 表頭拆掉後，得到 IPv6 封包。

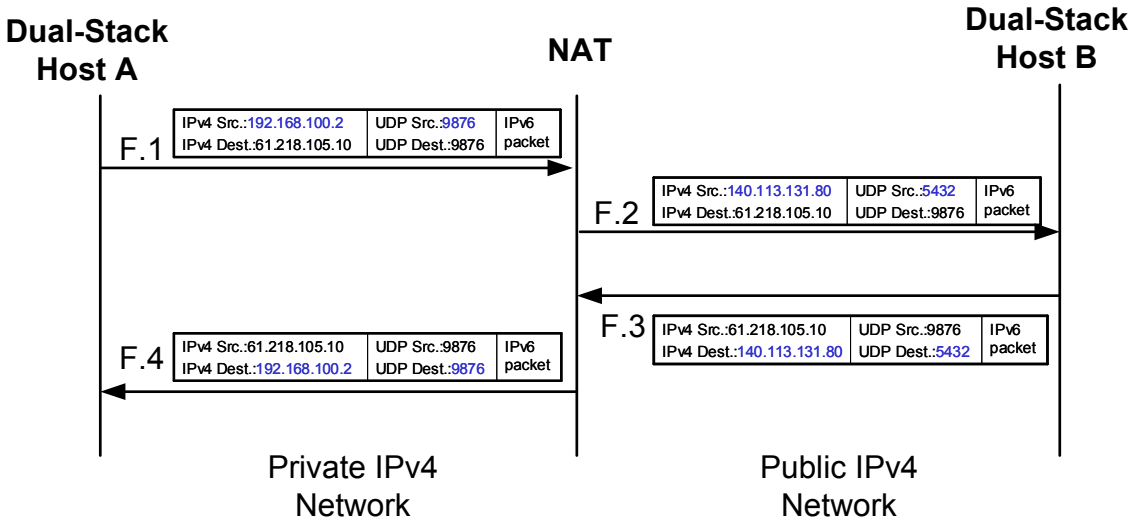

圖 2-7、Packets Transmission using UDP Tunnel

UDP Tunnel 可以直接使用 private IPv4 位址建立 Tunnel，不需要像 VPN Tunnel 一樣每個位在 NAT 環境(private 網路)的設備都先取得一個 global routable IPv4 位址。在 IPv4 位址不足的今天，顯然在這方面 UDP Tunnel 是較佳的解決方案。然而，使用 UDP Tunnel 也有一些缺點。例如上例中的 B 一開始沒辦法知道和 A 之間的 Tunnel 要建向哪個 IPv4 位址及 UDP port，必須等 A 先送 Tunnel 封包給 B，待 NAT 爲 A 建立位址對應後，IPv6 封包才得以從 B 傳送到 A。

2.2.3 Silkroad

Silkroad 是還在制定中的 Tunnel 標準，它也是使用 UDP Tunnel 穿越 NAT，不過在架構上比較複雜。在 Silkroad 的架構裡，有許多 Silkroad Access Router (SAR)連接 IPv6 網路和 IPv4 網路，位在 NAT 環境(private 網路)的設備必須選定一個 SAR 作爲它的轉送站。至於 NAT 後方設備要使用那一個 SAR 比較有效率，這個資訊則是由 Silkroad Navigator 負責提供。

CHAPTER 3

The Teredo Mechanism

Teredo 利用分散式的架構，配合 UDP Tunnel 傳送 IPv6 封包。它不需要修改 NAT 的設定，屬於 NAT-unaware Tunneling 的一種。

3.1 Operation Model

3.1.1 Components

如圖 3-1 所示，Teredo 的架構包含 **Teredo Client**、**Teredo Server** 和 **Teredo Relay** 三個元件。Teredo Client 安裝在 NAT 後方的電腦上，其功能爲建立和 Teredo Relay 之間的 Tunnel 以傳送 IPv6 封包。Teredo Server 同時連接 IPv4 網路和 IPv6 網路，其功能爲協助 Teredo client 取得 global IPv6 位址，以及協助 Teredo Client 建立和 Teredo Relay 之間的 Tunnel。Teredo Relay 爲連接 IPv4 網路和 IPv6 網路的 Relay Router，其功能爲轉送 IPv6 網路和 Teredo Client 之間的 IPv6 封包。

其中 Teredo Relay 必須維持和每個 Teredo Client 之間的 Tunnel 狀態，爲 stateful 設備。Teredo Server 不須記憶任何資訊，爲 stateless 設備。Teredo Server 的 stateless 和 Teredo Relay 的 stateful 特性，讓 Teredo 同時具備「管理集中」和「負載分散」這兩大優點：所有 Teredo Client 受到一個 Teredo Server 集中管理，在需要傳送 IPv6 封包時再動態尋找一個適當的 Teredo Relay 作爲 Relay Router。

此外，爲了讓 Teredo Client 和 IPv6 網路之間能更有效率的傳送封包，每個 Teredo Relay 皆會透過 IPv6 routing protocol 向 IPv6 網路宣傳(advertise) 3FFE:831F::/32 這個 IPv6 位址 prefix。如此，IPv6 網路送向 Teredo Client 的 IPv6 封包即可藉由 Teredo Relay 宣傳的 IPv6 位址 prefix，找到距離最近的 Teredo Relay 作爲 Relay Router。

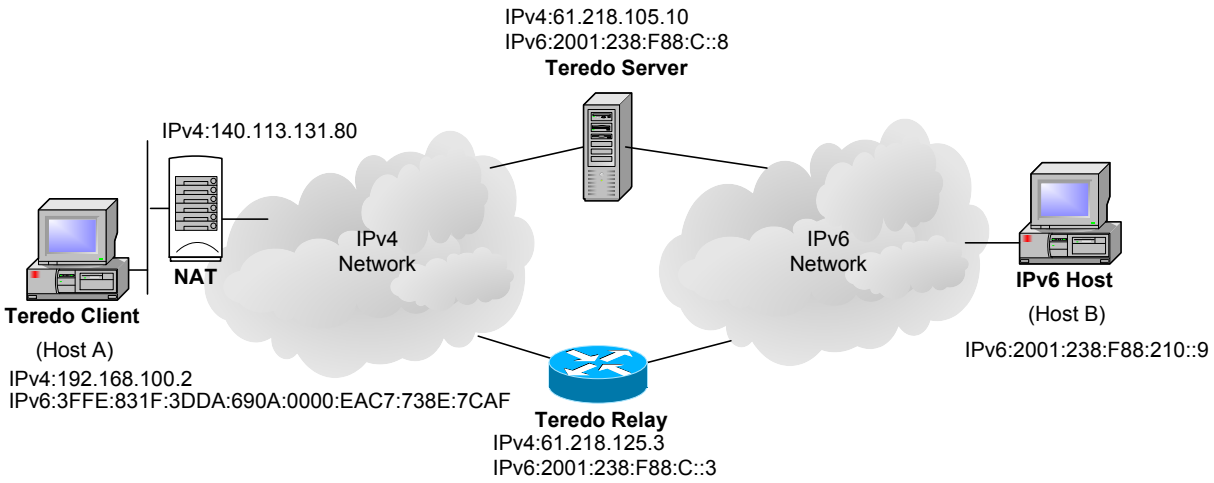

圖 3-1、Teredo Architecture

3.1.2 Packet Formats

Teredo 使用的 Tunnel 有 Simple encapsulation 和 Teredo encapsulation 兩種封裝格式(如圖 3-2 所示)。Simple encapsulation 將 IPv6 封包以 UDP 封包負載的型式傳送，主要用在傳送 Teredo Client 和 Teredo Relay 之間的 IPv6 封包。Teredo encapsulation 除了把 IPv6 封包當作 UDP 封包負載，還在 UDP 封包負載內夾帶額外的 Teredo data，這種 Tunnel 格式主要用在傳送建立 Tunnel 的資訊。

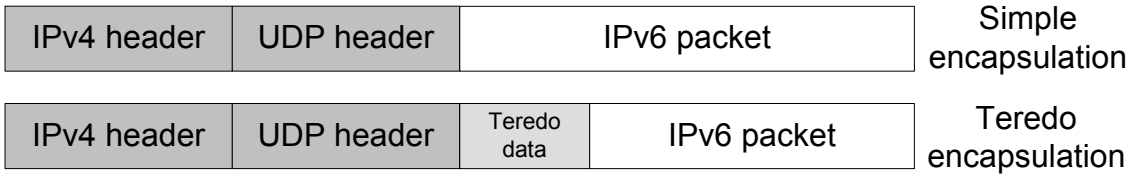

圖 3-2、Teredo Encapsulation Formats

在 Teredo 的架構裡，Teredo Server 和 Teredo Relay 必須使用 UDP port 3544 在 IPv4 網路收送 IPv6 封包(IPv6 封包在 UDP 負載中)，Teredo Client 可以使用任意 UDP port 在 IPv4 網路收送 IPv6 封包(IPv6 封包在 UDP 負載中)。Teredo Client 和 Teredo Relay 之間以 Simple encapsulation 傳送 IPv6 封包，Teredo Client 和 Teredo Server 之間或 Teredo Server 和 Teredo Relay 之間大多以 Teredo encapsulation 傳送 IPv6 封包(如圖 3-3 所示，圖中 Native 表示兩邊不將 IPv6 封包作任何 encapsulation，直接傳送 IPv6 封包)。

Teredo 另外定義了 Teredo Bubble，這種封包是只有 IPv6 表頭的 IPv6 封包(不帶任何負載的 IPv6 封包)。Teredo Bubble 以 Simple encapsulation 或 Teredo encapsulation 傳送，其用途爲建立 NAT 的位址對應(由 Teredo Client 從 private 網路向 public 網路送出 Teredo Bubble 時建立)。有關 Teredo Bubble 的運用方式，將在 3.2 節中說明。

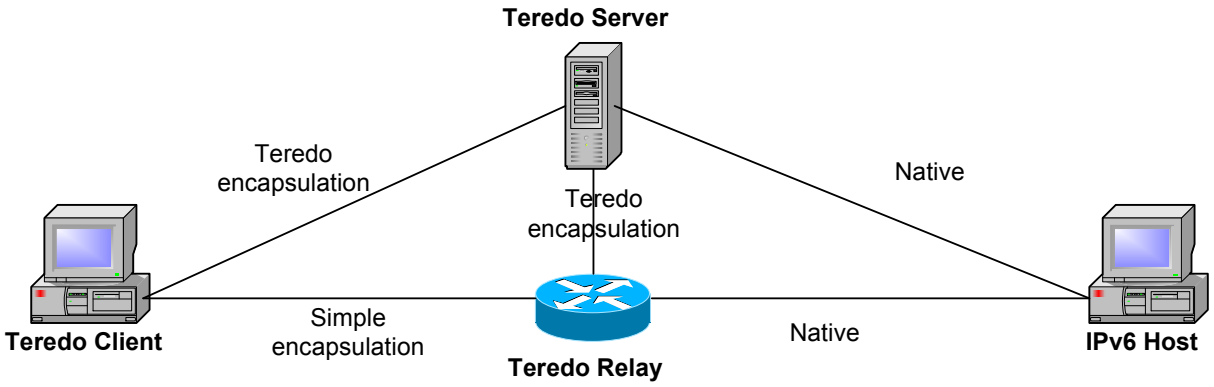

圖 3-3、Encapsulation Formats used between Teredo Nodes

3.1.3 Address Format

Teredo 對 IPv6 位址的編碼方式如圖 3-4 所示，其中各欄位所紀錄的資訊如下：

- **Teredo Prefix**

  32 位元的值爲 3FFE:831F::/32，標明此 IPv6 位址屬於一個 Teredo Client。

- **Teredo Server IPv4 address**

  32 位元紀錄 Teredo Server 的 IPv4 位址。

- **Flags**

  16 位元紀錄 Teredo Client 所在 NAT 的類型。當值爲 0x8000 時，表示 Teredo Client 位於 full cone NAT 後端，若爲 0x0000，則 Teredo Client 位於非 full cone NAT 後端。

- **Obfuscated Teredo Client Port**

  16 位元紀錄 Teredo Client 使用的 port 對應在 NAT 的 External port。

- **Obfuscated Teredo Client IPv4 address**

  32 位元紀錄 Teredo Client 使用的 IPv4 位址對應在 NAT 的 External IP 位址。

圖 3-4、Teredo IPv6 Address Format

其中 obfuscated 代表此欄位的內容爲原值和 1 作 XOR 的結果。舉例來說，若 Teredo Client 的 IPv4 位址對應在 NAT 的 External IPv4 位址爲 140.113.131.80 (0x8C718305)，Teredo Client 的 port 對應在 NAT 的 External port 爲 5432 (0x1538)，則其 Teredo IPv6 位址的 Obfuscated Teredo Client IPv4 位址欄位爲 0x738E7CFA (0x738E7CFA=0x8C718305⊕0xFFFFFFFF)，Obfuscated Teredo Client Port 欄位爲 0xEAC7 (0xEAC7=0x1538⊕0xFFFF)。

