國立交通大學

理學院網路學習學程

碩士論文

P2P 主動防禦系統之設計與實作

The Design and Implementation of Peer to Peer

Network Intrusion Prevention System

研 究 生：劉建德

指導教授：蔡文能　教授

中 華 民 國　九 十 八　年 五 月

P2P 網路主動防禦系統之設計與實作

The Design and Implementation of Peer to Peer Network Intrusion Prevention System

研 究 生：劉建德　　Student：Jiamn-Der Liu

指導教授：蔡文能　　Advisor：Wen-Nung Tsai

國 立 交 通 大 學

理學院網路學習學程

碩 士 論 文

A Thesis

Submitted to Degree Program of E-Learning

College of Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Degree Program of E-Learning

May 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年五月

P2P 網路主動防禦系統之設計與實作

學生：劉建德　　指導教授：蔡文能

國立交通大學理學院碩士在職專班網路學習組

摘　要

P2P（Peer to Peer）網路自 90 年代末開始崛起，隨著近期 P2P 應用範圍越來越廣泛，相對亦衍生出不少資安問題，諸如著作權、頻寬問題、病毒偽檔、個人隱私等。教育部於 2008 年正式行文各級學校禁止使用 P2P 軟體進行非法檔案傳輸，然而 P2P 軟體大量使用動態連接埠，傳統 Layer3 防火牆已經無法有效阻擋其連線。

本研究針對此問題，以自由軟體為基礎，設計一套傳輸層的防禦的方式（稱之為 T4-terminator，T4 代表 Transport Layer4），並引用兩套應用層的防禦套件（L7-filter、IPP2P），證明能有效禁止 P2P 連線行為。另外，本研究以免費軟體建構出測試環境，分別對三套防禦系統進行效能測試，最後分析其優缺點，提供網管人員管理 P2P 網路之參考。

關鍵字：P2P、IPS、Linux、L7-filter、IPP2P、T4-terminator、Benchmark

The Design and Implementation of Peer to Peer Network Intrusion Prevention System

Students：Jiamn-Der Liu　　Advisor：Wen-Nung Tsai

Degree Program of E-Learning
College of Science
National Chiao Tung University

ABSTRACT

P2P（Peer to Peer）applications have emerged since late 1990s. However, the widespread adoption of P2P applications lately have accounted for some concerns about information security, such as copyright、bandwidth、virus、individual privacy and so on. In 2008, Ministry of Education in Taiwan composed an official document, which stated schools at all levels should forbid the illegal usage of P2P file transfer. However, since P2P applications used dynamic ports in a large amount and thus the traditional Layer3 firewalls were unable to block them.

To resolve this problem, we designed an IPS based on transport layer inspection to drop P2P traffic.We named our IPS as T4-terminator(T4 for Transport Layer4).We also studied two other IPS,L7-filter and IPP2P,which are baesd on Open Source software . Furthermore, we also established a benchmarking environment with freeware, which is used to evaluate the performance of these approaches. The conclusions could offer a reference to MIS people for managing P2P network.

Keywords：P2P、IPS、Linux、L7-filter、IPP2P、T4-terminator、Benchmark

誌　謝

這篇論文能夠順利完成首先要感謝蔡文能老師細心的指導，蔡老師提綱挈領的功力高強，總是能輕易點出我疏漏之處，並指導我許多做研究的方法，令我受益匪淺。再者感謝我老鄰居兼老學長林盈達老師，提供不少寶貴的參考資料，對我實驗設計有很大的幫助。

其次在寫論文這一年，有許多朋友提供我不少協助，在此一併感謝：瑛旗、佳倫、裕峰、阿華、興能；以及同研討室的金庭，彼此砥礪督促，論文才能如期付梓。

最後謝謝我父母的鼓勵，太太及岳父母協助我照顧寶寶，因為你們的支持與包容，我才可以無後顧之憂地完成學業。

目　錄

表目錄

圖目錄

第一章 緒論

P2P（Peer to Peer）技術尚未出現之前，網際網路運作模式多採主從（client /server）架構。在90年代末P2P崛起，打破了這個傳統架構；且隨著近年來P2P的蓬勃發展，它所消耗的頻寬幾乎佔了學術網路或ISP業者大半的資源；此外，尚有智慧財產權等相關資安議題等待處理。

通常ISP業者會採用限制上傳速率的方式以降低P2P的衝擊，然而中小學環境與企業不同，中小學學生處於未成年階段，但是P2P網路上卻充斥著良莠不齊的資訊，這些未經篩選的資訊容易對青少年學生身心造成不良影響。另外，教育部亦明文禁止校園使用P2P軟體，因此網管人員必須設法阻擋P2P連線。

無論採用阻擋或者限制連線速率，都必須先準確的辨識出P2P連線，才有辦法進行管理。P2P剛開始發展時，各軟體大多使用固定的TCP或UDP埠號，因此很容易辨識。隨後P2P很快進化成連結埠供使用者自行設訂的方式，甚至使用動態連結埠或者Web port以躲過偵測，因此使得辨識工作變得困難。本研究以自由軟體為基礎，整合出三種不同的主動防禦方式，能夠有效地阻擋P2P的連線，還給中小學學術網路乾淨的空間。

1.1 研究背景與動機

P2P是種分散式的網路技術，在P2P網路中每台電腦同時扮演Client及Server的角色，不需透過中央伺服器就可以對等地互相分享資源，但此一劃時代的革命技術，雖然帶來了檔案交換的新氣象，卻也產生以下資安及著作權問題：

1、頻寬危機：P2P的搜尋洪流（Query Flooding）及上下載連線速率對等的方式，使得頻寬迅速被佔滿，造成網路壅塞。

2、著作權問題：P2P鼻祖Napster問世不久後，就遭到美國唱片業協會（RIAA）提出告訴，最後被迫終止服務。自Napster敗訴之後，P2P著作權訴訟大戰開始在世界各國點燃。2001年成功大學的學生下載MP3事件，最後財團法人國際唱片業交流基金會IFPI雖然撤告，但卻沒有為學術網路不當使用P2P的問題劃下句點。2005年台北地院對當時經營Kuro[21]的飛行網判決有罪，隔年9月飛行網以3.8億元與IFPI達成民事和解。2005年當Kuro多位員工遭判刑時，全球數碼的EzPeer[20]卻獲判無罪，成為全球唯一被判無罪的業者，EzPeer旋即在2006年6月與IFPI達成和解，並將品牌及服務轉讓給另一家公司，當時號稱是全球

第一家華人合法P2P音樂下載網站。結果在2008年，改版後的EzPeer Plus侵權問題仍舊嚴重，經舉發後第二度遭到台北地檢署起訴，官司目前仍在訴訟中。2009年4月，台灣最火熱的P2P軟體FOXY涉及侵權逾五十八億元，負責人遭板橋地檢署起訴，可見P2P衍生而來的侵犯著作權問題相當嚴重。

3、資料外洩：許多P2P使用者安裝軟體後不懂設定，造成硬碟資料全部被分享出去。新聞報導如軍方演習機密資料、警方筆錄、用戶帳號密碼、情侶自拍照等，均在使用者不知情的狀況下被分享出去，隱私完全暴露在網路上。

4、病毒與後門程式：部分P2P軟體缺乏檔案驗證或者評分回饋機制，因此網路上充斥許多惡意程式，卻是以假檔名的方式欺騙使用者下載，以致使用者在不知情的狀況下中毒，成為殭屍電腦，或遭植木馬盜帳號、密碼者不知凡幾。

目前全球最大的P2P用戶的軟體是KaZaA[24]，但在台灣現今最火紅的P2P軟體非FOXY[16]莫屬，FOXY採用Gnutella[37]分散式協定，它最大特色是快速、容易使用，除了在台灣大受歡迎外，香港、澳門亦有大量的用戶。這波FOXY的熱潮也蔓延到國中小校園網路，因為它找網路上的檔案就像使用Google搜尋一樣容易，並不需要像BitTorrent[17]必須先至相關論壇取得種子，此外操作介面也比eDonkey[19]/eMule[18]更簡單，下載資料又不像Kuro及EzPeer需付費，再加上用戶眾多下載檔案相當快速，於是許多國中小學生在無師自通下，學會了安裝FOXY來下載非法資料。

國中小的學生均處未成年階段，身心發展尚未臻健全，P2P網路上充斥著色情、盜版、病毒等良莠不齊的資訊，學生可能因為好奇而下載成人影片觀看，而導致性觀念偏差。班級電腦也常因為學生隨意安裝P2P軟體，在不知情的狀況下，下載夾帶病毒的檔案而導致系統損毀，讓網管人員疲於奔命；或者下載資料時佔用太多頻寬，拖垮學校網路效能。另外，採用P2P傳輸技術的IM(Instant Messenger)軟體如MSN Messenger及Yahoo即時通也讓學生趨之若鶩；通常教授資訊課程時，許多學生都會偷偷安裝通訊軟體來聊天，學習成效因而大打折扣。

IFPI於2007年11月13日發函教育部，正視學術網路的非法下載的問題；教育部也在2008年正式發文[1]要求各級學校確實清查並移除P2P軟體，所以網管人員必須對學校P2P連線設法加以限制，避免後續衍生的相關問題。

[1] 教育部民國 97 年 5 月 8 日台電字第 0970077254 號函，有鑑於不當檔案傳輸行為對資訊安全造成嚴重危害，要求各校清查非公務用的 P2P 分享軟體，並確實移除。

1.2 研究方法與範圍

本研究設定在國中小學術網路，以目前中小學學生最常使用的P2P軟體作為研究對象。另者因為中小學經費普遍不足，故本研究以Open Source的Linux為防火牆主體。由於P2P軟體具有動態選擇連接埠的特性，傳統Layer3的防火牆無法準確判斷，必須加上Layer7的封包檢驗才能有效辨別連線，所以本研究搭配同屬開放原始碼的L7-filter及IPP2P來達成應用層主動防禦的目標。再者部分P2P軟體提供加密連線，或者具備模糊運算技術，使得應用層封包呈現混亂的內容，無法準確辨識，本研究亦提出一套Layer4的防禦方式（稱之為T4 terminator，T4 for Transport Layer4），同樣也能達到阻擋的效果。

1.3 論文架構

本論文共分六章節，第一章為研究動機與目的，第二章介紹相關背景知識，包括TCP/IP理論與防火牆分析、Netfilter/IPTables的運作方式、L7-filter與IPP2P兩套外掛模組、P2P的演進、應用領域與運作架構。第三章探討與本研究相關的五篇期刊論文，分成以Signature-Based及Behavior-Based兩種不同方式來辨識P2P連線，最後是應用層防火牆的效能改善方法。第四章提出實驗設計，說明如何利用錄製的封包，控制在封閉的環境下進行實驗。第五章分析比較三種不同P2P防禦方式下，系統所呈現的效能及優缺點。第六章為結論及未來的研究方向。

第二章 背景知識

本章介紹與論文相關的技術及背景知識，2.1節介紹TCP/IP的技術及防火牆的種類及特性；2.2節介紹Linux核心及Netfilter與IPTables的相互應用；2.3節為論文實驗採用的L7-filter及IPP2P兩套擴充程式；2.4節為P2P的演進及運用；2.5為常見P2P運作架構。

2.1 TCP/IP 與防火牆

TCP/IP是目前網際網路應用最廣泛的協定，它具備強大的網路連結能力及免費開放的特色，因此越來越多作業系統都將它列為預設的網路協定。無論傳統防火牆或者應用層防火牆，甚至於IDS或者IPS系統，都是架構在TCP/IP的技術上面發展而來的。本節先介紹TCP/IP，接著說明防火牆如何在TCP/IP上面運作。

2.1.1 TCP/IP 基本概念及運作方式

1977年，國際標準組織(ISO，Internetational Standard Organization)開始發展一套標準化之通訊協定架構。於1984年頒布了OSI（Open System Inter-connection Reference Model）基本參考模型，訂定七個層次之功能標準，它規範了通信協定、資料傳輸、封裝及接收的規則。OSI七層由上而下分別為：應用層、展示層、會議層、傳輸層、網路層、資料連結層、實體層。

網際網路的先驅：美國高級研究計畫署(ARPANET，Advanced Research Project Agency NETwork)，最初是採用主機對主機(Host-to-Host)的網路控制協定(NCP，Network Control Protocol)，1974年後TCP/IP（Transmission Control Protocol/Internet Protocol）逐漸取代NCP協定，1983年TCP/IP成為網際網路的標準通訊協定，1990年代中期網路的蓬勃發展，TCP/IP成為全世界被利用最廣的通信協定。TCP/IP模型共分為四層，由上而下分別為：應用層、傳輸層、網路層、連結層。

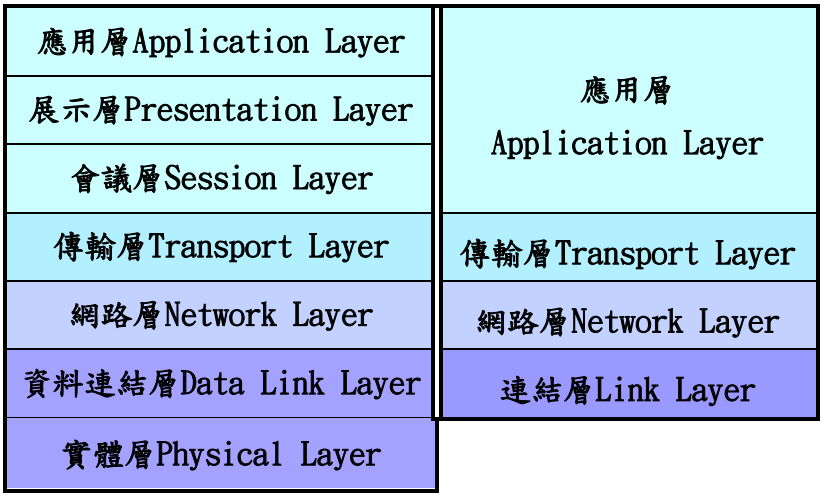

圖 2-1 OSI 與 TCP/IP 模型之對照

上圖為OSI七層及TCP/IP整合為四層之比較，可以看出OSI七層模型，在實作應用上，某些層級常會被整合在一起，因此原理及概念與TCP/IP四層是一樣的。以下就TCP/IP四層之功能與任務簡介：

1、應用層：應用層通訊協定主要在定義點對點之間資料交換的方法，常見的協定如HTTP、HTTPS、SMTP、POP3、FTP、SSH、TELNET等。

2、傳輸層：定義資料傳遞的方法，並規範兩個通訊協定，分別是TCP及UDP，如下說明：

（1）TCP（Transmission Control Protocol）：TCP連線起始會先進行「三交握」的動作，當接受方收到傳送方傳來訊息時，會送出一個確認訊息給對方，傳送方會等到收到確認訊息後，才會繼續傳送資料。傳送過程會檢查資料完整性及區段順序，因此TCP協定是可靠的連線方式。

（2）UDP（User Datagram Protocol）：UDP協定在傳送過程中，不會檢查資料的完整性，允許資料遺失，而且不管資料到達接受端的先後順序，因此如DNS查詢、Multicast、Broadcast、VoIP等，在傳輸資料發生遺漏時，並不會有太大影響，才會使用UDP協定。

由於兩個傳輸協定不同的特性，對一般P2P軟體來說，檔案查詢或者查詢回覆多半使用UDP協定，而檔案交換則採用TCP協定。

3、網路層：用在定義IP（Internet Protocol）、ICMP（Internet Control Message Protocol）、ARP（Address Resolution Protocol）、RARP（Reverse Address Resolution Protocol）四種協定，並定義出封包路由（Routing）的方法，讓不同網域的兩台電腦，可以跨越網路交換資料。

4、連結層：該層就是OSI的Data Link Layer，也就是網路最基礎的建設，包括乙

太網路(Ethernet)、光纖 (Fiber)、無線網路(Wireless)、訊框傳送(Frame Relay)、點對點實體網路 (PPP)，連結層最主要的目的在傳送及接收實體層所傳送的光電訊號。

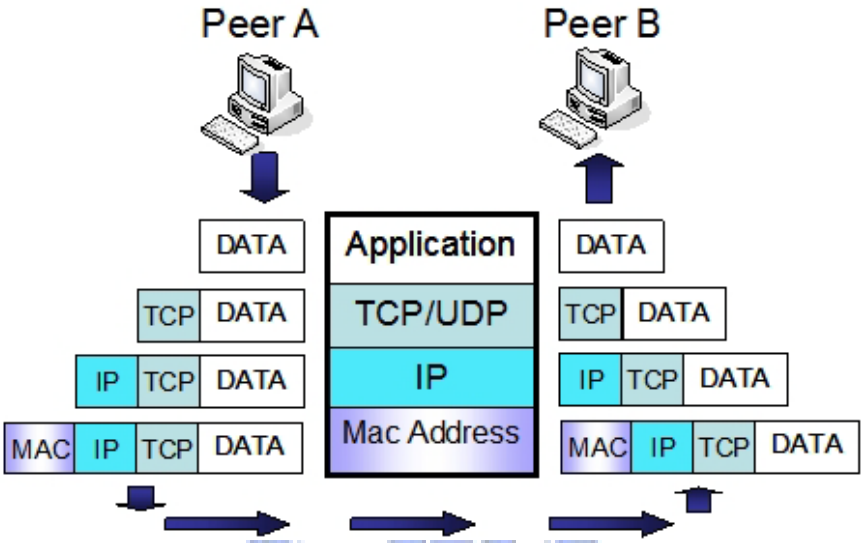

圖 2-2 封包傳遞的過程

上圖為封包(Packet)傳遞示意圖，當 Peer A 要和 PeerB 聯絡時，會把應用層資料交給下一層來處理。傳輸層接到應用層交下來的資料，會把它視為一個 Payload Data，然後幫它加上一個傳輸層的控制表頭，接著再往下一層傳遞，最後由連結層封裝後，把封包傳送出去。對方收到資料後，反覆剝去表頭及向上傳遞 Payload，最後到達應用層，完成了的交談。

2.1.2 防火牆技術分類

防火牆依過濾技術可分為封包過濾（Packet Level Filter）及應用層過濾（Application Level Filter）兩類，各有其使用時機與優缺點，分述如下：

1、封包過濾防火牆：即為傳統的Layer3防火牆，檢查最小單位為「一個封包」，單純處理封包IP及PORT的資訊，效能極高且應用廣泛，但缺點為檢查範圍只有一個封包，因此無法執行精準的過濾動作。

