IP 層網路服務品質測試

邱世華
中國文化大學資訊管理研究所
lchou@staff.pccu.edu.tw

蔡敦仁
中國文化大學資訊管理研究所
drtt@mail.pccu.edu.tw

摘要

目前正值網路交易時代，電子商務為各企業帶來無數的商機，但也同樣帶給現有的企業網路架構前所未有的衝擊，資訊傳遞的即時性及對資訊分類賦予不同之等級也愈顯重要。對 IP 網路上之服務品質(QoS)重視之程度日益增高，本論文將為 IP 層之 QoS 方案提出較佳的測試流程並討論現行方案所可能存在之問題。

關鍵字： RSVP、DiffServ、網路頻寬管理、服務品質(QoS)

一、前言

目前正值網路交易時代，網際網路為各企業帶來無數的商機，但也同樣帶給現有的企業網路架構前所未有的衝擊，因為電子商務不論是在企業間(B to B)或是企業與顧客間(B to C)的銷售行為，都是必需要建立在一個完善的網路架構上，也因此有許多的 IDC(Internet Data Center)成立，為的就是要提供各企業一個完善的網路空間。但即使是 IDC 產業，一樣只能提供一個無法保証的頻寬品質，如何將企業網路的品質提升到最大的效能是目前極大的挑戰之一。而 IP 層的網路服務品質(Quality of Service, QoS)便是以此種理念為目標而發展的，它早在 1997 年即被提出，但在實際的網路應用中，仍少有系統且完整的測試，對現行的網路架構上使用 QoS 尚未確切得知可能潛在的問題，本研究主要的動機即在此。

二、QoS 現行之解決方案

原有的 Internet 並未支援服務品質：像不同的服務分類、提供特別的支援給一些無法接受傳輸延遲的應用、對點對點的服務品質提供保証等，在 Internet 上原僅能提供最基本的服務品質—盡力服務(best-effort service)[3]，但隨著網際網路的發展，越來越多的應用程式需要不同等級的服務，像需要即時服務(real-time service)服務品質的網際網路電話、視訊會議等出現。為因應這些不同應用層的需求，Internet 之 QoS 也呈現出多元化的發展，目前大致可分成三種不同類型[1]：

1. 預測型的 QoS (predictive QoS)
2. 流量基準型 QoS(flow based QoS)
3. 非流量基準型 QoS (non flow based QoS)

(一)RSVP(resource ReSerVation Protocol)

流量基準型(flow based QoS)的方式是以佇列規約(queueing disciplines)來為訊流(flow)預留所需的頻寬，目前常用的佇列規約有以下幾種[2]：

1. CBQ(class base queueing)
2. PFIFO_Fast(Packet First-In-First-Out_Fast)
3. SFQ(Stochastic Fairness Queueing)
4. TBF(Token Bucket Filter)
5. RED(Random Early Detect)
6. IPQ(Ingress policer qdisc)
7. WRR(Weighted Round Robin)
8. DSMARK(DiffServ MARK)

RSVP 的設計上看起來只能適用於一般中小型的網路架構，無法使用於像 ISP 或骨幹網路等大型的網路架構。有鑑於此 IETF 組織在 1998 年 12 月時又提出另外一種解決方案，亦即下一節將討論的差別式服務(Differentiated Service)，此方案主要是將大部份的工作移至邊緣路由器(edge router)及頻寬代理器(bandwidth broker)上，讓核心路由器(core router)專心的處理資料封包轉送的工作。

(二) DiffServ(Differentiated Service)

DiffServ 是由 IETF Differentiated Service Working Group 所提出的一種網路流量控制機制，屬於非流量基準型(non flow based QoS)。它並不能提供端點對端點(end-to-end)的網路服務品質保証，只是依流量類別的不同讓優先權較高的訊流比優先權較低的訊流有更多的網路頻寬資源可用。它依 IPv4 封包標頭中的 TOS(type of service) byte 或 IPv6 中 traffic class(IETF 一開始定義為 Priority)欄位值來區分服務的類型，其中前面六個位元的部份，簡稱為 DSCP(DiffServ code point)，DiffServ 便是依此標籤決定服務的等級[4]。