會這麼做的原因，是由於某些廠牌的「Smart NAT」在收到由 private 網路往 public 網路送的封包時，會搜尋 IPv4 封包的負載，把資料中和封包來源位址相同的 32 位元(以及和來源 port 相同的 16 位元)根據位址對應表作替換，並在封包由 public 網路經過 NAT 向 private 網路時也作相反的替換。Teredo 把 Teredo IPv6 位址中 Teredo Client IPv4 位址欄位和 Teredo Client port 欄位 obfuscate，可以避免 Tunnel 封包的 Teredo IPv6 位址被 NAT 當作 IPv4 位址予以置換。

3.1.4 IPv6 Address Acquiring

Teredo Client 必須先向 Teredo Server 之間進行 *Qulfification Procedure*，藉此取得一個 global IPv6 位址後，才能開啓和 IPv6 網路之間的溝通。以圖 3-1 的網路環境爲例，Host A 的 IPv4 位址爲 192.168.100.2，Teredo Server 的 IPv4 位址爲 61.218.105.10。在 A 的 port 9876 執行 Teredo Client，Teredo Client 向 Teredo Server 進行 Qualification Procedure 的部份過程如圖 3-5，步驟如下：

Step G.1. Teredo Client 以 link local IPv6 位址發出 IPv6 Router Solicitation 訊息。此訊息以 Simple encapsulation 的格式封裝，從 UDP port 9876 向 Teredo Server 的 UDP port 3544 送出。

Step G.2. Teredo Client 送出的 UDP 封包被 NAT 攔截，NAT 爲此封包建立新的位址對應表(如表 3-1)。NAT 根據表 3-1 轉換封包欄位內容。

Step G.3. 當 Teredo Server 從 UDP port 3544 收到 Teredo Client 送來的 IPv6 Router Solicitaiton，Teredo Server 將此 UDP 封包的來源 IPv4 位址(140.113.131.80)和來源 port (5432)儲存在 Teredo data 中，利用 Teredo encapsulation 回送 IPv6 Router Advertisement；封包的目的 IPv4 位址、目的 port 設爲 Teredo Server 剛收到 Teredo Client 送來封包的來源 IPv4 位址(140.113.131.80)、來源 port (5432)。Teredo Server

送出的 IPv6 Router Advertisement 帶有一個 64 位元的 IPv6 位址 prefix “3FFE:831F:3DDA:690A::/64”，其中前 32 位元爲 Teredo prefix (0x3FFE831F)，後 32 位元爲 Teredo Server 的 IPv4 位址 (0x3DDA690A=61.218.105.10)。

Step G.4. 當 UDP 封包(Teredo encapsulation)經過 NAT 時，NAT 根據表 3-1 轉換封包欄位內容。封包被 A 的 Teredo Client 收下。Teredo Client 收到此封包後，將 Teredo data 中紀錄的 External port 值(5432=0x1538)和 External IPv4 位址值 (140.113.131.80=0x8C718350) obfuscate，得到 0xEAC7 (0xEAC7=0x1538⊕0xFFFF)和 0x738E7CAF (0x738E7CFA=0x8C718305⊕0xFFFFFFFF)，此爲其 IPv6 位址的後 48 位元。另外，根據 IPv6 Router Advertisement 中的 IPv6 位址 prefix 值，Teredo Client 得到 IPv6 位址的前 64 位元 0x3FFE831F3DDA690A。

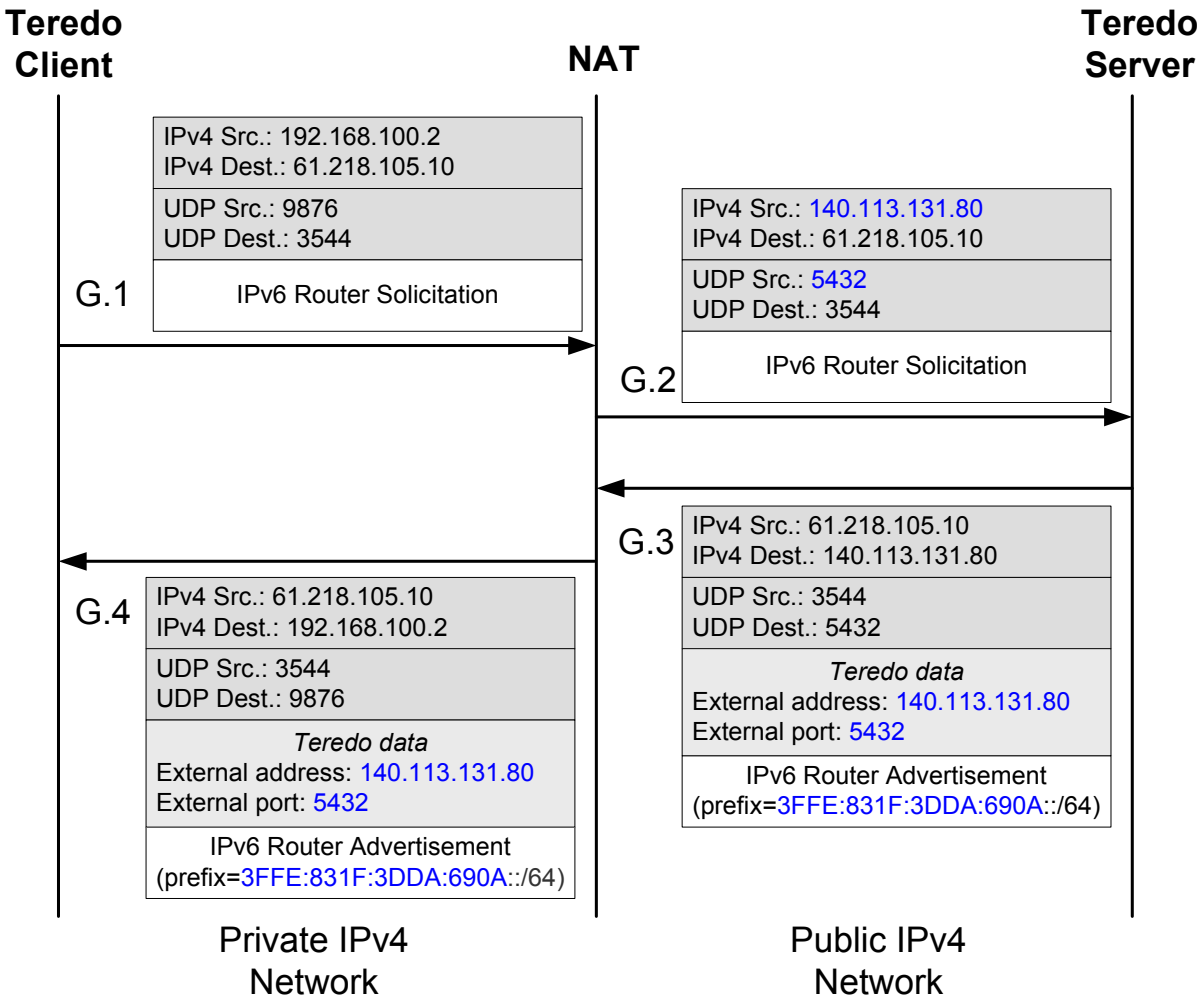

圖 3-5、Teredo Client IPv6 Address Acquiring

表 3-1、Address Mapping Table for Teredo Client-Teredo Server

Internal IP 位址	External IP 位址	Remote IP 位址	Transport protocol
192.168.100.2	140.113.131.80	61.218.105.10	UDP
Internal port	External port	Remote port	Lifetime
9876	5432	3544	

STUN (Simple Traversal of UDP Through Network Address Translators) [RFC 3489]規格中描述了位於 NAT 後端設備判斷 NAT 類型的方法。Teredo Client 依該流程，重複 **Steps G.1-G.4** 步驟和 Teredo Server 溝通，進而推算出 NAT 的類型，並得到 Teredo IPv6 位址中的 Flags 值。若 NAT 爲 full cone NAT，則 Teredo Client 判斷出 16 位元的 Flags 值應爲 0x8000。把所有從 Teredo Server 得到的資訊組合起來，Teredo Client 得到 3FFE:831F:3DDA:690A:8000:EAC7:738E:7CAF 這個 IPv6 位址。Teredo Client 將此 IPv6 位址設定在 A，往後 A 就可以利用這個 IPv6 位址透過 Teredo Client 和 IPv6 網路通訊了。

若這裡的 NAT 不爲 full cone NAT，則 16 位元的 Flags 值爲 0x0000，Teredo Client 得到 IPv6 位址爲 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF。

此外，位了保持 Teredo Client 的 IPv6 位址有效(意即 NAT 爲 Teredo Client-Teredo Server 之間建立的位址對應沒有改變，如此 Teredo IPv6 位址中夾帶的 obfuscated 資訊才有意義)，Teredo Client 會在 NAT 位址對應失效前向 Teredo Server 傳送訊息，以保持 NAT 位址對應存在。因爲 Teredo Client-Teredo Server 之間的位址對應一直被保持著，Teredo Server 可以隨時透過這個位址送訊息給 NAT 後端的 Teredo Client，這個特點對 Teredo Relay 傳送封包穿越 NAT 到 Teredo Client 有很大的幫助。此部份將在下面描述。

3.2 IPv6 Packet Transmission

Teredo 傳送 IPv6 封包的流程說明如下。假設在圖 3-1 的網路環境中，NAT 後端 Host A 和 IPv6 網路 Host B 之間以 Teredo 傳送 IPv6 封包。A 使用 UDP port 9876 執行 Teredo Client，和 Teredo Server 進行 Qualification Procedure 取得 IPv6 位址 “3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF”，並在 NAT 建立表 3-1 這個位址對應。B 的 IPv6 位址爲 2001:238:F88:210::9，Teredo Relay 的 IPv4 位址爲 61.218.125.3，NAT 的型態爲 port-restricted NAT。A 和 B 之間傳送 IPv6 封包有兩種不同的流程：若由 B 起始，其步驟爲 **Steps H.1-H.11**，若由 A 起始，其步驟爲 **Steps I.1-I.6**。

Case 1：由 public 網路上的節點 B 起始的封包傳送流程如圖 3-6 所示，詳述如下：

Step H.1. B 送 IPv6 封包給 A，IPv6 封包的來源位址及目的位址分別爲 2001:238:F88:210::9 及 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF。此封包透過 IPv6 routing protocol 被送往離 B 最近的 Teredo Relay。

Step H.2. Teredo Relay 收到此 IPv6 封包。若 Teredo Relay 曾轉送 IPv6 封包給 A，則 Teredo Relay 可以直接用 Simple encapsulation 轉送 IPv6 封包給 A (跳至 **Step H.7**)。若 Teredo Relay 不曾轉送 IPv6 封包給 A，基於 port-restricted NAT 的特性，必須要 Teredo Client 曾透過 NAT 與 Teredo Relay 通訊後，Teredo Relay 才能藉由 NAT 送資料給 Teredo Client。因此，Teredo Relay 先把要送給 Teredo Client 的 IPv6 封包暫存至 buffer，並向 Teredo Server 送出以 Teredo encapsulation 封裝的 Teredo Bubble。Teredo Bubble 的來源 IPv6 位址和目的 IPv6 位址分別爲 2001:238:F88:210::9 及 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF。Teredo Bubble 內 Teredo data 夾帶的來源 IPv4 位址和來源 port 爲 Teredo Relay 的 IPv4 位址(61.218.125.3)和 port 3544。Teredo encapsulation 封裝的 UDP 來源 port 和目

的 port 皆設爲 3544，IPv4 來源位址設爲 61.218.125.3，目的位址設爲 Teredo Server 的 IPv4 位址(可由目的 IPv6 位址中的 Teredo Server IPv4 位址欄位換算得知：0x3DDA690A=61.218.105.10)。

Step H.3. Teredo Server 從 port 3544 收到 Teredo Relay 送過來的 Teredo Bubble，TeredoServer 以 Teredo encapsulation 轉送此 Teredo Bubble 給 Teredo Client。Teredo Server 將自己的 IPv4 位址(61.218.105.10)及 port 3544 設爲 Teredo encapsulation 的來源 IPv4 位址及來源 port。目的 IPv4 位址和目的 port 則由目的 IPv6 位址計算得出(Obfuscated Teredo Client IPv4 位址欄位爲 0x738E7CAF，還原後得到 0x8C718350=140.113.131.80；Obfuscated Teredo Client Port 欄位爲 0xEAC7，還原後得到 0x1538=5432)。

Step H.4. NAT 收下內含 Teredo Bubble 的 UDP 封包。NAT 根據表 3-1 轉換封包欄位內容。

Step H.5. Teredo Client 收到 Teredo Relay 送來的 Teredo Bubble 後，用 Simple encapsulation 回送 Teredo Bubble 給 Teredo Relay。此 Teredo Bubble 的來源位址和目的位址分別爲 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF 和 2001:238:F88:210::9，Simple encapsulation 的來源 IPv4 位址和來源 port 設爲 192.168.100.2 和 9876，目的 IPv4 位址和目的 port 則設爲 Teredo data 中紀錄的來源 IPv4 位址(61.218.125.3)和來源 port (3544)值。

