行政院國家科學委員會專題研究計畫　成果報告

在Linux作業系統上研究與實作以政策為基礎的網路資源與安全管理(I)

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-002-103-
執行期間：91年08月01日至92年07月31日
執行單位：國立臺灣大學資訊管理學系暨研究所

計畫主持人：孫雅麗

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 93年2月4日

行政院國家科學委員會補助專題研究計畫成果報告

在 Linux 作業系統上研究與實作以政策為基礎的網路資源與安全管理(I)

計畫類別： ■個別型計畫　　□整合型計畫

計畫編號： NSC91-2213-E002-103

執行期間： 91 年 8 月 1 日至 92 年 7 月 31 日

計畫主持人：國立台灣大學資訊管理學系　孫雅麗教授

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：國立台灣大學資訊管理學系

中　華　民　國 92 年 7 月 10 日

在 Linux 作業系統上研究與實作以政策為基礎的網路資源與安全管理(I)

Design and Implementation of the Policy-based Network Resource and Traffic Manager

計畫類別：一般型計畫

計畫編號：NSC91-2213-E002-103

執行期限：910801-920731

子計畫主持人：孫雅麗教授

執行單位：國立台灣大學資訊管理學系

一. 中文摘要

在 Internet **風行的今日，上網已經變成日常生活中不可或缺的活動。然而，網路資源與流量並沒有妥善管理，許多重要的** Traffic **被一些不重要的** Traffic **搶走頻寬，往往造成較重要之應用程式的服務品質低落；另一方面，也由於** Internet **的廣泛連結，使得一些有心人士可以利用這便利的網路來從事非法的勾當，散佈病毒，癱瘓伺服器，使得網路使用者承受極大的損失。因此，如何妥善的管理網路資源的運作以提高服務品質（**QoS**），與防範網路攻擊是目前網路研究的重要課題。**

在這個計劃中，我們提出一套在 Linux**上的架構**(architecture)**，能很快檢驗每一個流經之資料封包**(data packet)**，依照已定之政策（**policy**）做出適切的動作。這樣一個架構，可以很方便的提供給各式應用來使用，例如防火牆（**firewall**），網路安全閘道**(network security gateway), QoS router, VPN **閘道，網路流量監視器等，甚至同時提供一個以上之應用。如加上適當的硬體，就可以成為一個獨立的機器。這個架構裡面含有** policy-based **封包分類器**(packet classifier)**，佇列管理**(queue management)**與流量管理**(Traffic Policing**，例如** RED)**，以及封包排程器**(Packet scheduler)**，包括我們最近所提出可以提供最大流速控制的** WF2Q-M**。另外我們將研發一套網路監視系統來紀錄所有流過之封包，並提供給以上之應用機器做即時反應，並向網路管理者發出警告。最後我們也將探討把整個系統移植到嵌入式系統上的問題，以適應更多的應用環境，以及降低整體系統的成本。**

關鍵詞：Linux**、服務品質保證、網路流量管理、封包分類、封包排程、佇列管理、網路資源管理、網路安全、**IPv6**、**Mobile IP**、無線區域網路、嵌入式系統。**

Abstract

Current Internet does not provide any kind of QoS guarantee for the network applications. Therefore, the network resources could be wasted by some unimportant applications. On the other hand, the extensive connectivity of the Internet also provides the malicious crackers the great environment to distribute the computer viruses and overload the servers by DDoS attack. In consequence, how to appropriately manage the network resources to enhance the QoS and to prevent network attack are the top issues of network researches.

In this project, we propose a novel QoS router architecture based on Linux. In this router, packets are classified and treated according to the user-defined policies. This architecture comprises many components: policy-based packet classifier, queue manager such as RED, traffic policer, and the Weighted Fair Queue packet scheduler with maximum rate control (W2FQ-M). Under such architecture, many applications could be implemented, such as: firewall, network security gateway, QoS router, VPN gateway and network monitor. With appropriate hardware support, all functions could be implemented in the same machine. Besides, we will also develop a network monitoring system to watch all the packets passing the networks and provide information to the QoS router to act promptly to some unusual network conditions. Finally, we will discuss some issues about porting the system to embedded devices to lower the cost of total system.

