國立交通大學

資訊科學系

碩 士 論 文

網 路 匿 名 連 線 機 制 研 究

Anonymous connection on the Internet

研 究 生：許志行

指導教授：楊 武 教授

中 華 民 國 九 十 四 年 六 月

網 路 匿 名 連 線 機 制 研 究

Anonymous connection on the Internet

研 究 生：許志行　　Student：Chih-Hsing Hsu

指導教授：楊　武　　Advisor：Wuu Yang

國 立 交 通 大 學

資 訊 科 學系

碩 士 論 文

A Thesis

Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

網路連線匿名機制

學生：許志行　　指導教授：楊　武 博士

國立交通大學資訊科學研究所

摘　　要

當電腦連上網路之後，與其他電腦互相溝通時，在彼此傳遞的封包中，都包含著會顯示出封包來源與目的位址的欄位。對內容加密僅可保障通訊內容不易被網路上的第三者得知，但是仍然無法隱藏通訊雙方的身分。若有第三者監聽到這些封包，就能得知是哪兩部電腦在傳遞資訊。

本論文研究的目標，主要是保護在網路上傳輸的封包中，通信雙方的實際身份，也就是 IP 位址，使其無法讓外部的觀察者獲知這些資訊，並且通訊目標也無法知道使用者的真實身份。因此首先要做到發送端所送出封包，不能直接顯露出傳送的目的 IP 位址。再藉由透過 proxy 的轉送方式進行連線，中間加入數部具有封包轉送功能的電腦，而整個封包傳遞過程再配合加密程序，來達到混淆監聽者，提高封包分析難度的目的。

Anonymous connection on the Internet

Student：Chih-Hsing Hsu **Advisor：Dr. Wuu Yang**

Department of Computer and Information Science

National Chiao Tung University

Abstract

As the computer is connected on internet to communicate with another machine, there are fields which indicate the locations of source and destination computers in the packets exchanged between the two computers. Encryption of the content in the packets can only ensure the information within not easily accessed by the third party, the encrypted data, however, is unable to conceal the identities of the two communicating parties. If a third party watches these packets on purpose, the two computers mutually interchanging information will be located.

The research goal of this thesis is meant to protect the two parties communicating on internet via transferring packets against revealing their real identities, namely IP address, hence such kind of information would not be gathered by the foreign intruders. To achieve this goal, the first step is to make the packets sent by source computer incapable of showing the IP address of the location of destination computer directly. Then the internet connection is built via transmitting the data through the proxy server, while several packet-forwarding-functioned computers are added to the route of connection to relay packets. And the whole packet-transferring process is combined with encryption procedure to yield confounding the intruder and heightening the complicities of packets thus making them difficult to be analyzed.

目錄

表目錄

圖目錄

第一章　序論

1.1　研究動機

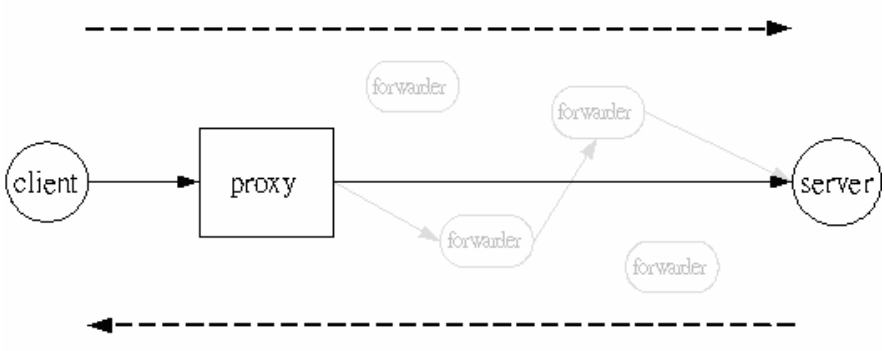
在一個 packet switched network 中，每個在上面傳遞的封包，都是由 IP header，後面接著傳送的資料所構成。因此，當電腦連上網路之後，與其他電腦互相溝通、傳遞訊息時，雙方互相交換的封包，在 header 中有欄位會顯示著封包來源與目的位址。對內容加密固然可以保障通訊內容不易被第三者知道，但是無法隱藏通訊雙方的身分。因此，在封包傳遞的過程中，若是其他人藉由封包擷取竊聽，就能得知通信雙方的位置與身分。

若是想保障隱私，做到匿名連線，則每個傳遞出去的封包，header 部分必須先經過處理再送出，不然在封包傳遞的過程中，若是有第三者監聽到這些封包，就能藉由封包內部資料得知是哪兩部電腦在傳遞資訊。並且，若是能夠做到再經由其他電腦，加以轉送傳遞這些封包，則更能達到混淆監聽者的目的。也因為透過封包標頭就能得知來源與目的的位置，因此針對封包內容加密，並無法避免攻擊者透過封包擷取竊聽，查出封包來源與目的位置。

1.2　研究目標

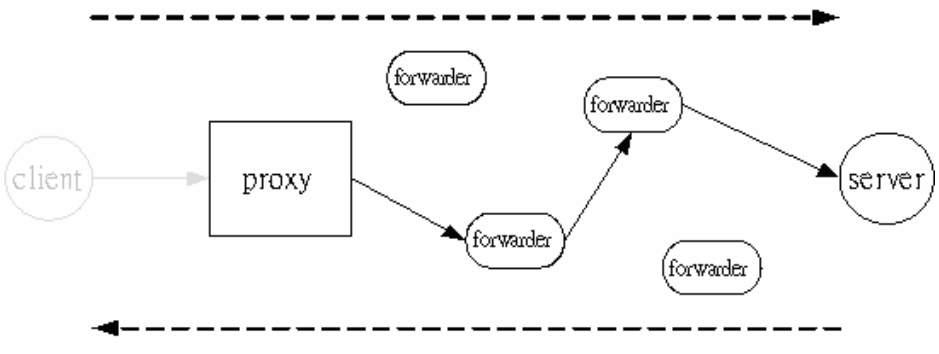
匿名通信，主要的目的是保護網路傳輸時通信實體的身份，也就是 IP 位址，使其無法讓外在的觀察者獲知這些資訊。因此針對在 Network Layer 中的封包，由於其 IP header 之中，會包含了來源 IP address 與目的 IP address，這些資訊使得若是封包遭第三者監聽，就能得知通訊雙方的位置，使得 client 而無法匿名連線通訊。因此我們第一個目標便是在 client 端，處理每個傳送送出去的封包。也就是當 client 送出封包時，在 IP header 的部分，不能直接顯露出目的 IP address，而實際目的 IP address，則隱藏在封包內部。這部分我們可以藉由先將封包傳遞

到 proxy，再由該 proxy 將此封包傳送給 server。另外就是，我們希望能夠做到僅修改 IP Layer 的方式，達到現有應用程式都不必修改的目標。



-圖 1.1 透過 proxy 連線-

而若只是 client 與 server 之間，搭配一部 proxy 來轉送封包，則第三者仍然很有機會，藉由擷取 proxy 收到與發送出的封包方式，來推測出實際在通訊的雙方 IP address。所以還要再配合多部電腦，負責轉遞封包，達到混淆監聽者的目的。因此我們另外再加上數部電腦，具有封包 forwarding 的功能，這些 forwarder 只有 proxy 知道他們的位置，當 proxy 收到封包時，他可以決定該封包接下來要經過哪些 forwarder，再將這些資訊包裝並加密在封包之中。Forwarder 收到的每一封包，經解密，得知封包接下來往哪傳遞的資訊後，便往下一部 forwarder 送出，直到原始的目的 server。



-圖 1.2 經過多部 forwarder 轉送-

因此整體而言共有以下幾個主要目標：

- 發送方匿名 sender anonymity，因此目標 server 不會知道真正的 client IP 位址。
- 提高封包擷取後分析的困難度。
- 混淆封包實際來源與目的位址。
- 不必修改上層的應用程式，僅修改 IP 層即可。

1.3 論文架構

本論文共分五章，第一章為序論，針對整篇論文做一個初步的基本介紹。在第二章的部分，會討論關於目前在 IP 層的匿名技術以及 TCP/IP 中對於封包與 IP 層中相關的技術。第三章中，說明關於系統架構、設計考量與原理，以及系統實作與執行流程。第四章則為實驗成果的說明。最後，在第五章做為本篇論文的結論，以及提出未來可能改進之處與發展的方向。

第二章　相關研究

包含網路隱私安全相關議題的研究、TCP/IP 資料儲存結構 mbuf、MTU 與 TCP checksum 的問題，與 Raw_sockets 方面的應用技術等。

2.1　Internet privacy 與 Anonymity

在 2.1 的部分，首先說明現階段網路匿名連線方面的研究，以及網路隱私安全相關的問題。

2.1.1　Anonymity

由於 anonymity 並無一明確的定義，因此在本論文中，所謂的匿名連線，大體上來的意義，即例如當甲電腦透過匿名的方式連線至乙電腦時，必須讓乙不知道此連線者的真正身分(位置)。並且，『甲連線到乙』這個動作，網路上不能有其他人能得知。

簡單的匿名構想，最早可以由透過一代理者的概念來達成，但是這種做法，代理者會知道甲與乙的身分，並且透過擷取代理者傳入與送出的封包，仍然有極高的機會分析出通訊者甲跟乙的實際位址身分。因此若是將封包透過多個網路上的節點來轉送，可以提高分析的難度。轉送的方式可以透過類似 IP in IP 的做法，並且搭配加密系統，在決定好封包未來移動路線與將會經過的節點之後，依序一層層使用不同的 key 加密，最後再送出。之後封包移動過程中，每經過一個節點，就對封包解密一次，查出封包下一個移動目的後送出，反覆這些動作直到封包到達最後目的地。

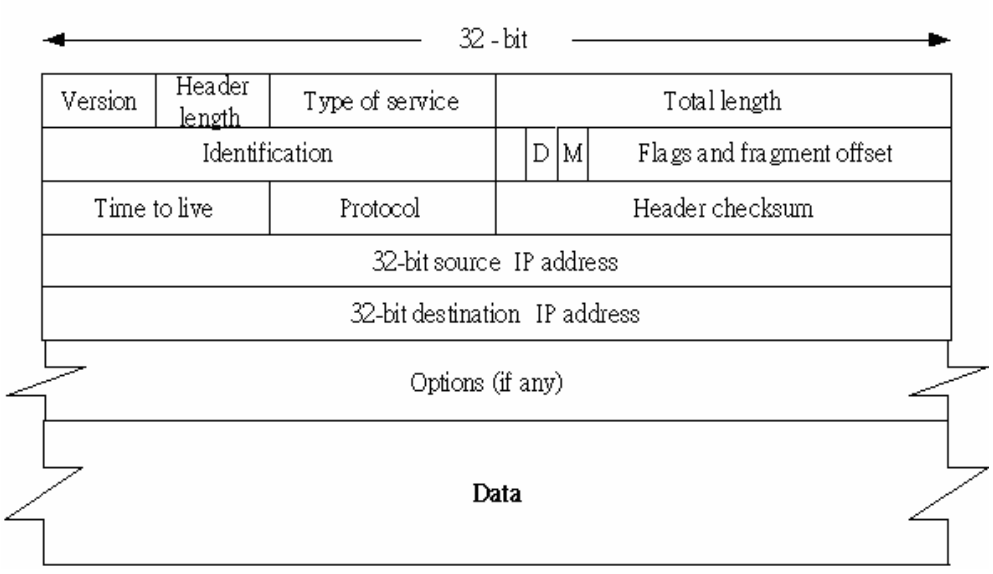
目前已有的匿名通訊包括 ANON.[3]、Crowds[6]、Mixes[4]、Onion Routing[8]、Tarzen[5]以及等，採用 rerouting 或是 padding 機制，來達到匿名保護。所謂的 rerouting，如同上述一般，即提供間接通信傳送，在一次通信傳輸中，配合多部 host 轉送訊息，由發送至接收構成一條由多個安全通道組成的

虛擬路徑，此為 rerouting path。在這個路徑上傳送的 IP 封包，外部的攻擊者無法獲得真實的發送端或是接收端的 IP address。使用這種方式，通信者的身份資訊(IP address)就可以有效地隱藏。

2.1.2 Internet Protocol 與 IP packet 的安全問題

所謂的 Internet，我們可以視為由許多 LAN 所互相連接而組合成的，而將這些 Internet 連結在一起的網路層協定，則稱為 IP (Internet Protocol)。其通訊的運作方式為：上層(transport layer)讀取 data stream，再將其分割成 segment，每段 segment 的大小訂為 1500bytes，再送往下層。當資料透過 Internet 傳送的過程之中，有機會被切割成更小塊。這些片段區塊傳送到目的電腦後，會在 Internet layer 中將資料重組還原，再送往上層處理。而 IP 的主要功能就是把封包送至目的位置。存在 IP 網路上的 host 與 router 都有其 IP address。Host 與 router 不同點在於有無 forward 封包的功能。

IP packet 是由一個 IP header 加上 data 所組成，*圖 2.1.1* 為一 IP 封包格式。



-圖 2.1.1 IP 封包格式-

當封包在網路上移動時，若遭到擷取、監聽，由於在 IP header 之中，有著 source IP address 與 destination IP address，因此監聽著就能藉由這些資訊，得知

傳送者與接收者的位置。

2.2 Mbufs 概述

在 TCP/IP 實作中，採用了 BSD(Berkeley Software Distribution)版的網路 buffer 管理方法。在 BSD 之中，是把網路資料，儲存在一個稱為『mbuf』的資料結構中，意思即為使用 mbuf 結構來做記憶體的分配。

每一個 mbuf 大小只有 128 bytes，因此如果想要放置的資料，大於 128 bytes 的話，則需要將資料切割後，分別存放在數個 mbuf 的記憶體結構，再將這些 mbuf 結構連結起來，如此才能夠容納下該筆大資料。

2.2.1 Mbufs 架構

- 型態 1.

  如*圖 2.2.1* 中，若指標 m_flags 等於 0 時，表示該 mbuf 僅含 data。該 mbuf 能存放的資料大小為 108- byte，且 m_data 指標是指向該 108-byte buffer。而 m_len 則顯示了該筆 data 的大小。

- 型態 2.

  如*圖 2.2.1*，若 m_flags = *M_PKTHDR*，代表此為一 packet header，意即包含一個 mbuf packet header，m_pkthdr.len 的值則代表整個 mbuf chain 的 total length，即這個 packet 的大小。m_pkfchdr.rcvif 在 packet 輸出時沒有使用，只有在接收 packet 時，用來指接收的 interface 之 ifnet 結構。

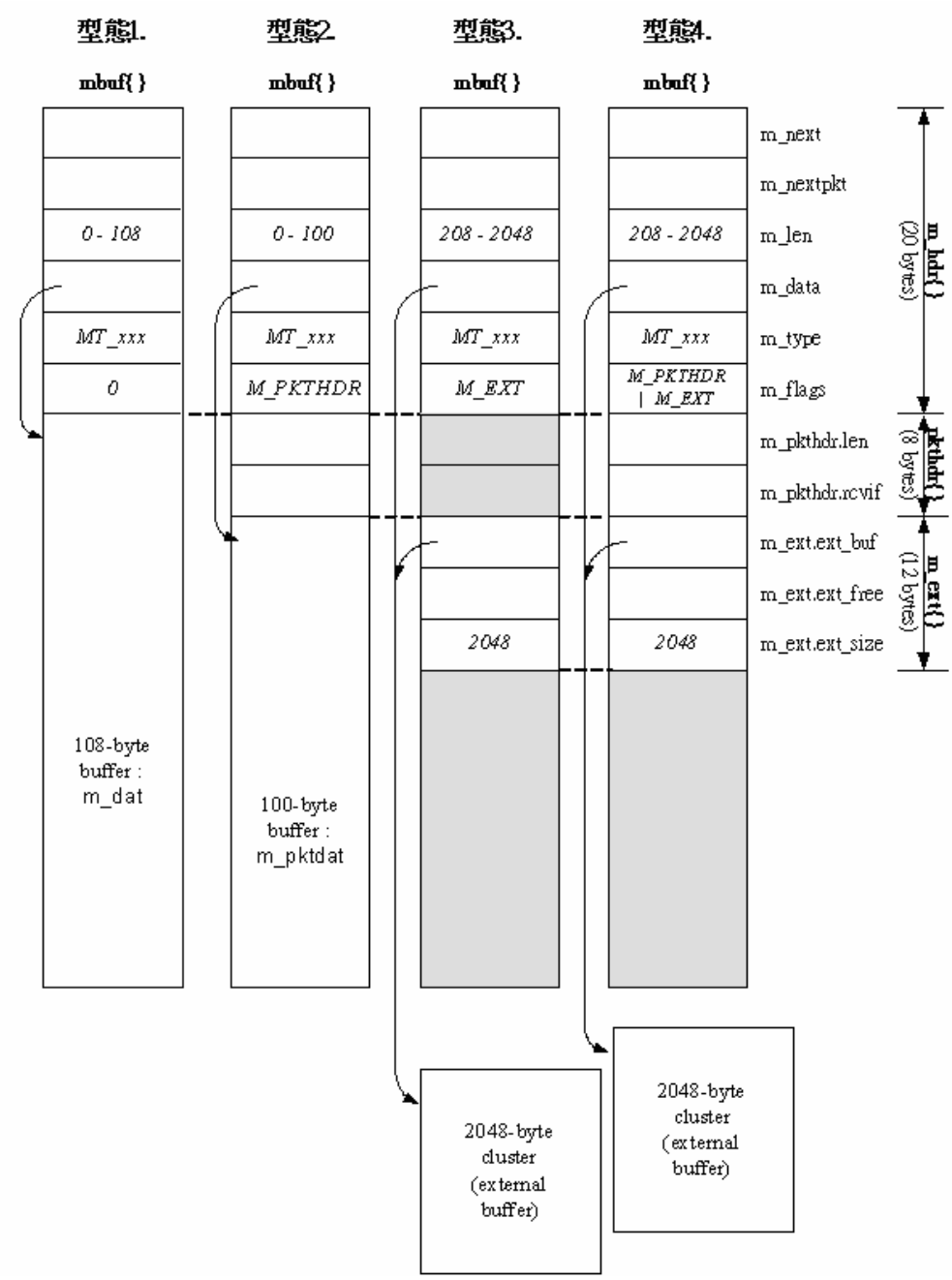
- 型態 3.