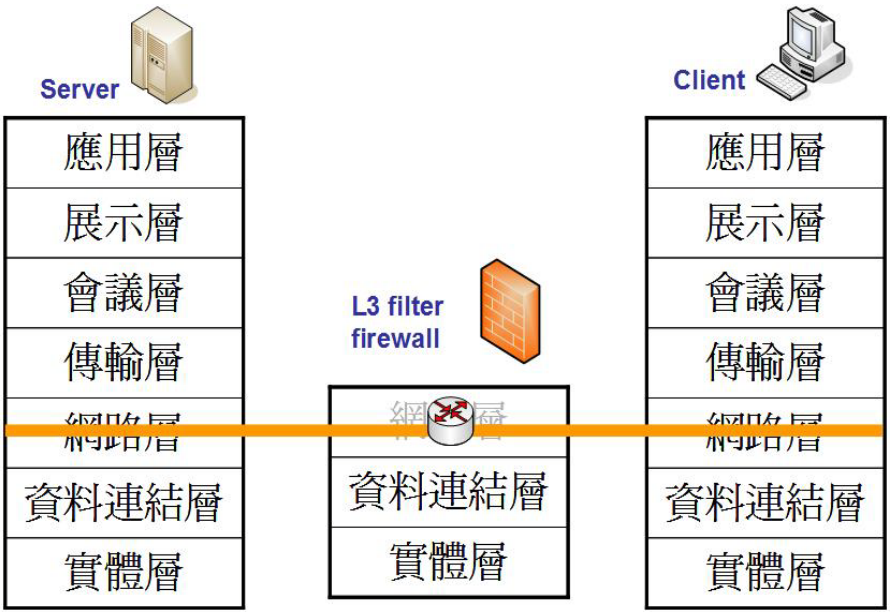

圖 2-3 封包過濾式 Layer3 防火牆

2、應用層防火牆：因為Layer3防火牆無法完整過濾進出Router的資訊，因此發展出Layer7的防火牆，通常此類防火牆會具有狀態檢測（Statefull Inspection）的機制，讓filter process能夠完整分析連線資訊，達成安全控管的目的。但此類防火牆最大問題在效能較低，而且過濾協定必須定期更新，才能辨識各式各樣的連線。

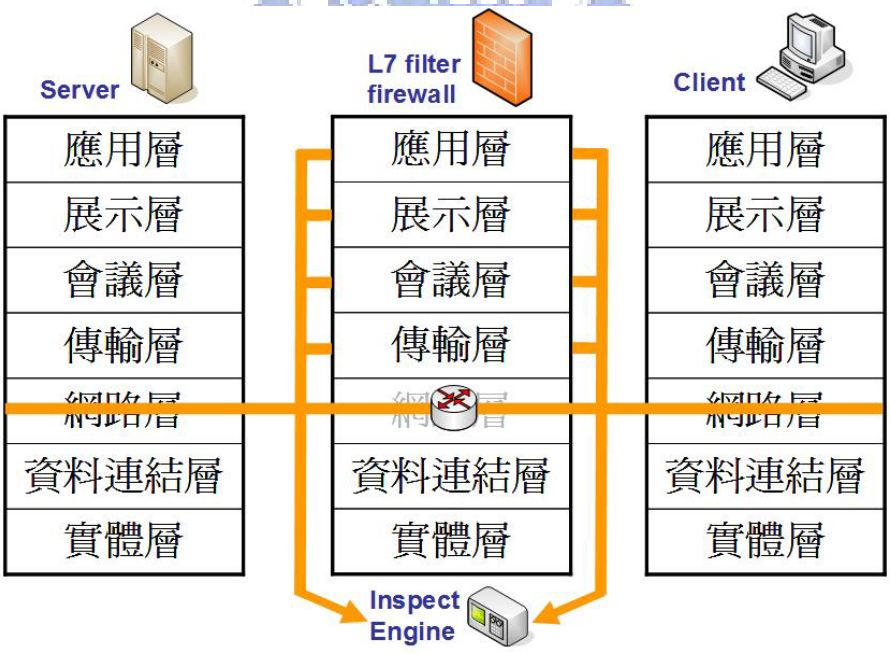

圖 2-4 狀態檢測式 Layer7 防火牆

2.1.3 IDS / IPS

IDS(Intrusion Detection System)特色在於「分析」與「警示」就如同 sniffer 軟體一樣監聽流過的封包，當發覺有不正常流量，旋即發出警訊通報系統管理者。但通常在發出警訊之時，病毒或者駭客正逐步攻陷了整個網域，等到管理者收到通報時，可能傷害已經造成了。在 OpenSource 的 IDS 當中，最著名的就是稍後會提及的 SNORT[28]。

IPS（Intrusion Prevention System）與 IDS 最大的不同在於 IPS 能夠在偵測到入侵的同時進行防禦。IPS 最大的特色就是「深層檢測（deep inspection）」和「在線模式（in-line mode）」，IPS 通常採用應用層的防火牆，以便偵測更完整的入侵資訊，一旦發現有異常網路行為，即可丟棄攻擊封包。

無論執行 IDS 或 IPS 都必須降低誤判（false postive）及漏判（false negative）的問題，此外如果 IDS / IPS 主機運算能力無法負荷高速網路，勢必 拖垮整個連線效能。

2.2 Netfilter/IPTables

Linux Kernel 第一代的防火牆是移植自 BSD 的 ipfw，後來在 Kernel 2.0(1996 年)後，開發了 ipfwadm 的工具程式，提供使用者更容易設定過濾規則。到了 Kernel 2.2（1998 年）推出了第二代的 ipchains，大幅提升防護功能。Kernel 2.4 之後，重新設計防火牆架構，開發出第三代 Netfilter/IPTables 模組，這版本的防火牆開始提供狀態檢測的機制，而且因為它開放程式碼，讓許多工程師投入外掛模組的編寫，其強大、穩定、高效能的特性，比起商用防火牆有過之而無不及。

Kernel Version	Firewall Engine
Kernel 2.0	ipfwadm
Kernel 2.2	Ipchains
Kernel 2.4 / 2.6	Netfilter / IPTABLES

表 2-1 核心版本與對應之防火牆機制

2.2.1 Netfilter 模組

如圖2-5所示，Netfilter[14]以模組的型態存在Linux核心中，封包進入核心後，提供5個檢查點（5 Hook points），分別是PRE_ROUTING(＃1)、LOCAL_IN(＃2)、FORWARD(＃3)、LOCAL_OUT(＃4)、POST_ROUTING(＃5)。每一個Hook都可以被上層的許多模組註冊使用，註冊時可以指定優先順序，系統會依照順序決定過濾的順序。

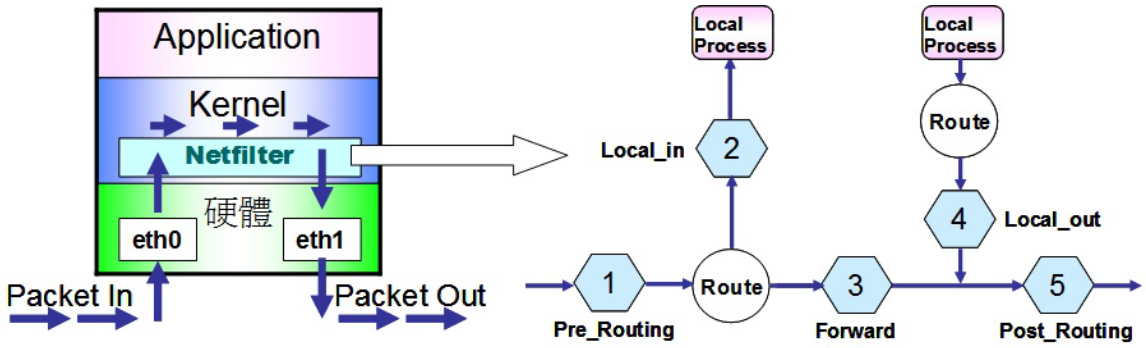

圖 2-5 Netfilter 在核心的位置及 5 個檢查點

PRE_ROUTING：封包在進入系統的檢查點；POST_ROUTING：封包離開系統的檢查點；LOCAL_IN：封包進入本機，給上層程式處理的檢查點；LOCAL_OUT：本機程式要傳送資料出去的檢查點；FORWARD：是當封包要做轉送時會經過的檢查點。

當有封包進入檢查點時，檢查點會啟動註冊的決策模組，如Connection Tracking、L7-filter、IPP2P等。同時也Netfilter 提供五種封包狀態回傳值，交給檢查點的決策模組使用，再藉由模組回傳之結果來決定封包的命運。五種回傳值如下所示：

1、NF_ACCEPT ：允許該封包送往下個Hook。

2、NF_DROP：拋棄該封包，不再繼續傳送。

3、NF_STOLEN：由檢查函式接管封包，不再繼續傳送。

4、NF_QUEUE：將封包放至佇列（queue）中，由上層的Userspace應用程式處理。

5、NF_REPEAT：封包再一次進入這個Hook檢查。

2.2.2 IPTables 模組

Netfilter位在下層系統核心，IPTables則是提供使用者操作的上層防火牆程式。如圖2-5及上個段落所述明，IPTables提供使用者向Netfilter架構中的hook點作註冊動作，所以每個通過核心的網路封包便可以送到IPTables，藉使用者定義IPTables規則來決定封包命運。目前IPTables內建了4個TABLE，各個TABLE的功能用途簡介如下表：

TABLE	Chain	Function
Filter	INPUT	執行封包過濾，可過濾TCP、UDP、ICMP、MacAddress封包
	FORWARD	
	OUTPUT	
Nat	PREROUTING	NAT（Network Address Translation）含SNAT及DNAT，可以提供IP偽裝，或者當成企業DMZ防火牆
	POSTROUTING	
	OUTPUT	
mangle	PREROUTING	可經由該機制修改封包內容
	INPUT	
	FORWARD	
	OUTPUT	
	POSTROUTING	
Raw	PREROUTING	加快穿越防火牆的速度，提高效能
	OUTPUT	

表 2-2 IPTables 內建之 TABLE 及功能

2.3 L7-filter 與 IPP2P

Netfilter/IPTables封包分析處理預設停留在Layer3、Layer4層級，但是現在網路應用軟體大量運用dynamic-port，因此必須將封包辨識提升至Layer7，才足以應付與日具增的P2P程式。L7-filter[12]及IPP2P[13]是本論文實驗所使用的Netfilter擴充模組，可以辨識常見的P2P連線。

2.3.1 L7-filter classifier

在2003年前，L7-filter[12]團隊原本在開發Linux kernel的QoS（Quality of Service）延伸模組，到2003年底才開始投入Netfilter的Layer7延伸過濾模組的開發，2005年初推出了v1.0版，截至本論文寫作期間，最新版本為v2.21。除了Kernelspace版本外，2005年12月也推出Userspace版本，目前為v0.11版。

L7-filter是透過V8 regular expression[38]字串比對方式來辨識封包，預設比對連線前10個封包或前2KBytes，此預設值可以讓使用者依需求自行更改。目前協定運作效果被區分為：Great、Good、Ok、Marginal、Poor五種等級，會如此區主要是因為有些協定(如HTTP、FTP、SMTP)是公開的，此類封包內容的關鍵字可以完全掌握。但是有些協定是不公開的，對於此類的協定只能藉由分析封包出現的關鍵字來做pattern，因此比較容易發生誤判，所以運作效果不是很好的便會歸到marginal或poor這兩類。

以msn為例，當我們使用msn帳號登入時，msn client會送出「*VER1 MSNP14 MSNP13 CVR0*」的封包字串給central server。歸納出各類網路程式每次送出的封包含哪些字串後，我們可以寫出一個pattern file，提供關鍵字讓L7-filter判斷，提供IPTables做後續的處理。

Msn-login
^ver [0-9]+ msnp[1-9][0-9]? *cvr0\x0d\x0a$
Msn- filetransfer
^ver [-~]*msnftp\x0d\x0aver msnftp\x0d\x0ausr
Yahoo Messenger
^(ymsg|ypns|yhoo).?.?.?.?.?.?.?[lwt].*\xc0\x80

表 2-3 L7-filter 的 pattern file 範例

上表列舉三個官方提供的定義檔範例，目前L7-filter官網提供辨識的網路協定多達109種，此外還能夠辨識16種檔案格式及2種malware病毒。若使用者覺得官方網站提供的寫法不妥，或是有其他網路協定尚未被納入規範，均可自行撰寫pattern file以供判斷。另外L7-filter已經結合了Connection Tracking的模組，可以透過「-j MARK」參數配合tc[1]指令方便地達到頻寬管理之目的。表2-4為對edonkey連線進行Qos的範例，set-mark的參數可隨意設定，之後便可交由tc管理。

[1] tc 指的是 traffic control，為 Linux Qos 常用的 Userspace 工具

```
iptables -t mangle -A POSTROUTING -m layer7 --l7proto edonkey -j MARK
--set-mark 1
```

表 2-4 L7-filter 的 QoS 範例

2.3.2 IPP2P classifier

IPP2P[13]是另一套能夠辨識Layer7封包的自由軟體，顧名思義即針對P2P連線所設計。此套件的運作方式和L7-filter相似，都是掛在Netfilter/IPTables上的延伸擴充模組，亦採用String-match的方式進行比對。不同的是L7-filter是有一個額外的pattern file，Netfilter會依照IPTables的指令去patter file呼叫適合的定義檔比對；但是IPP2P則是將String-match的內容以16進位碼直接寫入程式碼中，安裝時直接編譯到模組內。

相較於L7-filter的更新快速，IPP2P維護頻率顯得少很多，本論文寫作時官方網站的最新版本尚停留在2006年9月的v0.8.2版，支援的kernel version較少，太新或太舊的核心會造成編譯失敗。官方版本支援辨識的P2P協定有9種，若想要自行新增協定，必須先修改source code，再重新編譯後才能夠執行。

IPP2P不具有Connection Tracking的功能，連線比對採per packet match的方式，不像L7-filter只比對連線最前面的數據。也因為這特性，使得IPP2P不容易對整個連線進行頻寬管理。如下表所示，必須額外藉助CONNMARK模組來儲存（Rule＃03、Rule＃04）及還原（Rule＃01、Rule＃02）整個連線後，才能交由tc指令管理。

```
01# iptables -t mangle -A PREROUTING -p tcp -j CONNMARK --restore-mark
02# iptables -t mangle -A PREROUTING -p tcp -m mark ! --mark 0 -j ACCEPT
03# iptables -t mangle -A PREROUTING -p tcp -m ipp2p --ipp2p -j MARK
    --set-mark 1
04# iptables -t mangle -A PREROUTING -p tcp -m mark --mark 1 -j CONNMARK
    --save-mark
```

表 2-5 IPP2P 的 QoS 範例

2.4 P2P 的演進與運用

在1998年就讀於美國東北大學一年級的Shawn Fanning開發出Napster，同時也開啟了P2P技術的濫觴，隨後Napster版權官司的敗訴，並未停止P2P技術的發展。剛開始的P2P軟體多半使用固定的port來做資料傳送，2000年之後的P2P大量使用動態port，甚至採用連線加密技術以躲避偵測。P2P應用上也越來越多元，包括檔案傳輸、串流影音、即時通訊、線上遊戲、分散計算等，說明如下：

P2P Categories	Application Programs
P2P File Sharing	Foxy(Gnutella)、BT、eDonkey/eMule、Kuro、EzPeer、KaZaA、FastTrack、WinMX
Instant Message / VoIP	MSN、Yahoo Messenger、Skype、Google Talk、QQ
P2P Streaming	PPLive、PPStream、Sopcast、TVAnts、TVUPlay
P2P Distributed Computing	SETI@home、Datasynapse、Intel philanthropic P2P program

表 2-6 P2P 應用領域

1、P2P file sharing：運用於檔案傳遞，傳送類型多半為影片、音樂、圖片、軟體等，此類型的連線通常佔用ISP或學術網路最大量的頻寬，且最容易侵犯到智慧財產權。

2、Instant Message / Voice over IP：早期線上交談必須透過BBS電子布告欄或者線上CGI聊天室，後來ICQ、AOL、Microsoft NetMeeting普及後，才逐漸轉型為P2P即時通訊。目前MSN和Yahoo即時通是最受台灣學生歡迎的通訊軟體，而Skype採用GIPS iSAC and iLBC codecs編碼技術，採用應用層路由（Application Layer Routing）的輔助方式進行Voice over IP，穩定清晰的音質及便宜的通話費廣受業界的喜愛。

3、P2P Streaming：串流視訊需大量且即時的資料，才能提供穩定的畫面品質。一般VOD（Video on Demand）或MOD（Media on Demand）是採用中央伺服器提供影像，當使用者越多，會造成server的負載增加。而P2P Streaming技術則是運用眾多peers間的視訊串流互為sources，將central server的流量分散至觀看同一節目的peers，然後peers之間再共享串流，因此當某節目同

時越多人觀看，反而會得到越穩定的收看品質。

4、P2P Distributed Computing：P2P分散式計算計算最有名的就是SETI@home，SETI（Search for ExtraTerrestrial Intelligence）是美國NASA進行的國際性科學計畫，台灣亦有mirror站台[26]，可至網站免費下載程式，當螢幕保護裝置啟動時，電腦會進行協同運算，來分析由外太空傳來的無線電波協助尋找外星智慧。

2.5 P2P 的運作架構

大多數的網路服務都採用主從（client/server）架構，如FTP、WEB等服務，當越多client端加入連線，server會耗費更多運算資源及網路頻寬，而且client只能被動的接受server所提供的資源。但是P2P的技術打破了這個藩籬，每個peer都是對等的，同時扮演client以及server的角色，當越多節點加入這個網路，每個節點所能夠得到資源會越多。P2P運作架構可分成三類，分別是集中式、分散式、混合式，將於本節敘述。

2.5.1 集中式架構（Centralized）

Napster是這個系統架構的代表，此類P2P系統通常會使用固定的TCP port。新節點的加入首先必須連接中央索引伺服器（index server），然後回報其擁有的檔案列表，隨後各節點都透過詢問中央索引伺服器來得知擁有此檔案的使用者資訊，再依此資訊，直接向檔案所在端點要求下載。

集中式架構的優點在於實作上較簡單，搜尋檔案快速有效率；而其主要缺點在於一旦中央索引伺服器故障，整個系統便無法運作，而且伺服器必須處理大量客戶端的請求，容易形成系統運作的瓶頸。