二. 研究緣由、目的與成果

Ⅰ緣由

Linux**是一個功能強大的作業系統，而且由於其開放程式碼（**Open Source**）的特性，使得它在功能性以及穩定性方面都有極佳的水準。最近幾年，隨著網路的普遍使用以及資訊家電的崛起，**Linux **受到了大眾的矚目，許多廠商紛紛針對** Linux**來設計軟體，或是以** Linux **為基礎來發展嵌入式系統**(Embedded System)**，形成了一股** Linux**風潮，**Linux**儼然已經成為作業系統的主流。因此，鑑於** Linux**作業系統對於產業的重要性，深入瞭解** Linux**作業系統，並開發相關關鍵技術是個極為迫切的議題。鑑於目前網路的風行，網路資源的管理以及網路安全的維護越顯重要。故本計畫將以** Linux**作業系統為基礎平台，將本實驗室多年來在網路資源管理等相關議題的研究成果予以實作，並進一步研發** Linux**在網路應用上的支援。**

Ⅱ目的

現有的網路，尤其是對一個企業體而言，連外網路的頻寬通常相對於企業內部（enterprise intranet**）的頻寬相對小很多，如：**TANet **出國之頻寬、企業對外**

的網路等等。因此當資料匯集於對外出口時，壅塞隨即產生。因此，如何支援網路傳輸線頻寬共享 (Link Sharing)的理想就日益重要。換句話說，一個理想的網路要能夠提供一些機制，讓網路資源的共享可藉由政策考量，通訊協定及其資料型態或其他重要因素分類。它是一個以應用(application)、企業(enterprise)以及業務（business） 為導向的管理。過去的文獻資料顯示 Internet 的資料傳送效能（例如回應時間、throughput, etc.） 一直是不可預測。因此，如何提供使用者良好的上網傳輸品質與效能對網路的管理者是一大挑戰；這也是目前網路服務者最頭痛的問題之一。在這方面，相關議題包含網路傳輸品質的保證，最重要的是一個網路服務的提供者如何能保證他的客戶的 business-critical applications and traffic 能夠正確的、即時的傳遞到目的地。

除了上述的頻寬使用效率的問題之外，目前的網路由於 Internet 的普及化，雖然帶來了許多的便利，但也同時帶來了許多的危機。網路變成一些有心人士進行破壞的最佳利器。這些人利用網路散佈病毒，或是入侵他人的電腦竊取資料，甚至利用無辜者的電腦作為破壞的跳板。近來網路郵件病毒流竄，不但造成郵件伺服器效能大幅降低甚至當機，也使得使用者蒙受極大的損失；一些駭客（cracker）使用 DDOS 或是 DOS 來佔用網路頻寬、攻擊一些著名的網站、癱瘓伺服器效能，也會嚴重影響其他網路使用者的權力。由於目前連結上網際網路的電腦眾多，災害波及的範圍及層面也跟著加大。因此，能及早發現與阻擋這類攻擊是當前網路流量管理的重要課題之一。

因此，我們相信以檢驗每一封包內容之政策式階層式的頻寬共享 (Policy-based Hierarchical Link Sharing) 的架構是絕對必要的，因為組織的階層架構已經是一種常態，組織必須將網路資源供多個單位使用，而每個單位希望當網路發生擁塞時能得到保證的頻寬；相對的，當頻寬沒有被一單位使用而剩餘時，其剩餘頻寬應如何分配或是希望以何種方式供其他單位使用；甚至，可以去設定不同類別之間 traffic 的優先權，或是採用禁止、限制流量的方式來管制某些進入內部網路之中的 traffic，這些都是我們所謂的網路資源使用政策制訂的議題。傳統的存取控制列(access list)已不能滿足現今頻寬管理的需求，提供階層化的頻寬管理架構將使組織能更自然，更有彈性的管理網路頻寬。