  如*圖 2.2.1* 中，*M_PKTHDR* 若無使用，代表該 mbuf 不包含 packet header，但是其包含的 data 大小超過 208 bytes，表示 external buffer 被使用

(M_EXT is set)。原先用來存放 packet header value 大小的欄位仍然會配置，但是不使用。m_data 則是指向 external buffer。

- 型態 4.

如*圖 2.2.1*，第四種 mbuf 格式與第三種相同，但是多包含了 packet header，故其會使用 packet header value 大小的欄位。



-*圖 2.2.1* 四種 mbuf 型態-

mbufs 其餘指標說明：

<1> m_data 指標:可以指向該 buffer 中的任何一個記憶體位置(包含自己本身的 buffer 與 external buffer)。

<2> m_ext.ext_buf 指標：用來指向 external buffer 開始位置，m_ext.ext_size 則是指向 external buffer 存放資料的大小。

<3> m_next 指標：是用來連接 mbuf chain 的 (一個 mbuf chain 代表一個 packet)。

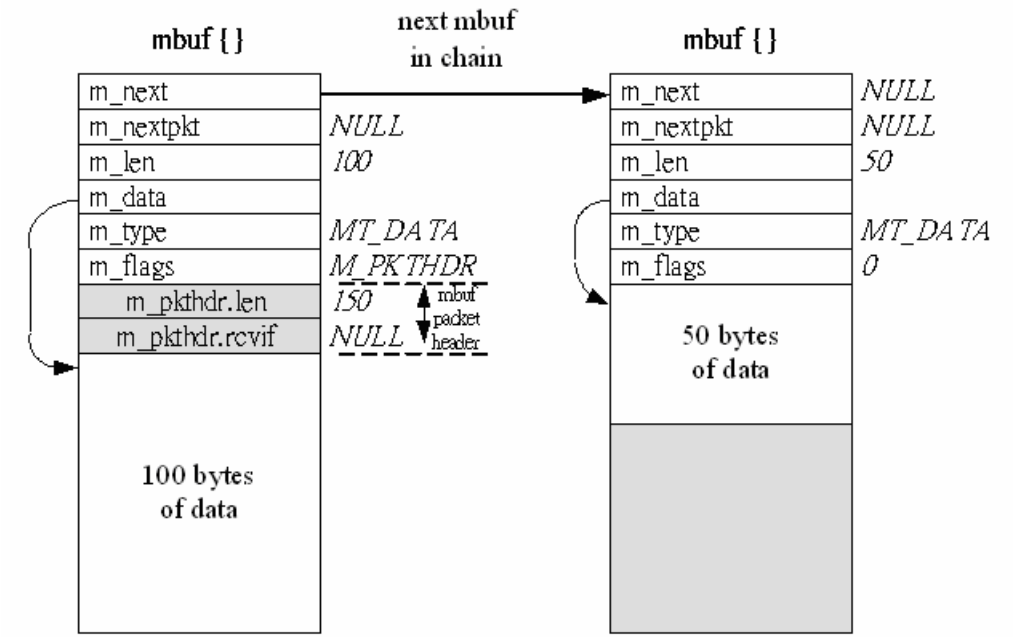
<4> m_nextpkt 指標： link 整個 mbuf chain。

2.2.2 Mbufs 運作方式

分為 mbufs 如何存放 data 以及 mbufs 加入 IP 與 UDP header 兩個例子來說明 mbuf 的運作方式。

1. mbufs 如何存放 data

mbufs 存放上層傳來的資料方式，以*圖 2.2.2* 說明如何存放一筆 150 bytes 大小的資料：



-*圖 2.2.2* mbuf 存放 data 格式-

<1> 根據 2.2.1 中所提到，一個 mbuf 能存放 108 bytes 大小的資料，因此若要放置 150 bytes 的資料，則必須使用兩的 mbufs 來存放。

<2> 由於第一個 mbuf 之中，需要存放 mbuf packet header 的資訊(m_pkthdr.len 與 m_pkthdr.rcvif) 共 8 bytes，因此在第一個 mbuf 中，真正能存放 data 的空間為 100 bytes，剩下的 50 bytes 則配置第二個 mbuf 來儲存。

<3> (a.) 第一個 mbuf ：

- m_next 指向第二個 mbuf 起始位置。
- m_len 值為 100 (100 bytes 的資料)。
- m_flags = M_PKTHDR

  (因為共兩個 mbufs，設定為 M_PKTHDR 則表示為起始 mbuf，意即包含 m_pkthdr.len 與 m_pkthdr.rcvif 兩個欄位來存放完整 mbuf packet header 的大小)。
- m_pkthdr.len = 150 (兩個 mbufs 共存放 150 bytes 資料)。

(b.) 第二個 mbuf ：

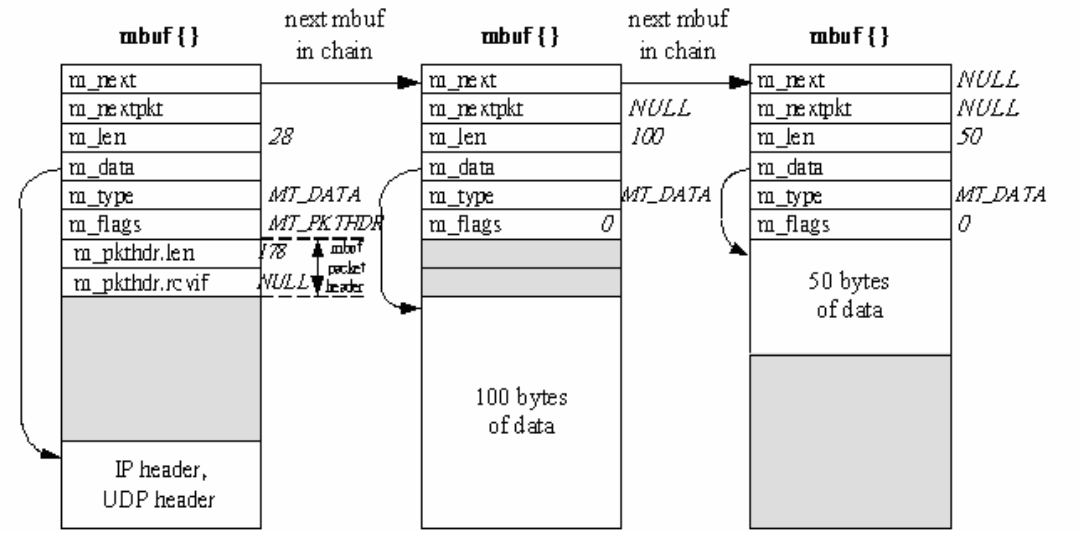
- m_next = NULL (最後一個 mbuf)。
- m_len = 50 (50 bytes 的資料)。
- m_flags = 0

2. mbufs 加入 IP 與 UDP header

mbufs 包裝成一個完整的 IP packet 的方式，以*圖 2.2.3* 來說明 (延續 mbufs 如何存放 data 之中，兩個已存放 150 bytes datas 的 mbufs)：

<1> 首先，新配置一個 mbuf，目的是用來存放 IP 與 UDP header(在此以 UDP 說明)。其中，令 m_next 指向 原先存放 150 bytes 的 mbuf chain，構成一包含三個 mbufs 的 mbuf chain。

<2> 由於新的 mbuf 成為該 mbuf chain 中的第一個 mbuf，因此將這個新的 mbuf 中 m_flags 設為 M_PKTHDR，而將原本第一個 mbuf 中 m_flags 設為 0。



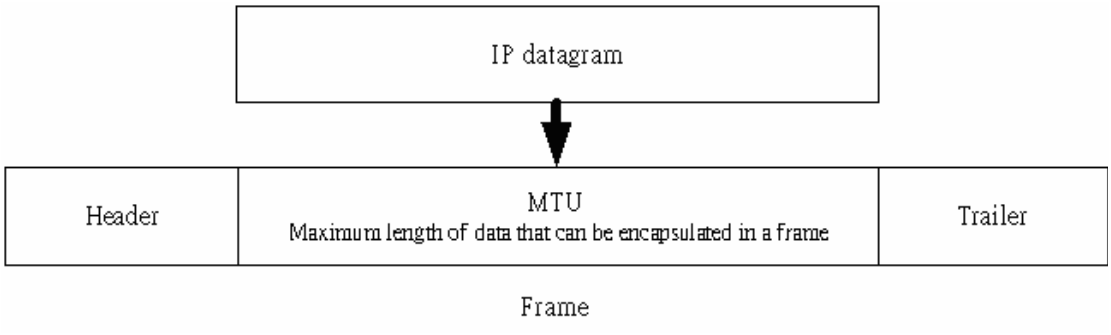
-*圖 2.2.3* mbuf *加入* UDP header-

<3> 最後，第一個 mbuf 中 m_pkthdr.len 設為 178，這是因為加入的 IP header 大小為 20 bytes，而 UDP header 大小為 8 bytes，加上原先的 datas 大小為 150 bytes，因此共為 178 bytes。

2.3 MTU 與 TCP checksum

2.3.1 MTU(maximum segment size)的作用與影響

當資料由 IP layer 傳至 DataLink layer 後，會被封裝成為 frame，由於 frame 的大小有所限制，因此傳入的 IP datagram 藉由 MTU 來限制大小，令其不得大於 MTU。



-*圖 2.3.1* IP datagram 受到 MTU 限制大小-

當 IP datagram 大於 MTU 時，必須進行 fragmentation。每一分割後的 fragment 都會有自己的 header。

2.3.2 TCP checksum 機制

TCP(Transmission Control Protocol) checksum 是用來確保在 TCP 連線機制下，所送出的封包在網路轉送時，驗證 TCP header 與 data 的部分是否有損壞。

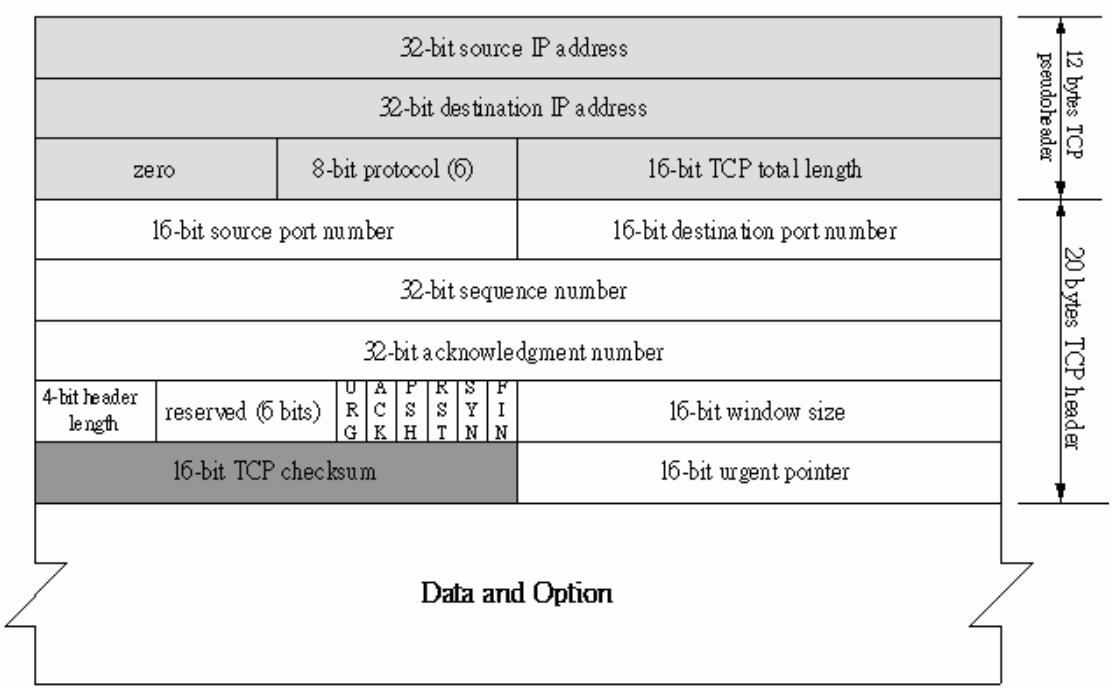
TCP checksum 可以保證正確的資料有：TCP pseudo-header、TCP header 與 data。其計算步驟為：

<1> 若封包 protocol 為 TCP (或 UDP，但是 UDP checksum 可填 0 表不算 checksum)，製作 TCP pseudo-header。

<2> checksum 欄位填 0。

<3> 以 0 來 padding 補足資料長度成 16 位元。

<4> 以 16 位元為單位，將 TCP pseudo-header、TCP header 與 data 的部分以 one's complement 相加。



-*圖 2.3.2* TCP header 中 checksum 欄位-

驗證的步驟則為：

<1> 若封包 protocol 為 TCP，製作 TCP pseudo-header。

<2> 以 0 來 padding 補足資料長度成 16 位元。

<3> 以 16 位元為單位，將 TCP pseudo-header、TCP header 與 data 的部分以 one's complement 相加。

<4> 計算結果為 0，則表示資料未受損。

2.4 Raw IP

2.4.1 Raw Sockets 概述

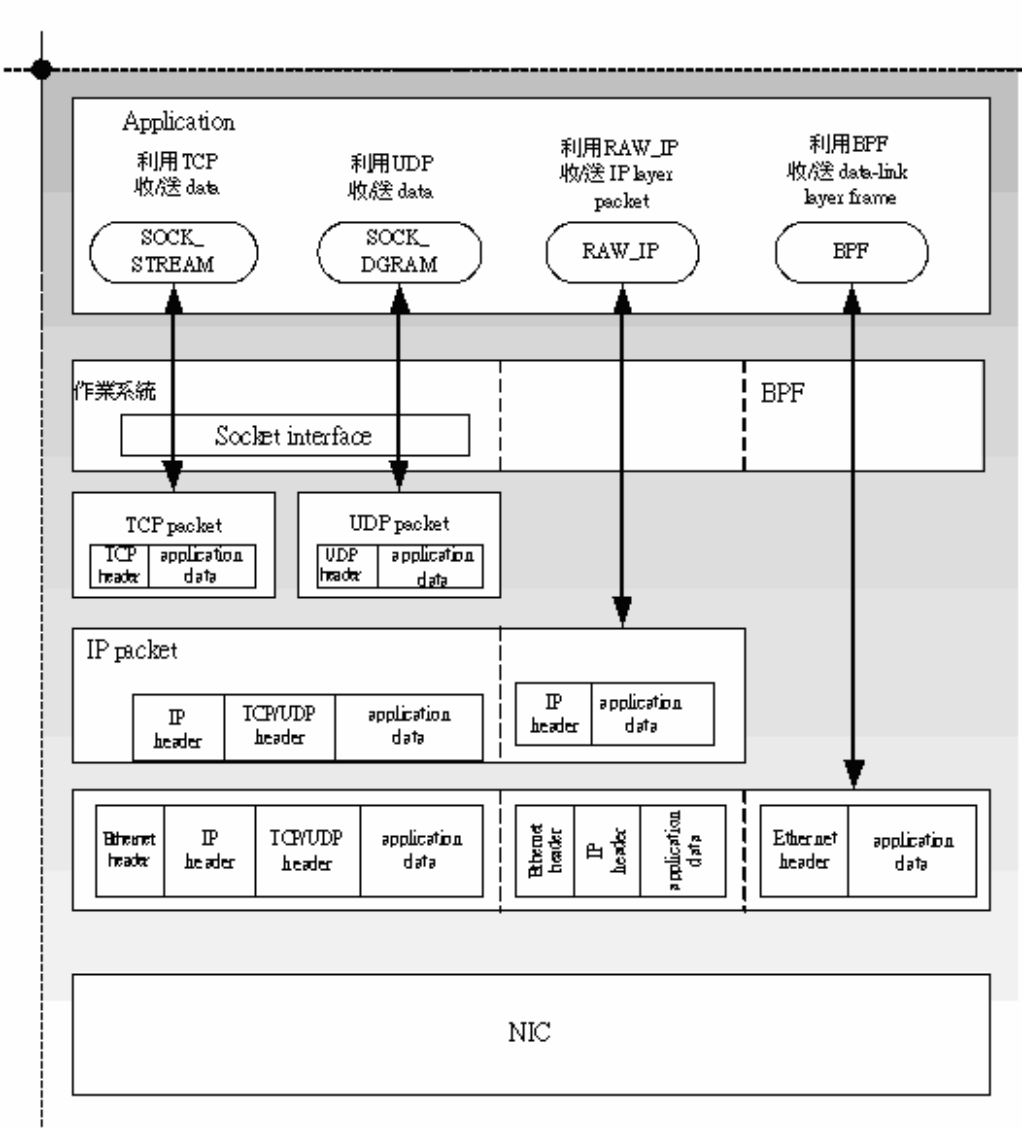
利用 Raw socket，我們可以讀寫 kernel 不處理之 IP protocol 欄位的 IP packet，並且可使用 IP_HDRINCL SOCKET 選項，建構自己的 IP header。