Step H.6. NAT 攔截內含 Teredo Bubble 的 Simple encapsulation 封包。NAT 根據此封包建立新的位址對應表(如表 3-2)，並在根據此對應表轉換封包欄位內容後，將此封包送給 Teredo Relay。

Step H.7.此時 NAT 已建立位址對應表。Teredo Relay 將 IPv6 封包用 Simple encapsulation 轉送 IPv6 封包至 Teredo Client(若有送向 Teredo Client 的 IPv6 封包被 Teredo Relay 暫存在 buffer，也在此時將 buffer 中的 IPv6 封包一併傳送)。Simple encapsulation 封包的來源 IPv4 位址和來源 port 爲 Teredo Relay 的 IPv4 位址(61.218.125.3)和 3544，目的 IPv4 位址和目的 port 則由目的 IPv6 位址(Teredo

Client 的 IPv6 位址)計算得到：還原目的 IPv6 位址的 Obfuscated Teredo Client IPv4 位址欄位得到目的 IPv4 位址爲 140.113.131.80 (0x738E7CAF⊕0xFFFFFFFF=0x8C718350=140.113.131.80)，還原目的 IPv6 位址的 Obfuscated Teredo Client Port 欄位得到目的 port 爲 5432 (0xEAC7⊕0xFFFF=0x1538=5432)。

Step H.8. NAT 收下內含 IPv6 封包的 UDP 封包，NAT 根據表 3-2 轉換封包欄位內容。

Step H.9. Teredo Client 收到從 B 送來，由 Teredo Relay 轉送的 Simple encapsulation 封包。Teredo Client 紀錄離 B 最近的 Teredo Relay 爲 61.218.125.3。往後當 A 要送 IPv6 封包給 B 時，Teredo Client 由此可知離 B 最近的 Teredo Relay 爲 61.218.125.3。Teredo Client 以 Simple encapsulation 傳送 IPv6 封包(設定來源 IPv4 位址和來源 port 分別爲 192.168.100.2 和 9876，目的位址和目的 port 分別爲 61.218.125.3 和 3544)。

Step H.10. NAT 收下內含 IPv6 封包的 Simple encapsulation 封包，NAT 根據表 3-2 轉換封包欄位內容。

Step H.11. Teredo Relay 收到此封包，將 Simple encapsulation 內的 IPv6 封包送至 IPv6 網路。此封包被 B 收下。

接著，A 和 B 之間就可以用 **Steps H.1, H.7-H.11** 的步驟互相傳送 IPv6 封包。

若 Teredo Relay 收到目的 IPv6 位址 Flags 值爲 0x8000 的 IPv6 封包(Flags 爲 0x8000 表示 NAT 上位址對應表爲表 3-3 這種型式，不限制任何 Remote 欄位的值)，Teredo Relay 可以直接用 **Steps H.1, H.7-H.11** 的流程將 IPv6 封包用 Simple encapsulation 轉送給 Teredo Client，不需爲了建立 Teredo Relay 的位址對應表傳送 Teredo Bubble，**Steps H.2-H.6** 這幾個步驟可以省略。

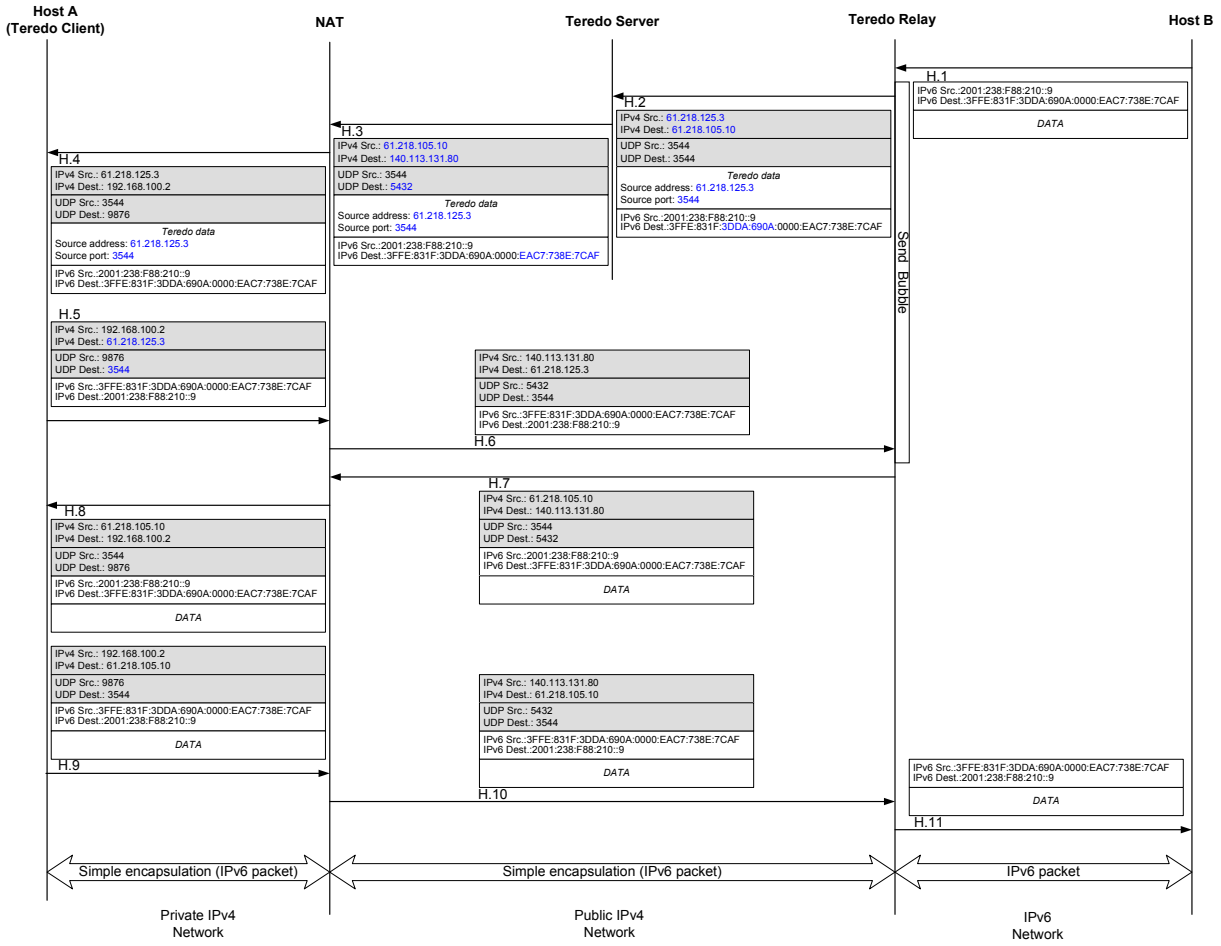

圖 3-6、Initial Steps for Packet Reception of Teredo Client

表 3-2、Address Mapping Table for Teredo Client-Teredo Relay

Internal IP 位址	External IP 位址	Remote IP 位址	Transport protocol
192.168.100.2	140.113.131.80	61.218.125.3	UDP
Internal port	External port	Remote port	Lifetime
9876	5432	3544	

表 3-3、Address Mapping Table for Teredo Client (on full cone NAT)

Internal IP 位址	External IP 位址	Remote IP 位址	Transport protocol
192.168.100.2	140.113.131.80	Any	UDP
Internal port	External port	Remote port	Lifetime
9876	5432	Any	

Case 2：由 A 起始 A 和 B 之間的封包傳送。(流程如圖 3-7)

Step I.1. A 把 IPv6 封包傳遞給 Teredo Client，由 Teredo Client 負責將封包送至 B。由於 B 不曾送 IPv6 封包給 Teredo Client，Teredo Client 不知道離 B 最近 Teredo Relay 的 IPv4 位址。A 送給 B 的 IPv6 封包會被存放至 Teredo Client 的 buffer。Teredo Client 用 Simple encapsulation 向 Teredo Server 送出目的 IPv6 位址爲 B、來源 IPv6 位址爲 A 的 ICMPv6 Echo Request。

Step I.2. 內含 ICMPv6 Echo Request 的 Simple encapsulation 封包被 NAT 攔截，NAT 根據表 3-1 將此封包作位址轉換後，將此封包送至 Teredo Server。

Step I.3. Teredo Server 從 port 3544 收到此封包，將 Simple encapsulation 裡的 ICMPv6 Echo Request 送至 IPv6 網路。

Step I.4. B 收到 Teredo Server 轉送過來的 ICMPv6 Echo Request，回送 ICMPv6 Echo Response 給 A。此 IPv6 封包透過 IPv6 routing protocol，找到離 B 最近的 Teredo Relay。(若此 Teredo Relay 不曾送 IPv6 封包給 Teredo Client，Teredo Relay 會執行傳送 Teredo Bubble 的流程，請參照 **Steps H.2-H.6**)

Step I.5. Teredo Relay 以 Simple encapsulation 方式將 ICMPv6 Echo Response 傳送給 Teredo Client。封包的目的 IPv4 位址和目的 port 設定和 **Step H.7** 相同。此 UDP 封包被送往 NAT。

Step I.6. 內含 ICMPv6 Echo Response 的 Simple encapsulation 封包被 NAT 攔截，NAT 根

據表 3-2 將此封包作位址轉換後，將此封包送至 Teredo Client。Teredo Client 收到來自 B 的 ICMPv6 Echo Response，Teredo Client 紀錄離 B 最近的 Teredo Relay 爲轉送此 ICMPv6 Echo Response 的 Teredo Relay (61.218.125.3)。

接著，A 和 B 之間就可以用 **Steps H.1, H.7-H.11** 的流程互相傳送 IPv6 封包。

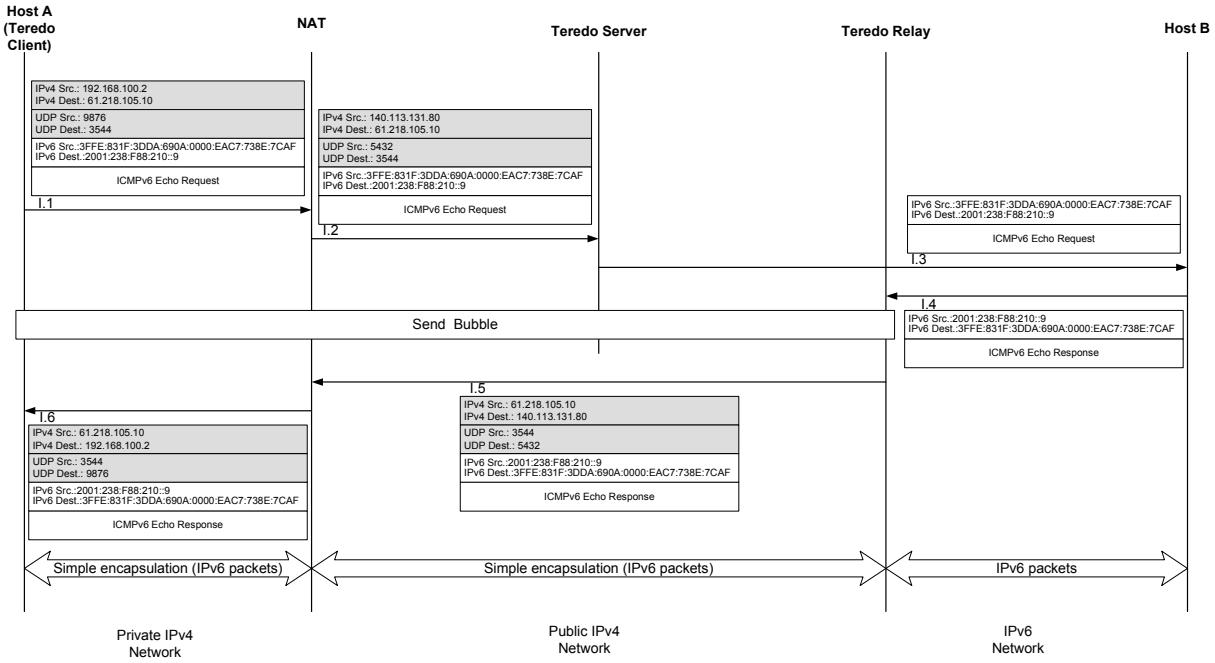

圖 3-7、Initial Steps for Packet Transmission from Teredo Client.