III成果

A. Design Goals

在整個系統的設計及實作中，我們必須設計實作出幾個基礎元件才可以達成我們的目標。我們的系統架構圖如圖(一)：首先，一個進入系統的封包，必須要經過封包分類器（Packet Classifier）的分類，才能夠得知其符合哪個 Policy 規範；然後，流量管理器（Traffic Policer）和佇列管理器（Queue Manager）會根據 Policy 的規格來決定是否丟棄這個封包；若這個封包成功通過，則封包排程器會根據 Policy 的規定來決定此封包所將接受到的待遇，以提供服務品質保證。

B. 文獻探討

封包分類器（Packet Classifier）

傳統的 router 只需要檢查 Source IP Address 來決定封包的下個目的地；然而，我們的 QoS router 是以 Policy 為基礎，為了提供更精細的分類，封包分類器需要檢查多個 IP Header 欄位才能夠辨識出每個封包所屬的 Policy，如：Source address and port，Destination address and port，和 Protocol ID；甚至如果要分得更為細微，可能還需要檢查 IP 封包所攜帶的內容（content）。

對於 Router 而言，檢查如此多個欄位，甚至檢查封包的內容，是個極大的負擔。因為經過 Router 的封包數目可能成千上萬，一旦系統來不及將這大量的封包及時分類，則後端的封包排程器作得再好也無用武之地，故封包分類器（Classifier）的效能變成為了整個系統效能的瓶頸。再者，由於我們的系統是以 policy 為基礎的系統，所以 rule 不一定都是 exact match，而有可能是個 range match，或是 prefix match。更加深了封包分類器的複雜性。傳統的循序式搜尋法是絕對不適用的，因為此搜尋法會隨著 Policy 數目的增加而降低其效能。為了解決這個問題，我們使用了 Bit Vector 封包分類器。

Bit Vector[4]是一個概念相當簡單，以 trie 為基礎的分類器，其使用了 divide-and-conquer 的方式來解決封包分類的問題。它會將 Policy database 中所有的 rule 的每一個欄位（如 source/ destination 的 address/port 等等）各建構出一棵 binary tree。在 tree 中，每一個 node 都會有一個 Bit vector，而 vector 中的每個 bit 就對應到每一個 rule。由於 rule 不一定是 exact match，而有可能是個 range 或是 prefix match，所以一個 rule 可能同時擁有多個 node。

當封包進入系統，它會取出這個封包之中所需要的資訊，然後逐一或是同時到每個 binary tree 中搜尋，當每個 tree 的搜尋完成，即表示演算法停留在某個 node 上面，然後傳回這個 node 所擁有的 bit vector 來代表符合的 rule 有哪些。最後再將所有的 bit vector 作 AND 的操作，我們就可以得知符合這個封包究竟符合哪些 rule 的規範。

由此可知，Bit Vector 封包分類器有下列優點：

1. 快速：由於是以 binary tree 為基礎，幾乎所有的動作都是 bit-wise 的 operation，所以分類速度極高。再者，bit vector 還可以在細分為 hierarchy 的架構來減少搜尋 bit vector 的時間。而且，它本身具有平行處理的特性，如果硬體有支援，速度可以更佳。
2. 高延展性（Scalability）：不論有多少 rule，他所花費的時間幾乎都是相同的，沒有傳統循序式搜尋會隨著 rule 數目增加而效能減低的問題。以 IPv4 的 source address 為例，這個 binary tree 的深度

（depth）就是 32，在 32 次比較之內的時間裡一定可以得到答案。

封包排程器（Packet Scheduler）

為了同時滿足不同應用(Real-time application)服務品質需求與 Best-Effort Traffic 傳輸，以及彈性的(flexible)階層式頻寬共享，我們提出一套階層性網路頻寬共享架構：Integrated Services 的精神是針對每個 session 或 connection 保留特定頻寬，而 Differentiated Services[9][10][11]則對於不同的服務類別給予不同服務保證；但在我們的網路資源共享階層架構下，可以同時支援 IETF 所制訂的 Integrated Services[8]及 Differentiated Services，即網路管理者可針對單一連線(connection)做頻寬的保留，或對特定服務類別提出服務品質需求。圖 2 是此網路資源共享階層示意圖，每個封包會被分類到一個最底層的節點(leaf node)，接受其應有的服務品質。在此架構下，網路管理者可針對其內部所有路由器之介面做資源管理之政策設定。