Raw Socket 提供三個正常 TCP 與 UDP socket 所沒有個功能：

<1> Raw Socket 提供我們讀寫 ICMPv4、IGMPv4，以及 ICMPv6 packet。

<2> 利用 Raw Socket，process 可以讀寫 kernel 不處理的 IPv4 protocol 欄位之 IP packet，由於大部分 kernel 只處理 protocol 欄位值為 1(ICMP)、2(IGMP)、6 (TCP)，和 17(UDP)，因此我們可以藉由 Raw Socket 來處理非上述 4 項 protocol 的 IP packet。

<3> 藉由 Raw Socket ，process 可以使用 IP_HDRINCL socket 這個選項，達成建立一個新的 IPv4 header 之目的。



-圖 2.4.1 建立封包的方式-

2.4.2 Raw Sockets 運作方式

- **Raw Socket Output**

Raw socket 的輸出受以下規則所控制：

<1> 正常的輸出是呼叫 sendto，並設定目的。

```
sendto(int sockfd, const void *buf, size_t nbytes, int flags,
                    struct sockaddr *dest, socklen_t    destlen);
```

<2> 如果沒有設定 IP_HDRINCL 選項，則要傳給 kernel 寫入的資料起始位址，來指定跟在 IP header 後的第一個位元組，這是因為 kernel 會自行建構 IP Header。

<3> 如果有設定 IP_HDRINCL 選項，則要傳給 kernel 寫入的資料起始位址，來指定 IP header 的第一個位元組。Process 要建構整個 IP header，除了：

(a) Identification 欄位，可以設為 0，讓 kernel 來設定這個欄位值。

(b)checksum 欄位，kernel 會計算該欄位值。

4. 如果封包的大小超過離開介面的 MTU，kernel 會再加以切割。

- **Raw Socket Input**

<1> Raw socket 的輸入，面臨到的第一個問題，即是：kernel 會將哪些收到的 IP packet 傳給 raw socket 呢？

(a) kernel 所收到的 UDP packet 及 TCP packet 不會傳給 Raw socket

(b)大部分的 ICMP packet 在 kernel 處理完 ICMP 訊息之後，會傳給 Raw socket

(c)所有的 IGMP packet 在 kernel 處理完 IGMP 訊息之後，會傳給 Raw socket

(d)所有包含 kernel 不了解之協定欄位的 IP packet 都會傳給 Raw socket

(e) Kernel 會一直等到所有的 IP fragment 抵達，再傳給 Raw socket

<2> 對每個 Raw socket 執行的檢查，共有 3 項，全部通過才會傳給該 socket：

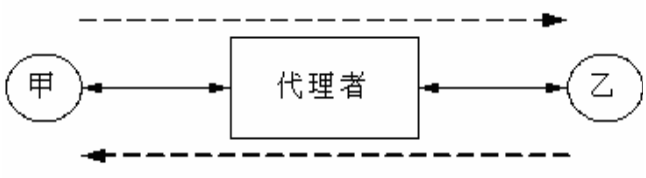
(a) 檢查 socket function 第三個參數---非 0。

(b) 如果有用 bind 繫結了一個本機的 IP address，則接收的 packet 目的 IP address 必須與其相同。

(c) 如果有用 connect 指定了一個遠端的 IP address，則接收的來源 IP address 同樣要符合該連線位址。

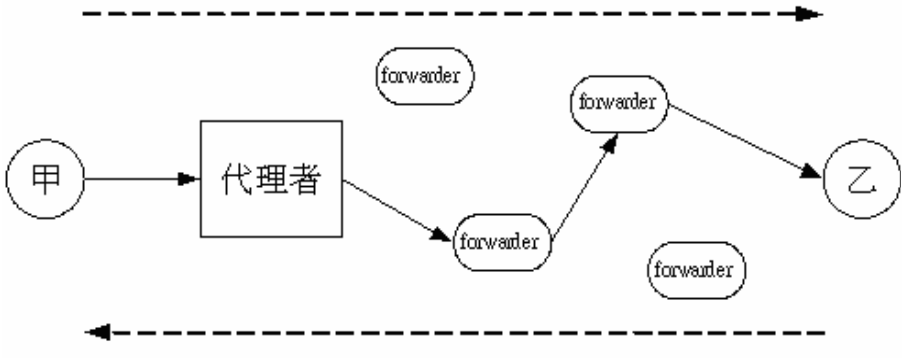
第三章　系統設計與實作

簡單的匿名概念，可以由通訊雙方之間，透過一個代理者來達成，如*圖 3.a* 中，甲與乙連線，甲方透過一代理者再連到乙，而乙則認為此連線要求者為代理者，而不知道此連線要求是甲所提出的。當甲透過代理者再連線到乙，雖然可以達到最基本匿名的要求，但是這種方式中代理者會知道甲與乙正在進行連線，並且，攻擊者仍然可以擷取代理者端封包進出的資訊，有機會推測出甲與乙正在進行通訊。



-圖 3.a 透過代理者達成的匿名連線-

因此，若是要避免攻擊者藉由擷取封包等方式，得知通訊雙方的真實位址，可以讓封包在傳送過程中，以透過幾部類似 forwarder 的機制，提高攻擊者找出實際在通訊雙方身分的難度，如*圖 3.b*。



-圖 3.b 代理者再經多部電腦轉送訊息-

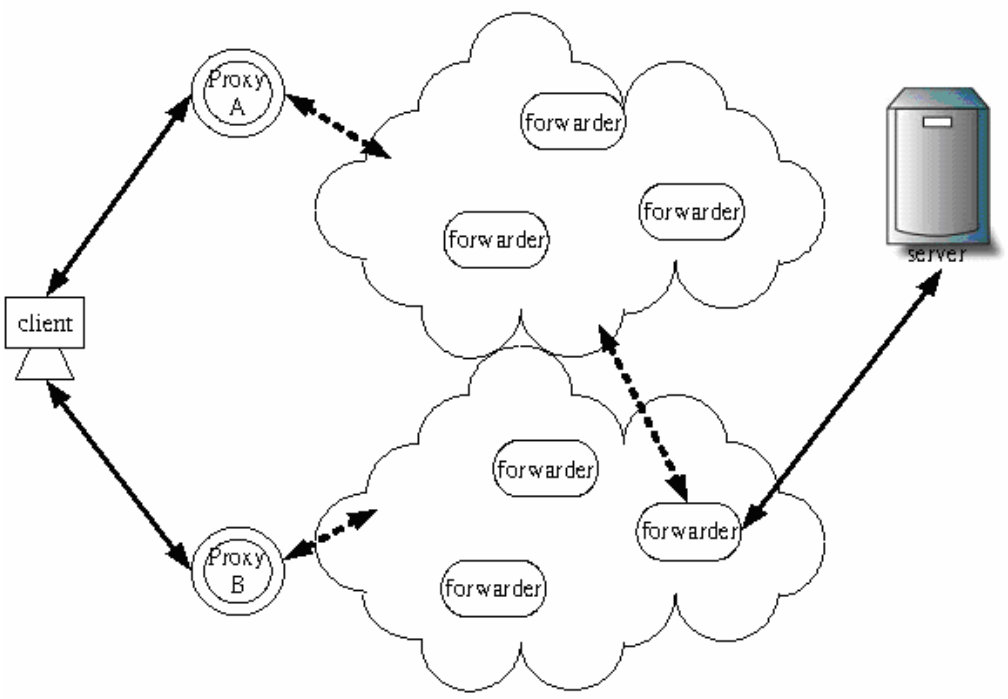
而本論文主要著重在甲與乙在連線過程中，能隱藏自己(甲)的位址。

本章主要說明系統如何設計，以及系統運作的流程：從 client 端送出的封包處理，再由 proxy 到各個 forwarder 中的轉送流程等。後面幾節則是說明如何實作這個系統。

3.1 系統架構概述

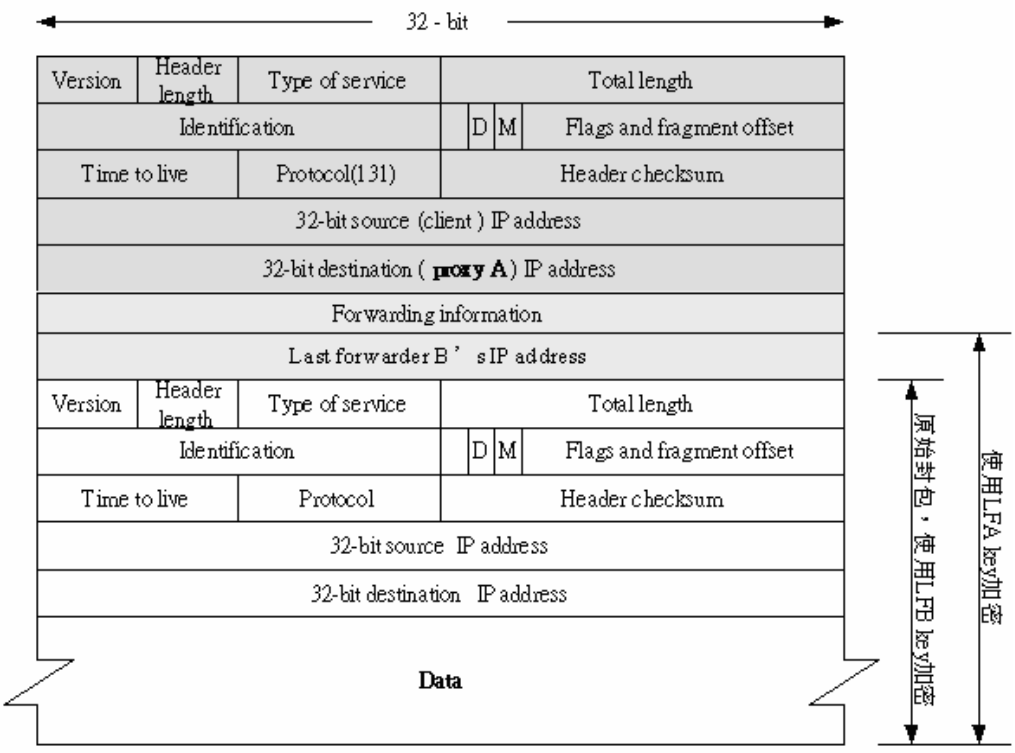
如*圖 3.1.1* 中，整個架構主要是由三個部分所組成，包括欲進行匿名連線的 host、決定封包行進路徑的 proxy，以及負責 packet forwarding 的數部 forwarder。主要的功能大致為：

- client：當開啟匿名連線功能後，接下來送出的封包會先送到 proxy，再透過 proxy 來傳遞。
- proxy：一部 proxy 紀錄著屬於該群組內所有的 forwarder 資訊。當 proxy 收到 client 送來的封包後，決定該封包接下來會經過哪幾部 forwarder，再將封包送出。也就是說，封包到達 server 之前，會經過哪些 forwarder 來轉送，是由 proxy 決定的。
- forwarder：接收到特定封包之後，會依封包內含的資訊，再將封包送往下一個目的地。因此，就 forwarder 的角度而言，只知道該封包是由何處傳來的，以及應繼續傳向何處，個別的 forwarder 並不知道該封包原始的來源以及真正的目的。這種設計方法可以增加整體系統的匿名性。



-圖 3.1.1 系統架構-

Client 開啟匿名連線時,會跟兩部以上的 proxy 發出請求,而 proxy 收到 client 的請求後，會在自己所屬群組中的挑選一部 forwarder，此 forwarder 為之後傳遞封包時離開該群組的最後一台 forwarder(last forwarder)，因為共有兩個群組，因此 proxyA 選出的 last forwarder 稱 LFA，proxyB 選出的 last forwarder 稱 LFB。因此 client 最後一共會得到的資訊為：隸屬於 proxyA，在此次匿名連線中被選為當作 last forwarder 的 IP 位址與 LFA.key，此即為 LFA 上的 symmetric key，還有隸屬於 proxy2，在此次匿名連線中被選為當作 last forwarder 的 IP 位址與 LFB.key，此即為 LFB 上的 symmetric key，client 再決定本次匿名連線要透過哪個 proxy 群組，依序層層包裝好封包並加密後再送出，如*圖 3.2.2*。



-*圖 3.1.2* client 送出的封包加密格式-

以下說明 client 與 server 溝通時封包傳遞的方式。假設 client 連至 server，過程中共需要兩個群組的 forwarder，透過兩部 proxy，以及不同群組的 forwarder 共三部，符號說明如下：

◆C：Client

◆PA：屬於群組 A 的 Proxy，知道目前屬於自己群組的 forwarder 有哪些，以及這些 forwarder 各自用來 forwarding 的 secret key。

◆F：第一個群組中的某部 forwarder (非 last forwarder)。

◆LFA：Last Forwarder A，為離開群組 A 前經過的最後一部 forwarder。

◆LFB：Last Forwarder B，為到達 server 前最後一部 forwarder，與 LFA 分屬不同群組。

◆S：client 的連線目標。

◆[X]{payload}[Y]：由 X 送往 Y 的封包。

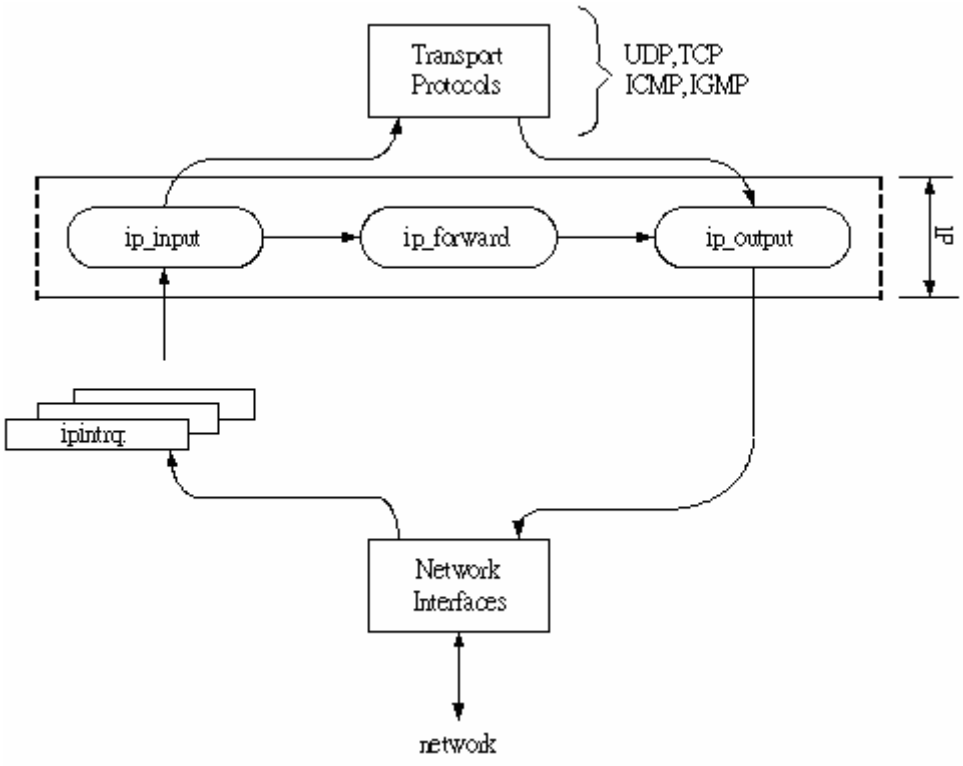
◆$(m)_x$：使用 X 所產生的 secret key 將訊息 m 加密，此 key 為專門用來 forwarding。