3.3 Features

Teredo 和第 2 章提到的幾個 Tunneling 方法相比，有下面幾個優點。

- **Robust**

  Teredo 以其廣佈的 Teredo Relay 轉送 IPv6 封包，若其中幾個 Teredo Relay 出問題，IPv6 封包可以透過 IPv6 routing 機制由另外一個 Teredo Relay 接續傳送，不會因爲其中一個網路轉送點的問題造成封包無法傳送。

- **Scalable**

  當 Teredo Relay 負載增加時，網路管理者可以藉由動態增加 Teredo Relay 的數量，讓 IPv6 封包選擇最近的 Teredo Relay 傳遞封包，以分散 Teredo Relay 的負載。當 Teredo Client 增多時，只需擴大佈建 Teredo Relay 即可，不需要更動到中間網路的設定。

- **Automatic**

  對 Teredo Client 而言，IPv6 位址的取得與 Tunnel 的建立，都不需要網路管理者針對使用者逐一設定，而可以透過 Teredo Server 的協助，由 Teredo Client 動態完成。

然而，Teredo 目前並沒有辦法穿越較複雜的 symmetric NAT。這仍是一個需要克服的問題。

CHAPTER 4

Three Teredo Implementations

於西元 2003 年，我們在 Linux 平台開發了 NICI-Teredo [NICI-Teredo]。NICI-Teredo 包含 Teredo Server 和 Teredo Relay 兩部份軟體，這兩部份可以被同時安裝在同一台主機上，也可以分別安裝在多個不同的主機。NICI-Terdo server 的軟體架構如圖 4-1 (a)，它是一個實作在 user-level 的 daemon。NICI-Teredo Relay 的軟體架構如圖 4-2 (a)，它包括 kernel-level 部分(圖 4-2 ③)和 user-level 部分(圖 4-2 ①和②)。kernel-level module 負責提供高速的 IPv6 封包轉送功能，並藉由 user-level 程式提供控制此 kernel-level module 的介面。

於西元 2003 年同年，6WIND、LIP6 和 Euronetlab 也協力開發了另外一個在 FreeBSD 平台的 Teredo 實作[6WIND]。到了西元 2004 年，一個跑在 Solaris、FreeBSD 和 Linux 的跨平台 Miredo-Teredo [MIredo]也被開發出來。這三個 Teredo 實作分別由三個不同的組織開發，彼此函式庫沒有任何關聯。本章就 Teredo 架構的 Teredo Server 和 Teredo Relay 兩個元件，分別介紹這三個 Teredo 實作不同的軟體架構。

4.1 Teredo Server

4.1.1 NICI-Teredo Server

NICI-Teredo Server (圖 4-1 (a))以一個 IPv6 raw Ethernet socket (圖 4-1 (b))和兩個 IPv4 UDP socket (圖 4-1 (c))收送封包。NICI-Teredo Server 的軟體架構可以分爲四個部份。

Packet processor (圖 4-1 ①) 負責 IPv6 封包的封裝，以及將封裝過後的 IPv4 UDP 封包解封裝成 IPv6 封包。*Dispatcher* (圖 4-1 ②)檢查 *packet processor* 送來的 IPv6 封包格式，並將通過查核的 IPv6 封包配送到適當的程式處理。*Qualification function* (圖 4-1 ③)提供和 Teredo Client 之間的 Qualification Procedure 功能(如第 3.1.4 節)。這個功能協助 Teredo Client 偵測其所在 NAT 的種類，以及選定一個 Teredo IPv6 位址使用。Teredo Server 利用圖 4-1 的路徑Ⓐ→Ⓑ→Ⓒ→Ⓓ→Ⓔ處理和 Teredo Client 之間進行 Qualification Procedure 的封包。當 Teredo Client 要和一般 IPv6 主機溝通(如第 3.2 節的 **case 2**)，*ICMPv6 relay function* (圖 4-1 ④)協助這個 Teredo Client 找到適當的 Teredo Relay 轉送封包。Teredo Server 利用圖 4-1 的路徑Ⓐ→Ⓑ→Ⓕ→Ⓖ轉送 ICMPv6 Echo Request (如 **Steps. I.1-I.3**)至 IPv6 網路，當 IPv6 主機回送 ICMPv6 Echo Response 時(如 **Steps. I.4-I.6**)，Teredo Client 藉此找到和此 IPv6 主機之間最適當的 Teredo Relay。

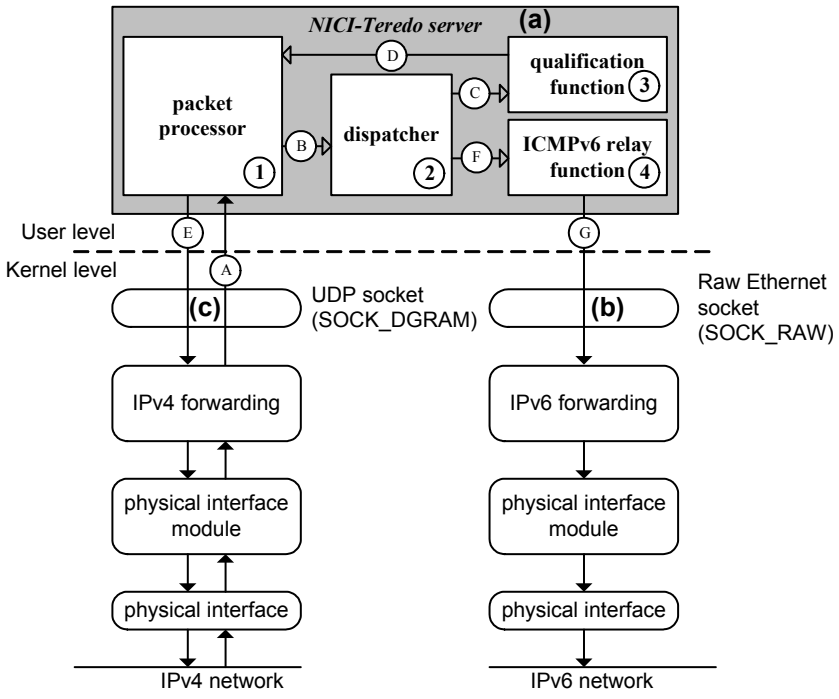

圖 4-1、Software Architecture of NICI-Teredo Server

4.1.2 6WIND-Teredo Server and Miredo-Teredo Server

由於 6WIND、Miredo 的 Teredo Server 設計和 NICI-Teredo Server 非常相似，這裡省略這兩個 Teredo Server 的軟體架構。

4.2 Teredo Relay

4.2.1 NICI-Teredo Relay

NICI-Teredo Relay (圖 4-2 (a))運用 Linux kernel 的 IPv6、IPv4 forwarding 機制(圖 4-2 (b)及(c))，以提供 IPv6 封包轉送功能。NICI-Teredo Relay 由三個 module 組成。*Prefix advertisement module* (圖 4-2 ①)向 IPv6 網路宣傳 3FFE:831F::/32 這個 IPv6 位址 prefix。透過這個 IPv6 位址 prefix 宣傳，由 IPv6 網路送向 Teredo Client 的封包可以被送到離封包來源最近的 Teredo Relay，並透過這個 Teredo Relay 轉送。*Routing management module* (圖 4-2 ②)初始化 Linux kernel 的封包轉送，它設定 IPv6 forwarding (圖 4-2 (b))將目的位址爲 Teredo Client 的 IPv6 封包(目的位址的 IPv6 位址 prefix 爲 3FFE:831F::/32)透過 *relay module* 轉送到 Teredo Client，並設定 IPv4 forwarding (圖 4-2 (c))把從 Teredo Client 送來的 IPv4 UDP 封包(封包的目的 UDP port 爲 3544)透過 *relay module* 轉送到 IPv6 網路。*Relay module* (圖 4-2 ③)以 `udpip6_tunnel_xmit()` (圖 4-2 Ⓐ) 和 `udpip6_rcv()` (圖 4-2 Ⓑ)這兩個回呼函式(callback function)提供封包的封裝(encapsulation)、解封裝(decapsulation)功能，而這兩個函式分別被 Linux kernel 的 IPv6 forwarding 及 IPv4 forwarding 呼叫。*Relay module* 呼叫 `IPTUNNEL_XMIT()`[1] (圖 4-2 Ⓒ)，透過 IPv4 forwarding 將封裝好的 IPv4 UDP 封包傳送至 Teredo Client。另一方面，*relay module* 呼叫 `netif_rx()`(圖 4-2 Ⓓ)，透過 IPv6 forwarding 將解封裝後的封包傳

[1] 依據 Linux 程式設計的習慣，大寫英文字母表示的函式代表巨集(macro)，如 `IPTUNNEL_XMIT()`，而用小寫英文字母表示的函式代表一般函式，如 `udpip6_rcv()`。同樣的表示法在後面介紹 6WIND 在 FreeBSD 的 Teredo Relay 實作時也會出現。

送至 IPv6 網路。

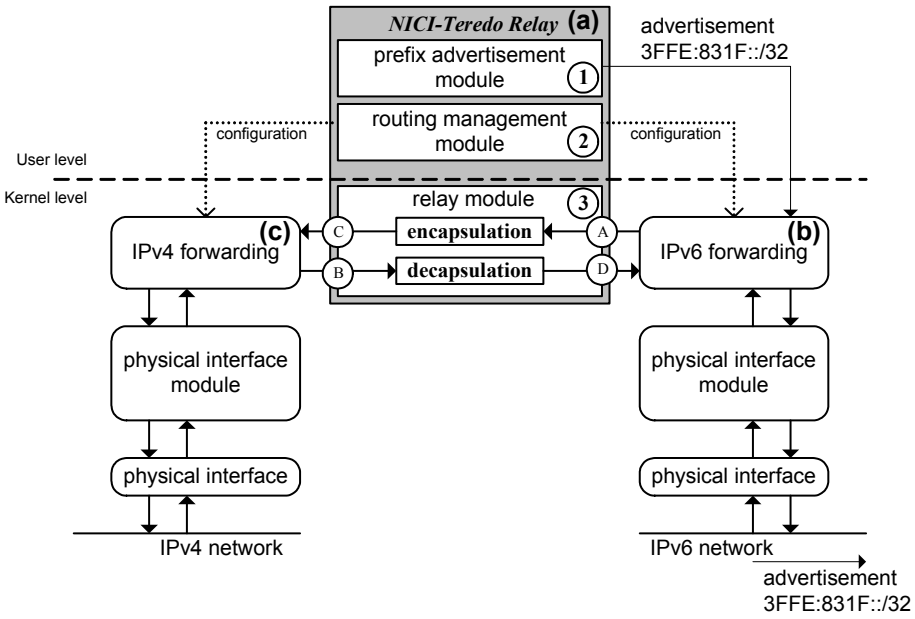

圖 4-2、Software Architecture of NICI-Teredo Relay

4.2.2 6WIND-Teredo Relay

6WIND-Teredo Relay (圖 4-3 (a))使用 FreeBSD kernel 內的 *netgraph* 網路子系統，配合運用同在 FreeBSD kernel 的 IPv6 forwarding 機制(圖 4-3 (b))和 socket 函式(圖 4-3 (c))，以提供封包轉送功能。和 NICI-Teredo Relay 不同之處，在於 NICI-Teredo Relay 使用單一 relay module (圖 4-2 ③)提供所有封包處理功能，而 6WIND-Teredo Relay 依賴多個 *netgraph* submodule (圖 4-3 ③)和 kernel IPv4 UDP socket 處理封包。6WIND-Teredo Relay 主要由三個部份組成。*Prefix advertisement module* (圖 4-3 ①)向 IPv6 網路宣傳 3FFE:831F::/32 這個 IPv6 位址 prefix，它和 NICI-Teredo Relay 的 prefix advertisement module 功能相同。*Routing management module* (圖 4-3 ②)初始化 FreeBSD kernel 的 IPv6 forwarding，設定它將目的爲 Teredo Client 的 IPv6 封包送至 *netgraph module*。*Netgraph module* (圖 4-3 ③)使用三個 submodule 處理 IPv6 封包。*NgN* submodule 利用 `netisr_dispatch()`和 `ng_iface_output()`函式連結到 IPv6 通訊協定堆疊 (protocol stack)以傳送 IPv6 封包。*Ksocket* submodule 以 port 3544 建立在 kernel 的 IPv4

UDP socket，並利用 `so_pru_sosend()`和 `so_pru_soreceive()`函式和 Teredo Client 之間收送 IPv4 UDP 封包。*Ksocket* submodule 將 IPv6 封包被當作這個 IPv4 UDP socket 的資料收送，封包的封裝以及封包解封裝都是透過這個 IPv4 UDP socket 完成。*Ng_teredo* submodule 透過 `NG_SEND_DATA()`和 `ng_teredo_rcvdata()`函式連結 *ngN* submodule 和 *ksocket* submodule，將 IPv6 封包在這兩個 submodule 之間傳送：被 *ngN* submodule 收到的 IPv6 封包透過 *ksocket* submodule 被寫到 IPv4 UDP socket，反之亦然。