我們底層的排程器(scheduler)是使用 WF2Q with Maximum rate control（簡稱為 WF2Q-M），故在每個階層(level)，每個節點在網路發生擁塞時，至少可享用它們所保留的最小頻寬保證；而對於受到最大傳送速度控制的服務類別而言，即使還有剩餘頻寬，它並不一定能用完全部的剩餘頻寬，因為它的最大傳送速度已受到限制。

傳統的 WF2Q 在網路發生擁塞時，可保證每個服務類別的最小保證頻寬；但在現今的網路應用中，只有保證最小頻寬是不夠的，在特殊情狀下，網路管理者即使在頻寬尚有剩餘時，仍想限制使用者的最大使用頻寬，其主要有以下三種動機：

1. 控制租線客戶最大網路流量。
2. 限制特定應用程式對外流量。
3. 限制分享剩餘頻寬。

除了最小保證頻寬，及控制最大傳送速度外，網路資源共享還需要優先順序(Priority)的機制，亦即當有優先順序較高的網路流量時，路由器就會先傳送優先順序較高的網路封包；當系統內都沒有優先順序比它高的網路流量時，路由器才會將它傳送出去。之所以網路流量區要區分優先順序，主要有以下兩個動機：

1. 使用者與網路應用程式之互動程度不同：使用者對不同應用程式的回應時間之敏感程度並不相同，因此，給予不同應用程式網路流量不同的優先順序是有其必要的，網路管理者可針對使用者對其反應時間比較敏感的應用程式之網路流量的優先順序設的比較高，如此一來其實際的反應時間就會較短。
2. 網路流量之重要程度不同：當網路發生擁塞時，重要的網路封包可以先行通過；因此，給那些較重要的網路流量較大的優先順序也是必要的。

因此，在我們的系統中，我們將 Priority 和 WF2Q 結合在一起，讓這個系統可以同時擁有這兩個排程器的優點。

佇列管理器（Queue Manager）

傳統對於佇列管理都是採用 Tail-Drop 方式來處理，也就是當新進的封包嘗試進入一個已經滿載的佇列中，它便會被丟棄。但是這個方法卻會對使用 TCP 的 traffic 造成嚴重的影響。由於 TCP 內部有複雜的壅塞控制（Congestion Control）的機制在運行，一旦封包連續地遺失，則 TCP 會自動調整其送出速率來減少壅塞持續的時間。而使用 Tail-Drop 機制的路由器在佇列滿載之後就不在允許任何封包進入佇列中，因此封包可能會被以連續的方式丟棄，如此一來 TCP 傳送端會誤以為網路發生嚴重壅塞而將速度降到最低，造成網路效能的低落。

為了解決這個問題，我們決定採用 RED（Random Early Detection）[12][13]的佇列管理方式，在佇列滿載之前就開始隨機地丟棄封包，避免封包被連續丟棄的情況發生。一旦佇列的長度超過最低門檻，系統就開始以機率來丟棄封包；一旦超過最高門檻，所以新進的封包全部會被丟棄。根據文獻上的探討，使用 RED 的確可以解決 Tail-Drop 對 TCP Traffic 造成的危害，提昇網路的效能。

預約保留及允入控制（Advance Reservation and Admission Control）

在我們的架構中，在對每種服務類別做保留頻寬時，都要經過 Admission Control 的控制，以確保所需要的保務品質在任何時間都可以被滿足。以往市面上的路由器或交換器(switch)，其頻寬的保留或服務類別的設定，都只限於即時設定（Immediate Reservation），亦即所有的設定於管理者設定後，立即成立，並不能提供預先設定或預約保留（Advance Reservation）[14]的功能，更不能提供週期性的預約（Periodical Reservation）。預約保留的功能將會受到越來越多的重視，主要有兩個原因：