Client 與 server 傳送訊息的流程如下：

1. C → PA：[C]{ (*LFB's IP addr.*, $(m\_req)_{lfb})_{lfa}$ }[PA]
2. P A→ F：[PA]{ (*LFA's IP addr.*, (*LF's IP addr.*, $\{m\_req)_{lfb})_{lfa}$)$_f$ }[F]
3. F→LFA：[F]{ *LF*, $(m\_req)_{lfb})_{f2}$ }[LFA]
4. LFA → LFB：[LFA]{ $(m\_req)_{lfb}$ }[LFB]
5. LFB → S：[LFB]{m_req}[S]
6. S → LFB：[S]{m_rep}[LFB]
7. LFB→LFA：[LFB]{ $(m\_rep)_{lfb}$} }[LFA]
8. LFA →PA： [LFA]{ $(m\_rep)_{lfb}$ }[PA]
9. PA → F：[PA]{ {*C's IP addr.*, $(m\_rep)_{lfb}$}$_f$}[F]
10. F→ C：[F]{ $(m\_rep)_{lfb}$ }[C]

3.2 IP layer 系統運作

在 TCP/IP 協定中，資料由應用程式 傳到 Transport layer 之後，經由 IP 層包裝成 IP 封包，再往 data-link 層送。由*圖 3.2.1* 我們可以看到，UDP/TCP/ICMP/IGMP 封包送至 IP layer 後，最後是在 ip_output function 做 IP packet 處理，包含 IP header 欄位的填寫、route selection 與 fragmentation 等。因此，若是想改變 host 送出的封包，我們可以修改 ip_output function 來達到目的。



-圖 3.2.1 IP 層運作架構-

在 IP 層裡，當接收到一個 IP 封包時，IP 層中的 ip_input function 會先檢查 IP header 中 protocol 欄位，一般而言 kernel 大部分情況都只處理 protocol 欄位為 1(ICMP)、2(IGMP)、6 (TCP)，和 17(UDP)的封包，因此我們可以透過使用 raw socket 的方式，在上層開啟一個 raw socket，來接收上述四種格式以外的封包。並且，藉由 raw socket 提供的功能，我們可以自己產生一個符合我們需要 IP 封

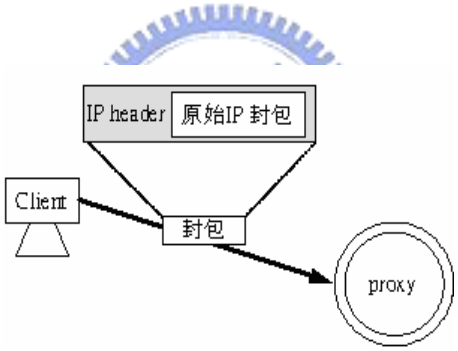
包(Raw IP)，並且自己填寫 IP header 欄位，包括像是 protocol、32-bit source IP address 以及 destination IP address 等欄位內容，之後計算 IP checksum 即可送出。

3.3 IP Packet 處理

分為『client 送出』的封包處理以及『 proxy 送出』的封包處理。

3.3.1 client 部分

由於我們希望 client 送出來的封包，都能先送給 proxy，但又不希望破壞原先的 IP header。因此，設計將原來的 IP packet 加上一個 IP header，以及一些之後轉遞時會用到的資訊，再送出去。故修改後的 IP packet 如*圖 3.3.1*：

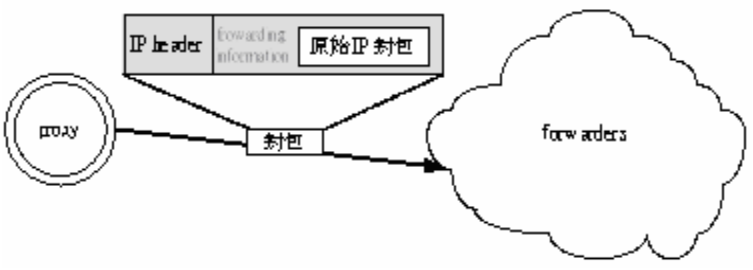


-*圖 3.3.1* client 送出給 proxy 的封包-

灰底部分為我們加上去的 IP header，白底則為原始 IP packet。封包傳給 proxy 後，再由 proxy 決定接下來該封包的行走路徑。

3.3.2 proxy 部分

當 proxy 收到 client 送來特定格式的 IP packet 後，會先紀錄一部分資訊，包含來源位址、來源 port 以及目的 port (TCP/UDP 封包)，之後根據目前網路上存活的 forwarder，決定封包移動的路徑，再將這些資訊依序層層包裝並加密，最後重新填寫 IP header，然後將封包送出。整個架構如*圖 3.3.2*。



-*圖 3.3.2* proxy 送給 forwarder 群的封包-

3.4 Packet Forwarding

a. request

使用專為 forwarding 用的 secret key 將封包解開一層，由 forwarding information 判斷是否為該封包傳遞過程中，扮演最後一台 forwardwer(last forwarder B，LFB)，若不是，則由 forwarding information 得知封包下一目的地，並往下一個目的地送出。若是，代表對該封包而言，此為 last forwarder，故在解密開封包後可知目的 server IP 位址，再將來源 IP 位址更改為此台 forwardwer 的 IP 位址，並重算 TCP checksum，最後送給目的 server。

b. reply

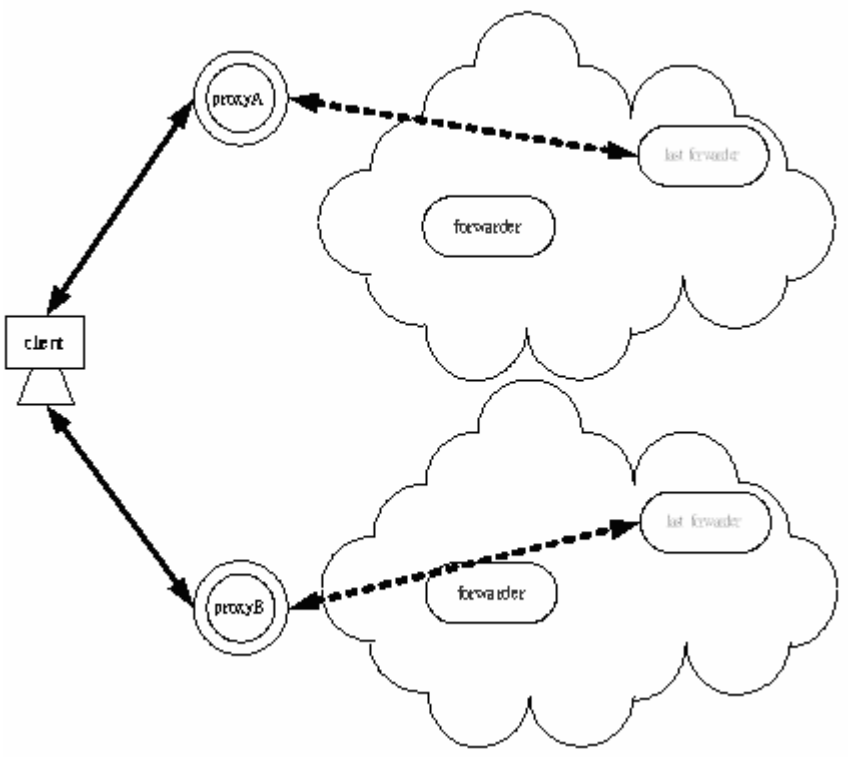
Last forwarder 收到 server 回覆的 packet 後，,使用 forwarder 本身的 secret key 將整個加密起來，外部再加上一層 IP header，將此封包傳給前一部 forwarder，即 LFA。而 LFA 收到此 reply 封包後，再轉送給自己所屬的 proxy，由該 proxy 找出真正的 client 位址，再透過其他 forwarder 轉送回給 client。

3.5 Proxy 問題、last forwarder 的表格與封包 padding

3.5.1 Proxy 中間傳遞 key 可能的問題

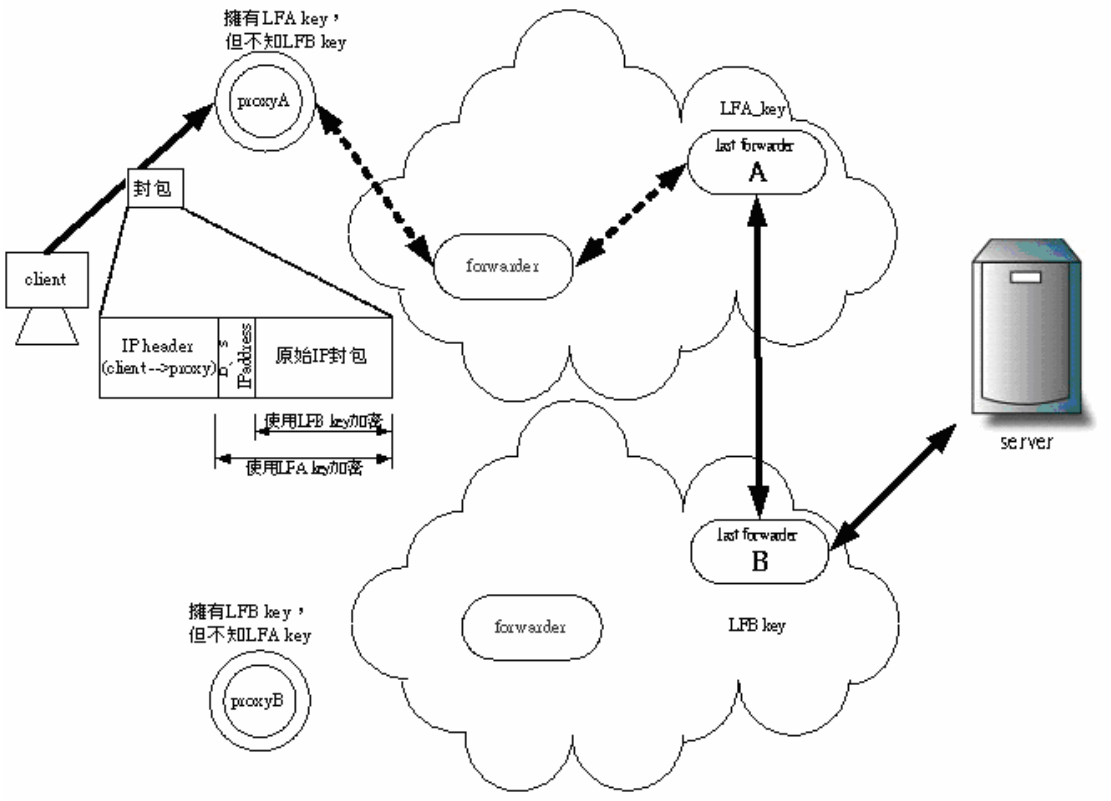
封包傳遞過程中，last forwarder 是由 proxy 決定的，並且，用來加密原始封包的 secret key 也是由 last forwarder 一開始產生後，經 proxy 轉送回給 client。因此 proxy 也是擁有該把 key。而當 client 接下來真正傳送資料時所送出的封包中，內部用來加密的 keyproxy 皆知道，如此會造成 proxy 可以將這些封包解密來得知連線目的 server 的真正位址。

因此若是 client 上的使用者若是想避免此情形發生，可以在一開始開啟匿名連線要求時，向不同群組的 proxy 發出要求，收到要求的 proxy 同樣會在自己的群組中選定一 last forwarder，被選定的 last forwarder 同樣都擁有一把 secret key，這把 key 會以 client 的 public key 加密後傳給 client。



-圖 3.5.1 client 發出的匿名連線請求-

接著 client 會收到兩個不同群組 proxy 所回應的資訊，包含該 proxy 中所選定 last forwarder 之一把 forwarding 用的 secret key，另外就是選定的 last forwarder 之 IP 位址。此時 client 可任意選定其中一部 proxy 做為第一個接收封包的目的地，由該 proxy 所屬的 forwarder 轉送封包後，經 last forwarder(同樣的群組)，再送給另外一個群組中被選定的 last forwarder，由另外一群組的 last forwarder 送給真正的目的 server。會採取這種方式，是因為這麼做可以避免加密最內層 IP 封包的 secret key，若經過的 proxy 也同樣知擁有該把 key，，則 client 所送出的封包，proxy 皆能解密開來得知真正的目的 server。



-圖 3.5.2 透過兩個群組的 forwarder 轉送封包-

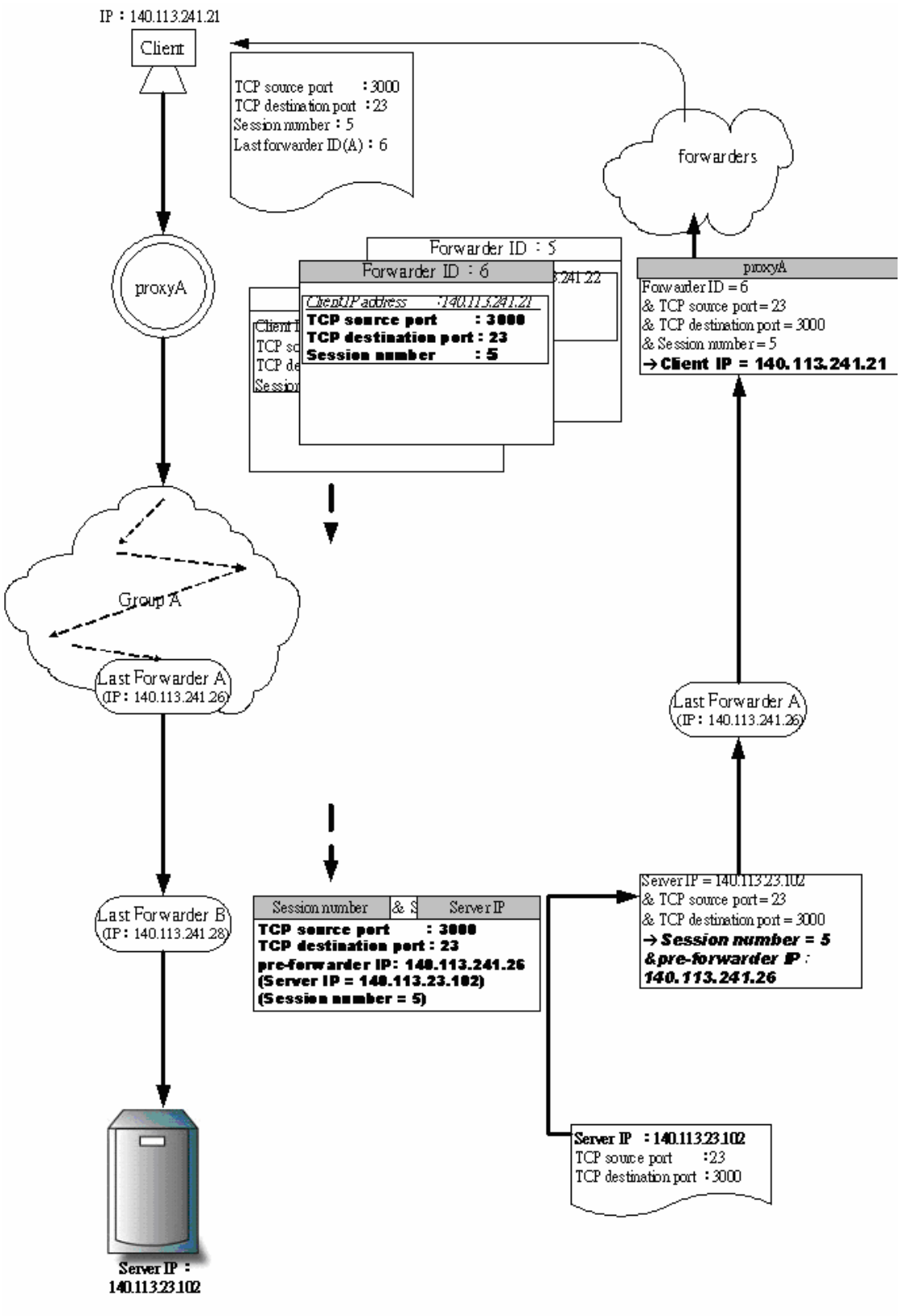
3.5.2 封包傳送時需建立的表格

由於我們希望能做到 client 的真實身分只有 proxy 能知道，並且 client 欲連線的目標 server 位址，我們又不希望被 proxy 得知。並且，client 的位址我們也不希望其他的 forwarders，包含 last forwarder 所知道。而每個 client 所送出的 TCP/UDP 封包中，含有連線所使用的 source port numbre 與 destination port number，因此我們可以藉由透過 source port 與 destination port，配合 proxy 針對每個要求匿名連線的 client 隨機配給的 session number，建立封包傳送往返時，查詢出真正目標位址的資訊。

proxyA 首先對於每個屬於自己群組的 forwarder 建立一個表格，當該 forwarder 被選為某次匿名連線的 last forwarder 時，所有的相關資訊，包括 client IP 位址、source port 與 destination port 便紀錄於此，另外 proxy 還會紀錄著 client 一開始要求匿名連線時產生的 session number(也可以視為在這一此的匿名連線中對此 client 所產生的 ID)。接著當封包移動到 LFB 時，在此也同樣會建立一組表格，內容紀錄了 session number、source port、destination port、pre-forwarder(即 LFA)， LFB 使用自己的 secret key(此 key 為 proxyA 所不知的)解開可以得到原始封包，就能知道目的 server IP 位址。