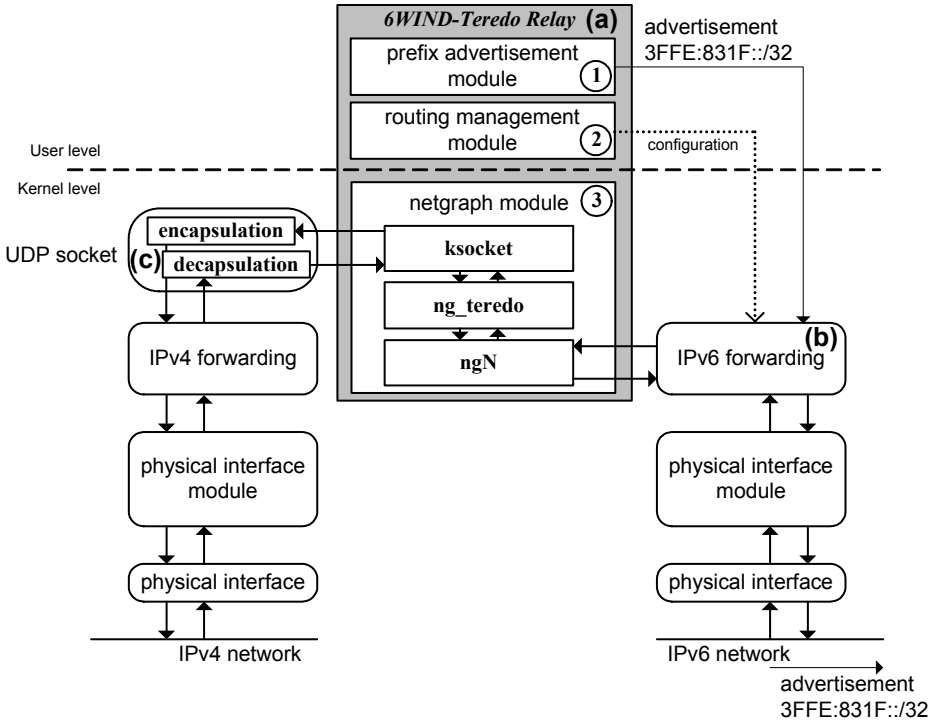

圖 4-3、Software Architecture of 6WIND-Teredo Relay

4.2.3 Miredo-Teredo Relay

Miredo-Teredo Relay (圖 4-4 (a))的軟體架構和前面兩個實作的差異較大。Miredo-Teredo Relay 透過 Linux kernel 的 IPv6 forwarding 機制(圖 4-4 (b))、IPv4 UDP socket (圖 4-4 (c))和 *tun module* (圖 4-4 (d))提供 IPv6 封包轉送的功能。和 NICI-Teredo Relay 的不同之處，在於 NICI-Teredo Relay 使用在 kernel-level 的 relay module (圖 4-2 ③)，而 Miredo-Teredo Relay 使用在 user-level 的 packet processing module (圖 4-4 ③)處理封包。Miredo-Teredo

Relay 可以分爲四個部份。*Prefix advertisement module* (圖 4-4 ①)向 IPv6 網路宣傳 IPv6 位址 prefix，*Routing management module* (圖 4-4 ②)初始化 Linux kernel 的 IPv6 forwarding，使送往 Teredo Client 的 IPv6 封包透過 *tun module* 轉送到目的地。*Prefix advertisement module* 和 *routing management module* 的功能和 NICI-Teredo Relay 以及 6WIND-Teredo Relay 對應的 module 功能相同。在 user-level 的 *packet processing module* (圖 4-4 ③)以 port 3544 建立 IPv4 UDP socket，並透過這個 socket 和 Teredo Client 之間收送 IPv4 UDP 封包。藉由將 IPv6 封包當作 IPv4 UDP socket 的資料傳送，IPv6 封包的封裝和解封裝都藉由此 socket 完成。*Packet processing module* 利用 `memcpy()`以及 `sendto()`、`recvfrom()`這兩個 socket 函式，將 IPv6 封包在 *tun module* 和 IPv4 UDP socket 之間傳送(圖 4-4 Ⓐ和Ⓑ)。透過上面描述，可以知道 Miredo-Teredo Relay 的軟體架構和 NICI-Teredo Relay 以及 6WIND-Teredo Relay 差異非常大。

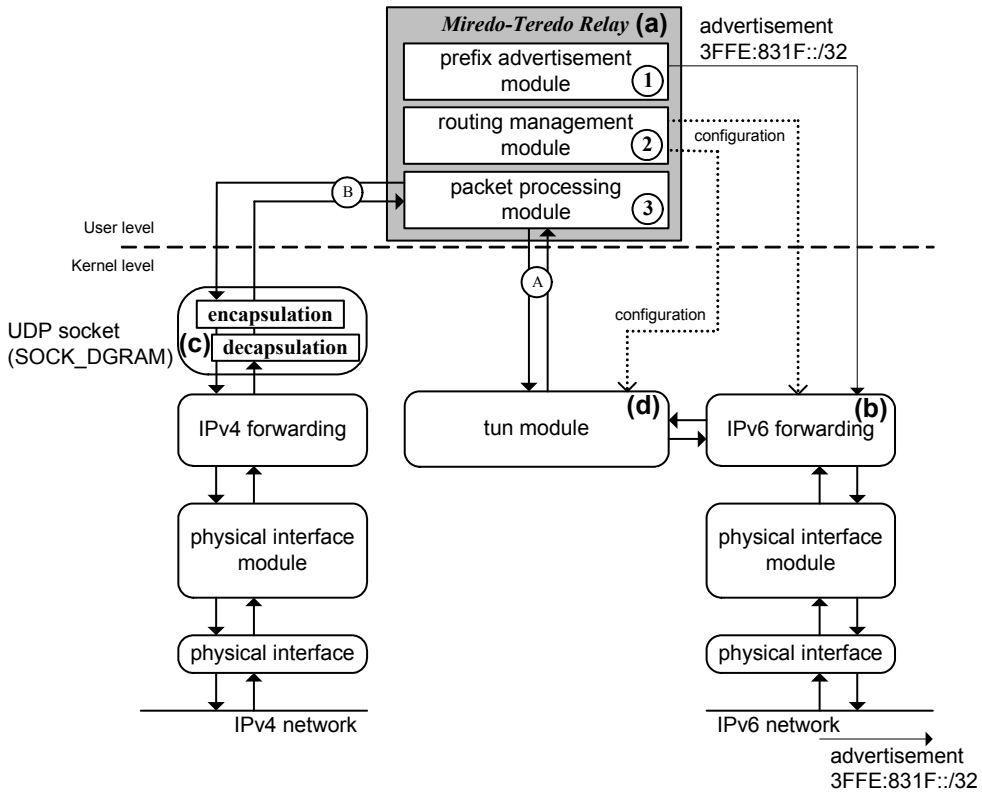

圖 4-4、Software Architecture of Miredo-Teredo Relay

Teredo Relay 軟體架構設計的不同，對 IPv6 封包處理效率影響非常大。由於 NICI-Teredo Relay 在 kernel-level 處理封包，省去封包在 user-level 和 kernel-level 之間拷貝所需的時間，因此 NICI-Teredo Relay 處理封包所需的時間較短。下一章將詳細描述這三種 Teredo Relay 之間的效能比較。

最後，需要注意的是，雖然 NICI-Teredo Relay 在實作時牽涉到 Linux kernel hacking，但它是以 LKM (loadable kernel module) [LKM]架構實作。亦即安裝 NICI-Teredo Relay 非常簡易，使用者完全不需要修改或重新編譯 Linux kernel，在使用上可以說是非常方便。

CHAPTER 5

Performance Measurement

在 Teredo 架構中，由於 Teredo Relay 處理 IPv6 網路和 Teredo Client 之間大量的封包轉送，相較於 Teredo Server 只處理少量封包，Tereso Relay 較有可能成爲 Teredo 架構裡的效能瓶頸。本章實際量測 NICI、6WIND 和 Miredo 這三個 Teredo Relay 處理 IPv6 封包耗費的時間，實地比較這三個 Teredo Relay 的效能。

5.1 Testing Environment

根據 RFC 2544 (Benchmarking Methodology for Network Interconnect Devices) [RFC2544] 的測試架構，我們將 Teredo 架構(如圖 3-1)中負責收送 IPv6 封包的 IPv6 Host 和 Teredo Client 以同一台主機模擬，架設出如圖 5-1 的測試環境。

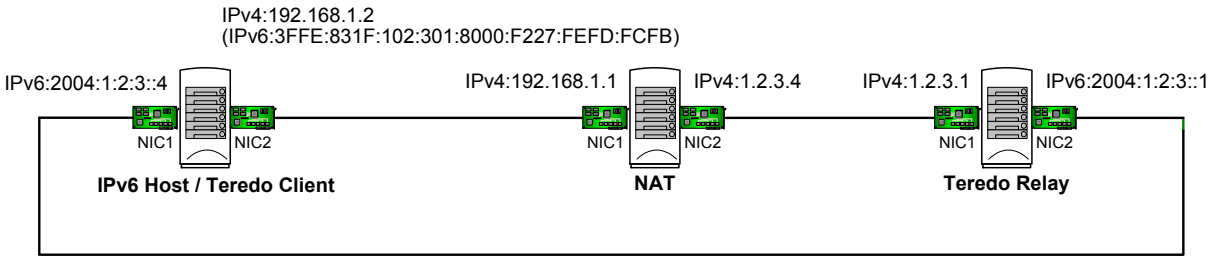

圖 5-1、Testing Environment

由於 IPv6 Host 和 Teredo Client 在同一台主機上，它們有完全一致的系統時間。藉此可

以精準地測量出 IPv6 Host 和 Teredo Client 之間傳送 IPv6 封包的 one way delay，不需要透過量測 IPv6 Host 和 Teredo Client 之間的 round-trip delay 間接估計 one way delay[2]。

在此測試環境中，三台主機的配備皆爲 AMD Athlon 1800+ CPU、256MB 記憶體和兩個 RealTek 8139 100BaseTX 網路介面。三台主機之間以 100BaseTX Ethernet 互相連接，Teredo Relay 和 IPv6 Host 之間爲 IPv6 網路，Teredo Client 和 NAT 之間以及 NAT 和 Teredo Relay 之間爲 IPv4 網路(IPv6 封包在這兩段網路中以 Simple encapsulation 方式傳送)。IPv6 Host/Teredo Client 以及 NAT 的作業系統爲 Linux RedHat 9。NAT 使用 RedHat 內部的 *iptables* [iptables]系統指令架設，並設定 NAT 型態爲 full cone NAT。在此爲了比較不同版本 Teredo Relay 處理 Teredo Tunneling 的效能，我們在 Teredo Relay 這台主機安裝 *LILO* 多重開機軟體，採用 Linux RedHat 9 及 FreeBSD 4.9 Release 作爲 Teredo Relay 的作業系統進行測試。其中 NICI 和 Miredo 使用 Linux，6WIND 使用 FreeBSD，測試以下版本的 Teredo Relay：NICI (version 0.3)、6WIND (version 1.13)、Miredo (version 0.3.0)。

5.2 Testing Scenarios

這裏使用 C 語言撰寫測試程式，配合 *Pcap* 函式庫[pcap]以監看主機的網路介面。利用 *Pcap* 函式庫擷取封包被作業系統收取、以及封包送出作業系統的時間，並透過計算封包出入作業系統的時間差，以得到主機處理封包耗費的時間[3]。依據 RFC 2544 的建議，我們針對封包大小爲 64 bytes、512 bytes、1280 bytes 的三種 IPv6 封包進行測試。以下分兩種情況測量 Teredo Relay 處理 IPv6 封包的時間，詳細測量方式如下。

---

[2] 這裡的 Teredo Client 是以 UDP daemon 模擬，因此 IPv6 Host 和 Teredo Client 之間不會有 IPv6 routing 的問題。

[3] *Pcap* 函式庫目前在 Linux 和 FreeBSD 平台都有實作。它整合 raw socket 的功能，提供應用程式監看網路封包的能力。

Scenario 1:

移除 Teredo Relay 主機作業系統中所有非必要執行程序(如 httpd, sendmail)，讓作業系統只具有 Teredo Relay 單一功能。在此前提下量測 Teredo Relay 在處理 IPv6 封包 en-tunnel、de-tunnel 兩種情況耗費的時間。這兩種情況分別是：由 IPv6 Host 送 IPv6 封包經過 Teredo Relay 到 Teredo Client (如圖 5-2)，以及反方向由 Teredo Client 送 IPv6 封包經過 Teredo Relay 到 IPv6 Host。測試時仿照 *ping* 指令測量 round-trip delay 的方式，我們每秒送出 1 個測試封包，測量 10000 個封包，最後統計量測結果。