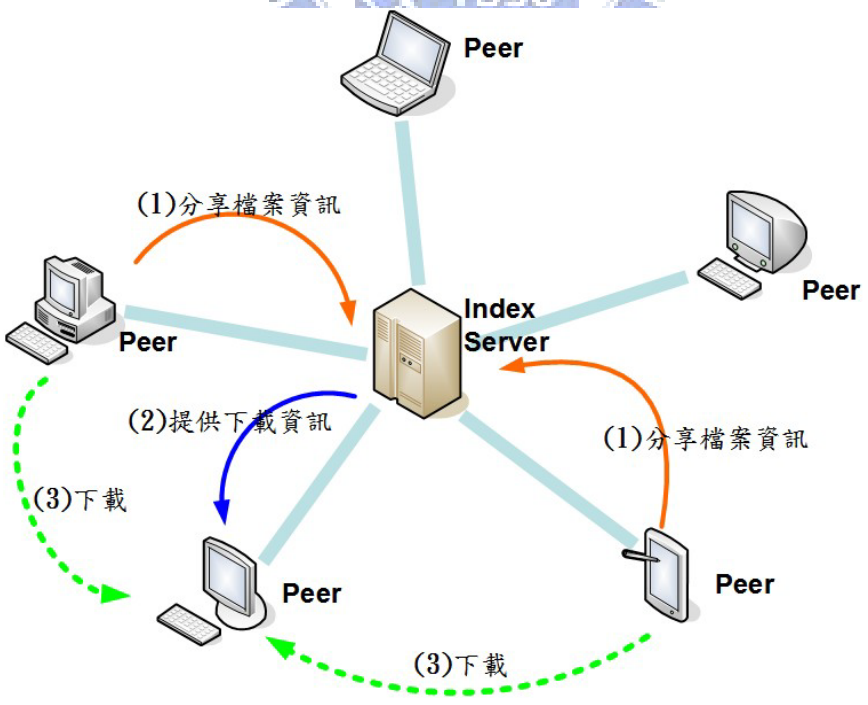

圖 2-6 集中式 P2P 架構

2.5.2 分散式架構（Decentralized）

分散式架構的代表是Gnutella[37]，為改良第一代架構的缺點及違法問題，此類P2P系統普遍採用dymanic port，或直接利用HTTP作為通信協定，以躲過防火牆偵測。另外也不需要任何中央伺服器，所有參與連線的端點同時扮演server與client的角色，地位完全對等。Gnutella系統定義了四種訊息格式：ping、pong係用來確認身份，而query及query-replies則用來搜尋檔案。

新的端點想要加入Gnutella網路，會先連結到鄰近任意節點，並且發出Ping訊息來表明自己身份，被連結到的節點會回應Pong訊息，內容包括IP、Port及分享的檔案數量及大小，同時被連結的節點會將該Ping訊息以遞迴（Recursion）的方式繼續往它的鄰居節點廣播，直到TTL(time-to-live)歸零。也因為這個方式，很容易造成搜尋洪流（Query Flooding），所以如何定義TTL值，或者限制遞迴的深度，以降低頻寬被佔滿的狀況，是分散式架構最大的挑戰。

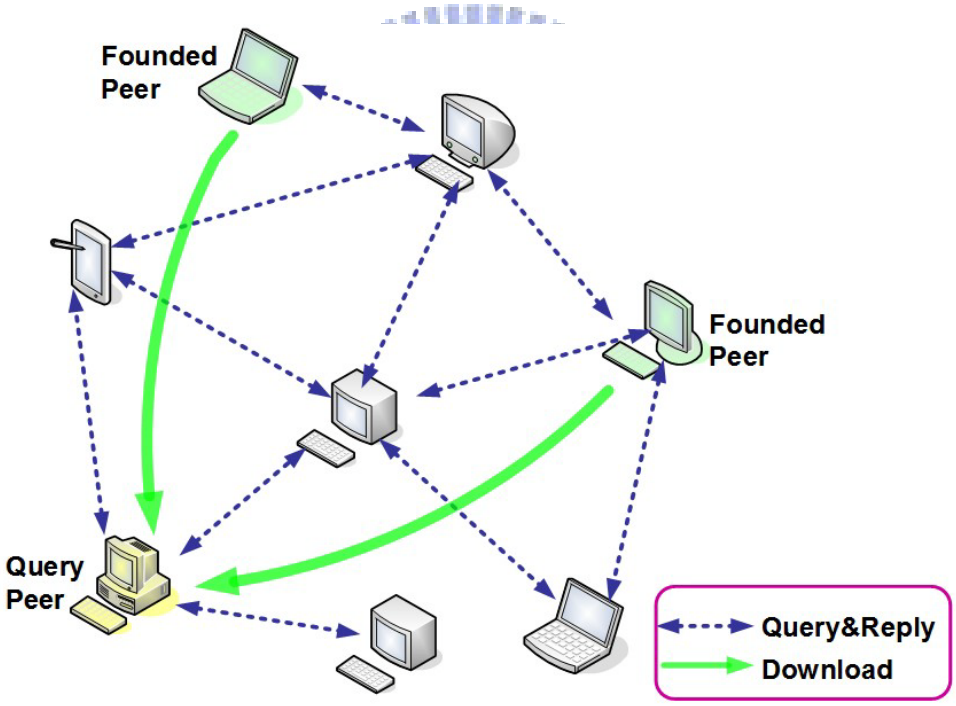

圖 2-7 分散式 P2P 架構

2.5.3 混合式架構（Hybrid）

混合式架構(如eDonkey[19]、FastTrack[23]、WinMX[22])是以分散式P2P為基礎，另外參考集中式P2P的特點，增加了Supernode這個角色。通常具有較大的頻寬及運算能力的節點會被選為Supernode，負責提供以它為中心之小型網路的各項服務，包括檔案索引及暫存等，因此讓整體網路傳輸更有效率。

為了避免防火牆阻擋，部分混合式P2P程式，如FastTrack、WinMX會採用連線加密技術，而eMule[18]在v0.47b版發表了模糊運算技術，讓封包看起來變成毫無規律的內容，這些方法都增加了封包辨識的困難度。

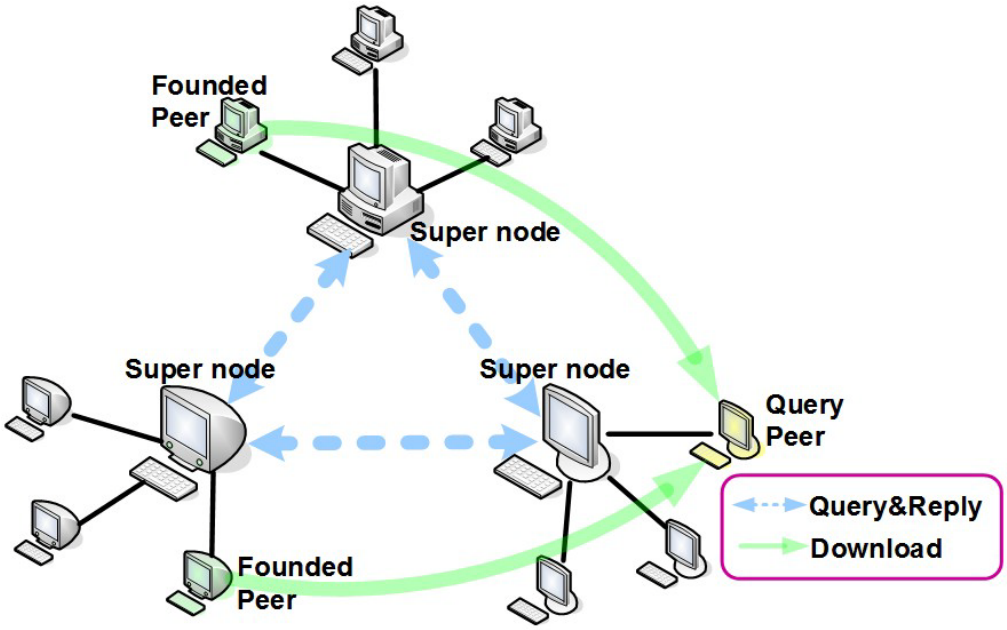

圖 2-8 混合式 P2P 架構

第三章 相關研究

本章就近年來諸多 P2P 辨識技術文章，擷取與本研究相關的文章討論。分別於 3.1 節討論封包特徵的辨識法；3.2 節中以 P2P 常見連線行為作為辨識法；3.3 節為改善 Linux 防火牆的過濾效能。

3.1 封包特徵辨識法之研究(Signature-Based)

P2P檔案分享軟體越來越盛行，佔用的網路頻寬越來越大，對於網管人員或是ISP業者必須先準確的辨別P2P流量，才能對P2P連線進行管理。因為P2P技術的進步，且應用越來越廣泛，以往藉由Port-Based來判斷P2P流量已經不夠準確，因此必須透過應用層封包分析技術(Application protocol analysis)才能達到辨識效果。

3.1.1 應用層特徵(Application Layer Signature)辨識

Subhabrata Sen等人[1]就目前熱門的P2P軟體，如Gnutella、eDonkey、BitTorrent、KaZaA、DirectConnect等，分析這些軟體所使用的通訊協定及封包的內容，歸納出各種應用程式特徵（Application Signature），藉以判斷此封包是否為P2P連線。

但是在大量封包中進行字串比對需要大量的計算時間，為了能夠在高速網路環境下快速地進行辨識，將所定義的P2P軟體特徵比對方式分成兩個部分：

（1）固定位移比對（Fixed Offset Match）：將P2P軟體特徵和TCP封包內容中，特定的Byte進行比對，比對方式為16進位具體代碼。

（2）變動位移比對（Variable Offset Match）：透過正規表示式（Regular expression）來比對。

各連線封包特徵分析及比對方式經歸納如下表：

連線軟體	封包特徵及比對方式
Gnutella	※ 封包開頭字串為“GNUTELLA”、“GET”、“HTTP” ※ 若封包開頭字串為“GET”、“HTTP”，則其內容必包函含下面字串“User-Agent: <name>”、“UserAgent:

	<name>”、“Server: <name>”
eDonkey	※ 封包開頭的第一個Byte為eDonkey marker，其值固定為0xE3 ※ 封包的第二到第四個表示封包內容的長度
DirectConnect	※ 封包開頭的第一個字元為“$”，最後一個字元為“\|” ※ 在“$”緊接一個以空白字元結束的字串，此字串為DirectConnect命令。
BitTorrent	※ 封包的第一個Byte，其值為0x19 ※ 接下來的19個Bytes，為一個長度19的“BitTorrent protocol”字串
KaZaA	※ 封包開頭字串為“GET”、“HTTP”，且封包中包含”X-Kazaa”字串

表3-1 常見P2P連線的封包特徵

將定義的P2P軟體特徵在Gigascope[1]中進行驗證，結果證明應用層特徵可精確的辨別出P2P軟體所產的流量，且誤判率(False Negative)均低於10%，對於DirectConnect甚至能做到零誤判的程度。且與Port-Based方法相比，Signature-Basd對於愈常使用non-standard port的P2P軟體有愈佳的辨識效果，尤其對於KaZaA高到3.47倍，如表3-2所示。

Protocol	All Connections	
	Port-based (MB)	Signature-based (%)
Gnutella	487.12	145
Kazaa	548.41	347.38
DirectConnect	2000.75	163.45
BitTorrent	54444.67	90.97
eDonkey	2149.84	102.37

表3-2 Port-based與Signatured-based辨識結果之比較

資料來源：Subhabrata Sen，et al. [1]

[1] Gigascope為一高速網路流量監測器，所能監測的網路流量最高可到達 OC-48（2.4Gbps）

3.1.2 SNORT 入侵偵測系統規則辨識

SNORT[28]這套入侵偵測系統(IDS,Intrusion Detection System)早在1988年就誕生了，為GPL & OpenSource模式的入侵偵測系統。2005年被防火牆大廠Check Point買下後，從此不再以GPL授權釋出，但仍然維持OpenSource模式。使用者可至官方網站免費下載安裝軟體，但是最新的規則庫必須花錢購買，若單純個人測試，官網仍免費提供比付費資料庫晚一個月的規則下載。

SNORT支援多種平台，又具備OpenSource模式，再加上規則編寫容易，擴充性極佳，所以廣受歡迎。[2]採用此套軟體，先以WinDump分析P2P程式的封包特徵，再以自編之SNORT rules來驗證其準確性。文中歸納P2P連線的週期行為大致上歸類為以下四點：

（1）Discovering and Booting：使用者開啟軟體登入網路，同時尋找線上節點。
（2）Sharing：送出自己分享的檔案列表。
（3）Querying and lookup：送出檔案請求查詢，網路上有此檔案的節點會回應。
（4）Downloading：peer to peer交換檔案。

以上述方法，分別對OpenNap、WinMX、FastTrack三套P2P程式進行分析。OpenNap偵測起始狀態及登入Server的封包訊息，FastTrack跟WinMX連線訊息有使用加密技術，但仍然可針對一些特徵提出警訊。說明如下：

1、OpenNap[25]（Open Source Napster Server）：

該程式即為Napster的修正版，同樣有central servers的架構，但是這套程式與Napster,Inc.無關，所使用的技術也不同。這是一種Open Source平台，適用於Unix及Windows系統，使用者可免費架設central server，提供WinMX及支援OpenNap的用戶登入。

使用時，clients與servers之間的溝通均為明碼傳送，因而整個連線行為及封包特徵較容易分析。SNORT規則共分六條，整理如下表：

編號	規則名稱	檢驗字串	位移比對及大小
1	catch the server welcome answer	〝VERSION〞	offset:4;depth:12
		〝SERVER〞	offset:11;depth:18
2	catch the client login message	〝WinMX〞	offset:4;nocase
3	catch the name of files shared by the client	〝|22 43 3a 5c|〞	offset:4;depth:9

4	Alert on query submit	〝 FILENAMECONTAINS〞	offset:4;depth:18
5	alert on download requests	〝GET〞	offset:0;depth:3; dsize:3
6	Alert on upload requests	〝SEND〞	offset:0;depth:4; dsize:4

表 3-3 SNORT Rules for OpenNap

2、WinMX Peer Network[22] （WPN）：

WinMX連線協定較OpenNap複雜許多，它不但支援登入OpenNap的伺服器群組，本身也提供一套WinMX對等網路傳輸協定，而且檔案搜尋、管理的功能非常強大。WinMX本身的網路傳輸技術採用decentralized以及encyrpted messages的方式，此外連線類型還區分兩種讓用戶選擇：

（1）Primary connections：適合給高頻寬使用者，具較佳搜尋效率及穩定性，使用該方式連線即可成為WPN的中繼cache server，還可開設聊天室。

（2）Secondary connections：適用低頻寬或者用戶，或者用戶連線被防火牆阻擋時，採用這方法將透過其他primary connections的cache servers來進行路由。

WinMX創辦公司Frontcode於2005年接獲美國唱片業協會（RIAA）要求在WinMX加入關鍵詞過濾功能，令使用者無法搜尋受版權保護的收費作品。在面臨法律訴訟的壓力下，Frontcode公司終於在2007年6月底將WinMX官方網址關閉，但是目前仍有不少私人架設的論壇提供升級及討論的服務。

由於WinMX Peer Network採TCP加密技術傳輸，因而增加辨識的困難。作者提出兩條SNORT規則，第一條偵測primary connection中的某固定大小為145 bytes的UDP封包，第二條僅偵測secondary connection是否建立，如下表說明：

編號	規則名稱	規則內容
1	alert on supernodes that answer to joining peer	msg：〝WPN Primary connection detected〞；dsixe：145；classtype：policy-violation
2	alert on secondary connection contacting WPN cache servers	msg：〝Probably WPN Secondary connection detected；flow：

		established；flags：PA；tag：host，1，packets，src″

表 3-4 SNORT Rules for WinMX

3、FastTrack[23]

FastTrack亦採用decentralized以及encrypted messages來連線，並且使用supernode的方式提供一般使用者作為indexing server。除此之外，FastTrack連線具有突破傳統防火牆封鎖的修正技術，透過supernode的仲介，讓防火牆背後的storing peer能提供檔案讓外面的requesting peer下載。至於下載請求則採用與HTTP非常類似的語法，再加上下載檔案前的溝通都採加密連線，增加不少判斷的困難。

採用FastTrack協定的軟體中，作者以最著名的KaZaA[24]為例。列舉出兩條規則：第一條規則判斷supernode回應使用者的UDP flooding；第二條判斷下載檔案時未加密的HTTP/1.1的字串，並加入搜尋「KazaaClient」的字串降低誤判率。如下表說明：

編號	規則名稱	檢驗字串	位移比對及大小
1	alert on supernodes that answer to joining peer	〝|28|″	offset：0；depth:1
		〝|4b 61 5a 61 41 00|″	offset:11；depth:17
2	Alert on sending a positive response to a request for a shared file	〝|48 54 54 50 2f 2f 31 2e 3120 32 30 30 20 4f 4b|″	offset:0；depth:15
		〝KazaaClient″	session：printable

表 3-5 SNORT Rules for WinMX

[2]並未針對文章中提出的規則在真實網路中進行判斷準確率的檢驗，但作者運用了規則編寫容易，且擴充性佳的SNORT作為P2P連線偵測系統。網管人員可依照該方法自行定義規則，用來辨識其他P2P連線。

3.2 連線行為辨識法之研究(Behavior-Based)

應用層封包辨識的效果是P2P連線辨識法中最準確的，但是這種方法最大的問題在於（1）應用層防火牆處理效能較低（2）需隨時更新辨識資料庫。因此本節的研究以P2P常出現的連線行為來進行辨識，盡量能在不減低防火牆負擔的狀況下有效的辨識P2P連線。

3.2.1 傳輸層（Transport Layer）的辨識

第一代P2P軟體通常使用固定port進行連線，但隨著技術進步，現今P2P連線已經很難單純地透過固定連接埠來辨別。3.1節介紹Payload Analysis的P2P辨識方法，這種方式優點為辨識率準確，但仍有一些限制：

（1）Throughput：Layer7防火牆因為需大量封包比對，與僅進行IP及Port比對的Layer3防火牆相比，負擔相對提高。

（2）HTTP Request：許多P2P協定使用HTTP協定傳檔，使我們不容易分辨P2P或WEB連線，通常Payload Analysis只分析連線最前面8~16bytes的封包，容易發生漏判（False Negative）的狀況，如”HTTP/1.1 206 Partial Content”的字串都可能在HTTP或P2P連線中產生。

（3）Encryted：message加密或先透過SSL的連線都將使Payload Analysis失效。

（4）Others P2P protocols：P2P應用廣泛，封包辨識法無法保證能成功辨識新的協定，必須隨時更新定義檔。

為改善這些缺點，[3]藉由觀察P2P常見的行為樣式（connection pattern），提出一套PTP演算法（P2P Traffic Profiling Algorithm），利用傳輸層的辨別，可以不需檢查任何封包的內容，快速地檢驗出P2P flow。PTP演算法包含兩個原則，分別是TCP/UDP IP pairs及{IP,Port} pairs，分述如下：

1、TCP/UDP IP pairs：

大部分的P2P軟體都會同時使用TCP及UDP傳輸協定，通常利用UDP來進行traffic control、queries或query-replies，TCP則用來下載或上傳資料。除了P2P連線之外，其他會同時用到TCP/UDP來進行傳輸的軟體很少，表3-6列了此類軟體的TCP/UDP port，提供我們進行辨別時，將這些例外port摒除，避免造成誤判。

Ports	Application
135,137,139,445	NETBIOS
53	DNS
123	NTP
500	ISAKMP
554,7070,1755,6970,5000,5001	streaming
7000, 7514, 6667	IRC
6112, 6868, 6899	gaming

表 3-6 同時使用 TCP/UDP 的非 P2P 應用程式

資料來源：Thomas Karagiannis，et al. [3]