而當 LFB 收到 server 回應的封包後，由 server IP 位址、source port 與 destination port 查出對應的 session number，將 session number 資訊包裝至封包中，再回傳給 LFA，經 LFA 再回傳給 proxyA。

proxyA 收到由 LFA 所送來的封包之後，藉由該部 last forwarder_ID、source port、destination port 與 session number，查出 client 的 IP 位址，將這些資訊包裝至封包中，再透過一些 forwarders 轉送給 client。整個查表傳送的流程如*圖 3.5.3* 所示。

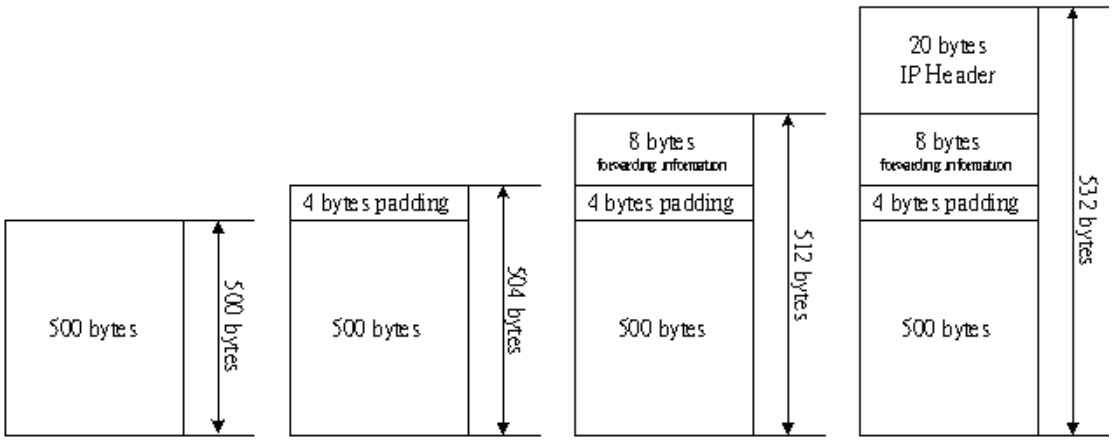


-圖 3.5.3 封包往返與查表-

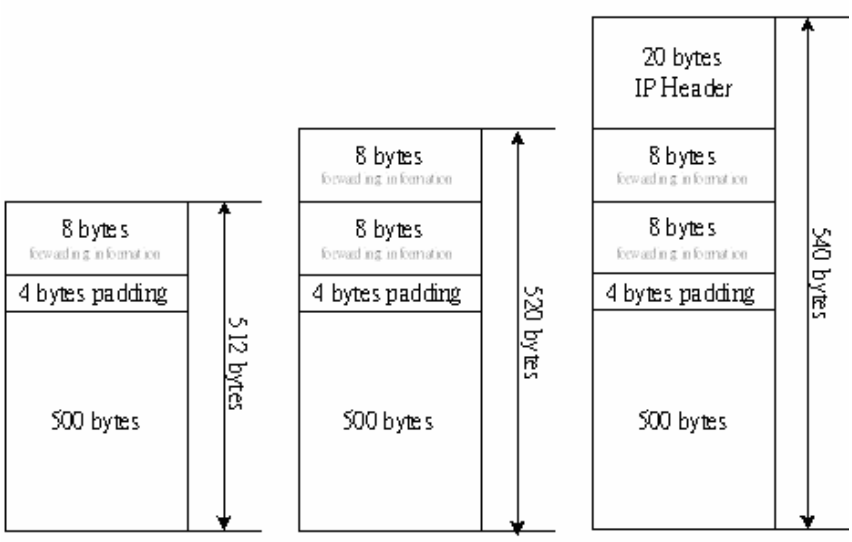
3.5.3 封包 padding

封包每經過一個 forwarder 後，會有一大小為 8 bytes 的 forwarding information 被去除，使得封包變小，並且這個封包每經過一部 forwarder 後便會變小 8 bytes，例如一個大小為 540 bytes 的封包，進入第一個 forwarder 後，出來的封包尺寸便由原先的 540 bytes 縮小為 532 bytes，再進入第二個 forwarder 後，出來的封包大小會再變小為 524 bytes，如此容易讓攻擊者有較多的資訊來分析封包行經的路徑，因此整個封包傳遞過程，除了最後的 LFB to server 以外，其他所有 forwarder 間的封包傳遞，封包大小應保持一致。

例如，當有一筆資料大小為 500 bytes 的訊息要傳遞時，首先，因為 DES 加密系統是以 8 bytes 為單位，因此 500 mod 8 = 4，8 – 4 = 4，因此需要 padding 4bytes，再加入 8 bytes 的 forwarding information，共為 512bytes，如*圖 3.5.4*；而若是希望封包傳遞過程中再多經過一部 forwarder，則可以再加入一 8 bytes 的 forwarding information，由於每多加一 forwarding information 皆為 8 bytes，因此不須再做 padding，如*圖 3.5.5*。

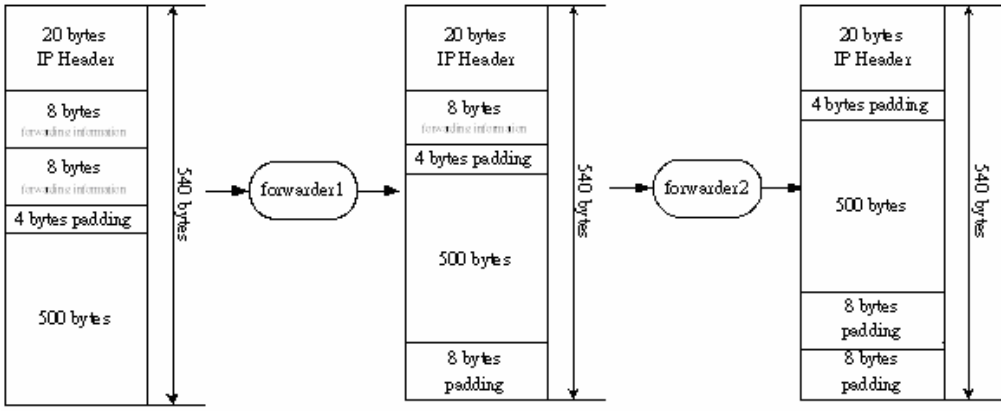


-圖 3.5.4 加密時封包大小需 padding 成 8 的倍數-



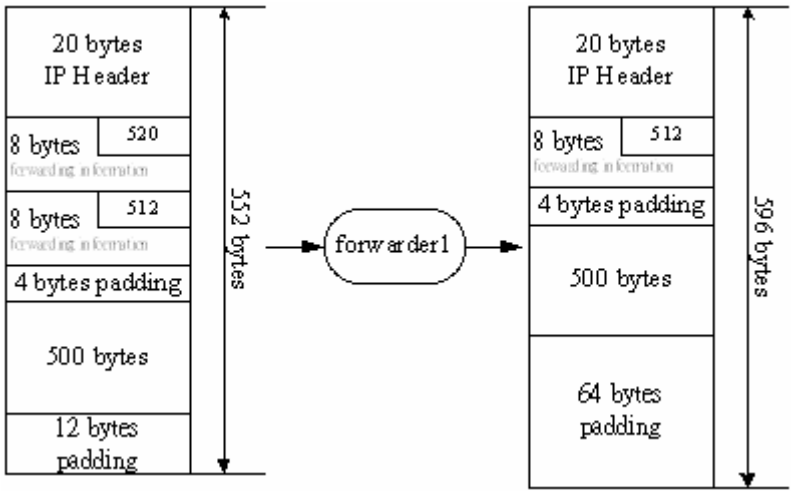
-圖 3.5.5 增加多筆 forwarding information-

當第一部 forwarder 收到如圖 3.5.5 最右邊，大小為 540 bytes 的封包後，先扣除 20 bytes 的 IP header，解密 IP header 之後的 520 bytes 資料，則可以看到第一組 forwarder information，由這裡可以知道實際資料大小為 512 bytes，剛好為 520 bytes 扣除 8 bytes 的 forwarder information 的大小，也就是說並沒有做 padding，之後當封包要由 forwarder1 送出給 forwarder2 時，512 bytes 的資料之後，由於剛剛去除了 8 bytes 的 forwarder information，因此必須再 padding 8 bytes 的大小，最後再加上一 IP header，512 + 8 + 20 = 540 bytes，如*圖 3.5.6* 所示，這麼做可以令同一個封包進入 forwarder 再轉遞出去後，封包大小不會改變。



-圖 3.5.6 每移除一筆 forwarding information *需再* padding-

或是動態決定 padding 的大小，使得封包進入 forwarder 與送出時的大小皆不相同，使封包在進出之間難以辨別相關性，更能提升攻擊者分析封包的難度，如*圖 3.5.7*。



-圖 3.5.7 動態決定 padding 大小-

3.6 系統實作

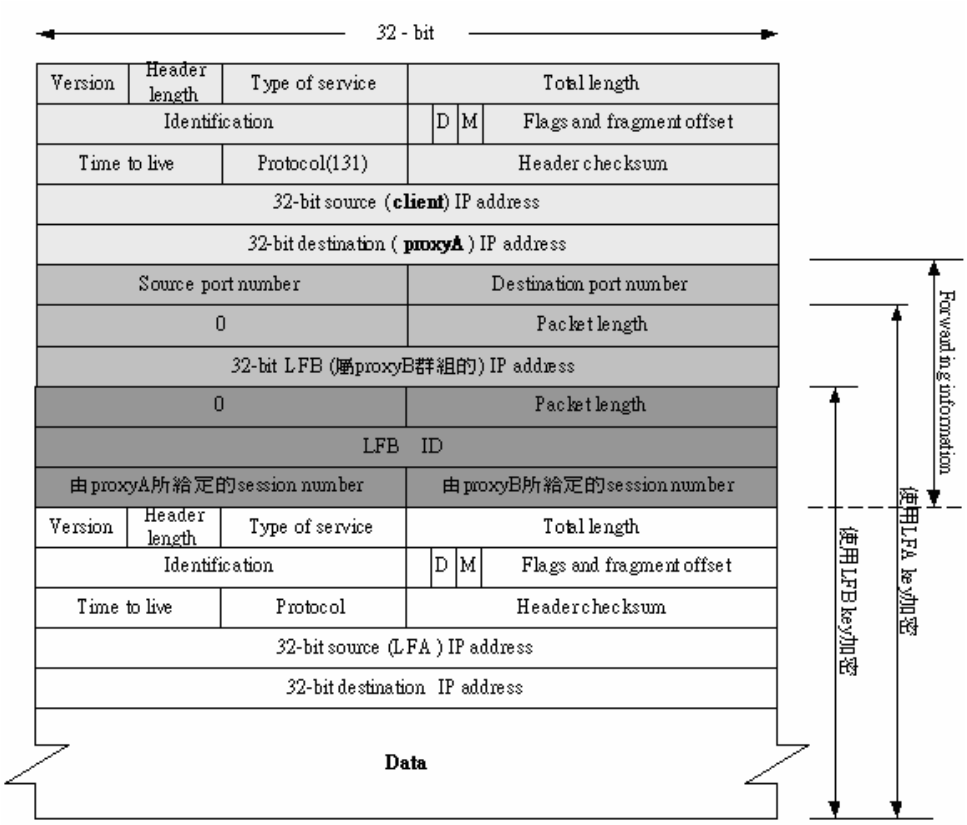
整體系統架構如*圖 3.1.1*，因此實作上共分為三個部分，分為 client、proxy 與 forwarder。

3.6.1 Client 端

在 client 端，我們必須處理的情形，可分為匿名封包的送出，以及收取匿名封包，而當 client 開啟匿名連線時，會向兩部以上 proxy 發出請求，兩部 proxy 分別會選定一台 last forwarder，在此假定由 proxyA 群組中選到的 last forwarder 稱為 LFA，另外一 proxyB 群組中選到的 last forwarder 稱為 LFB2，而 client 之後會收到由 proxyA 回應的資訊為：LFA 的 IP 位址、一把 LFA 上用來 forwarding 的 secret key，在此令其為 LFA key，以及 LFA 在 proxyA 註冊時的 ID 編號，另外就是 proxyA 對於這次 client 要求匿名連線給定的一個 session number。同樣的也會從 proxyB 收到資訊為:LFB 的 IP 位址、一把 LFB 上用來 forwarding 的 secret key，在此稱其為 LFB key1，以及 LFA 在 proxyA 註冊時的 ID 編號，還有就是

proxyB 對於這次 client 要求匿名連線給定的一個 session number。最後 client 再自己選定往與 server 溝通的封包時要送給哪部 proxy。本論文以下內容中接假定 client 會選定 proxyA 做為轉送封包的第一站，而最後送達 server 前的一部 forwarder 則為 LFB。下面說明與 server 溝通時送出封包的格式。

一個原始的 IP packet 格式如*圖 2.1.1*，由於我們目標是將封包傳送目標改為 proxyA，因此對原來的 IP packet 再加上一 IP header 以及 forwarding information。格式如*圖 3.5.1*：



-*圖 3.6.1* client 送出要求 server 的封包格式-

淺灰底部分為我們加上去的 IP header，深灰則為 forwarding information，白底則為原始 IP packet。

『forwarding information』欄位說明(由上而下)：

・Source port & Destination port number：大小為共為 32-bit，由原始 TCP/UDP 封包中複製出一份相同的 Source port & Destination port number，目的是讓 proxyA 接收到該封包後可以取得這些資訊，以便將來轉送 server 回覆的封包給 client。

・Packet type & packet length：大小為共為 32-bit，前半段 16-bit 填 0(request)，接著後面 16-bit 則填寫未加密前封包的長度(計算為這裡的 0 & packet length 到封包尾端)。

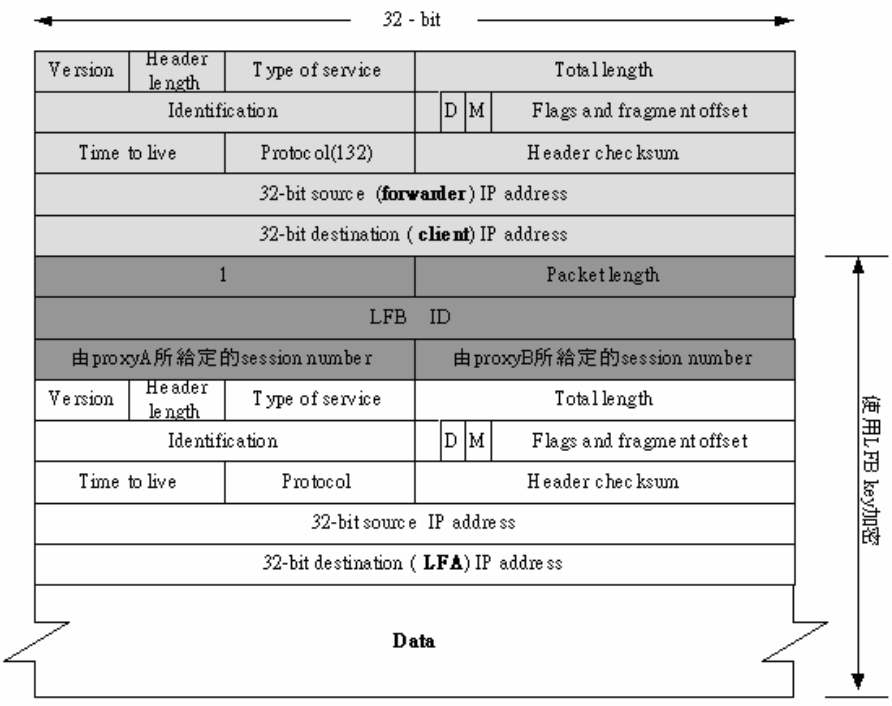
・32-bit LFB IP address：放置 LFB 的 IP 位址。

・Packet type & packet length：大小為共為 32-bit，前半段 16-bit 填 0(request)，接著後面 16-bit 則填寫未加密前封包的長度(計算為這裡的 session number & packet length 到封包尾端)。