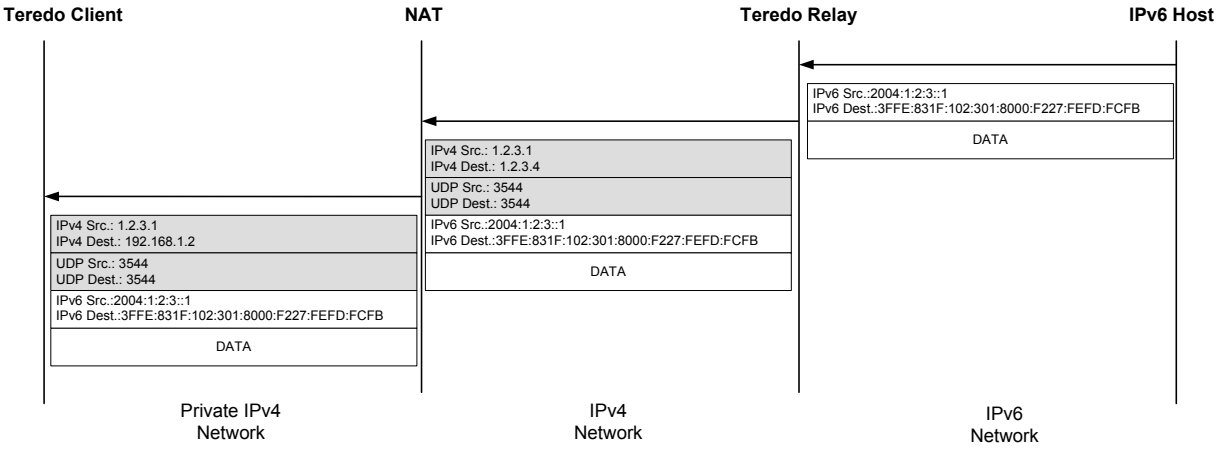

圖 5-2、Packet Transmission from IPv6 Host

圖 5-3 標明 IPv6 封包由 IPv6 Host 送至 Teredo Client (如圖 5-2)途中所耗費的各段時間。其中(1)、(5)和(9)是主機將封包傳送至網路耗費的時間，(2)、(6)和(10)是封包在網路上傳輸耗費的時間，(3)、(7)和(11)是主機從網路接收封包耗費的時間，(4)是 Teredo Relay 主機將 IPv6 封包 en-tunnel 爲 Simple encapsulation 耗費的時間，(8)是 NAT 主機轉換 IPv4 封包表頭耗費的時間。

對應到圖 5-3，此 scenario 中要量測的時間爲(4)，以及 Teredo Relay 處理反方向 de-tunnel 花的時間。

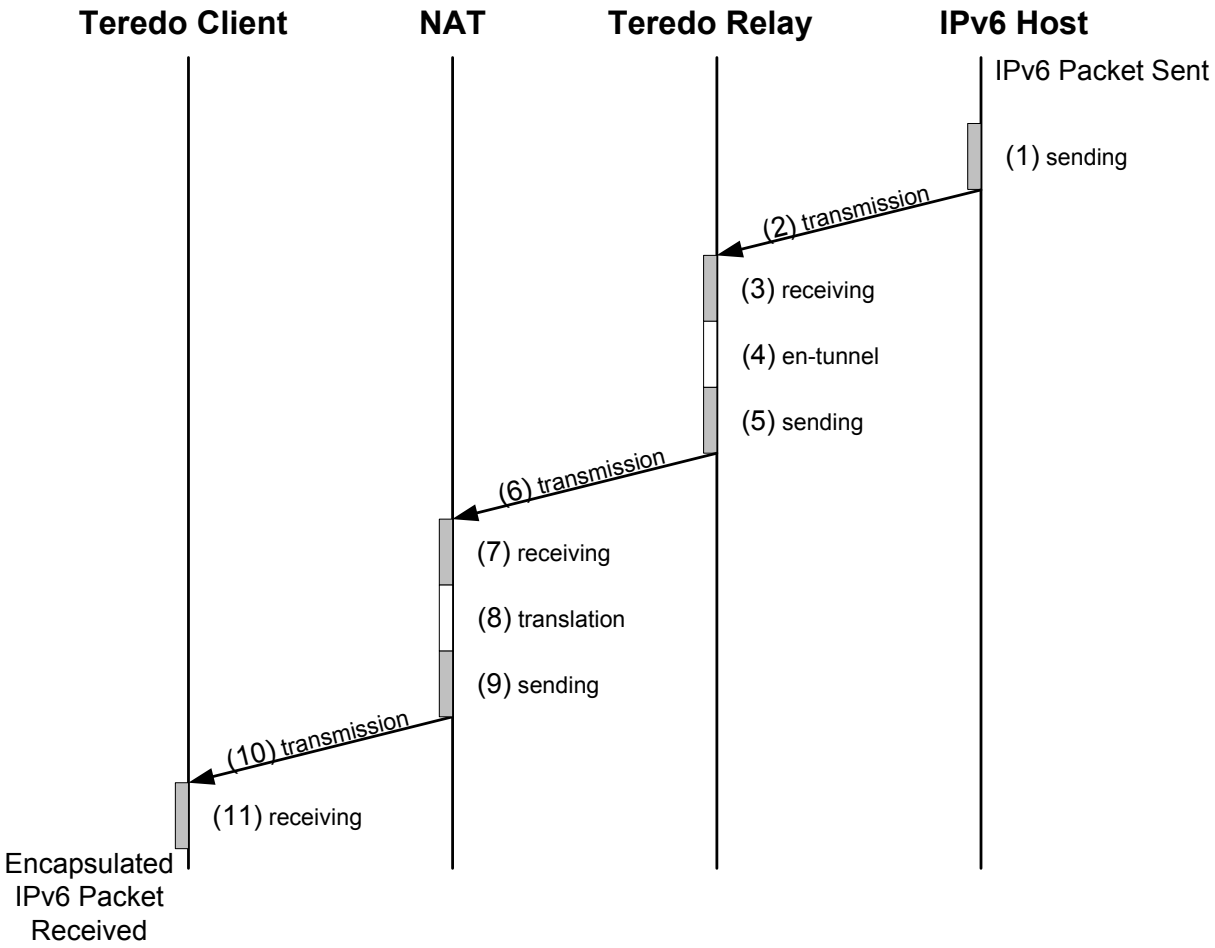

圖 5-3、Packet Transmission Overhead

Scenario 2:

在 scenario 1，量測時 CPU 的負載值爲 0。在 scenario 2 中，我們在作業系統使用 dummy loop 增加 CPU 負載值，藉以量測 Teredo Relay 處理 IPv6 封包 en-tunnel (如圖 5-3 的(4))、de-tunnel 所耗費的時間。此 scenario 同樣移除 Teredo Relay 作業系統中所有非必要執行程序，另外藉執行 dummy loop 產生 Teredo Relay 系統大量負載，並量測在此情況下 Teredo Relay 處理 IPv6 封包的能力。

Dummy loop 是會一個產生大量 CPU 運算的程式，其程式碼如下：

```
int main() { int i; for (;;) { ++i; } return 0; }
```

每在作業系統多執行一個 dummy loop 程式，Teredo Relay 的 CPU 負載值就會加 1 (此 CPU 負載值由 Linux 和 FreeBSD 底下的 *uptime* 指令得到)。

5.3 Results

將 NICI-Teredo Relay，Miredo-Teredo Relay 和 6WIND-Teredo Relay 以 5.2 節的方法量測，得到的結果如下。

Scenario 1：

當 IPv6 封包大小爲 1280 bytes 時，Teredo Relay 將 IPv6 封包 en-tunnel 爲 Simple encapsulation 所需的時間分佈如圖 5-4。此表的橫軸是 Teredo Relay 處理 IPv6 封包耗費的時間(以μs 爲單位)，縱軸是此時間出現次數佔總測量次數的百分比。在 10000 次測量中，NICI-Teredo Relay 耗費的時間分布在 7μs~9μs 之間(平均值爲 8.06μs)，6WIND-Teredo Relay 耗費的時間分布在 11μs~15μs 之間(平均值爲 14.00μs)，Miredo-Teredo Relay 耗費的時間分布在 34μs~40μs 之間(平均值爲 33.08μs)。

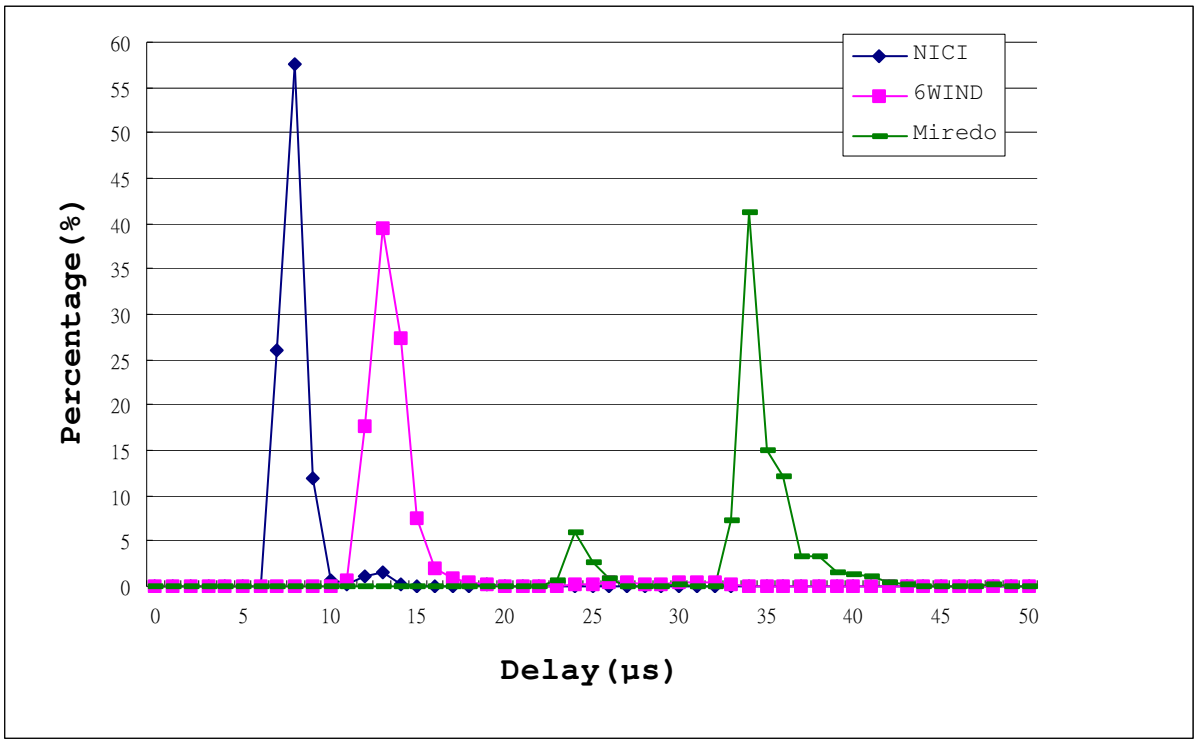

圖 5-4、En-tunnel Delay Histograms of Teredo Relay (1280 bytes)

量測 NICI-Teredo Relay、6WIND-Teredo Relay、Miredo-Teredo Relay 處理 IPv6 封包 de-tunnel 的結果，也有和圖 5-4 類似的時間分佈，它們耗費時間的平均值分別爲 9.24µs、14.80µs 和 34.85µs。表 5-1 列出當測試 IPv6 封包大小爲 64bytes 或 1280bytes 時，各個 Teredo Relay 處理 IPv6 封包 en-tunnel、de-tunnel 的平均時間。針對各個 Teredo Relay 處理不同封包大小的詳細時間分布圖可以在本文最後的 **Appendix A** 找到。

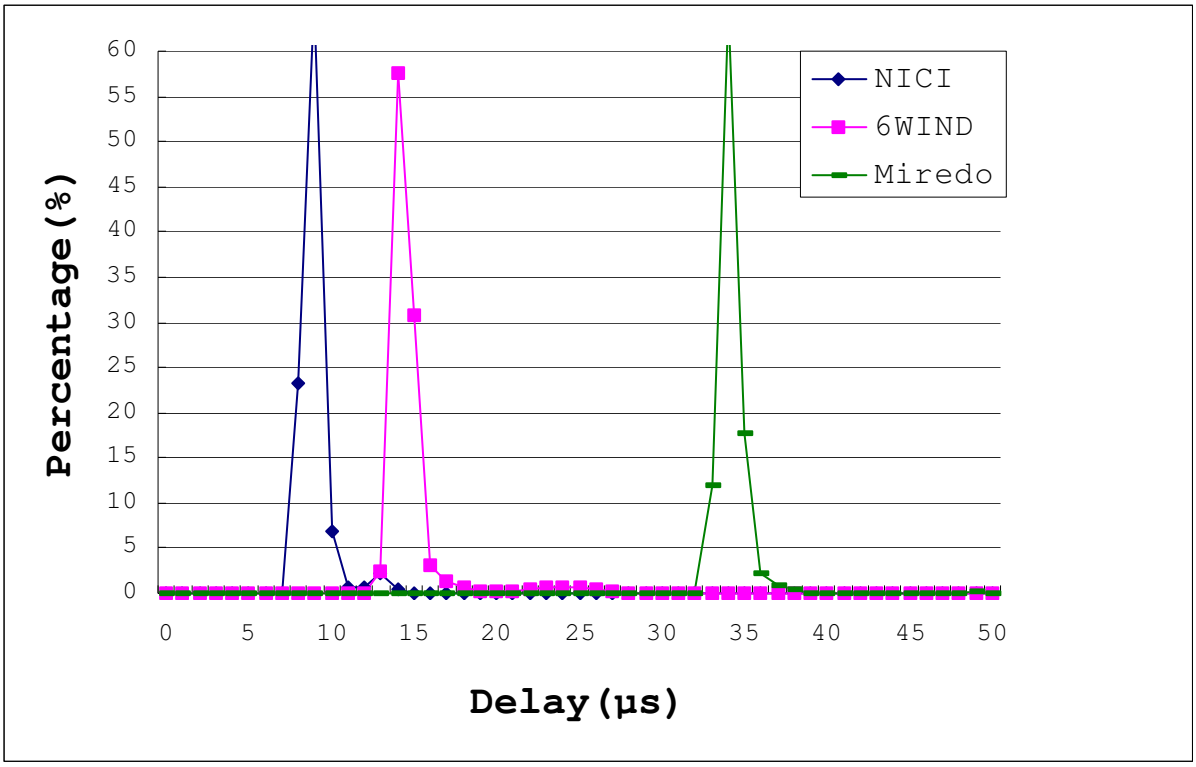

圖 5-5、De-tunnel Delay Histograms of Teredo Relay (1280 bytes)