2、{IP,Port} pairs：

近來P2P網路朝向分散式（distributed network）或混合式（hybird network）網路模式發展。無論哪種模式，當一個新節點要加入網路時，通常會跟一些cache servers進行連線，這些節點稱為superpeers/ supernodes，能夠把自己暫存的網路節點列表，傳送給新節點，以加快連線速度。當新節點與superpeers建立連線後，superpeers同時也會將新節點的連線資訊傳送出去，也就是將IP及連接埠（{IP, port} pair）廣播至整個網路中，以便讓網路其他的節點可以與其通訊。整個連線過程如下圖所示：

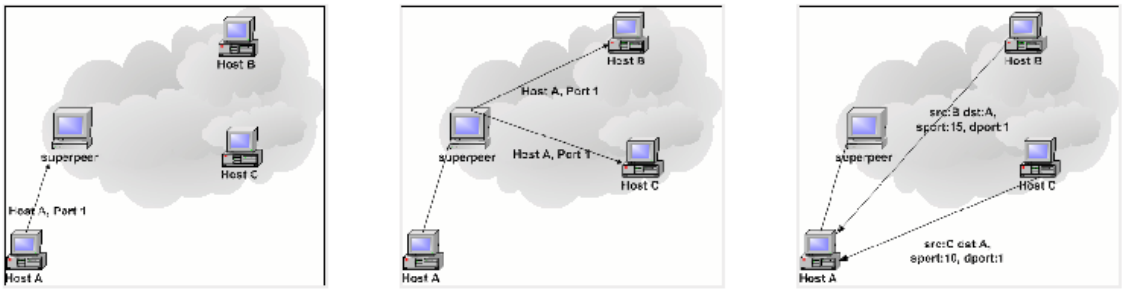

圖3-1 {IP,port}pairs of Host A

N個不同IPs與A連線，同時也會有N個不同的foreign used ports用來與A連線

資料來源：Thomas Karagiannis，et al. [3]

利用上述的兩種原則，PTP演算法提出的P2P判斷步驟如下：

（1）辨識source-destination IP pairs同時使用TCP/UDP傳輸協定。

（2）摒除表3-6少數同時使用TCP/UDP的程式，如DNS、NETBIOS、IRC、gaming、streaming等，假設發現某連線同時使用TCP/UDP pairs，而且s-d ports沒在表中，我們可以合理懷疑該連線為P2P flow。

（3）接著檢驗其{IP,Port} pairs，如果超過20個連線，且持續5分鐘以上，即可斷定為P2P連線。

最後將PTP演算法與封包比對法相比，證明這種non-payload驗證方法相當有效，其實驗結果如下：

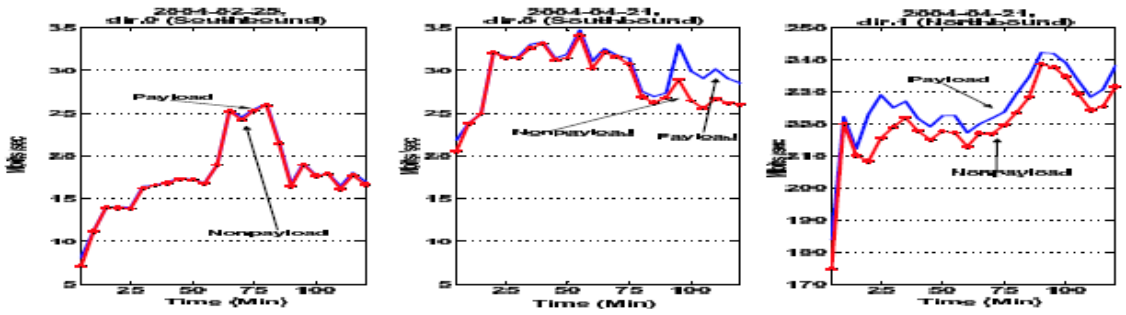

圖 3-2 PTP 演算法與封包分析法的準確率比較

資料來源：Thomas Karagiannis，et al. [3]

上線為封包比對法，下線為PTP演算法，在三種不同連線速度的網路環境檢驗，PTP演算法準確率都極高，即使速度高達220Mbps（最右圖），兩者相較之下仍有超過95%的準確率。

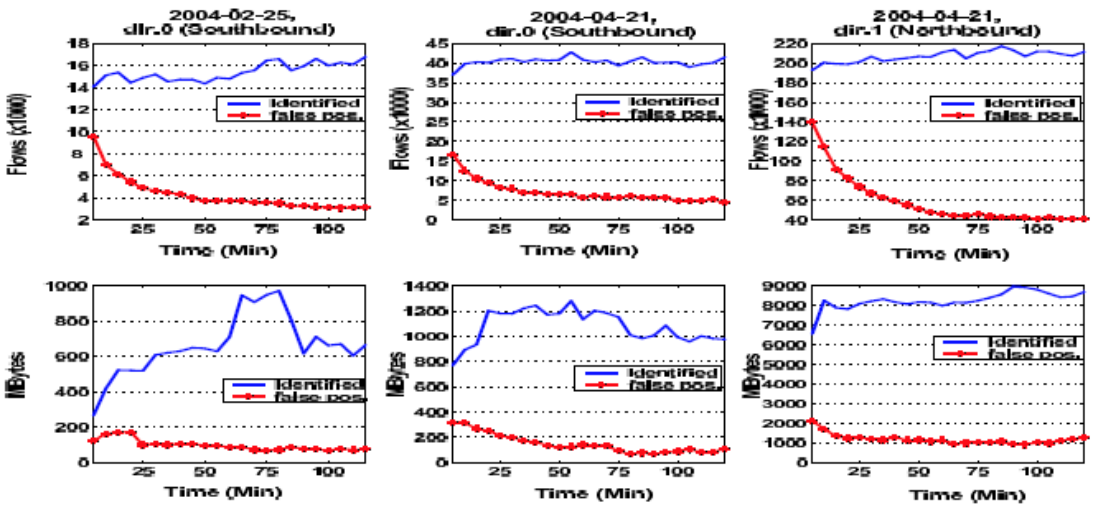

圖 3-3 PTP 演算法的誤判率

資料來源：Thomas Karagiannis，et al. [3]

上線為正確P2P flows，下線為誤判率（False positives），PTP演算法誤判率僅8%~12%。

該演算法乃依照P2P軟體在網路中連線的行為模式來判定，其優勢有（1）不

用檢驗任何payload，提升防火牆效率（2）可辨識未知的P2P協定（3）辨識準確率高且誤判率低(4)連線加密亦能判斷。但是網 路中某些連線行為模式與{IP,Port} pairs相同，卻不是P2P的應用。例如web server、mail server的連線行為就是server的連線資訊可能同時和多個相異的{IP,Port} pairs有關連。未來可以將這類連線透過well-known port排除掉，以減低誤判率。

3.2.2 封包重送的辨識法（Based on the Content Redistribution）

P2P網路中，每個節點同時扮演著server及client的角色，因此當節點下載到某段資料之後，通常會立刻上傳給有此檔案需求的其他節點。[4]根據這個特性，提出一套封包偵測方式，可以不需要比對封包內容，即可判斷出已知或未知的P2P協定。

下圖以BitTorrent為例，Tracker即是supernode，連線分享步驟概分為（1）當peer1有file1欲分享時，它會把訊息傳送給Tracker（2）Tracker接著把這訊息通知想要下載file1的節點peer2（3）當peer2下載完一小片段後，接著peer也3透過Tracker的仲介，向peer2取得剛完成下載的片段檔案。

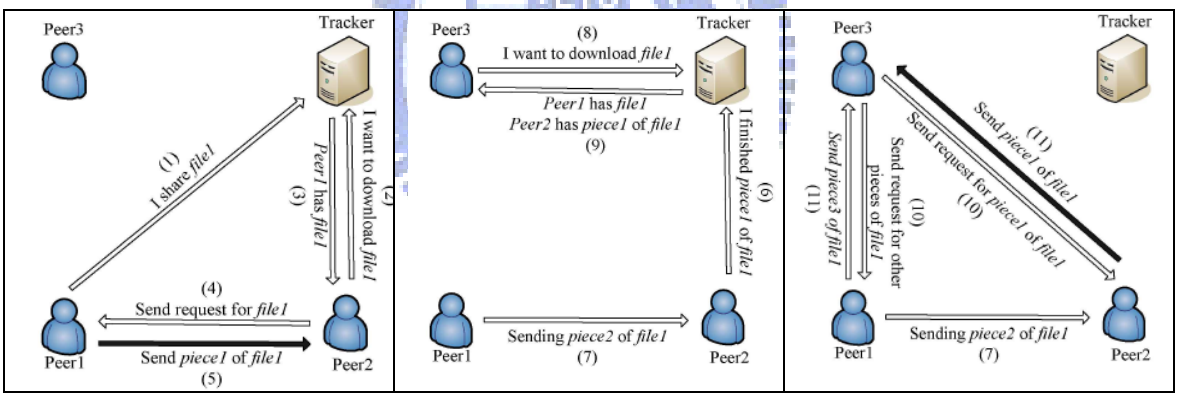

圖 3-4 BitTorrent 下載與上傳的模擬圖

資料來源：Xing Lu，et al. [4]

在P2P網路中，每個參與連線的節點至少都有一個listening TCP port，提供其他節點來溝通或下載資料，雖然這個port是動態的，而且每個參與連線的節點的port number可能都不同，但是當節點開啟P2P應用程式加入連線後，這個port就會維持固定，直到本次連線結束為止。

[4]找出不同節點各自的監聽埠，透過以圖3-5及圖3-6，來辨別P2P flow：

（1）peer_table:我們找出所有節點的listening port並記錄成peer_table，表內容包含<ip,port>。當TCP封包到達該peer，該封包的<sIP,sPort>或<dIP,dPort>可在peer_table找到，我們記錄該封包所屬的flow為P2P flow

（2）content_hash_table:為了更新peer_table，當收到TCP封包時，將封包最前面k-bytes字串的payload映射至content_hash_table。當有相同的content被同一個節點接收後又送出，該content及flow我們都判定為P2P flow，同時將該content的<sIP sPort>s記錄至peer_table中。

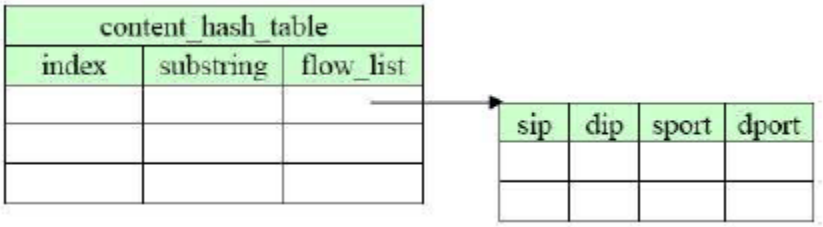

圖 3-5 Content_hash_table 的表格內容

```
1)  for each packet
2)     if packet.protocol == TCP
3)        flow = (packet.sip, packet.dip, packet.sport,
    packet.dport)
4)        if not find(flow, flow_table)
5)           insert(flow, flow_table)
6)        if find(packet.sip, packet.sport, peer_table)
7)           the flow is judged as a P2P flow
8)        if find(packet.dip, packet.dport, peer_table)
9)           the flow is judged as a P2P flow
10)       content = packet.first-k-byte-string_of_payload
11)       if item = find(content, content_hash_table)
12)          if not find(flow, item.flow_list)
13)             insert(flow, item.flow_list)
14)          if packet.sip appear in item.flow_list as dip
15)             flows in item.flow_list are P2P flows
16)             for each flow in item.flow_list
17)                insert(flow.sip, flow.sport, peer_table)
18)       else
19)          insert(content, content_hash_table)
```

圖 3-6 演算法虛擬碼

資料來源：Xing Lu，et al. [4]

最後以北京清華大學學生宿舍網路為實驗環境，進行兩次實驗，把 content redistribution 與 signature-based 的方式互相比較，證明該方法對 BitTorrent 及 PPLive 有極準確判斷率，對於 Maze 及 Xuelei 準確率較低，但總誤判率（False Postive Rate）均低至 1.5%左右。如表 3-7 實驗結果所示：

trace1		
P2P Protocol	**Total Bytes (by signature-based classifier)**	**Accuracy (%)**
BitTorrent	6038858664	90.96
PPLive	2314489902	98.96
xunlei	1146679677	25.93
Maze	8389816003	13.12
False Positive Rate (%)	1.49	

trace2		
P2P Protocol	**Total Bytes (by signature-based classifier)**	**Accuracy (%)**
BitTorrent	5337658347	92.42
PPLive	4252107517	99.31
xunlei	1400498437	45.67
Maze	11350333966	22.77
False Positive Rate (%)	1.52	

表 3-7 封包重送辨識法的準確率及誤判率

資料來源：Xing Lu，et al. [4]

Maze（天網）及 Xuelei（迅雷）均為中國大陸境內相當流行的 P2P 下載程式，其判斷效率不佳的原因在於，Maze 是 CERNET（China Education and Research Network）[27]常用的分享軟體，它提供 peers 透過區網（同一個 class C 或 class B）分享檔案的功能，類似微軟的網路芳鄰，由於 LAN（Local Area Network）分享的速度及效率均遠高於 WAN（Wide Area Network），因此 peer 會在短時間內完成檔案下載工作，不像其他 P2P 軟體，必須得透過 WAN 端數個不同 peers 分成片段上下載。

Xuelei 這套程式可提供 peer 下載 web server 的檔案，類似像 archie 搜尋 ftp 檔案，也因為檔案來源幾乎都是集中式的 web server，並非 P2P 常見的分散式傳輸架構，也因為 Maze 及 Xuelei 並沒有具備一般 P2P 網路「highly sought-after rosources」的特性，因此以封包重送辨識法的準確率就顯得不足。

該演算法優勢為不需要透過封包比對，即可有效辨識已知或未知的 P2P 連線，但缺點在於端看連線過程是否加密。假設連線過程只有 message stream 加密，而檔案傳輸過程不加密，如 WinMX、FastTrack 等，此方法仍可有效判別。但若連線全程加密，因為各連線之間金鑰不同，本演算法將會完全失效。

3.3 應用層防火牆之效能改善

封包進出防火牆必須進行規則比對，傳統以 Rule-based 的 Layer3 或 Layer4 的防火牆，當規則越來越多時，效能會慢慢降低，如果加上 Layer7 防火牆，須耗費更多的記憶體及 CPU 來處理，效能會變得更加低落。

L7-filter 雖然只比對連線前 10 封包，但是後續每個封包仍需要經過所有規則比對，才能決定封包命運。因此 Signature-based 方式辨識 P2P 連線雖然準確率高，但是效能問題卻變成相對衍生的困擾，[5]提出一套改善方法，修改 Netfilter 的 pre-routing hook，把已經辨認過的連線標記放進一個表中，讓後續進來的封包先查詢該表是否已經有經過比對，如果有就直接從表中讀出比對的結果，反之才會去比對系統的防火牆規則。

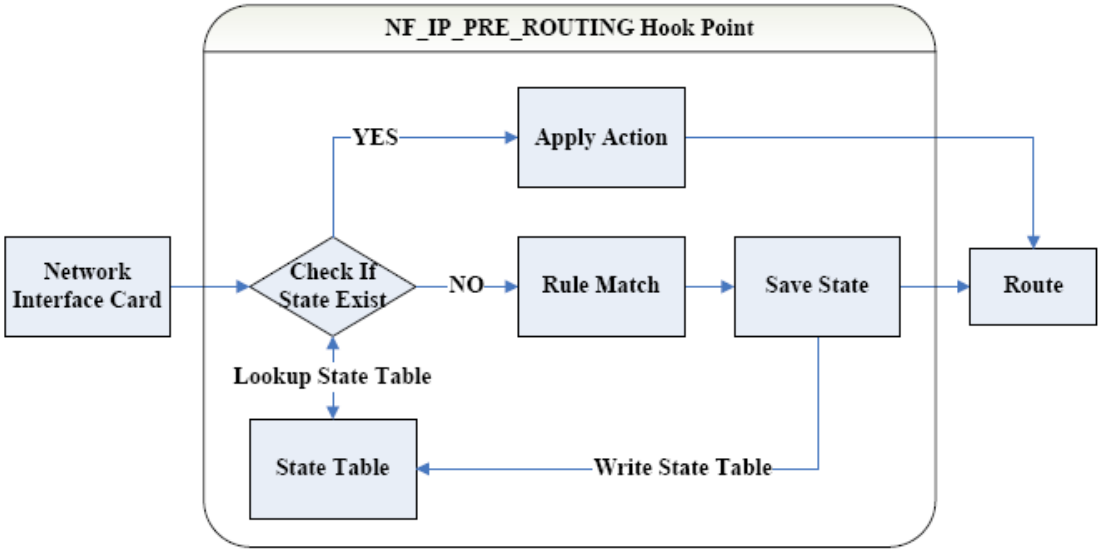

圖 3-7 具狀態檢測的 Netfilter
資料來源：Bing-Heng Peng，et al. [5]

圖3-7中，當第一次比對出結果後，隨即將該封包的連線狀態旗標儲存在state table中，隨後的封包可以先查詢該表有無自己連線的比對結果。此外並設定一個比對門檻值8，當某連線前8個封包比對所有規則後仍然沒有結果，就可以把預設決策設定為ACCEPT或是DROP，寫入對應的Conntrack Entry中，避免該連線後續封包仍繼續進行比對。

除了ACCEPT與允許DROP之外，另外還增加STATE目標函數；當封包比對出結果後，再運用相關參數，將結果存入Conntrack中，相關參數如下表說明：

Parameter	Description
--accept	允許封包通過，並在對應之 Conntrack 狀態欄標記為“accept”。
--drop	不允許封包通過，並在對應之 Conntrack 狀態欄標記為“drop”。
--mark X	允許封包通過，並在對應之 Conntrack 狀態欄標記 mark 值“X”，提供如 QoS 額外之用

表 3-8 STATE 目標函數之參數
資料來源：Bing-Heng Peng，et al. [5]

測試環境為Linux kernel 2.6.14、L7-filter 1.4，硬體為SmartBits 6000C，將防火牆規則分成50、100與200條，並確保封包要比對到最後一條規則。另外也增加無規則的測試當作對照組。分別測試了封包大小為128、512與1518 bytes之下，防火牆最大處理效能，實驗結果如下表所示：

Packet Size：128 Bytes			Packet Size：512 Bytes			Packet Size：1518 Bytes		
Rules	Throughput (Mbps)		Rules	Throughput (Mbps)		Rules	Throughput (Mbps)	
	Original L7-filter	Improve-ment		Original L7-filter	Improve-ment		Original L7-filter	Improve-ment
0	249.77	**249.77**	0	837.58	**837.58**	0	1000	**1000**
50	126.02	**249.77**	50	427.66	**837.58**	50	1000	**1000**
100	79.61	**249.77**	100	279.69	**837.58**	100	744.77	**1000**
200	40.94	**249.77**	200	156.95	**837.58**	200	450.86	**1000**