・LFB's ID：大小為 32-bit，此值是 proxyB 給定的。

・Session number：共 32-bit，前 16-bit 放 proxyA 指定的 session number，後 16-bit 則是存放 proxyB 所指定的 session number。

除此之外，client 尚還需處理 server 回覆(reply)的封包。Client 收到回覆的封包格式如*圖 3.6.2*：



-*圖 3.6.2* client 收到 server 回覆的封包格式-

判斷是否為 reply 封包，我們在本論文中是以當封包的 IP protocol 欄位為 132

時，則 client 即在 ip_input fountion 中處理該封包。先在 ip_input function 中去掉 20bytes IP header，再使用 LFB key 解開剩下的部分，然後檢查 session number 欄位與 last forwader ID 是否正確，若是都正確，則根據 length 欄位去除因加密所 padding 的部分，得到原始 server 回應封包，接著於原始封包 IP header 中 desintation IP address 欄位中填入 client 自己的 IP address，並重算 TCP checksum，或是令當檢查 TCP checksum 時令其跳過不檢查(這是因為在 IP 層重算，之後檢查一定還是正確的)。

3.6.2 Proxy

Proxy 主要會收到兩種形式的封包，一種為 client 送過來的；另外一種則為 server 回覆 client 的封包，此類封包是由 last forwarder 轉送過來的。而由 proxy 送出的封包，也分為轉送 client_to_server (request)與 server_to_client(reply)的兩種。

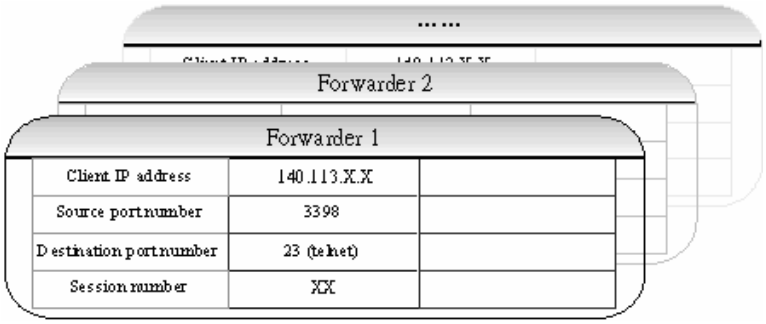
<1>接收到 client 送來的封包

分為兩部分。首先我們在上層，開啟一個 raw_sock，等待接收 IP protocol 為 131 的封包，當收到此類封包後，根據一開始 client 要求匿名連線時所給定的 last forwarder，對其建立不同的 table，其中紀錄 client 的 IP address 與、Source port number、Destination port number 與 session number，再處理後續 forwarding 的部分。

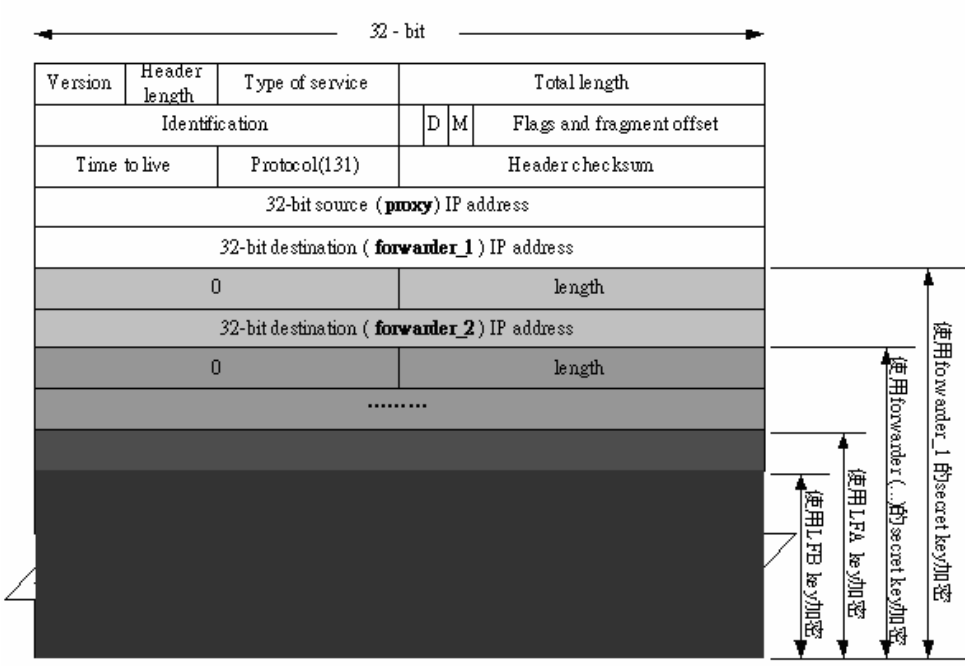
<2>送給 forwarder 的封包

proxy 會先對每個 forwarder 建立一個表格，之後當收到 client 送來的封包時(格式如*表 3.6.1*)，proxy 會根據先前指定給該部 client 的 last forwarder，找出該 forwarder 的表格，將 forwarding information 中的資訊，也就是 source port 與 desionation port number 紀錄下來，還有該部 client 在本次匿名連線中所對應的 session number 也一併記下，如*圖 3.6.3*。接著選擇數台目前存活 forwarder 的位

址，依照安排的移動順序包裝並層層加密起來，最後加上一 IP header 後再送出，整個封包的格式如*圖 3.6.4* 所示。



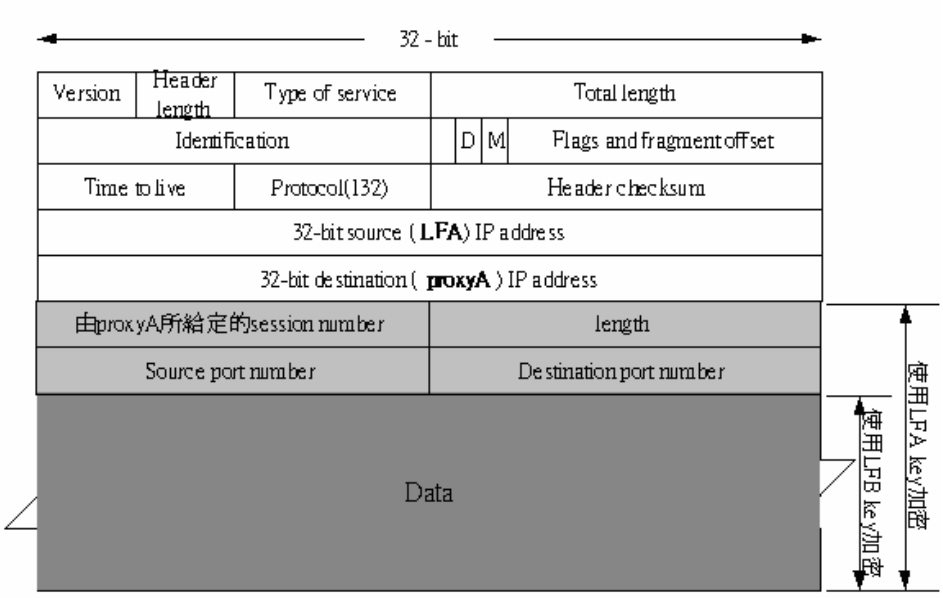
-*圖 3.6.3* last forwarder *表格*-



-*圖 3.6.4* proxy 傳給 forwarder 封包格式-

<3>接收到 server 回覆，由某部 forwarder 所轉送來的封包

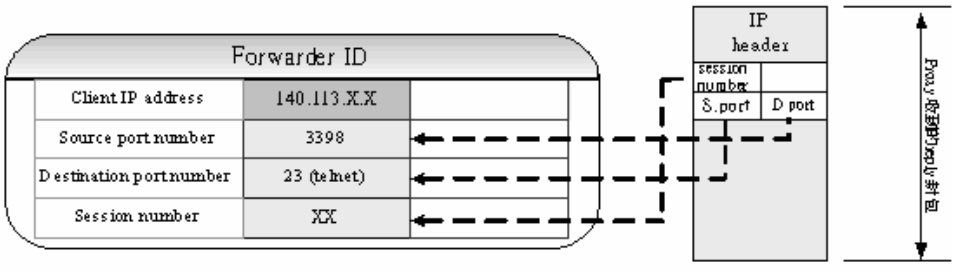
與接收到 client 送來的封包類似，在上層開啟一個 raw_sock，等待接收 IP protocol 為 132 的封包，當收到此類封包後，根據來源 IP 位址找出對應的 secret key，再由 IP header(20 bytes)加上 source port&destination port(4 bytes)再加上 1 byte 的 session number 與 16-bit 的『0』後面開始，這把 key 解密，之後再處理後續回送給 client 的部分。收到的封包格式如*圖 3.6.5*。，



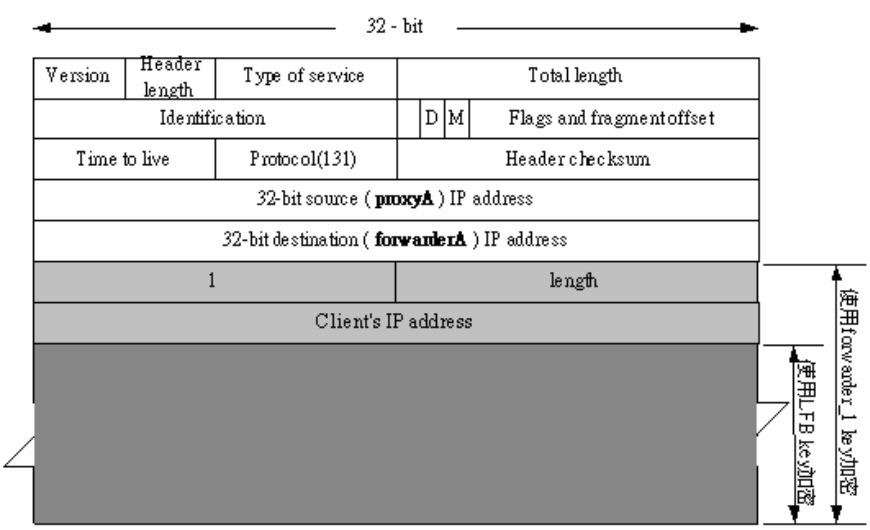
-*圖 3.6.5* forwarder 傳給 proxy 的封包格式-

<4>送回給 client 的回覆封包(replay packet)

收到「<3>接收到 server 回覆，由某部 forwarder 所轉送來的封包」後，如同上述所說一般，先拆開一層 IP header，然後根據來源 IP 位址找出對應的 secret key 解密，得到 Source port number、Destination port number 與 session 找出對應的 last forwarder ID 表格，查表比對 source port number、destination port number 與 session number 來決定回送的 client 位址(參考*圖 3.6.6*)，接著再任意選擇數部 forwarders，如同先前轉送 client 送給 server 的 request 封包一般，reply 封包也是令其再透過數部 forwarder 轉送回 client，完整送出的封包格式如*圖 3.6.7* 所示。



-*圖 3.6.6* proxy 查表找出對應 client IP 位址-



-*圖 3.6.7* proxy 回傳給 client 的封包格式-

3.6.3 Forwardwer

Forwarder 主要包含以下數種功能：

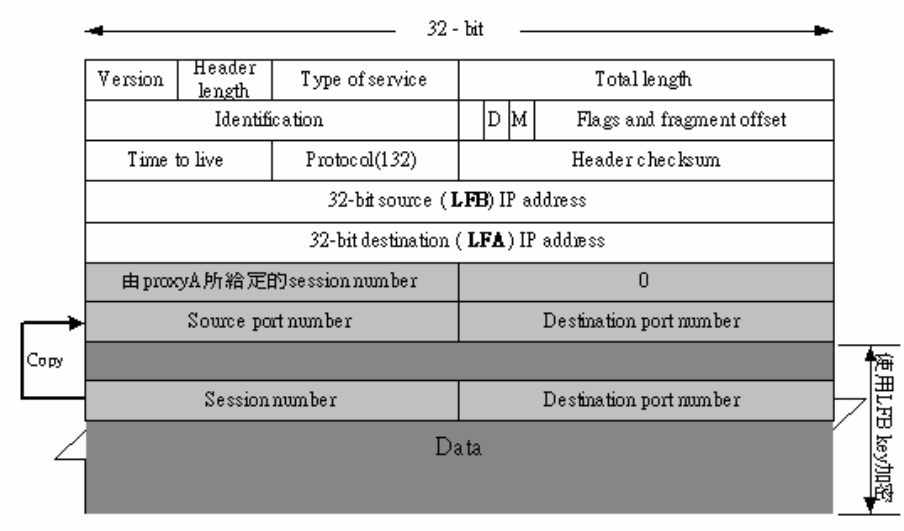
<1> 向 proxy 提出註冊請求，以便加入該群組，包含取得 proxy 的 public key，以及告知 proxy 自己的 secret key，這把 key 主要是做為封包 forwarding 時所使用，並且該把 key 是藉由 proxy 的 public key 加密後再回傳給 proxy。另外就是每隔一段時間對 proxy 送出存活通知。

<2> 收到特定封包(IP protocol 為 131 或 132)後，若 IP protocol 為 131，表示從 IP header(封包 20 bytes 後)開始使用自己的 key 解開，並根據解開後得到的資訊找出封包接下來轉送的目的；若是 IP protocol 為 132，則將收到的封包由 IP header 之後全部以自己的 key 加密起來，再轉送給 proxy，送出的封包格式如同圖 *3.6.5*。

<3> 依照封包訊息，決定該封包的處理方式，包含再次 forwarding、將 server 回應的 reply 封包回送給前一部 forwarder，或是擔任連線 last forwarder 時將該封包送給目的 server，與收到 last forwarder 傳來的封包(LFB to LFA)四種：

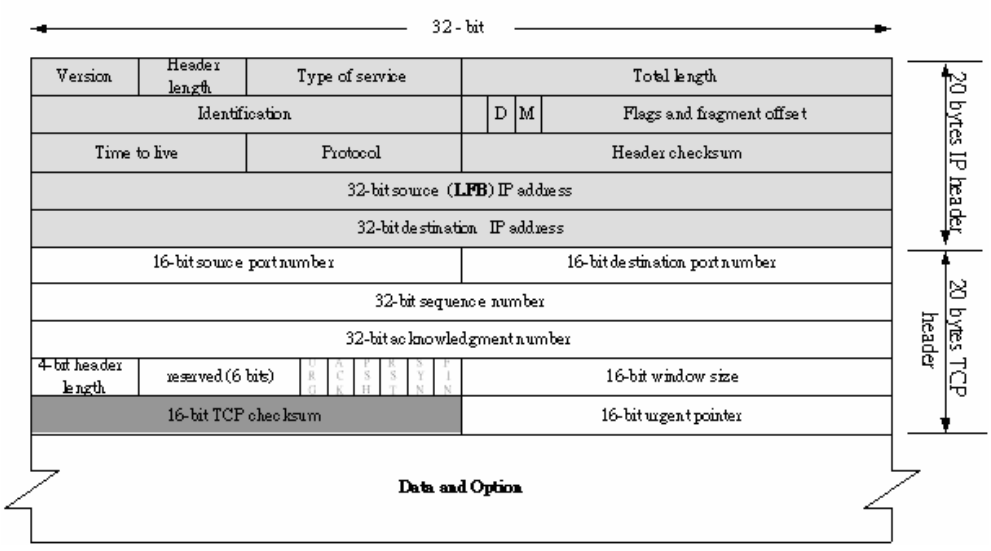
(1)擔任 last forwarder(指 LFB)的 forwarder 收到 server 回應的封包後，查表

找出前一部 forwarder 的 IP 位址，然後重新包裝封包如*圖 3.6.8*。



-*圖 3.6.8* LFB 回傳給 LFA 的封包格式-

(2)傳送給 server，對 TCP header 中 source port number 部分，我們必須先以 session number 建立一個表格，內容紀錄了原始 TCP source port、destination port、pre-forwarder IP address，以及 server IP address，最後重算 TCP checksum 再送給 server，送出的封包格式如*圖 3.6.9* 所示。



-*圖 3.6.9* 重算 TCP checksum-

(3)收到 last forwarder(即 LFB)傳來的封包，格式如圖 3.6.8，則再將該封包加密一層後傳給 proxy，送出的封包格式如*圖 3.6.5* 所示。