表 5-1、Average Processing Latency

Teredo Relay	Avg. Latency of En-tunnel (µs)			Avg. Latency of De-tunnel (µs)		
	64 bytes	512 bytes	1280 bytes	64 bytes	512 bytes	1280 bytes
NICI	7.69	7.82	8.06	8.77	9.10	9.24
6WIND	13.30	13.53	14.00	14.34	14.62	14.80
Miredo	23.05	25.46	33.08	24.90	33.14	34.85

表 5-1 顯示不管封包的大小爲何，NICI-Teredo Relay 和 6WIND-Teredo Relay 處理封包耗費的時間幾乎維持一定，兩種 Teredo Relay 處理 IPv6 封包 en-tunnel、de-tunnel 的速度都很穩定，而 Miredo-Teredo Relay 處理封包的時間根據不同封包大小而有所變動。其中 NICI-Teredo Relay 處理 en-tunnel、de-tunnel 的速度是三個 Teredo Relay 裡面最快的。

percentage = (*Other-Teredo_Processing_Time* - *NICI-Teredo_Processing-Time*)

* 100% / *Other-Teredo_Processing_Time*.....(1)

由公式(1)計算，可得到在 64 bytes IPv6 封包 en-tunnel 的處理，NICI-Teredo Relay 比 6WIND-Teredo Relay 快了 42% ((13.30 - 7.69) * 100% / 13.30)。總括來說，NICI-Teredo Relay 在 en-tunnel 方面比 6WIND-Teredo Relay 快了 42%，比 Miredo-Teredo Relay 快了 67-76%，de-tunnel 方面比 6WIND-Teredo Relay 快了 38-39%，比 Miredo-Teredo Relay 足足快了 65-73%。

Scenario 2：

圖 5-6 爲 NICI-Teredo Relay 在不同 CPU 負載值下處理 IPv6 封包 en-tunnel 的平均時間。橫軸爲 CPU 負載值，縱軸爲時間(以μs 爲單位)。由表中可以得知，NICI-Teredo Relay 和 6WIND-Teredo Relay 處理 IPv6 封包所耗費的時間皆不會被 CPU 負載值的大小影響，當系統 CPU 負載值非常大時，它們將封包 en-tunnel 成 IPv6 封包所耗費的時間還是維持定值。不過，Miredo-Teredo Relay 處理 IPv6 封包所耗的時間會依系統 CPU 負載值的增加而變長。

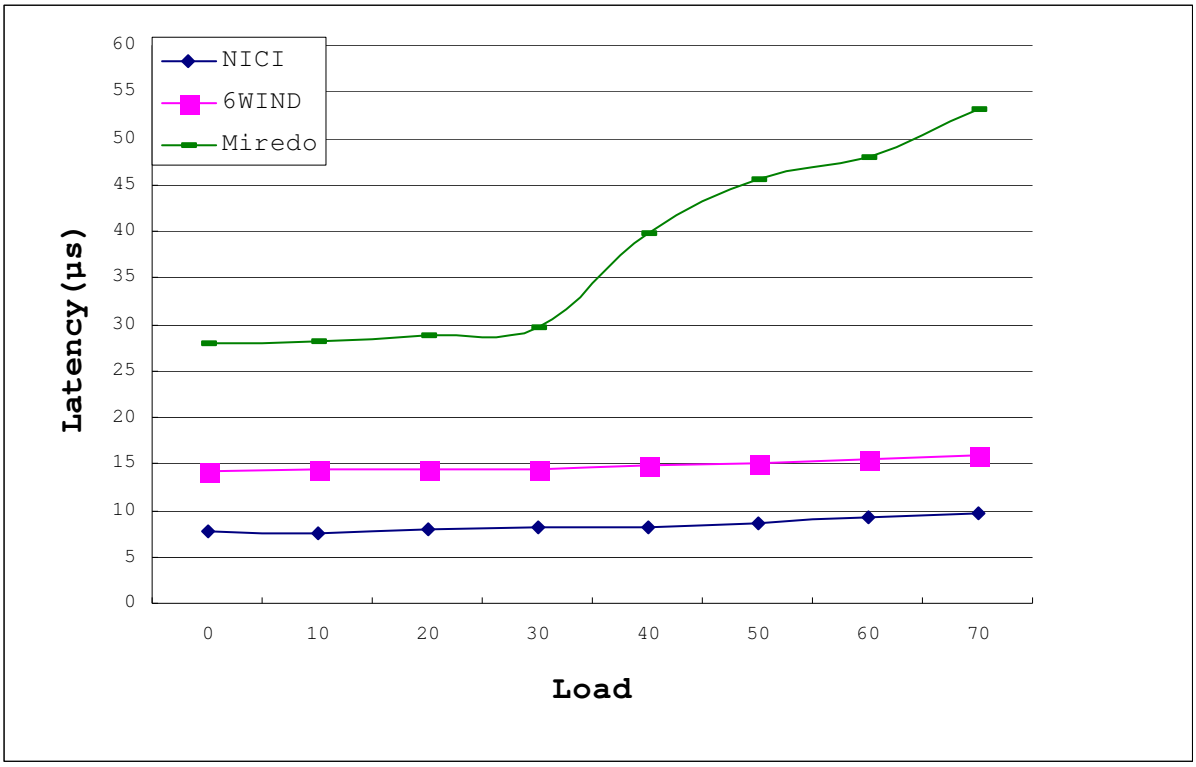

圖 5-6、En-tunnel Delay Distribution of Teredo Relay with Dummy Loops

在對封包 de-tunnel 方面，圖 5-7 顯示不論 CPU 的負載值爲何，NICI-Teredo Relay 和 6WIND-Teredo Relay 處理 IPv6 封包的時間幾乎維持不變，而 Miredo-Teredo Relay 處理 IPv6 封包的時間會隨著 CPU 負載值而變大。由此可以驗證 NICI-Teredo Relay 是一個非常穩定的系統。

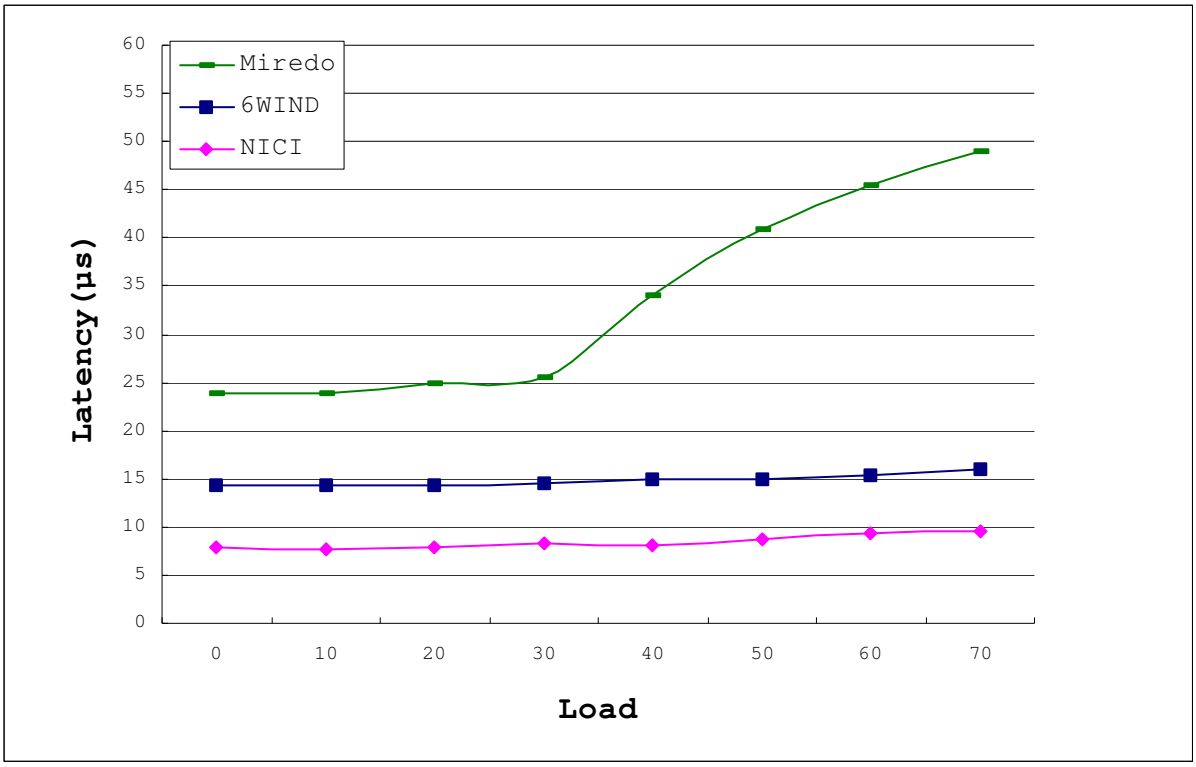

圖 5-7、De-tunnel Delay Distribution of Teredo Relay with Dummy Loops

CHAPTER 6

Conclusion

本文整理出目前各種修正 Tunneling 機制，讓 NAT 後端 IPv6 設備得以連上世界上其他 IPv6 網路。我們將這些方法分類爲 NAT-aware Tunneling 和 NAT-unaware Tunneling 兩大類。Teredo 屬於 NAT-unaware Tunneling 的方法之一。以其分散式的架構，Teredo 具備提升 NAT 後端 IPv6 設備和 IPv6 網路之間 IPv6 封包傳送效率的能力。

我們在 Linux 平台開發了 NICI-Tededo，這是第一個在 Linux 平台的 Teredo 實作。以 Linux 系統普及和架設簡易的特性，跑在 Linux 平台的 NICI-Teredo 讓 Teredo 這個 Tunneling 技術更容易被推廣開來，提供處於 NAT 環境廣大 IPv6 使用者一個簡易連上世界上其他 IPv6 網路的媒介。

最後，我們指出 Teredo 並不能用在所有種類的 NAT。NAT 根據位址對應及存取控制的不同[RFC 3489]，可以分爲 full cone NAT、restricted cone NAT、port-restricted cone NAT 和 symmetric NAT 這四種。雖然 Teredo 可以成功穿越前面三種 NAT，但 Teredo 在面對 symmetric NAT 時，仍是一籌莫展。這是一個值得深入研究的方向。

Appendix A

Average Processing Delay

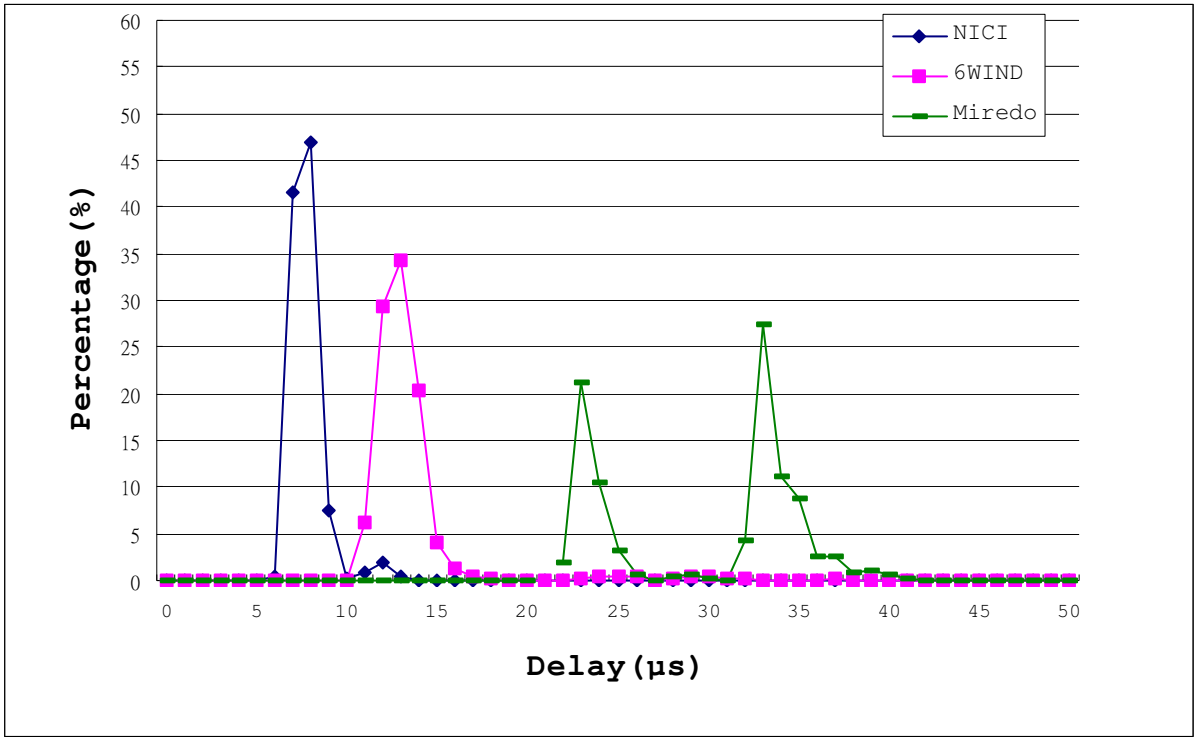

圖 A-1、En-tunnel Delay Histograms of Teredo Relay (512 bytes)