DiffServ 在網路間的傳送是透過外圍的 edge router 和內部 core router 做適當的工作分配，以達到分工的目的，edge router 主要是負責標記(marking)-寫入 TOS 來替換成 DSCP 及分類(classifier)的工作，core router 則是以 edge router 處理過的 flow 進行整流(shape)及丟棄/控管(drop/police)的動作(如圖 1 所示)。

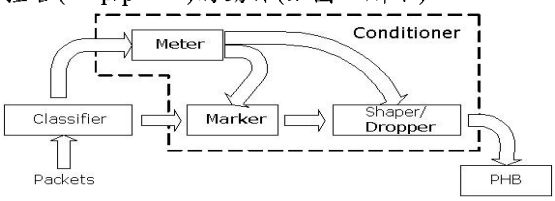


圖 1[5]

（三）現今較常見到的測試方式

因為仍無普遍支援在 IP 網路上實行 QoS 的方式，因此目前可找到的 IP QoS 測試大都屬於單純的測試，都只針對部份功能作測試，未提供完整且有效的測試模式。如 Linux 中的 iproute2(本研究所使用之軟體)的使用者在 LARTC 群組[6]的討論區中相互檢討其設定及產生出的結果數據，但大都是零星及隨興的一般測試，或單單在個人所服務公司需求的條件下所進行之測試，其中只有少數曾做整理，像 http://users.belgacom.net/staf/[7]，將個人所測試的結果簡單的報告，但並未做有效的整理。

三、環境建置

（一）測試環境

目前可在 IP 層架構提供 QoS 功能且較常見的有 ALTQ 及 iproute2 兩套 QoS 管理及設定工具，在 Windows 2000 Server 版本中雖然也有提供類似 QoS 的功能(QoS management)，但在測試後發現其功能僅能協助提供最陽春的頻寬限制，也就是一般最常見之以鎖 IP 的方式來控制頻寬，因此在此測試中並不考慮以此種方式作為 QoS 的測試。ALTQ 是針對 BSD(像 Free BSD、NET BSD、Open BSD)系統所設計的，另外一套 iproute2 則是以 Linux 平台為主(像 Open Linux、Suse、RedHat)，而因為此測試環境是在以 Linux RedHat 7.0 為系統平台下作業，且 IETF 所公開之測試軟體亦是以 iproute2 為 QoS 管控工具[2]，因此我們採用 iproute2 套件作為測試工具。我們將測試主要分為兩大步驟(因為在 QoS 功能上可分為 IntServ、DiffServ 兩種最主要的分類)，分別以 RSVP 中其中一種 CBQ(class base queuing)模式及 DiffServ 的主要機制—以 DSCP 取代 TOS byte 的方式，對頻寬及封包中的 header 做標記(mark)動作以觀察其結果並測量其效能。而使用做測量流量變化的，則是一套 TTT(tele traffic trapper)的工具，它是一套即時監控遠端流量並加以圖形化的軟體，選擇此套工具的主要原因是因為此套軟體有監控流量歷史記錄的功能，對整個頻寬即時的變動可以有較明顯升高或滑落的趨勢圖，在測試的過程中可以藉由歷史記錄找到頻寬轉變的關鍵點，此外它還可以在同一圖形表現不同 IP 或 protocol 的趨勢；另外封包截取則使用 Sniffer，目的在於確定 TOS byte 會被更改為較新的 DSCP 格式(符合 AF 值[8])，且儲存在 flow label 的欄位[9]，詳細的測試清單及測試環境請參考圖 2 和表 1。