1. 目前許多即時的網路應用程式是需要多人同時參與的。因此頻寬的使用變成計劃性的行為，故使用者必需事先預約保留頻寬，以確保在預定時間時有足夠的頻寬以供使用。
2. 許多頻寬的使用牽涉到週期性活動：現今許多企業網路是向 ISP 租專線使用，為了使其網路資源運作的更有效率，它們可能需要週期性的契約。另外，企業可能也希望在上班時間及下班時間採取不同的資源分享政策。因此提供週期性的預約保留是相當重要的。

故在我們的系統中的政策管理器，將提供即時保留、預約保留、週期性保留的功能。在管理者做完設定後，系統會判斷其為立即保留，或是預約保留，而馬上做檢查的工作，看頻寬是否足夠供此新頻寬的保留。如果管理者有使用週期性保留的功能，系統會檢查每個欲保留的時段，若有發生資源不足的情形，會馬上回應管理者，使其提前採取應變措施，以方便網路的管理。

Firewall

防火牆與QoS Router有許多共同的基本功能，如policy-based packet classifier。一般而言，Firewall對於網路內部用戶而言是透明的，資料一進一出，也就是說Firewall只是個Bridge，用來過濾網路封包用；但是，Router必須要將網路切割程許多子網路才能夠達到這個功能，因此影響範圍極大，終端用戶的網路設定必須更改，甚至連IP位址都必須更動。為了網路管理的方面，我們必須要把在IP層工作的QoS Router變成類似QoS Bridge Router[16]，提供通透性讓網管人員不需要去切割網路才能享用我們提供的先進功能。目前，雖然Linux已經有提供類似的功能，但是尚屬實驗階段。因此，我們可能需要去研究如何更動一些系統核心，讓這個功能更穩定。

另外，Firewall的政策設定和QoS的設定有些許的不同，使用的方式也不盡相同。如：QoS的政策為提供適當的服務品質給特定的封包；但是，就防火牆的需求來說，封包只有被丟棄或是允許通過兩個選擇。這是個政策原則的問題。如何將兩種政策完美地整合在一個統一的見面中也是個重要的議題，這方面可能需要深入的研究才能夠訂定出一個妥善的Semantic。

C. 實作

主要分成兩方面：在 Kernel 方面，我們已實作出 WF2Q-M 以及 Priority 封包排程器、RED traffic Policing 以及 Bit Vector 封包分類器。在 User Space 方面，我們開發出政策管理器供使用者設定政策，由於是 Web 介面，所以這邊使用到 CGI（Common Gateway Interface）和 Java Script 來撰寫，儘可能達到使網管人員可以對於現在的設定一目了然，以及輕易的就可以對 QoS Router 的做出最佳的設定。另一方面，我們還撰寫了中介程式來作為政策管理器和核心程式溝通的管道，這程式需要提供允入控制以及預約保留的功能，因此以 Multi-thread 的方式來撰寫，以達到最佳的程式執行能力。以下分別說明我們已經完成之政策管理器及 QoS Router 相關的核心元件：

I. 政策管理器（Policy Manager）：這裡包含了使用者介面以及政策資料庫的維護、和 QoS Router 的程式介面；為了簡化網路管理員的操作，政策管理器必須提供 Web 介面。在本計劃中，我們依功能不同而設計了不同的 Web 介面給網路管理員使用，如圖 3,4,5,6，在本系統裡，提供了一個樹狀結構供網管人員可以輕易的明瞭各 Interface 和各 policy/rule 之間的關連性。

II. QoS Router Prototype 的核心元件：除了位於 Kernel 的封包處理器、封包排程器之外，必須存在一個中介程式來作為政策管理器與 QoS Router 核心程式溝通管道。由於封包處理器、封包排程器都必須實作於 Linux Kernel 之中，這個位於 User Space 的中介程式必須具備存取 Kernel Space 資料的能力，此外允入控制以及預約保留的機制也由這個程式提供。

於本系統中，封包分類器可過濾出我們所不想要之封包，將其丟棄，而對於符合特定 policy 的封包，則移至封包排程器進行排程，並於適當時機送出，以符合網管人員對於 firewall 及 QoS 的要求。

四、參考文獻

[1] The Linux Homepage, http://www.linux.org/

[2] M Beck, et al. , “Linux Kernel Internals, Second Edition”, Addison Wesley 1997.