(4)將封包再轉送給下一個 forwarder，先去除一個 IP header，再使用自己的

secret key 解開封包，可以得到該封包的下一個目的位址，重新包裝 IP header 後再送出。

3.7 執行流程

3.7.1 啟動匿名連線

分別向兩部(或兩部以上)proxy 發出匿名連線請求，proxy 收到請求後，從該群組的所有 forwarder 中選出一部 last forwarder，並配置一 session number 給該部提出匿名連線請求的 client，同樣的另外一部收到 client 提出匿名連線要求的 proxy 也一樣。

完整流程敘述如下：

<1>client 告知 proxyA 與 proxyB 將進行匿名連線，並把 client 自己的 public key 送給 proxyA 與 proxyB。

<2>proxyA 得知有一 client 將進行匿名連線時，先由該群組中選定一台 forwardwer 做為連線代理之 last forwarder，在此步驟中，由 proxyA 決定 last forwarder 是哪一台，而不是讓 client 決定 last forwarder 的原因在於，只有 proxyA 知道有哪些 forwarder 可用，client 並不知道，因此 last forwarder 必須交由 proxyA 來決定，之後並針對 client 這次的匿名連線產生一 session number。同樣的 proxyB 也會於所屬群組中挑選出一部 last forwarder，以及一組 session number。

<3>最後將這些資訊，包括選定的 last forwarder_ID、last forwarder IP 位址、last forwarder 的 secret key 以及 proxy 所配置的 session number，使用 client 的 public key 加密起來回傳給 client。

<4>client 再決定之後與 server 傳遞封包要先送給哪部 proxy，若是選定 proxyA，則之後封包移動路徑為

client → proxyA →{數部為 proxyA 群組中的 forwarder} → LFA → LFB → server ，

若是選定為 proxyB，則之後封包移動路徑為

client → proxyB →{數部為 proxyB 群組中的 forwarder } → LFB → LFA → server 。

3.7.2 傳遞封包

<1> client 開啟匿名連線功能後，所有封包皆會在 ip output function 中先經過處理，再透過 proxy 去轉送。

<2>proxyA 收到 client 送來的封包後，依照 forwarding information 中的內容，先檢查 source port number 與 destination port number，接著再選擇數台 forwarder(在此以經一部 forwarder1 後到達 LFA 為例)，依序包裝好 forwarder IP address 並加密，再送出給第一部 forwarder1。

<3>Forwarder1 收到 proxy 送來的封包後，先解開一層，再由解開的封包內部資訊中得知下一個目的地為 LFA，去除存放這些資訊的內容後重新填寫 IP header 後再送出。

<4>LFALFA 收到 forwarder1 送來的封包後，同樣先解開一層，並且由內部資訊中得到下一個目的地為 LFB，同樣的，去除存放這些資訊的內容後重新填寫 IP header 後再送出。

<5>LFB 收到 LFA 送來的封包後，同樣先解開一層，由內部資訊中得知該封包必須經自己轉送給 server，因此改變原始 IP header 中 source IP address，重算 TCP checksum 後再送出給真正的目的 server。

<6>當 server 收到由 last forwarder 送出給他的封包後，之後回覆的封包會送給 LFB，這是因為 server 會認為此連線是由 LFB 所要求的。而 LFB 收到 server 回應的封包後，經由查表比對 session number、source port、destination port 與 pre-forwarder IP 位址，再將封包送回給 LFA。

<7>LFA 收到後，再回送給 proxyA。

<8>proxyA 收到此封包後，找出該封包真正的目的位址(client)之後，再選擇數部 forwarders 來轉送該封包(在此僅選擇一部 forwarder1)，將轉送的 IP 位址資訊包裝好，再送給該 forwarder1，透過該 forwarder1 將此封包轉送給 client。

3.7.3 結束匿名連線

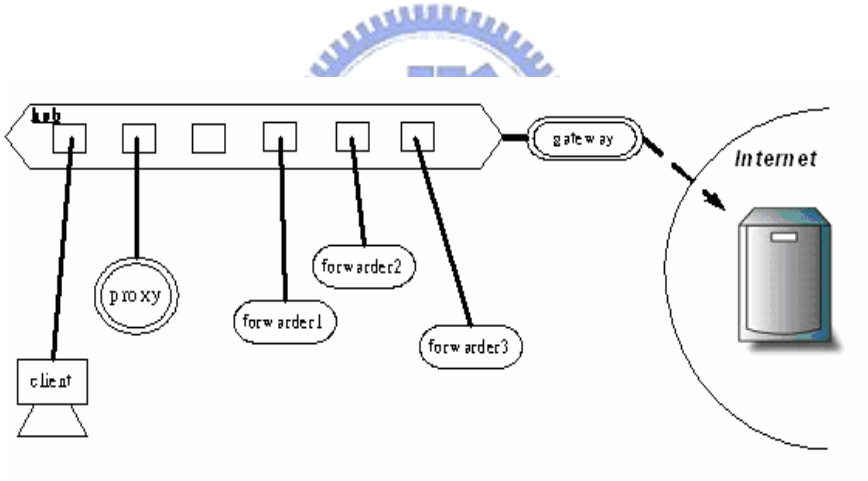
當 client 欲結束匿名連線時，會先送出通知給 proxyA 與 proxyB，proxyA 與 proxyB 再分別通知 LFA 與 LFB 令其釋放對應 session number 相關的表格，最後 proxy 再清除對應 forwarder 上的資訊。

第四章　實驗成果

分為「匿名實驗測試」與「效能實驗結果」兩部分。

4.1　實驗設計

為了展示實際匿名封包傳輸的情形，我們可藉由 Ethereal 這套程式聽取在 client、proxy 與各個 forwarder 中封包的傳送情形，以及 server 對於此次連線，判定送出連線要求的 client 其 IP address 為何。整個實驗的系統架構圖如同*圖 4.1.1* 所示，所有電腦都接在同一個 hub 上方便捕捉封包觀察，再連到 Internet 上，存取遠端的 server，此部分是以交通大學資訊科學系計算機中心的工作站 (140.113.23.102)做為連線測試。



-圖 4.1.1 實驗系統架構配置圖-

表 4-1 實驗設備

client	FreeBsd 5.1,　AMD athlon XP 1800+,　256 MB of RAM
proxy	FreeBsd 5.1,　AMD Duron 1.8Ghz,　128 MB of RAM
forwarder1	FreeBsd 5.1,　Intel CELERON 333 Mhz, 128 MB of RAM
forwarder2	FreeBsd 5.1,　Intel CELERON 300 Mhz, 128 MB of RAM
forwarder3	FreeBsd 5.1,　Intel PentiumIII 700Mhz,　128 MB of RAM

其中 proxy 扮演 proxyA，forwarder1 與 forwarder2 同屬 group A，forwarder2 扮演 LFA；而 forwarder3 則同時扮演 proxyB 與 LFB。

4.2 實驗結果

4.2.1 匿名性

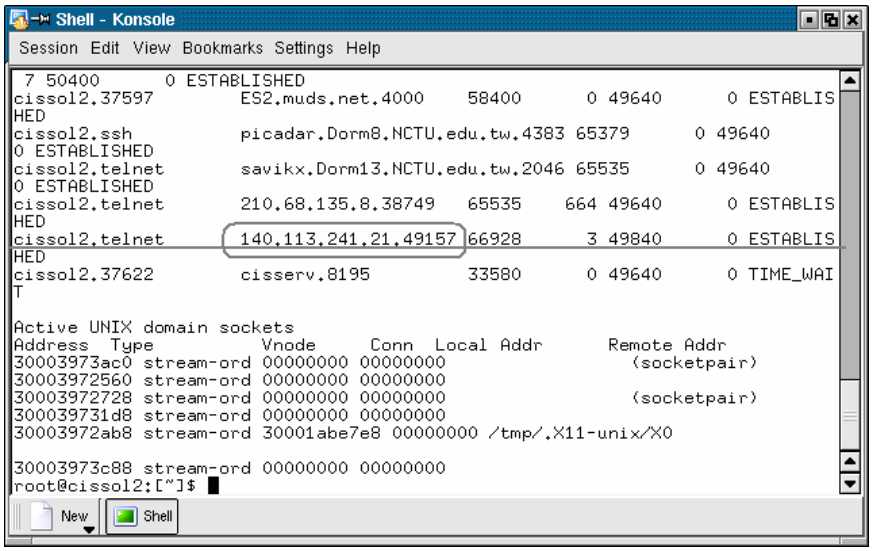
由於原始的 IP header 是一直都放在後面 proxy router 及 forwarder 所造的 IP header 之後，並且加密，所以無論在何處擷都取封包都無法得知真正發送端與接收端的 IP，而截取到的封包 IP Address 只有以下情況：

表 4-2 封包匿名性

1	source	client IP address	destination	proxy IP address
2	source	proxy IP address	destination	forwarder IP address
3	source	forwarder IP address	destination	forwarder IP address
4	source	forwarder IP address	destination	server IP address
5	source	forwarder IP address	destination	proxy IP address
6	source	forwarder IP address	destination	client IP address

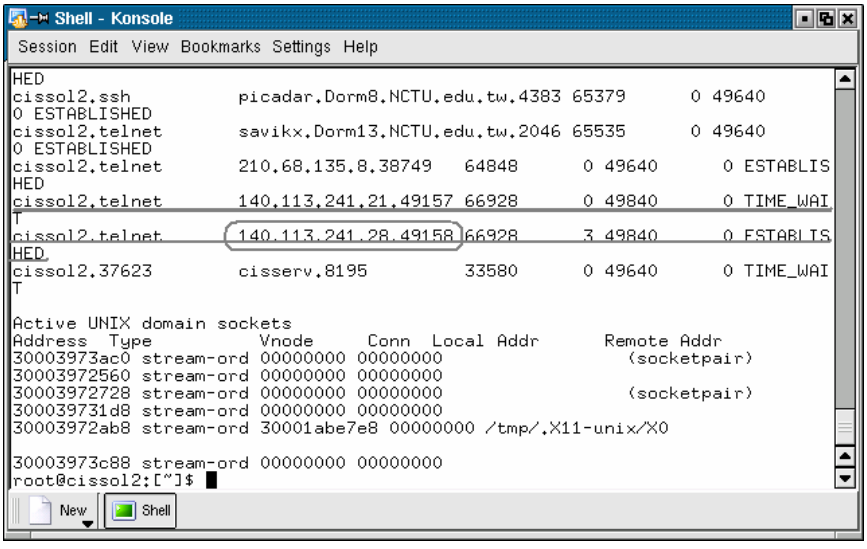
1. Telnet 連線測試：

連線目標 server 為交通大學資訊科學系計算機中心工作站 cissol2 ，IP 位址為 140.113.23.102，client 的 IP 位址則為 140.113.241.21。以 telnet 連上 server，連上後下 netatat 指令，可以得知目前該 server 與哪些 client 建立連線，如*圖 4.2.1* 中，畫上底線部份，我們可知目前 client 正藉由 140.113.241.21 這個 IP 位址與 server 連線中。



-*圖 4.2.1* Telnet 正常連線結果-

接著我們啟動匿名連線，同樣再 telnet 到 server 上，一樣執行 netstst 指令，結果如*圖 4.2.2* 中，可以發現目前與 server 進行連線的 client 中，有一 IP 位址為 140.113.241.28 的電腦，而該部電腦正是其中一部 forwarder，並且可以確定的是，在該部電腦(140.113.241.28)上，目前並無執行 telnet 連線到該 server 之上，可知我們成功的透過其他電腦連上了 server，並且 server 無法確切得知真正連線的 client IP 位址。

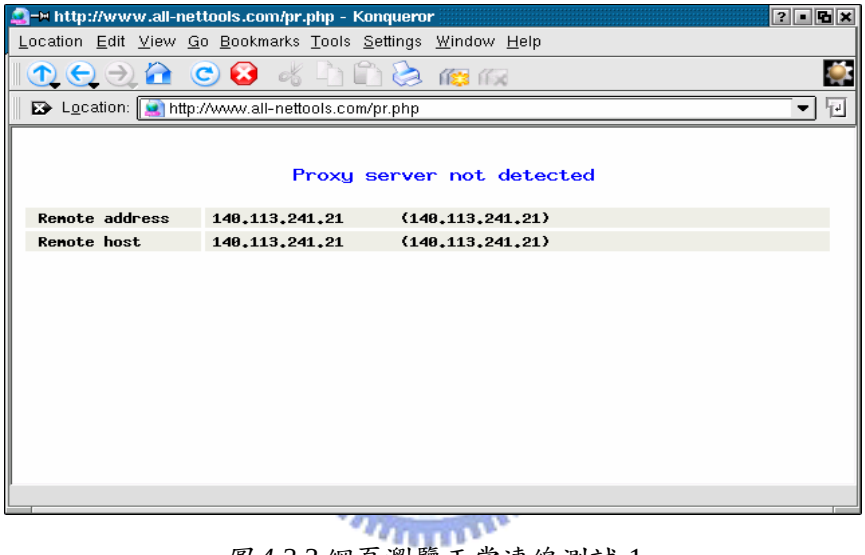


-*圖 4.2.2* Telnet 匿名連線測試-

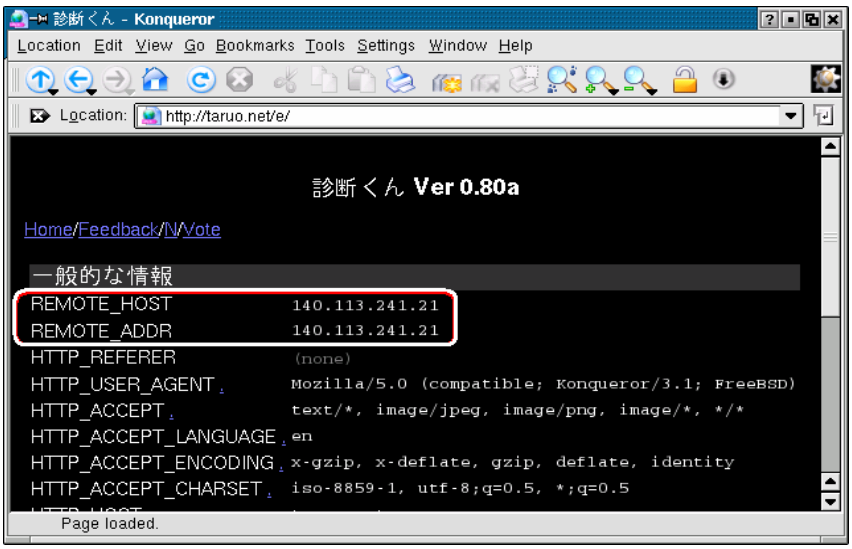
2. HTTP(Hypertext Transfer Protocol)網頁連線測試

client 連上某特定網頁，會回報該 client 目前連線所使用的 IP 位址為何，例如 http://www.all-nettools.com/pr.php、http://taruo.net/e/ … 等，藉由這類的網頁，使用者可以測試 client 連到該 server 時，有否透過 proxy 或是直接連上的。

我們分別測試了這兩個網頁，一開始是採用正常模式連線，得到的結果如同*圖 4.2.3*、*圖 4.2.4*，server 回報皆為目前 client 使用的 IP 位址 140.113.241.21。