表 3-9 三種不同大小封包的測試數據
資料來源：Bing-Heng Peng，et al. [5]

由實驗結果可以看出，當過濾規則變多時，文中提出的改善機制能夠有效改善過濾效能，不會因為過濾速度過慢，嚴重拖垮網路效能。

第四章 實驗設計

本實驗設計採「仿真」的方式進行，針對中小學學生最常使用的 P2P 軟體進行實體連線的封包錄製及重播。中小學經費較不足，因此本實驗全部採用免費軟體進行，而且為了讓實驗變因更容易掌握，本章節提出一套封閉環境的架構方法，讓雜訊干擾的減至最低，以利實驗進行順利。

4.1 模擬（simulation）或仿真（emulation）

一般網路實驗可分為模擬（simulation）或仿真（emulation）兩類，差別在於「模擬」選取整個系統環境的某些行為特徵，用另一系統來表示它們的過程；而「仿真」則為較高階段的模擬技術，讓被模仿的系統更貼近真實的環境。著名模擬軟體如 NS2[39]或交通大學開發的 NCTU-ns[40]，它們共同特色在於能夠使用單一電腦的 CPU 來模擬多台網路上不同的節點，並透過軟體本身專屬的語法及支援的網路協定，讓模擬出來的節點們互相溝通，此方式可大幅節省真實環境硬體的開發成本。

而本實驗設計採用「仿真」方式進行，原因在於P2P軟體種類相當多元，不同軟體連線時所採用的協定也是百家爭鳴，所以到目前為止沒有任何的模擬軟體可以正確地產生P2P流量以供壓力測試[6]。除此之外，網路上亦也有些免費的packet builder/generator的流量製造工具程式，提供使用者定義封包內容之後發送，但是此類方式仍然無法貼切地表示出完整連線的互動過程。因而本實驗選擇將整個連線過程的封包錄製下來，再透過播放軟體重送至網路環境中，讓防火牆彷彿置身於真實網路環境中，來達到仿真以及效能測試的目的。

4.2 封包錄製

本實驗欲比較 T4-terminator、L7-filter、IPP2P 三種不同方式下，對於 P2P 連線主動防禦的效能。為了控制每次實驗時，P2P 的連線情形均需相同，因此預先將真實環境的 P2P 流量錄製下來，實驗時再進行重播的動作來模擬相同的連線。

4.2.1 P2P 連線軟體的選擇

P2P 軟體相當多，本實驗選擇中小學學生最常用的五項軟體進行研究，分別是 FOXY、BitTorrent、eMule、MSN Messenger、Yahoo 即時通。FOXY 採用 Gnutella 協定，它目前是國內最多用戶的檔案分享軟體，操作介面極為簡單，所以大部分學生都在無師自通下學會了安裝 FOXY，及抓取違法版權的資料。FOXY 由於用戶多，所以造成的資訊洪流最嚴重，不但大量地阻塞上傳頻寬，而且使得電腦效能大幅下降，更重要的是上面假檔充斥，病毒、色情資訊多到氾濫的地步。BT 與 eMule 因為操作方式較為複雜，使用的學生人數比較少，但這兩套軟體也是盜版電影與盜版音樂的大溫床，亦列為禁止的目標。

MSN Messenger及Yahoo即時通是佔有率極高的IM軟體，通常老師上資訊課時，許多學生都會偷偷安裝該類軟體來聊天，無法專心聽講，學習成效大打折扣；而這類軟體很容易使用TCP 80 port來逃避防火牆監督，因此必須透過應用層辨識方式加以攔截。

4.2.2 錄製工具

封包錄製或者監看的工具程式不少，依使用者不同需求而發展出不同的功能。功能最基本的，在UNIX上面常見的為Tcpdump，而Windows版本則為Windump，兩者皆為文字操作介面，需要下達指令才能操作。Wireshark[29]則是另一套跨平台的封包錄製工具，它前身即為大名鼎鼎的Ethereal，它不但免費且功能相當強大，支援十多種其他錄製工具的檔案格式。除上述軟體外，在Windows上面還有兩套免費錄製工具，Packetyzer以及本實驗採用的Microsoft Network Monitor[30]。

免費的錄製程式著重在解碼封包內容，在視窗中顯示出封包各欄位的詳細內容，以供使用者分析。而商用的工具著重在「監控」與「統計」，能夠在線監控網路狀況，並即時提供大量的統計分析圖，支援的網路協定也相對地比免費軟體來得多，著名的商業軟體如Sniffer Pro、Capsa等。

本實驗採用Microsoft Network Monitor Ver3.2為封包錄製工具，它是微軟提供的免費封包錄製軟體。大部分錄製軟體都依照封包進出網卡的先後順序排列，所以同一時間在視窗畫面上會參雜著許多不同程式的封包；而Network Monitor最大的不同點，在於它可以把封包按照本身歸屬的應用程式做出分類，因此使用者能夠排除其他封包干擾，錄製該程式對外互動的完整過程。

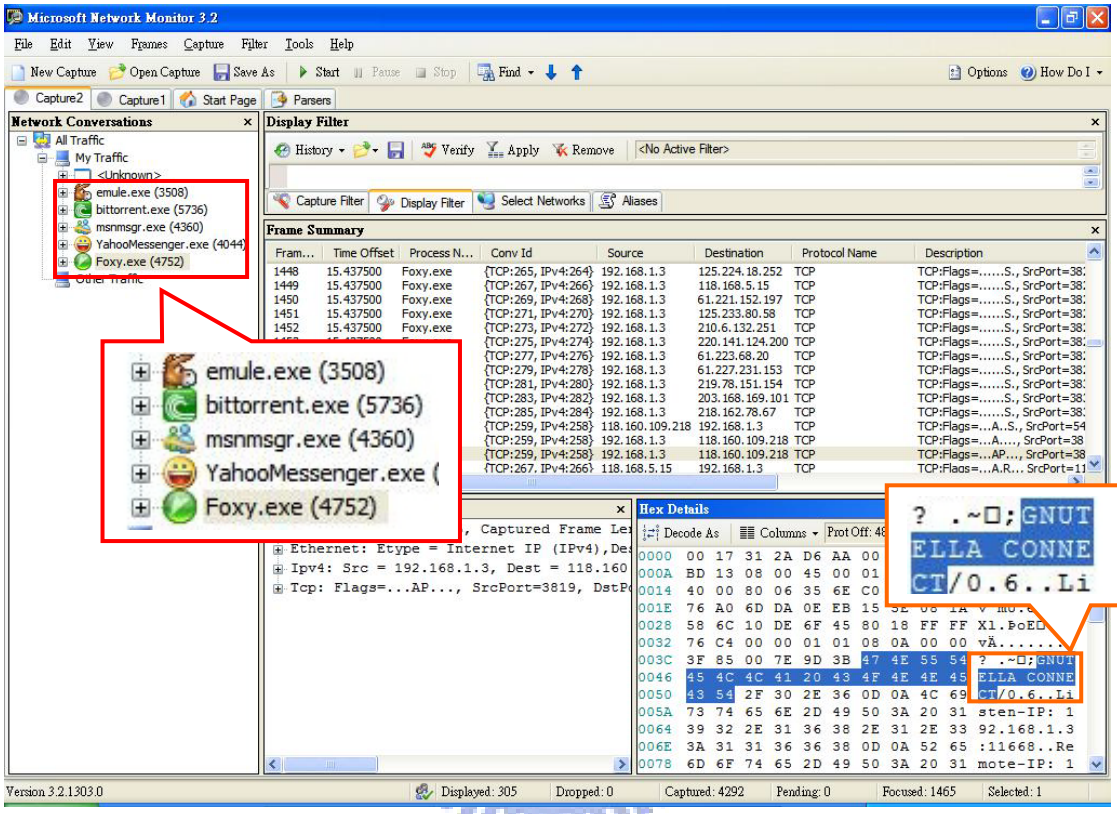

圖 4-1 Microsoft Network Monitor 執行畫面

由上圖執行畫面可看出，各連線的封包都被區隔分類，不但更容易觀察連線行為，而且可以選擇單獨儲存某連線的封包；而右下角的小窗格可看出Foxy傳輸採用Guntella協定。

4.2.3 錄製方式

通常來說，P2P連線的Payload Signature會出現在連線起始階段的前幾個封包，連線建立之後的封包大部分都是資料的傳遞，所以擷取請求建立連線階段的封包比較具有代表意義。

本實驗欲錄製五種P2P連線的起始封包，必須先啟用L7-filter或IPP2P進行連線阻擋，才能順利擷取其請求連線的起始封包。由於IPP2P不支援IM通訊協定，所

以選擇支援較多協定的L7-filter為過濾工具，將防火牆規則設定如下表：

```
01# iptables -N BAN
02# iptables -A FORWARD -m layer7 --l7proto gnutella -j BAN
03# iptables -A FORWARD -m layer7 --l7proto bittorrent -j BAN
04# iptables -A FORWARD -m layer7 --l7proto edonkey -j BAN
05# iptables -A FORWARD -m layer7 --l7proto yahoo -j BAN
06# iptables -A FORWARD -m layer7 --l7proto msnmessenger -j BAN
07# iptables -A BAN -j ULOG
08# iptables -A BAN -j DROP
```

表 4-1 IPTables L7-filter Rules of packet capturing

IPTables指令若不指定table的話，預設值為filter table，在第二章有提過filter table共有三個chain，分別是INPUT chain、FORWARD chain、OUTPUT chain。而Rule＃01先在filter table自行定義一個名為「BAN」的chain，為了要方便觀察紀錄被DROP的封包。Rule＃02到Rule＃06表示在FORWARD chain上面使用L7 extension match module，分別對五種不同連線協定進行應用層封包辨認，條件符合者將它移至BAN chain，Rule＃07將BAN的資料再搬到ULOG這個記錄檔，最後Rules＃08將BAN chain的封包全部丟棄。

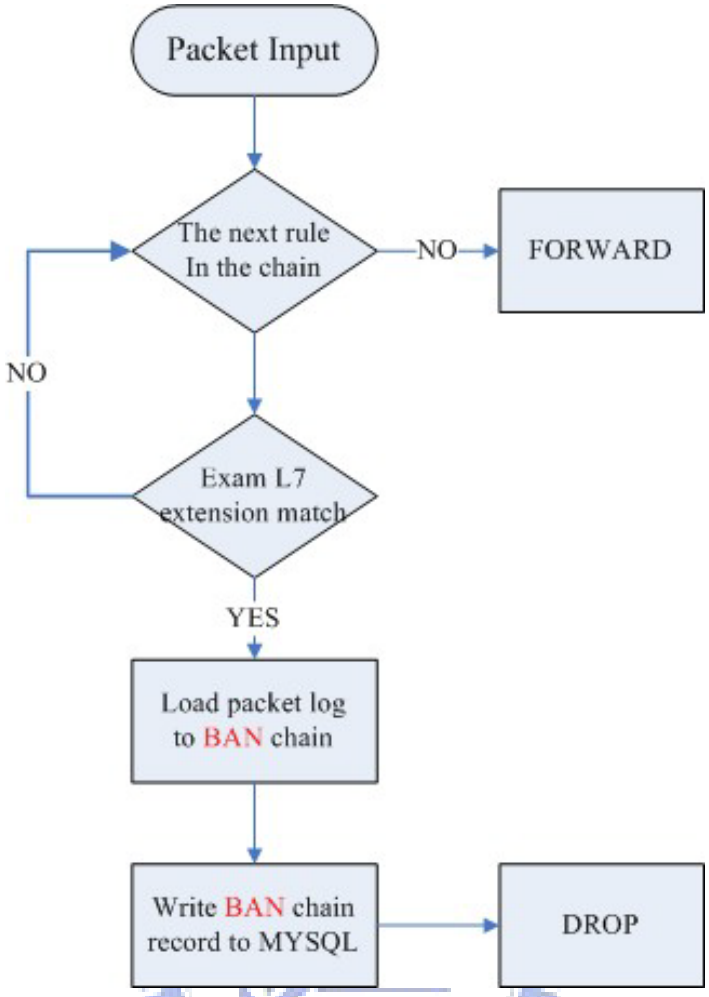

圖 4-2 Flowchart of packet capturing

上圖是整個指令的處理流程，雖然Netfilter能以內建計數器[1]及LOG模組[2]，透過系統核心的syslogd來記錄封包資訊，但也因為透過syslogd，所以會跟系統其他log資訊混雜在/var/log/messages裡面，雖然可利用某些方式分離，但是閱讀上仍然很吃力。

[1] 使用 iptables -L -n -v 指令觀看 Netfilter 的統計數字
[2] 使用-j LOG 與其附加參數（-j LOG --log-level 等）紀錄封包資訊

ULOG是ulogd[31]安裝編譯後在IPTables模組產生的target function，目的也在記錄辨識到的封包資訊，它與LOG不同之處在於ULOG將log交由UserSpace Program處理，再把資訊完整寫入MySQL或Postgresql等資料庫中，最後透過PHP編寫的分析工具Nulog[32]，以網頁介面呈現，讓使用者在錄製封包時，能夠同步觀察被DROP封包的各種屬性。

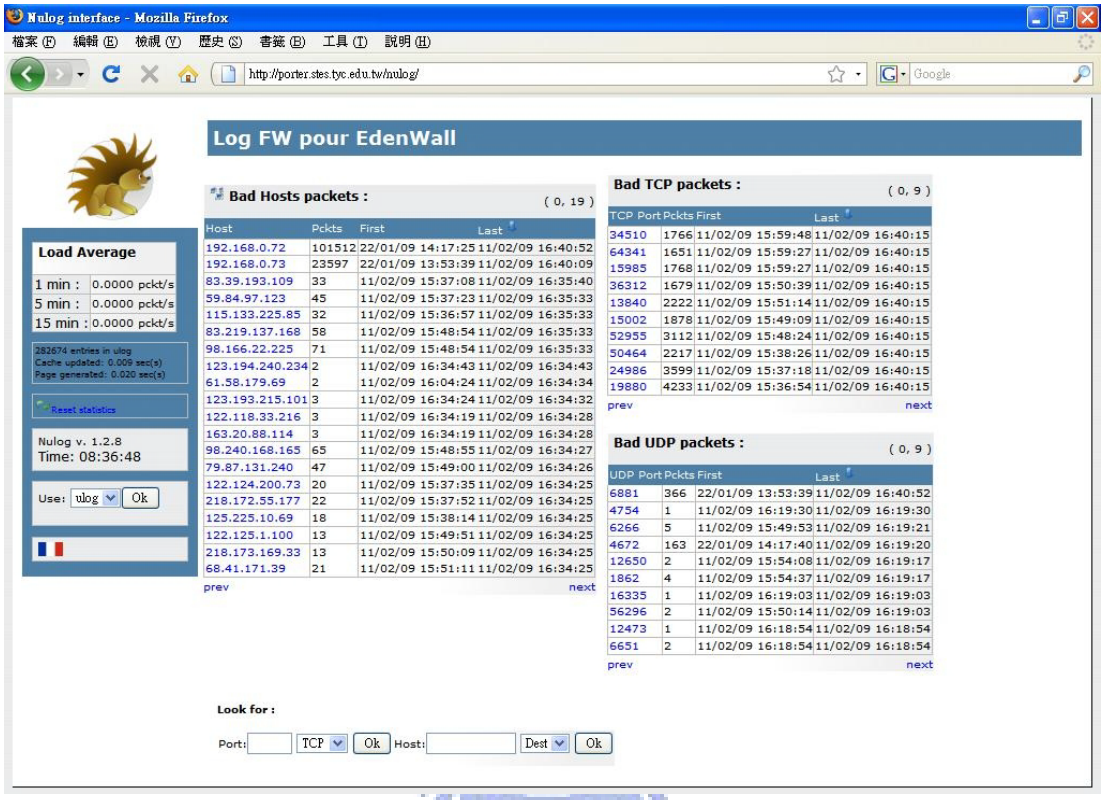

圖 4-3 Nulog 的執行畫面

實際錄製時，如下圖所示，將五種軟體安裝在五台 PC 執行，每種軟體個別錄製 15 分鐘，最後封包統計結果如表 4-2，因為每種軟體特性不同，產出的封包數量差異性也不同，很明顯看出最容易產生資訊洪流的 FOXY、BT 無論產生之封包總數、總量、封包平均大小都是最大。

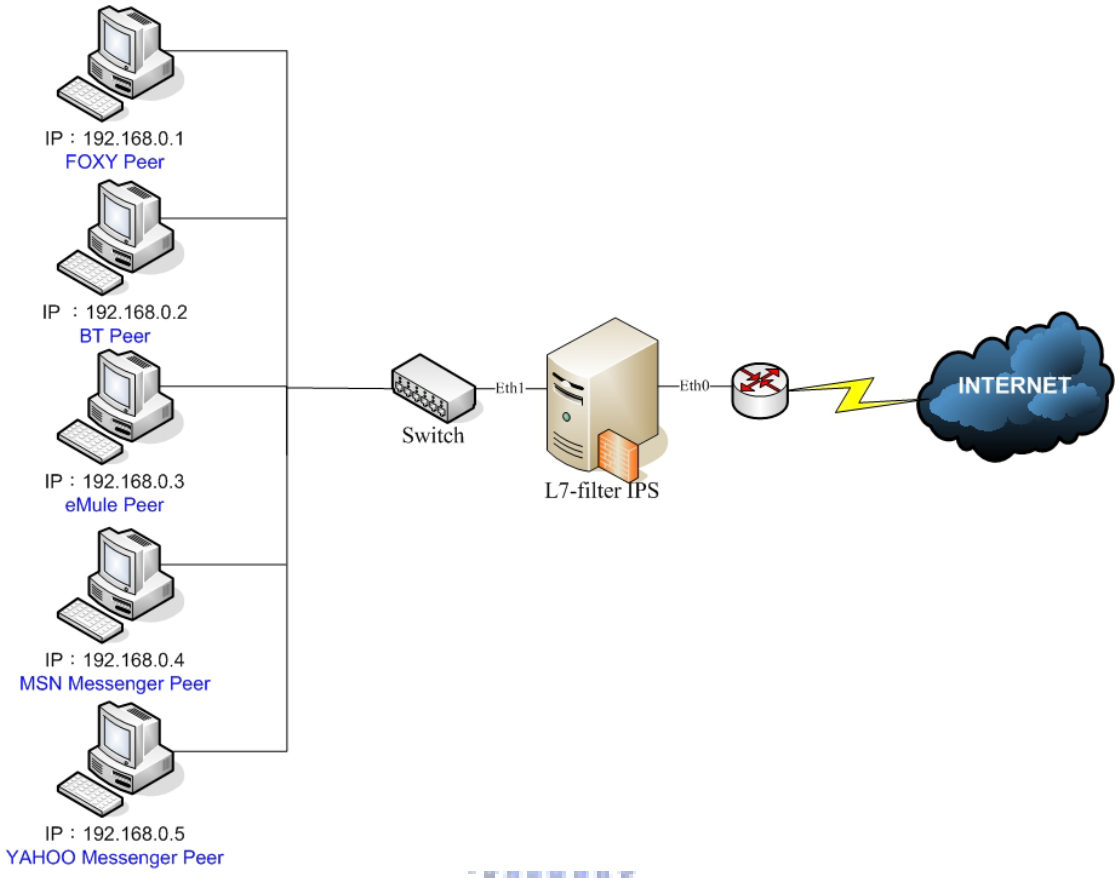

圖 4-4 封包錄製配置圖

Application	Packets Number	Total Size	Average Size
FOXY	4586	1.67 MB	0.37 KB
BitTorrent	4583	1.61 MB	0.36 KB
eMule	2710	674 KB	0.25 KB
MSN Messenger	313	98.9 KB	0.32 KB
Yahoo Messenger	354	79.2 KB	0.22 KB