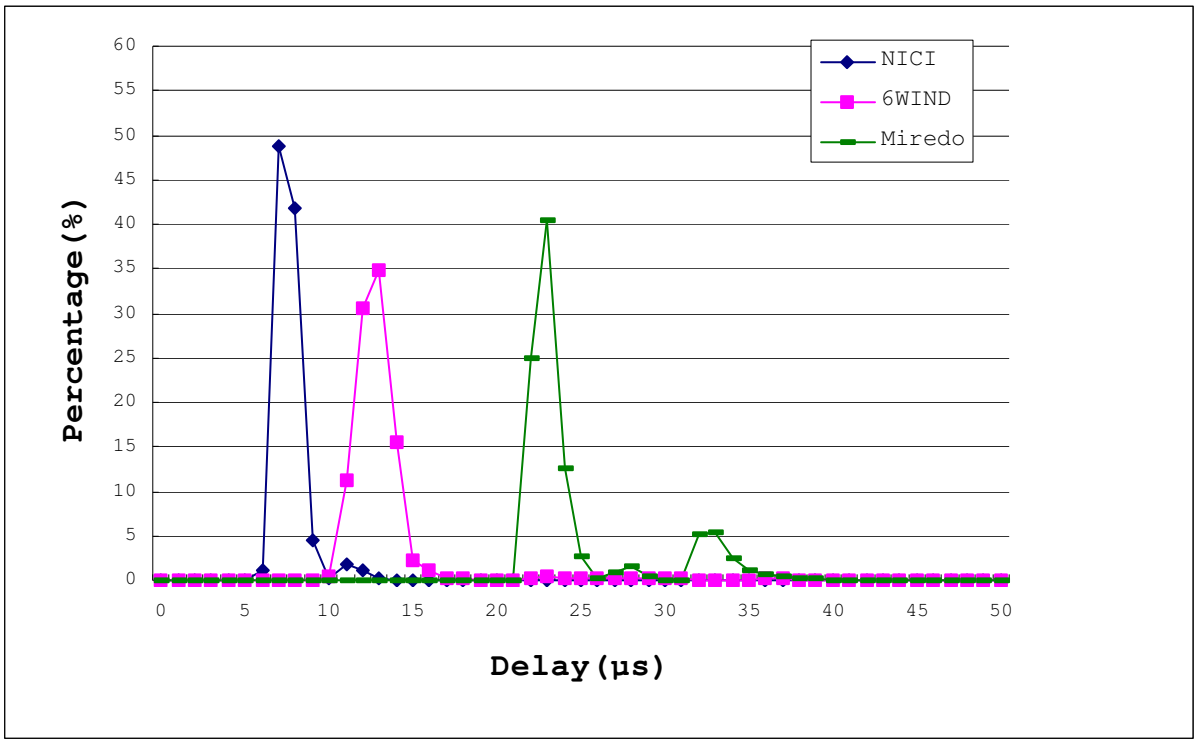

圖 A-2、En-tunnel Delay Histograms of Teredo Relay (64 bytes)

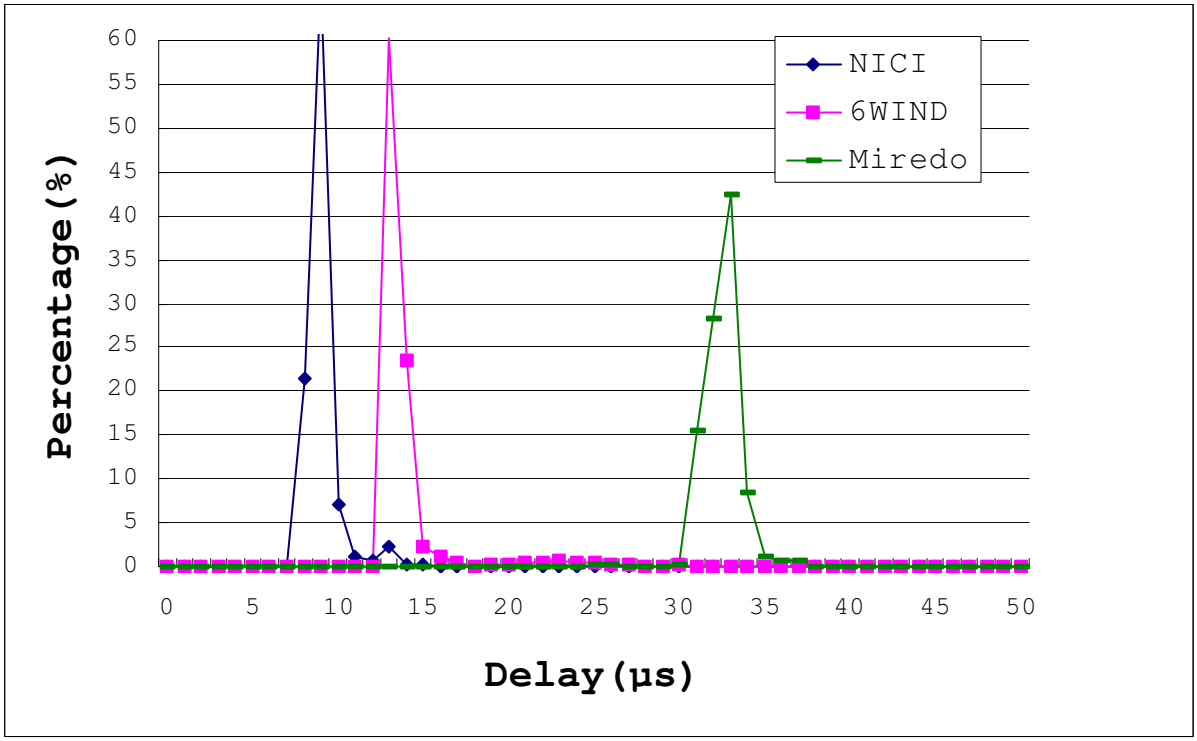

圖 A-3、De-tunnel Delay Histograms of Teredo Relay (512 bytes)

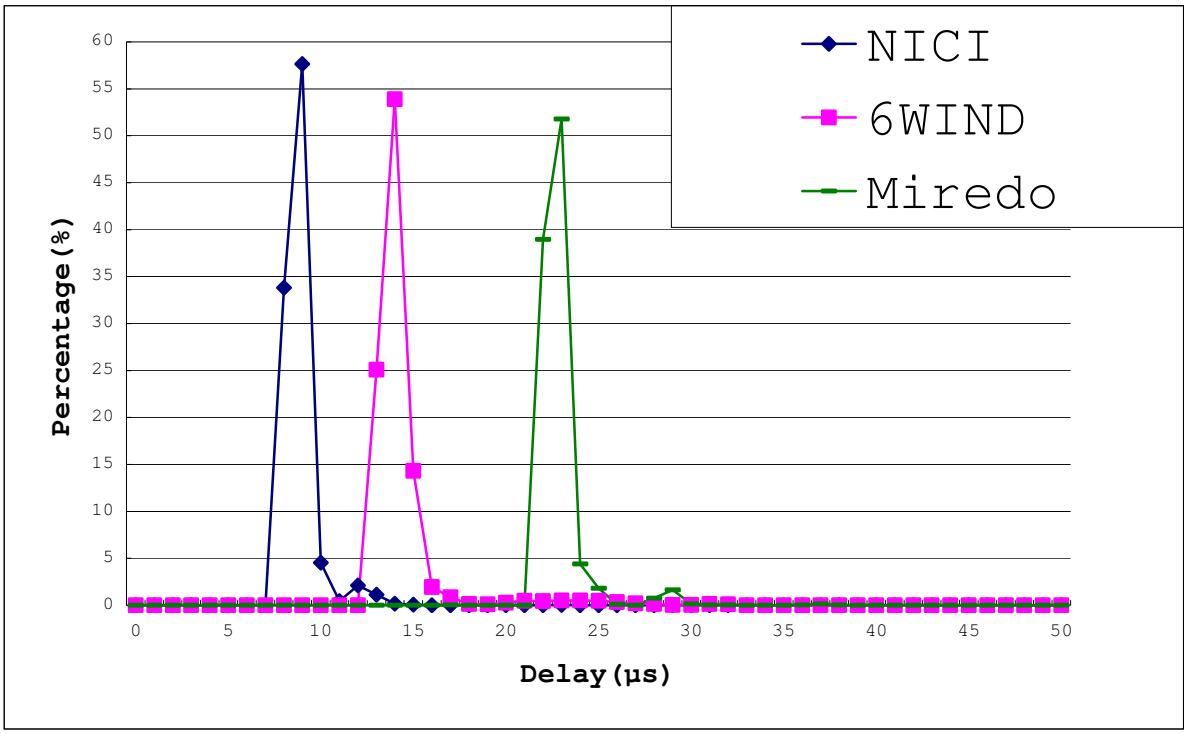

圖 A-4、De-tunnel Delay Histograms of Teredo Relay (64 bytes)

Acknowledgement

This work was sponsored in part by NICI IPv6 R&D Division, NSC Excellence project NSC93-2752-E-0090005-PAE, NSC 93-2213-E-009-100, NTP VoIP Project under grant number NSC 92-2219-E-009-032, and ITRI/NCTU Joint Research Center.

References

[6WIND-Teredo] Teredo for FreeBSD, http://www-rp.lip6.fr/teredo/

[iptables] The netfilter/iptables project, http://www.netfilter.org/

[Lin] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architectures*. John Wiley & Sons, 2001.

[LKM] B. Henderson, Linux Loadable Kernel Module HOWTO, http://www.linux.org/docs/ldp/howto/Module-HOWTO/

[Miredo-Teredo] Miredo: Teredo for Linux, http://www.simphalempin.com/dev/miredo/

[NICI-Teredo] S.-M. Huang and Q. Wu, "Implementation of Teredo - Tunneling IPv6 through NATs", Technical Report for National Information and Communication Initiative (NICI) IPv6 R&D Division, Taiwan, ROC, 2003.

[pcap] Tcpdump and libpcap, http://www.tcpdump.org/

[PF41] M. Simons-Nikolova and X.Bodlaender-Pu, **"**PF 41-packet forwarding for seamless use of IPv6 devices behind IPv4-only NAT gateways**",** *IEEE Transactions on Consumer Electronics*, **51**(1):76-79, February 2005.

[RFC 2473] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification", IETF RFC 2473, December 1998.

[RFC 2544] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", IETF RFC 2544, March 1999.

[RFC 2764] B. Gleeson, A. Lin, J. Heinanen, G. Armitage and A. Malis, "A Framework for IP Based Virtual Private Networks", IETF RFC 2746, February 2000.

[RFC 2893] R. Gilligan and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", IETF RFC 2893, August 2000.

[RFC 3022] P. Srisuresh and K. Egevang, “Traditional IP Network Address Translator (Traditional NAT)”, IETF RFC 3022, January 2001.

[RFC 3053] A. Durand, P. Fasano, I. Guardini and D. Lento, “IPv6 Tunnel Broker”, IETF RFC 3053, January 2001.

[RFC 3056] B. Carpenter and K. Moore, “Connection of IPv6 Domains via IPv4 Clouds”, IETF RFC 3056, February 2001.

[RFC 3489] J. Rosenberg, J. Weinberger, C. Huitema and R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, IETF RFC 3489, March 2003.

[RFC 3519] H. Levkowetz and S. Vaarala, “Mobile IP Traversal of Network Address Translation (NAT) Devices”, IETF RFC 3519, April 2003.

[Silkroad] M. Liu, X. Wu, Y. Cai, M. Jin and D. Li, “Tunneling IPv6 with private IPv4 addresses through NAT devices”, Internet Draft, draft-liumin-v6ops-silkroad-02.txt (expired), November 2004.

[Teredo] C. Huitema, “Teredo: Tunneling IPv6 over UDP through NATs”, Internet draft, draft-huitema-v6ops-teredo-05.txt (work in progress), April 2005.