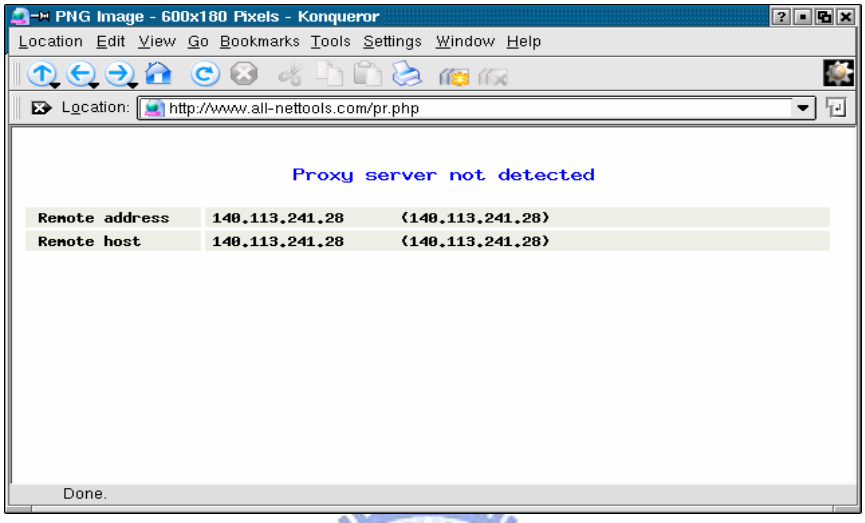


-*圖 4.2.3* 網頁瀏覽正常連線測試-1-

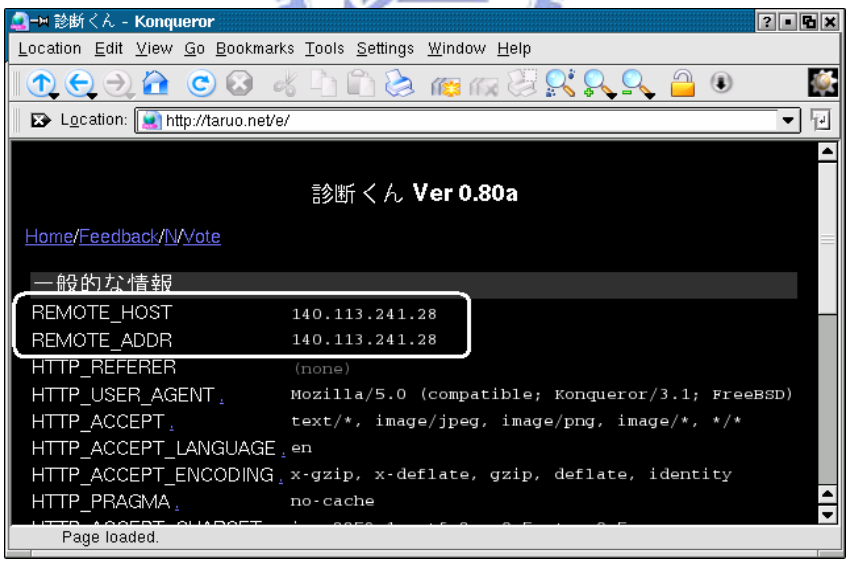


-*圖 4.2.4* 網頁瀏覽正常連線測試-2-

接著我們啟動匿名連線，重新連上網頁，得到的結果如*圖 4.2.5 與圖 4.2.6*，回報的結果，都認定連線 client 位址為 140.113.241.28，這是某部 forwarder 的 IP 位址，而非 client 真實的 IP 位址。



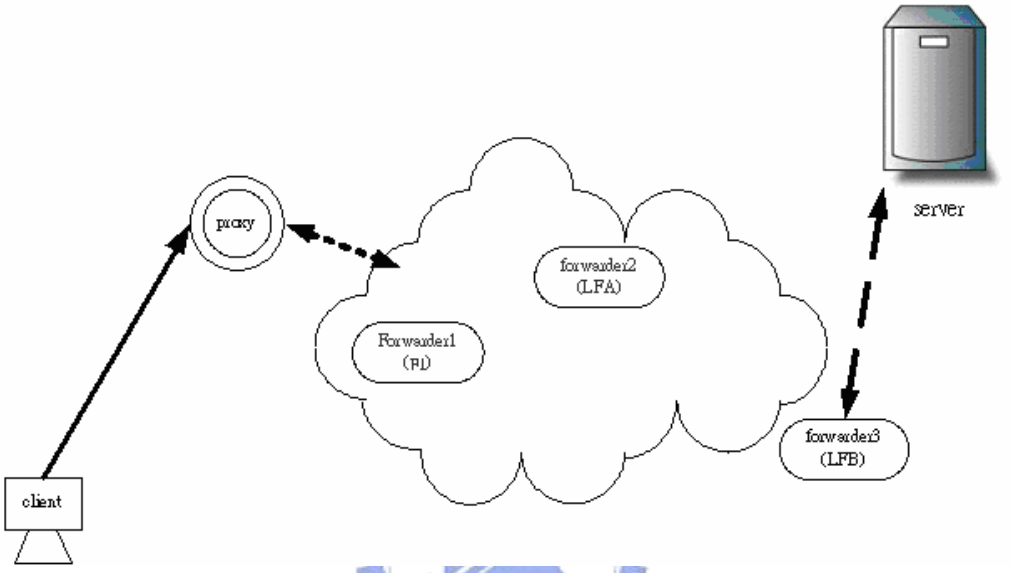
-*圖 4.2.5* 網頁瀏覽匿名測試-1-



-*圖 4.2.6* 網頁瀏覽匿名測試-2-

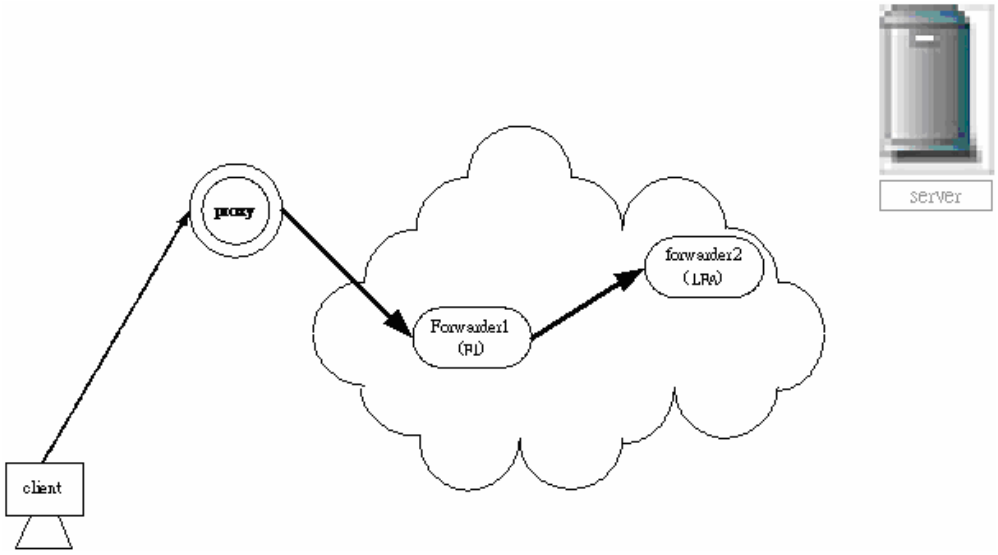
匿名性分析：

以 client 的觀點來講，整個連線過程中 client 知道 server 存在的位置，也知道 proxy 的位址，但 client 無法得知封包傳遞過程中，會經過哪些 forwarder，僅會知道先經 proxyA 後再走到 LFA，而到達 LFA 之前還會經過哪些 forwarder 就不是 client 能決定的了，如*圖 4.2.7* 所示。



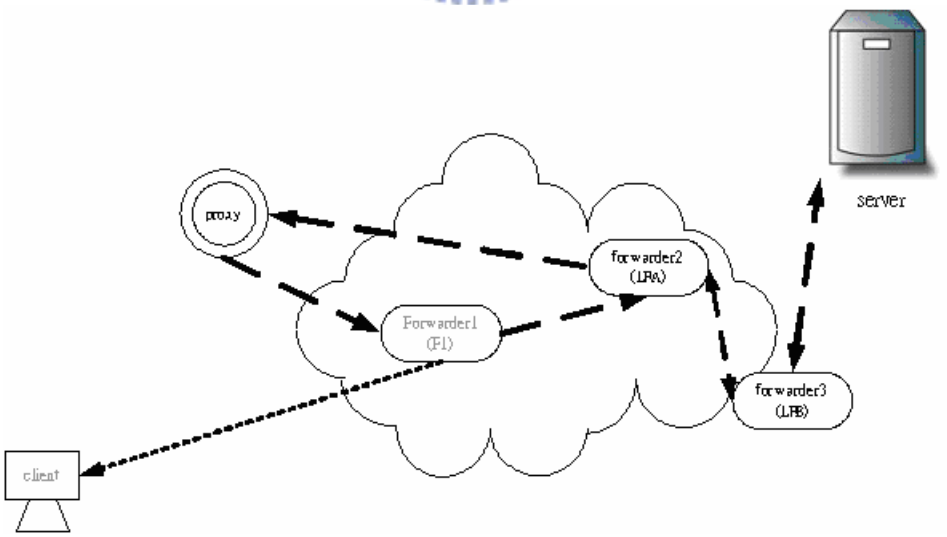
-*圖 4.2.7* 匿名性-client 的觀點-

而若是以 proxy 的觀點，則 proxy 可以知道每個轉送的封包，client(來源)的實際位置，以及接下來會經過哪些 forwarder 轉遞該封包(這是由 proxy 自行決定的)，另外就是封包於該群組中到達的最後一台 forwarder，即 LFA，但是對於該封包實際的目的地(server)與 LFA 之後還會送到哪(在此指 LFB)並無法得知，如*圖 4.2.8* 所示。



-*圖 4.2.8* 匿名性-proxy 的觀點-

中間的 forwarder，若只是單存負責轉送封包，則該 forwarder 僅會知道此封包是由哪個 node 送過來的，以及接下來要轉送給誰。而最後做為 last forwarder 的 node，其獲得的資訊則為：目的 server 位址以及要求連線的來源位址，其中來源在 last forwarder 的認知中是為 proxy，而非真正 client，如*圖 4.2.9* 所示。



-*圖 4.2.9* 匿名性-forwarder 的觀點-

4.2.2 效能

目前加密部分的實作是使用 DES 加密系統，有關於加解密這部分的功能，可使用其他方式替換，或是完全不加密。

1. FTP 效能測試：

連線目標 server 為交通大學資訊科學系計算機中心工作站 cissol2 ，IP 位址為 140.113.23.102，client 的 IP 位址則為 140.113.241.21，上傳檔案大小為 15MB。Client 直接連上 server 時，上傳速度約為 9MB/sec.，因此不到兩秒便能完成上傳。實驗方式是中間經過 proxy 與兩部 forwarder(分別為 LFA 與 LFB) ：

Client → proxy → LFA → LFB → Server

Server → LFB → LFA → proxy → Client

<1> 最初版本中，上傳效能非常不理想，使用 Ethereal 捕捉封包分析後發現，由於 proxy 經過多層包裝再送出的 IP 封包，其大小會超過 1500 bytes，因此必須再做 fragment，這會使封包量會倍增，例如，client 送給 proxy 一個 IP 封包，大小是 1496 bytes，經 proxy 處理，加入之後 forwarders 的位址，使得該 IP 封包由原先的 1496 bytes 變大為 1504 bytes，因此送出的封包會分割成兩個，使得效能會降低。而 IP 封包大小是根據系統中 MTU 值來決定，因此我們可以藉由限制 client 發送的 IP 封包大小，例如令 client 送出的封包大小最大限制為 1492 bytes，之後該封包即便在 proxy 增加資訊，仍能避免封包大小超過 1500 bytes，如此則可免除送給 proxy 再轉送出去的封包會有 fragment 的問題發生。

<2> 修正過後 MTU 造成的 fragment 問題後，測試結果為：若是封包完全不加密，

則上傳速度降為約 6.01MB/sec.，而若是再加一部 forwarder，則上傳速度約剩 4.9MB/sec.。若是有加密，則上傳速度降到 1.2MB/sec.，若是再加入一部 forwarder，則上傳速度約剩 0.99MB/sec.。

表 4-3 上傳效能- ftp 測試

正常連線	無加密	加密
9.1MB/sec.	4.9MB/sec.	1.2MB/sec.

2. ICMP 封包測試：

此部分我們測試由 client 送一 ICMP echo request 封包給 server，等待 server 回應 echo reply 封包，紀錄 round-trip time，分別送出封包大小為 64bytes、512bytes 與 1024bytes，接著再修該系統，測試經過一部 proxy 與一部 forwarder 的情況，以及使用原系統經一部 proxy 與兩部 forwarders 的情況。最後數據顯示出封包再經過這些 forwarder 與 proxy 之後延遲的時間。

- Hop 2 hosts

(client→proxy→F1→server, server→F1→proxy→client)

表 4-4 ICMP 測試 -1

Packet Size	**Latency (ms)**
64	0.96
512	1.901
1024	2.4

- Hop 3 hosts

(client→proxy→LFA→LFB→server, server→LFB→LFA→proxy→client)

表 4-5 ICMP 測試 -2

Packet Size	**Latency (ms)**
64	1.649
512	2.2
1024	2.681

第五章　結論

在本篇論文中，我們設計了一個讓接受連線的伺服器無法得知真正使用者身份的系統，並且提高攻擊者藉由 traffic analysis 來判斷通訊雙方身分的難度。

5.1 系統特點

過去單純使用 proxy 做為匿名連線的方法，proxy 本身會得知通訊雙方的身分，並且攻擊者有機會藉由分析 proxy 傳入與送出的封包找出通訊者的身分。因此，若是再藉由多部 forwarder 轉送，以及加密系統，也就是對每個封包再做加密，且將通訊雙方身分的資訊拆開分散給 proxy 以外的 forwarder，可以提高匿名連線系統的安全度。

由於我們設計的方向是著重在不修改上層的應用程式，所有傳送與接收的資料皆以一個個封包的觀點來看，因此位在上層的應用程式，都能夠直接使用本系統而不需另外再做修改。

5.2 外來發展與改進方向

- 封包往返效率問題

系統實作中封包傳遞轉送過程中，在 proxy 與 last forwarder 中皆會建立表格索引，以便回覆 server 回應的封包給真正的 client，當初這麼設計的目的是希望 forwarder 的功能可以放入 router 中來運作，因此是以封包為單位來傳遞資訊，但是封包一個個加解密，對一般的電腦佔用的資源不算低，因此目前系統測試的效能不盡理想。

- 匿名連線佔用的資源

一次匿名連線，至少需要對兩組 proxy 發出請求，過程中至少會產生兩組 session number 與佔用兩部 last forwarder，消耗的資源不小。

- 系統中 proxy 造成的瓶頸

系統中扮演 proxy 的電腦負擔很大，除了必須維護紀錄目前可使用的 forwarder 資訊，又需要處理所有匿名的封包，包含轉送以及回傳封包的資訊，這方面希望可以再修改。

- key 的產生問題

forwarder 上的 key，目前我們的設計是各個 forwarder 各自產生，再告知 proxy，並且若是有變動則再通知 proxy 更新，而若是有 forwarder 被選為當作 last forwarder 時，則在該段時間內不能更新 key，這個部分容易產生漏洞，因此若是由修正由 client 先決定好數把 key，再依決定好的路徑一一告知各個 forwarder，應該更能提升安全度。

- 匿名封包格式

目前在系統中傳送的封包，都只是單純的使用特定 protocol 的 IP 格式封包，雖然封包內部有加密，使攻擊者看不到 IP header 後面的資訊，但是仍可輕易判斷出哪些封包是匿名封包，因此希望能改進這個部分，例如再將封包偽裝成 UDP 格式等方式，來提高封包分析的難度。

參考文獻

[1] Ansari, S.; Rajeev, S.G.; Chandrashekar, H.S., “Packet sniffing: a brief introduction**”,** Potentials, IEEE Volume 21, Issue 5, Dec. 2002-Jan. 2003 Page(s):17 - 19

[2] Behrouz A. Forouzan , “TCP/IP Protocol Suite, 2/e” McGraw-Hill, 2002

[3] Chen-Mou Cheng; Kung, H.T.; Koan-Sin Tan; Bradner, S. , “ANON: an IP-layer anonymizing infrastructure”, in *proceedings DARPA Information Survivability Conference and Exposition*, Volume: 2 , 22-24 April 2003

[4] Marc Rennhard ; Bernhard Plattner, “Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detectionWorkshop On Privacy In The Electronic Society.”, Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society.

[5] Michael J. Freedman ; Robert Morris , “Tarzan: a peer-to-peer anonymizing network layer”, Conference on Computer and Communications Security. Proceedings of the 9th ACM conference on Computer and communications security.

[6] Michael K. Reiter ; Aviel D. Rubin , “Crowds: anonymity for Web transactions”, ACM Transactions on Information and System Security (TISSEC) Volume 1 , Issue 1 (November 1998)

[7] Michael Lucas, Jordan Hubbard, “Absolute BSD: The Ultimate Guide to FreeBSD”, No Starch Press, 2002

[8] Syverson, P., “Onion routing for resistance to traffic analysis”, in *proceedings DARPA Information Survivability Conference and Exposition*, 2003. Volume 2, 22-24 April 2003 Page(s):108 - 110 vol.2

[9] Shields C, Levine BN., “A protocol for anonymous communication over the Internet.”, In: Proc. of the 7th ACM Conf. on Computer and Communication Security. 2000. 33~42. http://citeseer.nj.nec.com/shields00protocol.html

[10] W. Richard Stevens, “UNIX Network Programming, Volume 1 , 2/e”, October 1997

[11] Wright, Gary R. and W. Richard Stevens , “TCP/IP Illustrated-Volume 2: The Implementation.” Reading, MA: Addison-Wesley, 1995.

[12] Anonymous remailer.
http://www.lcs.mit.edu/research/anonymous.html

[13] Lucent personalized Web assistant.
http://www.bell-labs.com/projects/lpwa

[14] 村山公保著, 李傳亮 編譯, “TCP/IP 網路實驗程式設計”, 全華科技, 2001