表 4-2 錄製 15 分鐘後的封包總量

4.3 封包重播

封包播放工具比擷取工具來得少，較常使用在 UNIX 上有 Tcpreplay、Tomahawk 兩者均為 Open Source 的自由軟體，兩者均需透過指令操作，並沒有 GUI 介面；Tcpreplay 雖有 Win32 版本，但仍須使用 C 語言編譯後才能執行。

而具有 GUI 操作介面的有 Packet Player[33]、Traffic IQ Pro，兩者都是 Windows 的應用程式，Traffic IQ Pro 是商用軟體，它提供封包改寫以及封包發送時同步報告的功能，但是試用版僅支援罕見的 kar 封包紀錄格式。本實驗使用自由軟體 Packet Player，操作相當方便，支援播放的檔案格式也相當多，操作畫面如下圖。

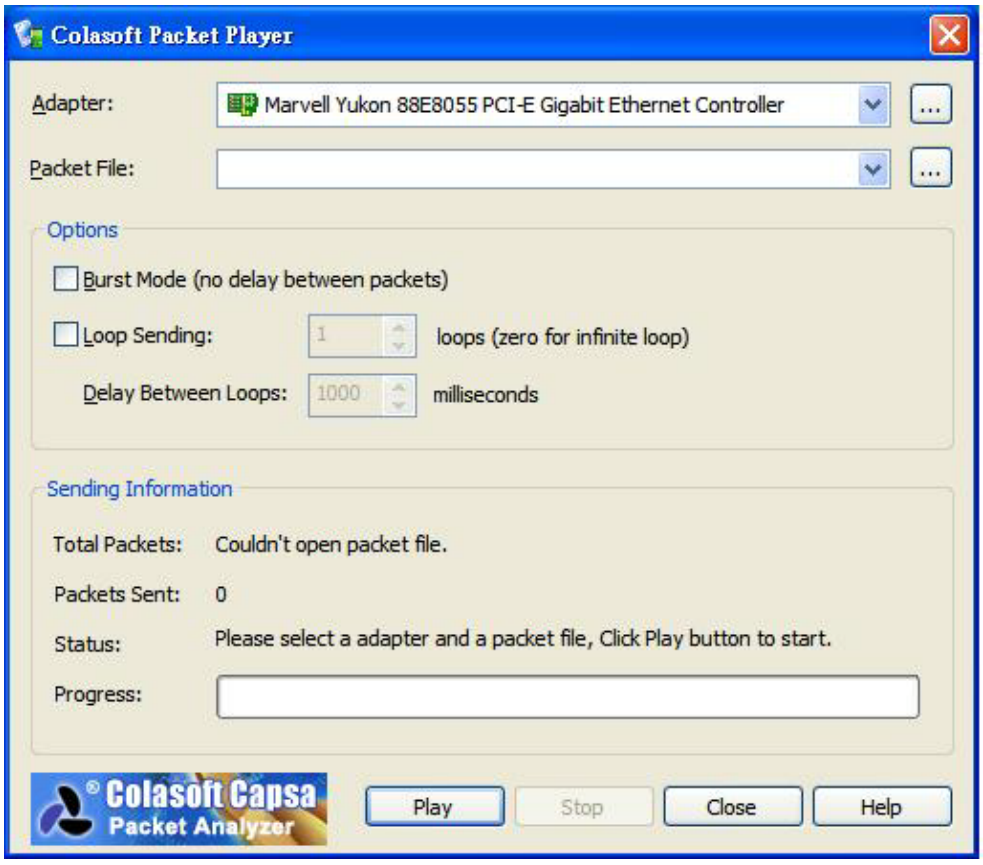

圖 4-5 Packet Player 操作畫面

Packet Player 重播時是依照錄製的時間戳記將封包依序送出，最大特色在「Burst Mode」及「Loop Sending」。「Burst Mode」顧名思義為暴衝模式，該方式能夠將封包傳遞之間的間隔時間修正到趨近於零，等於一瞬間就可以把原本排程數分鐘甚至數小時的封包量全部送出，製造待測物的壓力；「Loop Sending」可由 one loop 設定至 infinite loop，讓同樣的播放動作反覆不斷地進行，配合「Burst Mode」可讓網卡在短時間內衝出極高的流量，以供防火牆壓力測試。

4.4 效能測試

提到網路設備效能測試硬體，最常被使用的就是知名大廠 SPRIRENT 公司的 SmartBits 與 IxChariot 公司的 IXIA。兩大系列的產品都可以借由機身擴充槽插上介面卡，再接上待測物進行仿真壓力測試，但是這類設備要價不斐，並非一般中小學經費能夠負荷。所以本研究利用 IxChariot 公司提供的 Qcheck[34]及 Endpoint[35]這兩套免費軟體作為效能測試工具。

圖 4-6 Qcheck 執行畫面

Qcheck 與 Endpoint 支援四種協定（TCP、UDP、SPX、IPX）及提供四種測試功能（Response Time、Throughput、Streaming、Traceroute）。Qcheck 是前端使用者 console 軟體，目前僅提供 Win 作業系統版本，它必須配合後端待測物背景執行的 Endpoint 程式，才能測出各項數據。Endpoint 支援多達 17 種的作業系統，並提供測試時 localhost to remote 及 remote to remote 兩種方式，使用上具有相當大的彈性。

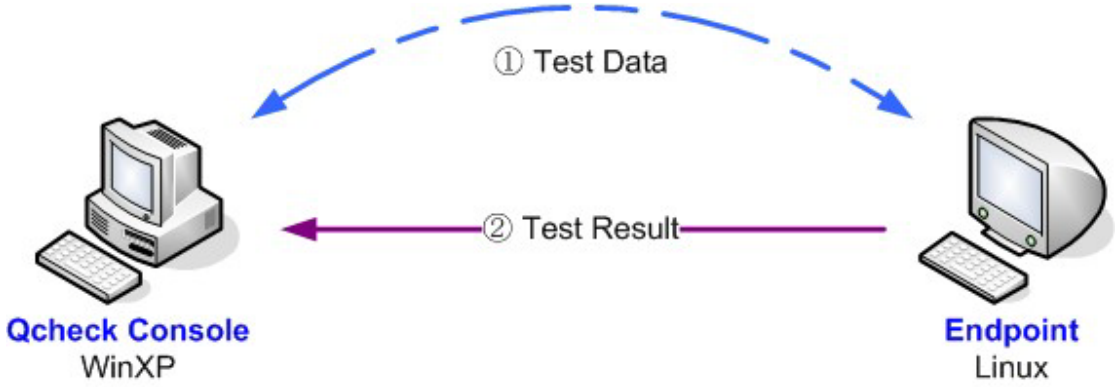

圖 4-7 Localhost to Remote 效能測試圖

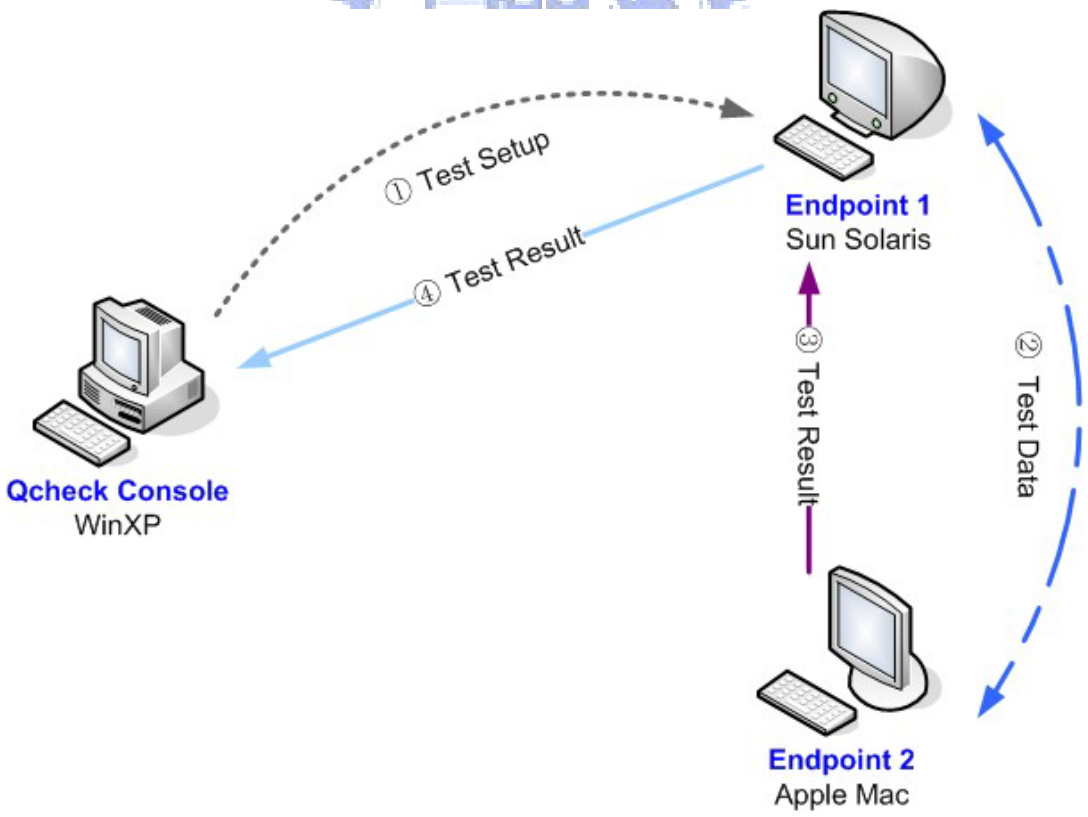

圖 4-8 Remote to Remote 效能測試圖

第五章 系統實作與效能評估

本章節說明實作過程，將實驗分為兩大組，分別是實驗組的 T4-terminator、L7-filter、IPP2P 及對照組的 Pure NAT；實驗組均設定 8 條防火牆規則，對照組則單純進行 NAT 工作，觀察在大量流量壓力下，防火牆各項效能數據，並加以分析評估。

5.1 實驗環境

圖 5-1 為實驗環境的配置，與第四章封包擷取配置相較，多了一台 WEB Server 及效能測試電腦。

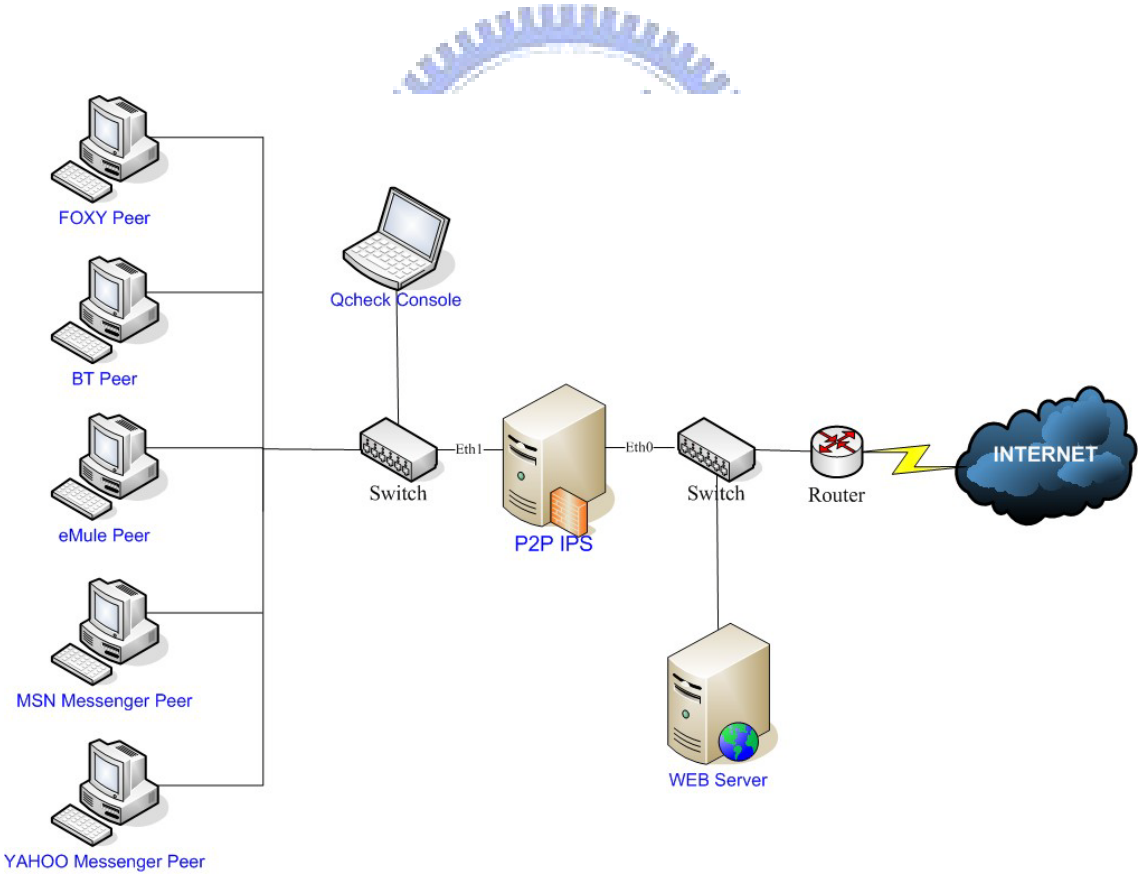

圖 5-1 實驗環境配置圖

實驗環境中 Clients 與 Servers 之硬體規格及作業系統：

P2P Clients		Qcheck Console	
CPU	Intel Celeron 2.8 G	CPU	Intel Core2 Duo1.83 G
RAM	512 MB	RAM	1 GB
OS	WinXP Pro SP3	OS	WinXP Pro SP3
LAN	SiS900 10/100	LAN	Marvell Yukon 100/1000

P2P IPS		Web Server	
CPU	Intel Pentium4 3.2 G	CPU	Intel Xeon 3.2 G
RAM	1 GB	RAM	1 GB
OS	CentOS 5.1	OS	FreeBSD 6.2-RC2
Kernel	V2.6.23.10	Web Server	Apache-1.3.37
IPTables	V1.3.8		
WAN	Broadcom NetXtreme 100/1000	WAN	Intel PRO/1000 GT
LAN	Realtek8139 10/100		

表 5-1 硬體設備與作業系統規格表

5.2 實驗方法與步驟

五個 P2P Clients 端將預錄好的封包檔案，以 Burst Mode 及 Infinite Loop 的方式重播，並設定每個 Loop 之間沒有任何延遲，讓網卡打出不間斷的連續流量來加壓 P2P IPS。另外許多 P2P 程式運用大量的 HTTP-like 的協定來傳輸檔案[6]，如 FastTrack（KaZaA）、Gnutella（Foxy）、Xuelei 等，因此本實驗另建置一台 Web Server，網頁上放置如表 5-2 所示五個 ISO 檔案，每個 client 端均對五個 ISO 檔開啟 HTTP 下載連線，模擬 P2P 連線時，大型封包傳輸的情況，增加 IPS 判斷封包內容的工作。

File name	Size
KNOPPIX-5.3.1-DVD-EN.iso	4.14 GB
CentOS-5.2-i386-bin-DVD.iso	3.74 GB
Fedora-9-i396-DVD.iso	3.33 GB
FreeBSD-7.1-Release-i386-DVD.iso	1.77 GB
FreeBSD-6.4-Release-i386-DVD.iso	1.70 GB

表 5-2 Web Server 五個 ISO 檔案

在前一章節以 L7-filter 進行封包擷取，同時以 Nulog＋ULOG＋MySQL 觀察記錄被擋下封包的資訊，但在實作效能測試時，直接將條件比對符合的封包予以丟棄，以節省系統資源。

在防火牆規則數方面，因 L7-filter 支援多達 109 種，而 IPP2P 支援的協定僅有 9 種，而且 IPP2P 所支援的 9 種協定中，WinMX 協定在 L7-filter 卻又不支援，為了控制實驗變因，於是本實驗取兩軟體最大交集共 8 種協定進行測試。另外在 T4-terminator 部分，IPTables 規則數亦同樣設定為 8 條，除了開放 WEB、SMTP、POP3 連線外，其餘 TCP 封包只要 5 分鐘內連線超過 30 次，均視為 P2P 封包加以阻擋。

L7-filter 及 IPP2P 相互交集的 8 種協定中，Gnutella（Foxy）、eMule /eDonkey、BitTorrent 是國內學生較常用的 P2P 軟體，其他如 FastTrack（KaZaA）、Direct Connect、AppleJuice、SoulSeek、Ares 則是其他國家流行的連線軟體。這 8 種協定並不包含所有的 Instant Messenger，但進行實驗時仍會把先前預錄的 MSN、YAHOO 即時通封包打出，因為這些封包仍會在所有的 IPTables

規則中逐一進行比對，對效能測試仍具有意義。

實驗進行時共分成兩大組，先進行對照組的 Pure NAT，再進行實驗組的 T4-terminator、L7-filter、IPP2P。每一組再分別進行 3 種效能測試（Benchmark）：

1、Throughput：分別以 5 種不同 Data Size（50kB、100kB、250kB、500kB、1000kB 測試 IPS 的最大處理能力(Throughput)，每 10 秒鐘測一次值，每個 Data Size 各測量 20 次後取其平均。

2、Response Time：反應時間（Response Time）與延遲時間（Latency）最大差異在於「Latency」指的是單一事件的延遲，而「Response Time」是指會造成延遲的所有因素的總和(overall processing delay, made up of individual latencies)。Qcheck 軟體設計的目的在測量遠端網路狀況，通常測量封包會跨過數個網域，所以才會定義為 Response Time。本實驗環境封閉，且 Console 端與待測物 IPS 僅相隔一個 switch，所以測出之 Response Time 幾乎等同於 Latency，但本測試名稱仍採用軟體命名的 Response Time。實驗時分別以 5 種不同 Data Size（2kB、4kB、8kB、16kB、32kB）測試 IPS 的延遲時間，重複次數（Iterations）設成 5，每 10 秒鐘測一次，每個 Data Size 各測量 20 次後取其平均。