[3] Alessandro Rubini, “Linux Device Driver”, O’ Reilly 1998.

[4] T.V. Lakshman and D. Stiliadis, “High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching”, ACM SIGCOMM 1998, Aug 1998.

[5] A. K. Parekh and R. G. Gallager, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case”, IEEE/ACM Transactions on Networking, Vol. 1, No. 3, pp.344-357, June 1993.

[6] J. C. R. Bennett and H. Zhang, “WF2Q: Worse-case Fair Weighted Fair Queueing”, INFOCOM’ 96, 1996, pp.120-128.

[7] H. Zhang, “Service Disciplines for Guaranteed Performance Service in Packet-Switching Network”, Proc. IEEE, Vol. 83, October 1995, pp. 1374-1396.

[8] J. Wroclawski, “Specification of the Controlled-Load Network Element Service”, IETF RFC 2211, September 1997.

[9] S. Blake, et. al., “An Architecture for Differentiated Services,” IETF RFC2475, December 1998.

[10] J. Heinanen, et. al., “Assured Forwarding PHB Group,” IETF RFC2597, June 1999.

[11] V. Jacobson, et. al.,”An Expedited Forwarding PHB,” IETF RFC2598, June 1999.

[12] Floyd, S., and Jacobson, V., “Random Early Detection gateways for Congestion Avoidance”, Vol.1 No.4, p. 397-413, August 1993.

[13] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, “Recommendations on Queue Management and Congestion Avoidance in the Internet”, RFC 2309. Internet Engineering Task Force (IETF), April 1998.

[14] Olov Schelén, Andreas Nilsson, Joakim Norrgård, Stephen Pink, “Performance of QoS Agents for Provisioning Network Resources”, Internet Workshop on Quality of Service 1999, June 1999.

[15] Cisco, “Netflow”, http://www.cisco.com/warp/public/732/Tech/netflow/.

[16] Peter Breuer, “Linux Bridge+Firewall Mini-HOWTO version 1.2.0”, http://www.linuxdoc.org/HOWTO/mini/

Bridge+Firewall.html.

[17] S. Deering and R. Hinden. “Internet Protocol, Version 6 (IPv6) Specification”. RFC 2460, IETF, December 1998.

[18] S. Hinden and S. Deering. “IP Version 6 Addressing Architecture”. RFC 2373, IETF, July 1998.

[19] T. Narten, E. Nordmark, and W. Simpson. “Neighbor Discovery for IP version 6 (IPv6)”. RFC 2461, IETF , December 1998.

[20] S. Thomson and T. Narten. “IPv6 Stateless Address Auto-configuration”. RFC 2462, IETF , December 1998.

[21] Han-Chien Chao,Yen-Ming Chu and Mu-Tai Lin,“Cellular Mobile IPv6: The Implication of the Next-Generation wireless Network Design,” IEEE Transaction on Consumer Electronics, vol.46, Issue.3, pp. 656 –663, August. 2000.

[22] Worrall, K.P. ,” *The impact of IPv6 on wireless networks*”, 3G Mobile Communication Technologies, 2001. Second International Conference on (Conf. Publ. No. 477) , 2001

[23] ANSI/IEEE. 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 1999.