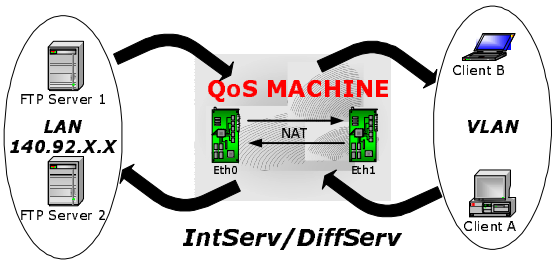


圖 2[10]

表 1　QoS Test Bed

QoS Machine(NAT Server)				
CPU	RAM	網卡		HD
PENTIUM II-MMX 300	64MB	NE2000(ISA)*2		2.5GB
OS	Kernel	Monitor Tool	Traffic Control	NAT Tool
Linux RedHat 7.1	2.4.2-2	ttt、sniffer	Iproute2	iptables

Client A(PC)		
CPU	RAM	網卡
PENTIUM 200	64MB	Realtek RTL8139
HD	OS	Kernel
6GB	Linux RedHat 7.1	2.4.2-2

Client B(Notebook)		
Model	CPU	RAM
ASUS L8400B	PENTIUM III 500	127MB
網卡	HD	OS
D-Link DFE 530TX	13GB	Windows 98

由圖中可看出測試區中只對兩 router 間的區域做 QoS 的控管，其理由是由於 QoS 機制只會對 outbound 的 flow 加以控管，而並不會對 client A 或 client B 兩端之封包加以處理。因此只有在兩 router 間傳送封包或 router 傳送封包給 client A 及 client B 的時候才會受到 RSVP 或 DiffServ 的影響(改變其 TOS byte 的值)。真正屬於 QoS 的範圍是只有兩 router 間流動的封包。測試的過程可分為幾個步驟：

一、Sniffer 與 TTT 流量監控之比較。

二、單純的 CBQ 功能測試。

三、單純的 TBF 功能測試。
四、CBQ 參數調整結果。
五、頻寬侵占行為測試。
六、多點同時傳送之 CBQ 功能測試。
七、保留頻寬效能測試。
八、CBQ Class 樹狀結構測試。
九、TOS Byte 轉為 DSCP 之格式轉換測試。

（二）測試過程

1. Sniffer 與 TTT 流量監控之比較

檢驗測試工具在實驗中之真確性是必要的。不論是哪一套軟體，在記錄流量的歷史必需要呈現真確的整體趨勢，也就是同樣的流量經由不同但皆宣稱具真確性的軟體記錄後，即使結果的大小有些微的差距(可能是因記錄時的時間區間或預設的封包計算單位不同所導致)，但它們趨勢線的起伏應該是一致的，如此才可証明所截取的流量趨勢是正確的。因此第一個實驗步驟首先要針對流量的監控軟體加以比較，將實驗中所要使用的 TTT 軟體與目前市場上較為人熟悉的 Sniffer 比較其流量記錄後的趨勢線，以確定 TTT 記錄流量趨勢的真確性。測試方法為在 Client A 向 FTP Server 1 抓取檔案時，同時以 TTT 和 Sniffer 監控其歷史流量，結果分別如圖 3-2 及圖 3-3 所示。