3、CPU Utilization：為比較Burst Mode對IPS加壓的效果，本實驗再分成兩組，第一組按照封包錄製時的時間軸循序播放，第二組才以Burst Mode加壓播放。兩組均使用sar（System Activity Repoter）指令[1]記錄每秒鐘CPU使用率，連續記錄 500 秒後，取其之最大值、最小值、平均值。

[1] # sar 1 500 > /root/filename，「1」指記錄的間隔秒數，「500」指連續記錄之時間，並將結果輸出成檔案

5.2.1 實驗一：T4-terminator

在[3]中作者提出一套 PTP 運算方法，使用傳輸層（Transport Layer）來辨識 P2P 連線，藉著檢驗{IP,Port} pairs 的連線數量，如果連線數超過 20 條，而且持續時間超過 5 分鐘，即判定為 P2P 連線。而在第二章（圖 2-6、圖 2-7）也清楚呈現，無論分散式或者混合式的 P2P 架構，一個新加入的節點，很快地就會跟鄰近節點產生連線。因此本實驗依照 P2P 連線的特性及[3]的方式加以簡化，提出一套 Layer4 的 P2P 限制方式，本研究將其命名為 **T4-terminator(T4 for Transport Layer4)**，設計之防禦規則如下表所示。

```
00# modprobe ipt_recent ip_list_tot=1000 ipt_pkt_list_tot=50
01# iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 -j ACCEPT
02# iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 80 -j ACCEPT
03# iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 25 -j ACCEPT
04# iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 25 -j ACCEPT
05# iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 110 -j ACCEPT
06# iptables -A FORWARD -i eth0 -o eth1 -p tcp --sport 110 -j ACCEPT
07# iptables -A FORWARD -p tcp -m state --state NEW,ESTABLISHED, \
    RELATED -m recent --name P2P_db --update --second 300 --hitcount 30 \
    -j DROP
08# iptables -A FORWARD -p tcp -m recent --set --name P2P_db
```

表 5-3 T4-terminator Rules

因為第 7、8 條規則會動用到 recent 模組，但 recent 資料庫預設最多僅記錄 100 個 IP，而且每個 IP 預設僅能記錄 20 個封包資訊；當資料庫滿載時，新資訊會取代舊資訊，這對於 NAT 主機要服務整個 Class C 的區網，及對外上百甚至上千個連線來說，顯然是不夠用的，所以第 0 條規則以手動方式載入 recent 模組，並指定其參數，讓記錄的 IP 數量擴充到 1000 個，每 IP 追蹤的封包擴充到 50 個。

第 8 條規則運用 recent 模組的 set 參數，把符合該規則的 TCP 封包記錄到自行定義之 P2P_db 資料庫；第 7 條規則用到參數 update，它會不斷比對更新 P2P_db 資料庫的資訊，若資訊不存在，則交給第 8 條規則把該資訊寫進資料庫。第 7 條規則 second 300 及 hitcount 30 的意思為 300 秒內重複次數 30 次，亦即只要 5 分鐘內超過 30 次 TCP 連線，無論狀態是 NEW、ESTABLISHED、RELATED 的連線，隨後的封包均採 DROP 方式處理。

第 1~6 條規則目的在當開放 WEB、SMTP、POP3 連線，因為 IPTables 採「first match」的關係，所以這三種連線的 TCP 封包不會被後續的規則 DROP 掉，讓 Client 端嘗試 P2P 連線被禁止時，至少還能維持網頁、郵件收發的正常連線。

整個執行流程如圖 5-2 所示，本方式最大的優勢在於不檢驗 Payload Signature，有效提高 IPS 的過濾速度。另外對於採加密連線，或模糊演算技術，甚至未知特徵的 P2P 連線亦能進行阻擋，尤其對於中小學學生熱愛的 P2P 線上對戰遊戲阻擋效果甚佳。此方式最大缺點在於會漏判部分採 80 port 連線的 P2P flow，或者誤判其他非 P2P 的 TCP 連線，但漏判、誤判的問題在本研究中並未加以探討。

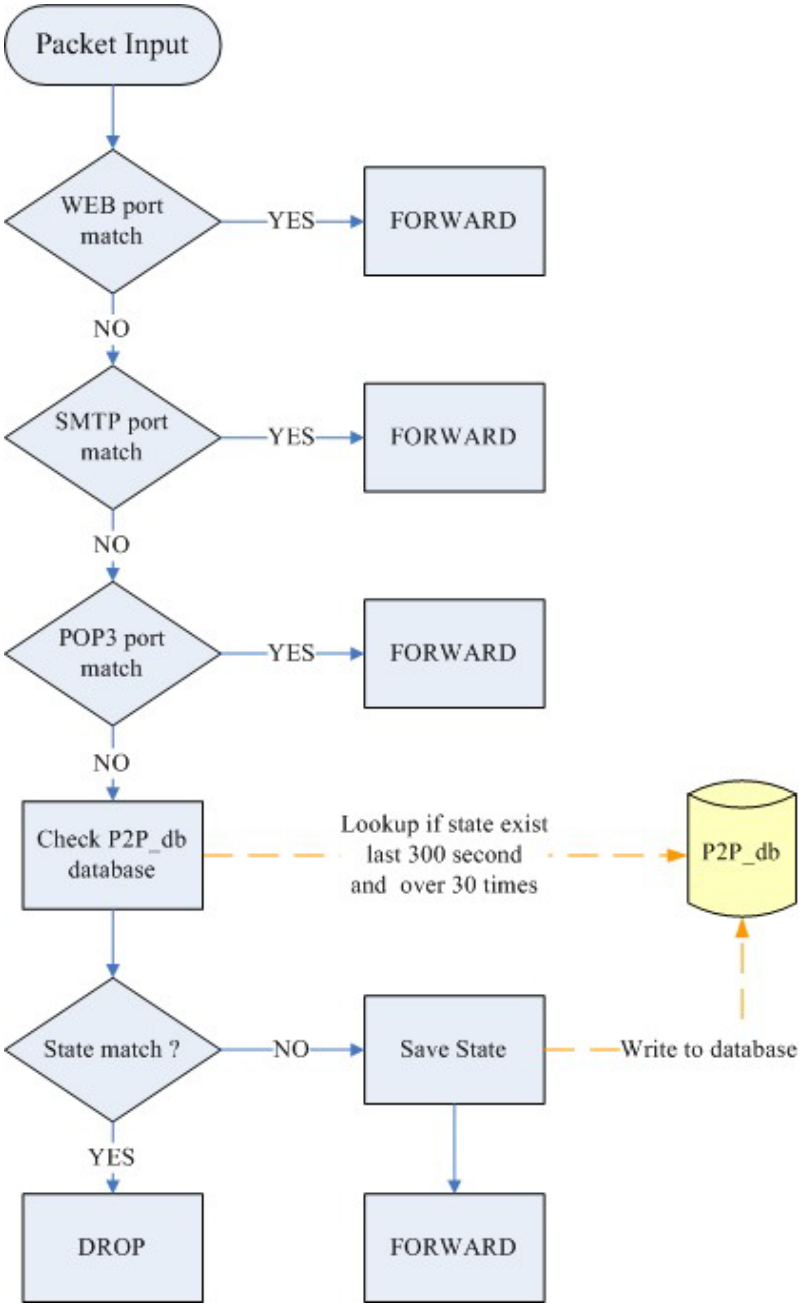

圖 5-2 T4-terminator flowchart

5.2.2 實驗二：L7-filter classifier

L7-filter classifier 提供 Kernelspace 及 Userspace 版，實驗使用 Kernelspace v2.20 版本，Protocol definitions 為 v2008-12-18。實驗定義 IPS 規則如下表，因為 IPTables 規則比對是採「first match」逐一比對，必須比對到符合條件的規則，才會決定封包的命運，所以將封包數量最多（4586 個）的 FOXY 放最後，封包數較少（2710 個）的 eMule 放第六條。

```
01# iptables -A FORWARD -m layer7 --l7proto directconnect -j DROP
02# iptables -A FORWARD -m layer7 --l7proto soulseek -j DROP
03# iptables -A FORWARD -m layer7 --l7proto applejuice -j DROP
04# iptables -A FORWARD -m layer7 --l7proto ares -j DROP
05# iptables -A FORWARD -m layer7 --l7proto fasttrack -j DROP
06# iptables -A FORWARD -m layer7 --l7proto edonkey -j DROP
07# iptables -A FORWARD -m layer7 --l7proto bittorrent -j DROP
08# iptables -A FORWARD -m layer7 --l7proto gnutella -j DROP
```

表 5-4 L7-filter Rules

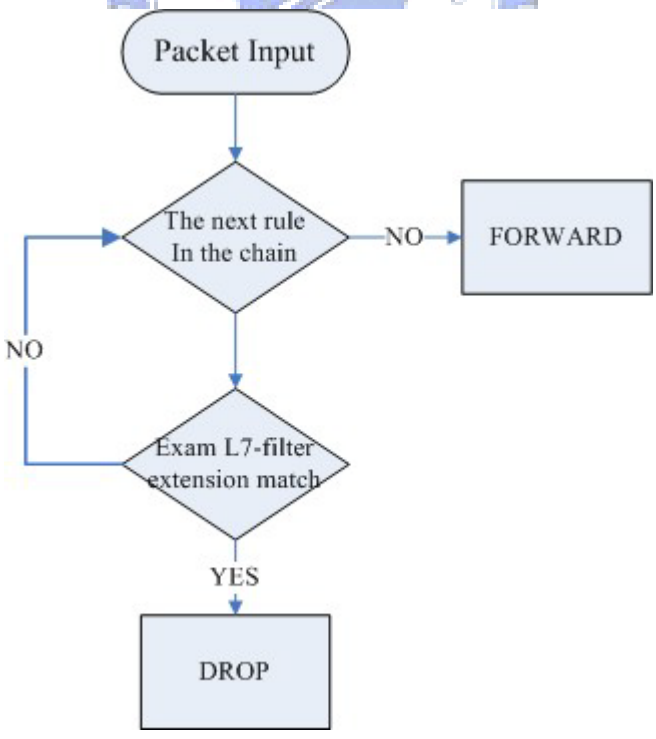

圖 5-3 L7-filter Rules flowchart

5.2.3 實驗三：IPP2P classifier

本實驗使用 IPP2P v0.8.2，不同 L7-filter 採呼叫外部 Protocol definitions，IPP2P 的 String-match 是以 16 進位碼直接隨程式碼編譯到模組中。實驗定義 IPS 規則如下表，所有規則順序都跟 L7-filter 相同。

```
01# iptables -A FORWARD -m ipp2p --dc -j DROP
02# iptables -A FORWARD -m ipp2p --soul -j DROP
03# iptables -A FORWARD -m ipp2p --apple -j DROP
04# iptables -A FORWARD -m ipp2p --ares -j DROP
05# iptables -A FORWARD -m ipp2p --kazaa -j DROP
06# iptables -A FORWARD -m ipp2p --edk -j DROP
07# iptables -A FORWARD -m ipp2p --bit -j DROP
08# iptables -A FORWARD -m ipp2p --gnu -j DROP
```

表 5-5 IPP2P Rules

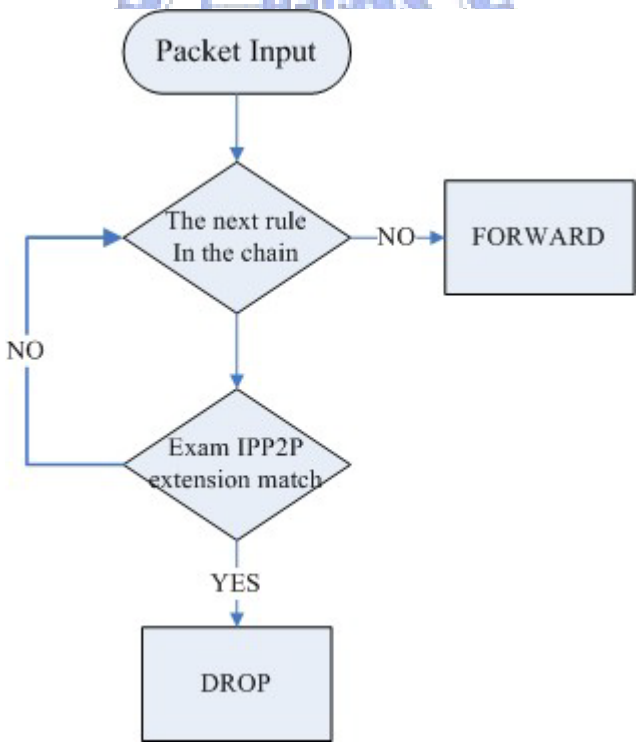

圖 5-4 IPP2P Rules flowchart

5.3 實驗結果與效能評估

5.3.1 Throughput

以 5 種不同 Packet Size 測試後，實驗結果均呈現最大處理能力由高至低為：T4-terminator ＞ PureNAT ＞ IPP2P ＞ L7-filter，各項實驗數據如以下圖表：

Throughput （Mbps）					
	50 kBytes	100 kBytes	250 kBytes	500 kBytes	1000 kBytes
PureNAT	2.26	3.36	6.32	8.14	10.06
T4-terminator	2.31	3.90	6.72	8.29	10.25
L7-filter	1.14	1.89	3.46	4.55	5.47
IPP2P	1.18	1.97	3.55	4.87	5.63

表 5-6 Throughput of different Packet Size

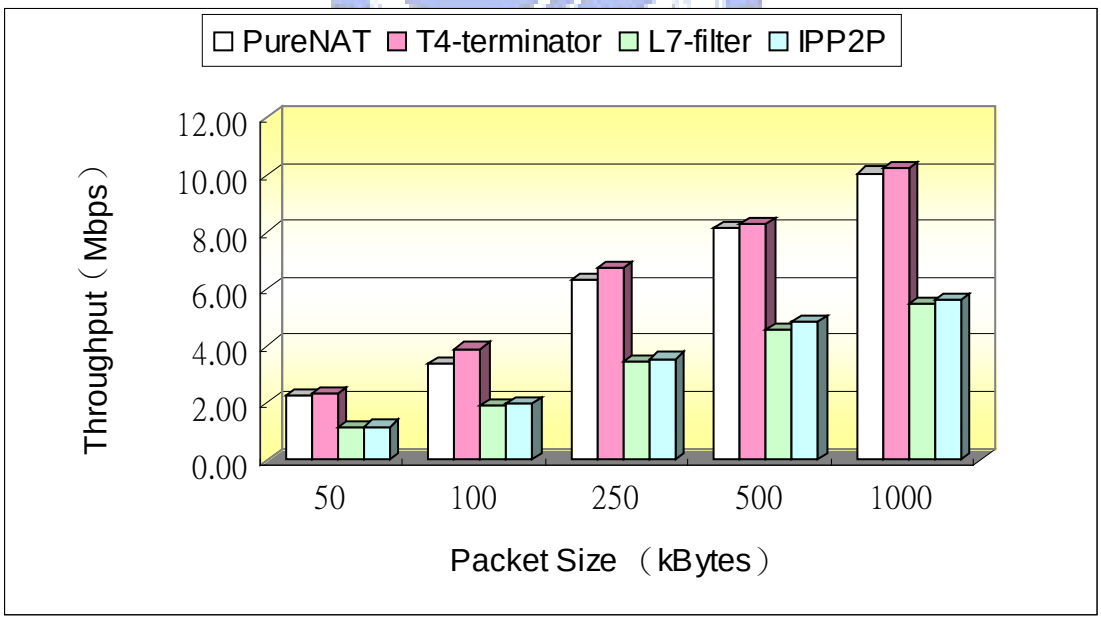

圖 5-5 Throughput of different Packet Size

5.3.2 Response Time

以 5 種不同 Packet Size 測試後，實驗結果呈現兩種排序：

1、Packet Size 為 2kB、4kB、32kB，反應時間由快至慢為：

PureNAT 、 T4-terminator 、 IPP2P 、 L7-filter。

2、Packet Size 為 8kB、16kB 時，反應時間由快至慢為：

T4-terminator 、 PureNAT 、 IPP2P 、 L7-filter。

這部分 PureNAT 與 T4-terminator 互有領先，各項實驗數據如以下圖表：

Response Time （msec）					
	2 kBytes	4 kBytes	8 kBytes	16 kBytes	32 kBytes
PureNAT	36.60	74.50	83.20	112.13	142.93
T4-terminator	38.60	76.07	75.40	111.67	151.33
L7-filter	72.30	141.90	143.10	214.70	284.50
IPP2P	71.00	126.20	142.00	207.80	267.00

表 5-7 Response Time of different Packet Size

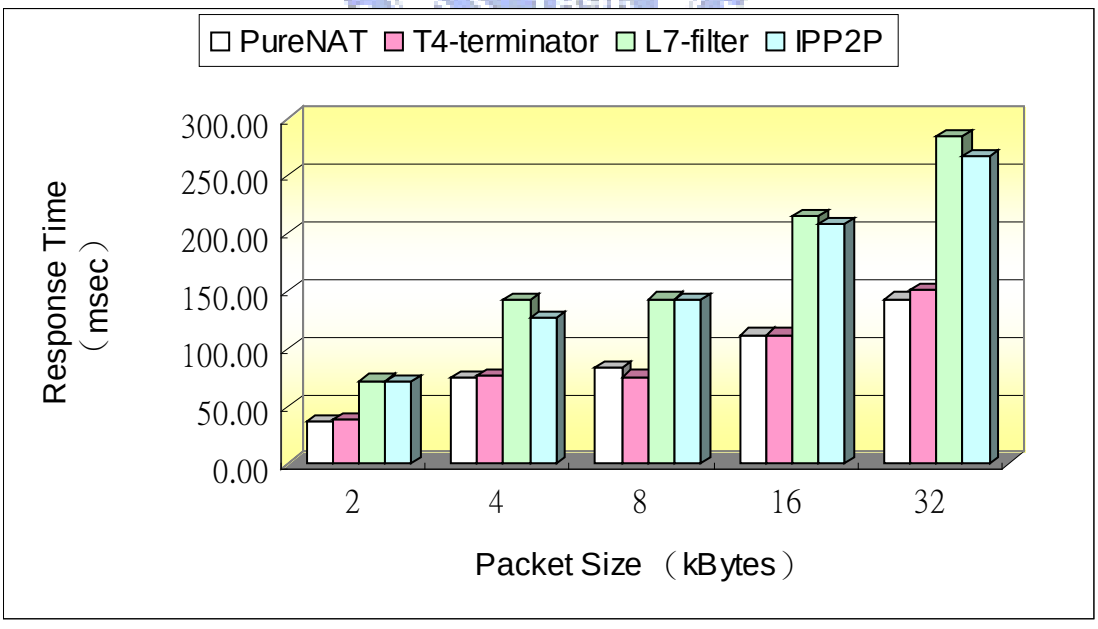

圖 5-6 Response Time of different Packet Size

5.3.3 CPU Utilization

為瞭解僅記錄 500 秒的 CPU 使用率是否具有代表性，因此本實驗進行前，先將預錄封包進行 500 秒的 Burst Mode Replay，記錄 500 秒之後各軟體重播次數如下表：