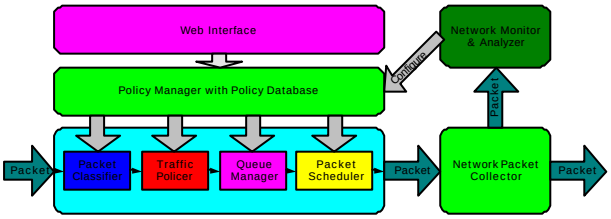

Figure 1. **系統架構圖**

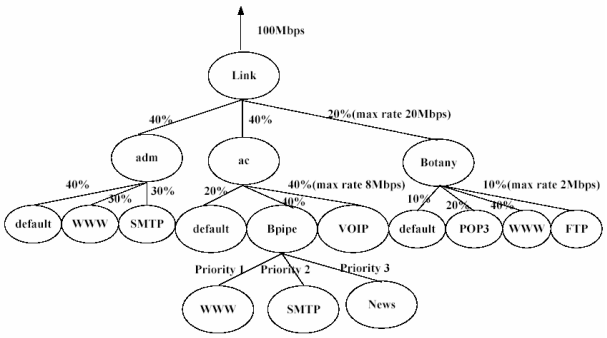

Figure 2.**網路資源共享階層示意圖。**

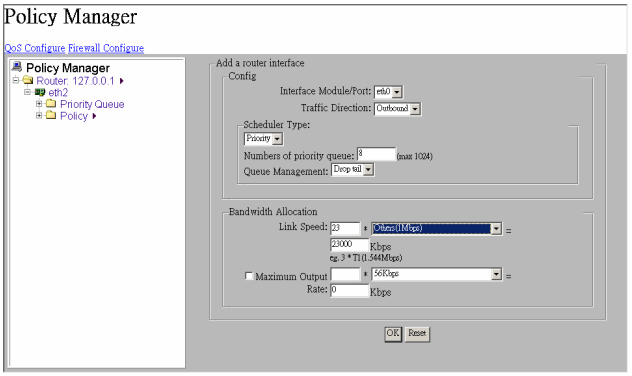

Figure 3. QoS Router **設定** Interface **介面**

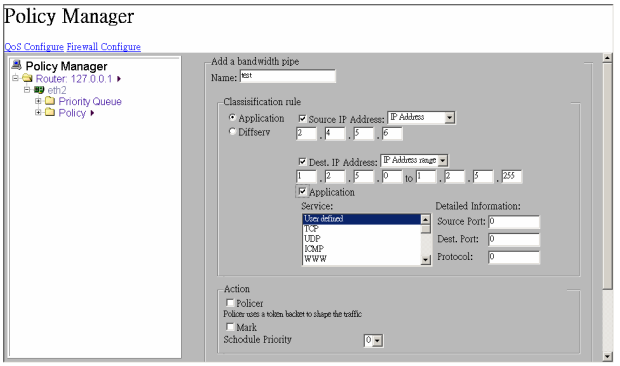

Figure 4. QoS Router **設定** policy **之介面**

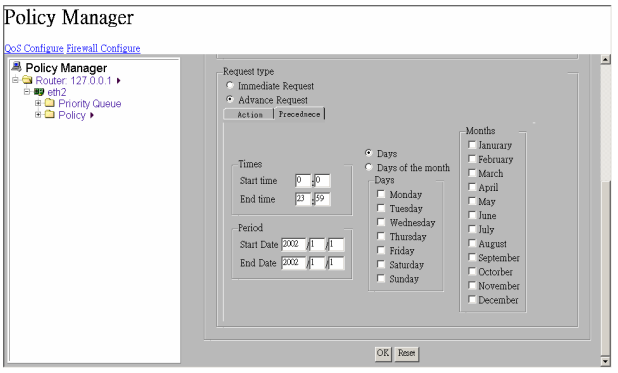

Figure 5. QoS Router **設定** Advance Request **介面(可設定某特定時段或是某些固定時段執行該** policy)

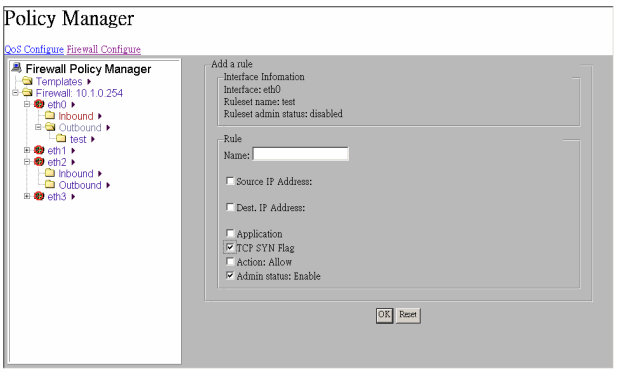

Figure 6. Firewall **設定** rule **之介面**