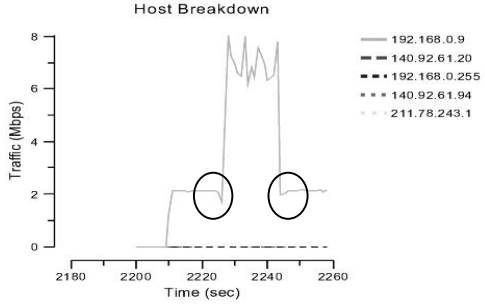


圖 3　TTT 流量圖

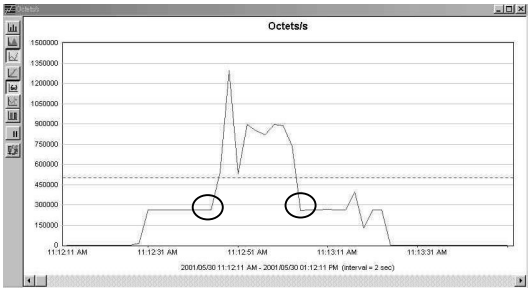


圖 4　Sniffer 流量圖

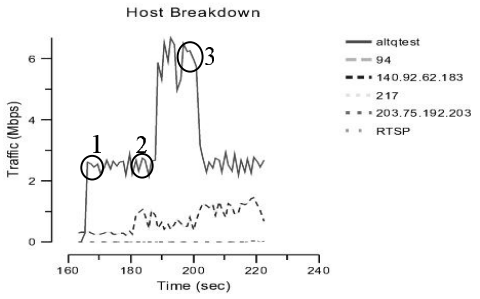
由圖 3 及圖 4 中可知兩軟體所得到的歷史趨勢圖結果是相當一致的，且由兩圖中的轉折點可得知兩者流量瞬間變化的時間點是相同的。在測試中只簡單的使用 CBQ 中限制頻寬功能(2Mbps)的有無，來凸顯出其高低起伏的曲線，並未考慮其它因素的影響。

2. 單純的 CBQ 功能測試

在此測試過程中，只單純的從 Client A 向 FTP Server 1 請求資料，以確定 CBQ 的運作狀態。其測試參數限制及結果報告如下(表 2 及圖 5、圖 6)：

表 2　CBQ 測試參數設定

avpkt (平均封包大小)	整體頻寬	速率(rate)	備考
1000 (byte)	10Mbps	2Mbps	在所有的測試過程中全部整體頻寬都是 10Mbps



1→執行 CBQ　2→解除 CBQ　3→執行 CBQ

圖 5　CBQ TEST 1 (CBQ avpkt=1000)

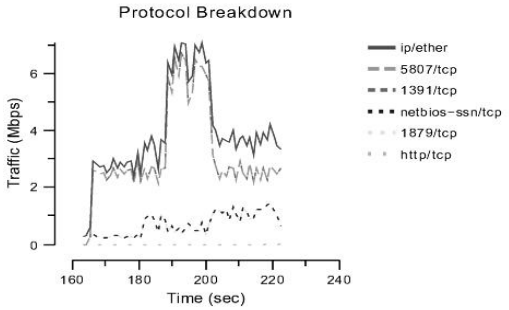


圖 6　CBQ TEST 2 (CBQ avpkt=1000)

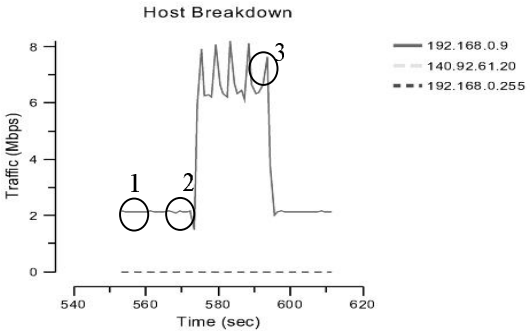
從測試的結果可以看出實體的頻寬確實有受到 CBQ 的參數所影響，但在參數中所設定的速率(rate)為 2Mbps，但結果卻是 2.4Mbps 左右，明顯的比設定值要來的高。因此下面兩個步驟是將 Queueing 的方式改變(改為 TBF)及在參數部份稍做調整，以觀察結果是否有所不同。

3. 單純的 TBF 功能測試

此測試的目標在於確定速率之所以會有差異的原因是否是出在 CBQ 本身參數的部份，因此採用 Queueing Discipline 中另一種單純限制頻寬的 TBF Queueing，表 3 及圖 7 即為測試參數限制及結果。