Application	Replay times
Foxy	9298
eMule	13274
BitTorrent	8761
MSN Messenger	34341
YAHOO! Messenger	33814

表 5-8 統計 500 秒 Burst Mode 封包重播之次數

由表 5-8 得知，原先各軟體錄製十五分鐘的封包，使用 Burst Mode Replay 在短短 500 秒內，重播次數由八千多次至三萬三千多次不等，可證明本實驗記錄 500 秒的 CPU Utilization 已具有足夠的代表性。

另外，為比較有無使用 Burst Mode 對 IPS 加壓的效果，本實驗再分成兩組，第一組按照封包錄製時的時間軸循序播放，第二組才以 Burst Mode 播放，其實驗結果如後詳述。

首先第一組進行封包依錄製時間軸播放的方式，以 sar 指令連續記錄 CPU 使用率 500 秒鐘後，由實驗圖表可看出雖然 L7-filter 及 IPP2P 的 CPU 使用率高於 T4-terminator，但是之間差異極微。主要原因來自依時間軸重播的封包，其固定單位時間內送出的封包數量甚少，尚不足以對 CPU 的運算能力構成壓力。

CPU Utilization（%）			
	Minimum	Maximum	Average
PureNAT	17.00	30.30	21.58
T4-terminator	17.17	30.30	21.40
L7-filter	19.00	32.00	22.93
IPP2P	19.00	30.69	23.04

表 5-9 CPU Utilization without Burst Mode

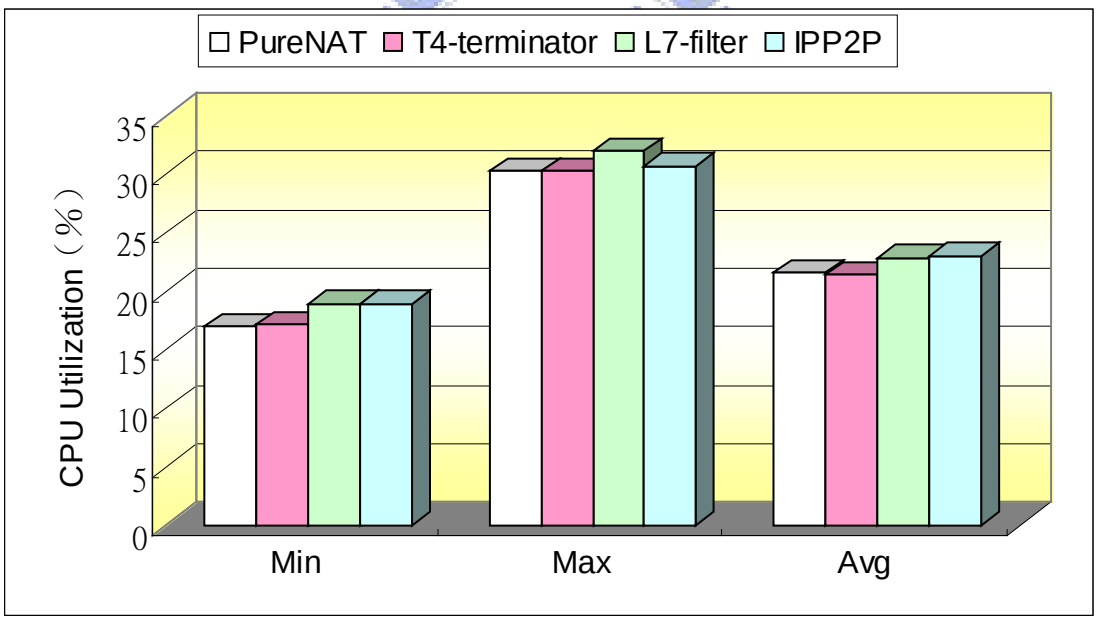

圖 5-7 CPU Utilization without Burst Mode

第二組以 Burst Mode Replay 再次實驗後，可看出 CPU 使用率均明顯提高，尤其以 L7-filter 最為顯著，高出同樣採用 Layer7 延伸比對的 IPP2P 不少。實驗數據顯示，無論最大值、最小值、平均值，其使用率由少至多排序均呈現：PureNAT ＜ T4-terminator ＜ IPP2P ＜ L7-filter。

CPU Utilization （%）			
	Minimum	Maximum	Average
PureNAT	41.00	55.00	45.97
T4-terminator	42.57	57.00	49.06
L7-filter	73.53	81.19	77.79
IPP2P	44.55	59.41	49.23

表 5-10 CPU Utilization with Burst Mode

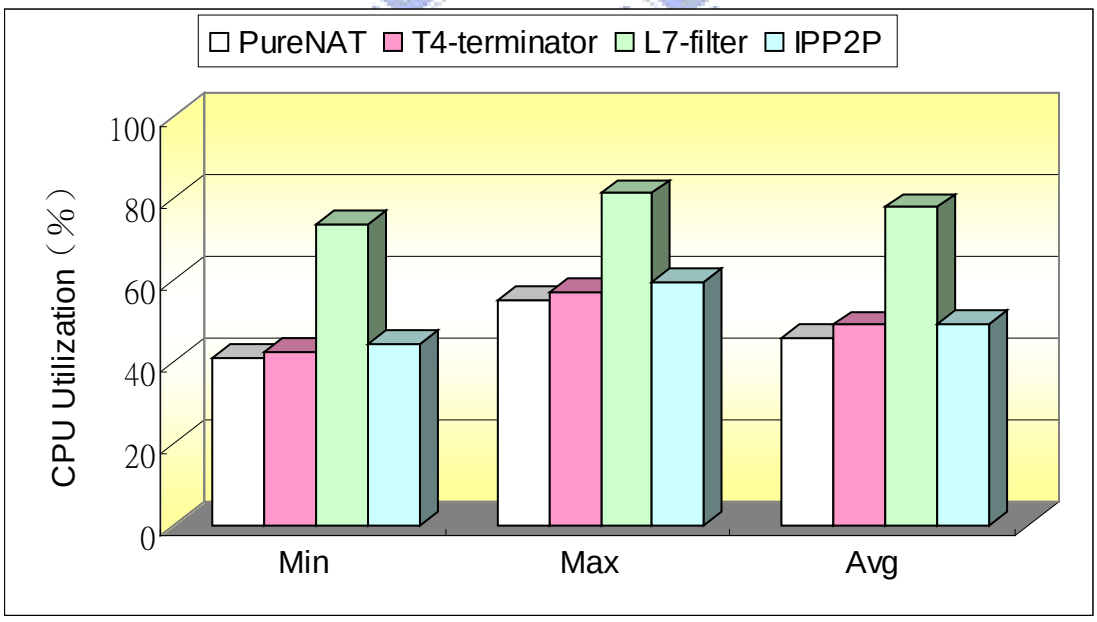

圖 5-8 CPU Utilization with Burst Mode

5.4 分析與比較

從各項實驗數據顯示，本研究提出的T4-terminator阻擋方式，縱使IPS規則同為8條，但是各項效能表現幾乎不亞於PureNAT，其中最大處理能力甚至優於PureNAT，原因在於PureNAT要對所有穿透封包進行目的端DNAT（Destination NAT）及來源端SNAT（Source NAT）封包偽裝轉換，核心必須記錄每個封包的正確的歸屬IP；而相對本研究的T4-terminator規則，第8條規則雖然必須消耗資源，將新的TCP封包資訊記錄到P2P_db這個檔案，但是當資料記錄符合「5分鐘內超過30條TCP連線」的條件後，後續封包就直接被核心丟棄，這些大量被丟棄的封包，可節省系統資源，不必再進行DNAT、SNAT動作。另外T4-terminator前6條規則的比對是採用Layer3的Port Number比對方式，對系統資源的消耗並不多，因此在這消長互現的狀況下，呈現出T4-terminator IPS極佳的處理效能。

而在L7-filter與IPP2P方面，兩者雖然都使用Kernelspace版，並採Layer7的Payload Signature檢驗，但各項測試數據均顯示IPP2P效能優於L7-filter，尤其CPU Utilization的表現與T4-terminator可說是不相上下。下表將研究成果配合其運作原理做一整理：

	L7-filter	IPP2P	T4-terminator
CPU Utilization	最高	很低	很低
Response Time	較長	中等	最短
Throughput	較低	中等	最佳
Kernelspace version	是	是	是
Userspace version	是	否	-
運作方式	Netfilter/IPTables Extension match	Netfilter/IPTables Extension match	Netfilter/IPTables
支援Kernel版本數量	多	少	-

支援IPTables版本數量	多	少	-
封包比對層級	Layer7	Layer7	Layer3/Layer4
Payload Signature 演算法	Regular Expression	16進位具體代碼	-
比對方式	每條連線前10個封包或前2048 Bytes	每個封包均比對	每個封包均比對
封包處理速度	慢	快	極快
辨識通訊協定數量	109(包含其他協定)	9(全是P2P協定)	-
特徵定義檔位置	Extension pattern file	Write in Code	-
自行定義協定的方便性	僅需修改pattern file,具彈性且方便	需修改source code，再重新編譯	-
Quality of Service/ Traffic Control	已結合conntrack，方便於Qos	不具conntrack，較不方便進行QoS	只能全禁止或全放行
對已連線且傳輸中的P2P flow能否禁止	效果不佳	效果不佳	立即生效
辨別加密連線或模糊演算	否	否	可
辨別未知協定	否	否	可

表 5-11 三種 P2P 主動防禦方式之比較

總結上表可看出3種防禦方式各有其優缺點：L7-filter更新快、支援新版系統核心、辨識範圍廣、擴充性高、容易進行頻寬控管，雖然執行效能稍低，但是使用上彈性很大，因此不少商用IPS都以L7-filter來作為產品的開發核心。IPP2P雖然執行效率極佳，但最大缺點在官網接近三年未發佈新版本，以致新版的Netfilter/IPTables均無法支援，自行擴充協定也不方便。T4-terminator的方式最大優勢在防火牆效能幾乎不受影響下，卻能夠有效封鎖P2P連線，但缺點在於「誤判」其他非P2P之連線，或者「漏判」少數採80 port傳輸的P2P flow。

此外L7-filter及IPP2P兩種Layer7特徵防禦方式，在實作時發現，對於已進行傳輸中的P2P flow，阻擋效果都不佳。亦即client端已經開啟P2P程式分享檔案後，無論再執行L7-filter或IPP2P規則，都無法有效中途抑止傳輸行為。原因在於兩者定義的封包特徵，絕大多數都在連線建立之初已溝通完畢，一旦連線建立後，交換的封包多為大型檔案資料，導致阻擋效果不佳，因此client端通常還是可以順利地把進行檔案交換。另外碰到加密、模糊演算、未知協定等P2P連線，兩種Layer7的辨識方法都無法加以防禦；但是上述問題，採用本研究提出的T4-terminator防禦方式，都可以迅速迎刃而解。但若網管人員不想全面禁止P2P flow，而是採用頻寬管理的方式，本研究的T4-terminator就無法順利達成。

第六章 結論與未來方向

近來P2P網路蓬勃的發展，隨之而來的頻寬問題、資通安全危機層出不窮；另外智慧財產權的法律訴訟亦在世界各國陸續點燃。各家ISP業者通常以QoS的管理方式以減低P2P對頻寬的衝擊，而我國教育部則是明文禁止各級學校禁用P2P軟體來下載非法檔案。

隨著P2P技術進步，傳統防火牆已經很難加以抵擋，必須透過Layer7的辨識系統進行防禦，然而此類的商用IPS要價不斐，對於經費拮据的國中小學來說，是一筆很大的負擔。本研究採用自由軟體實作一套IPS系統，以極低的成本達到P2P連線的主動防禦效果。

6.1 研究成果

1、仿真、免費的測試環境：通常商用網路設備的測試儀器，如SmartBits或IXIA動輒百萬，絕非中小學或一般企業可以負荷。本研究分析比較許多免費測試軟體之後，整合出一套適合於效能評估，且操作方便的仿真環境。本實驗環境不僅限於P2P的測試，同時也適用其他網路設備的效能測試。

2、迅速、有效的T4主動防禦：因為Layer7的比對需花費較多的系統資源，一旦區網電腦增加，或者流量變大，將使IPS必須維護更多的連線，以導致網路速度變慢；另者Layer7缺點在於對於封包加密、連線採用模糊演算技術，甚至未知特徵的P2P連線均無法正確判斷。本研究運用IPTables內建的connection tracking功能及recent模組參數，提出T4-terminator的防禦方式，對於加密、已連線並傳輸中、未知協定的P2P連線都可以有效阻擋。而且T4的方式不需要進行封包內容比對，可以大幅提升IPS效能。

3、精準、彈性的Layer7比對模組：無論L7-filter或者IPP2P都具有Open Source的特性，可讓使用者自行定義封包特徵，外掛在Netfilter/IPTables上面進行延伸比對，有效地阻擋P2P連線。

6.2 未來方向

本論文提出的方案可以有效阻擋目前常見的P2P連線，但尚有未臻完美之處，未來本研究可朝三個方向進行：

1、整合T4及Layer7，提升IPS效能及防禦準確率：本研究提出的T4防禦規則採用「寧可錯殺一百、也不可放過一個」嚴格規範，雖然P2P防禦效果極佳，但相對也會導致其他正常連線封包遭到丟棄，誤判率的問題有待探討。而放行WEB 80 port的方式亦可能讓少數P2P連線順利穿越IPS，因此網管者可視學校需要，修正防火牆內容，讓T4-terminator及Layer7的規則互相搭配，達到有效防禦的目的。

2、整合IDS及編寫CGI介面，提高安全性及便利操作：IPTables規則語法需要一段時間的學習才能上手，較不利網管人員交接，未來可利用PHP或Perl等語言撰寫操作介面，讓不熟悉語法的網管更方便操作；還可以整合其他IDS（Intrusion Detecion System）如SNORT的辨識規則，抵擋WAN端的各類攻擊，以及判斷LAN端的網路病毒的行為，搭配IPTables加以封鎖，提高資訊環境安全性。

3、運用負載平衡開發分散式IPS系統，維持網路品質：當LAN端電腦越來越多，或者網路環境採GigaNet高速傳輸時，單一台IPS的運算能力會拖垮整體效能，未來系統可以運用負載平衡演算法，將封包導向分散式系統進行平行處理，維持網路連線品質。

參考文獻

[1] Sen, S., O. Spatscheck, and D. Wang. *Accurate, scalable in-network identification of p2p traffic using application signatures*. 2004: ACM New York, NY, USA.

[2] Spognardi, A., A. Lucarelli, and R. Di Pietro. *A methodology for P2P file-sharing traffic detection*. in *Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on*. 2005.

[3] Thomas, K., et al., *Transport layer identification of P2P traffic*, in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. 2004, ACM: Taormina, Sicily, Italy.

[4] Xing, L., D. Haixin, and L. Xing. *Identification of P2P traffic based on the content redistribution characteristic*. in *Communications and Information Technologies, 2007. ISCIT '07. International Symposium on*. 2007.

[5] Bing-Heng Peng, H.-J.L., Huan-Yun Wei, *Performance Enhancement over Linux Content Filtering*. Journal of Information Technology and Applications, 2007. **2**: p. 157-163.

[6] Ying-Dar LIN, M., Po-Ching LIN, Meng-Fu TSAI, Tsao-Jiang CHANG,and Yuan-Cheng LAI, *kP2PADM: An In-Kernel Architecture of P2P Management Gateway*. IEICE TRANS. INF. & SYST, 2008. **VOL.E91–D**.

[7] Constantinou, F. and P. Mavrommatis. *Identifying Known and Unknown Peer-to-Peer Traffic*. in *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on*. 2006.

[8] Dedinski, I., et al. *Cross-Layer Peer-to-Peer Traffic Identification and Optimization Based on Active Networking*. 2005.

[9] Haoyi, W. and N. Ishikawa. *Design and implementation of a simulator for peer-to-peer networks: optimal-sim*. in *Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on*. 2005.

[10] Schollmeier, R. and G. Schollmeier. *Why peer-to-peer (P2P) does scale: an analysis of P2P traffic patterns*. in *Peer-to-Peer Computing, 2002. (P2P 2002). Proceedings. Second International Conference on*. 2002.

[11] 陳勇勳, *Linux 網路安全技術與實現*. 1 ed. 2008, Taipei: 悅知文化.

[12] *L7-filter Classifier*. Available from: http://l7-filter.sourceforge.net/.

[13] *IPP2P Classifier*. Available from: http://www.ipp2p.org/.
[14] *Linux Netfilter/IPTables framework*. Available from: http://www.netfilter.org/
[15] *IPTables Tutorial*. Available from: http://iptables-tutorial.frozentux.net/iptables-tutorial.html.
[16] *FOXY*. Available from: http://tw.myfoxy.net.
[17] *BitTorrent*. Available from: http://www.bittorrent.com/.
[18] *eMule*. Available from: http://www.emule-project.net/.
[19] *eDonkey*. Available from: http://www.edonkey2000.com/
[20] *EzPeer*. Available from: http://web.ezpeer.com/.
[21] *Kuro*. Available from: http://www.kuro.com.tw.
[22] *WinMX*. Available from: http://winmxworld.com/.
[23] *FastTrack*. Available from: http://developer.berlios.de/projects/gift-fasttrack/.
[24] *KaZaA*. Available from: http://www.kazaa.com/.
[25] *OpenNap*. Available from: http://opennap.sourceforge.net/.
[26] *SETI@Home*. Available from: http://setitaiwan.tripod.com/MIRROR/
[27] *CERNET,"China Education and Research Network"*. Available from: http://www.edu.cn/.
[28] *SNORT*. Available from: http://www.snort.org/.
[29] *Wireshark*. Available from: http://www.wireshark.org/.
[30] *Microsoft Network Monitor*. Available from: http://www.microsoft.com/downloads/.
[31] *ulogd*. Available from: http://www.netfilter.org/projects/ulogd/index.html.
[32] *Nulog*. Available from: http://software.inl.fr/trac/wiki/EdenWall/NuLog.
[33] *Colasoft Packet Player*. Available from: http://www.colasoft.com/packet_player/.
[34] *IxChariot Qcheck*. Available from: http://www.ixchariot.com/products/datasheets/qcheck.html.
[35] *IXIA Endpoint Library*. Available from: http://www.ixiacom.com/support/endpoint_library/.
[36] *RFC 1180 - A TCP/IP Tutorial*. Available from: http://www.ietf.org/rfc/rfc1180.txt.
[37] *RFC-Gnutella*. Available from: http://rfc-gnutella.sourceforge.net/.
[38] *V8 regular expression*. Available from: http://l7-filter.sourceforge.net/V8regex.
[39] *NS2*. Available from: http://www.isi.edu/nsnam/ns/.
[40] *NCTUns*. Available from: http://nsl10.csie.nctu.edu.tw/.