表 3　TBF 測試參數設定

avpkt (平均封包大小)	整體頻寬	速率(rate)
1000 (byte)	10Mbps	2Mbps



1→執行 TBF 2→解除 TBF 3→執行 TBF

圖 7　TBF TEST 1 (TBF avpkt=1000)

從實驗中發現當 CBQ 及 TBF 的 avpkt 同樣是 1000 的參數設定下，TBF 的速率是正確的 2Mbps，因此下一步驟將會將 CBQ 的 avpkt 調整後做測試。

4. CBQ 參數調整結果

此步驟是希望可找出影響 CBQ 速率不正確的參數，並加以改進，其測試限制及結果報告如下(表 4 及圖 8、圖 9、圖 10)：

表 4　CBQ 測試參數調整設定

avpkt (平均封包大小)	整體頻寬	速率(rate)
300、500、800 (byte)	10Mbps	2Mbps

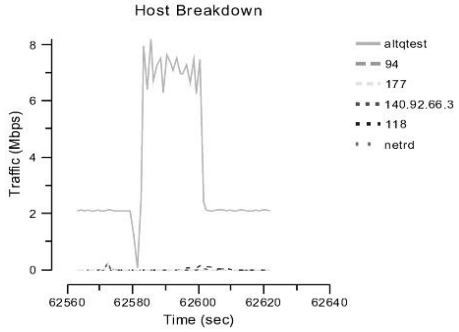


圖 8　CBQ TEST 3 (CBQ avpkt=300)

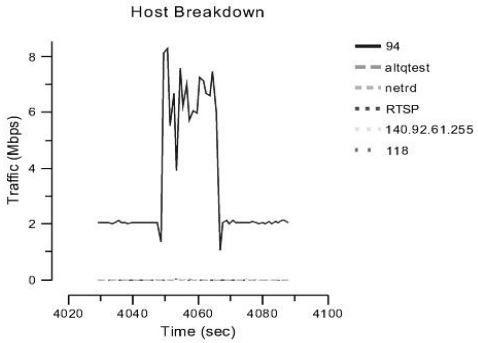


圖 9　CBQ TEST 4 (CBQ avpkt=500)

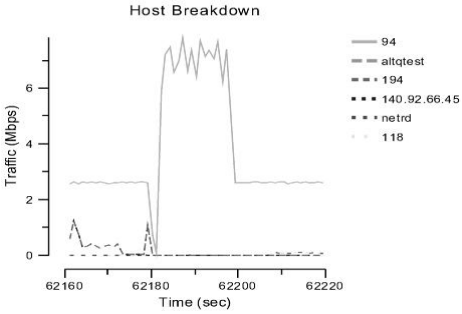


圖 10　CBQ TEST 5 (CBQ avpkt=800)

從測試的結果中發現在平均封包大小等於 500 的時候其 CBQ 的精確度最高，可以最接近 2Mbps 的速率達到頻寬的限制，大於或小於 500 的環境下都會讓精準度失準。而且當全部頻寬為 4Mbps 、平均封包大小等於 500 時測試的結果仍然相同(如圖 11)，精準度依然存在。但之前 TBF 以平均封包大小等於 1000 的情況下並未改變其精準度，因此在此多做了一次當 TBF 的平均封包大小等於 500 時看是否會影響其速率，其結果如圖 12。

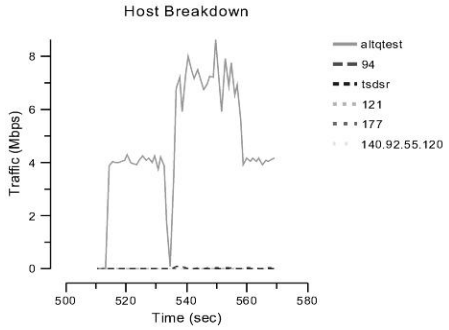


圖 11　CBQ TEST 6 (CBQ avpkt=500；Rate=4Mbit)

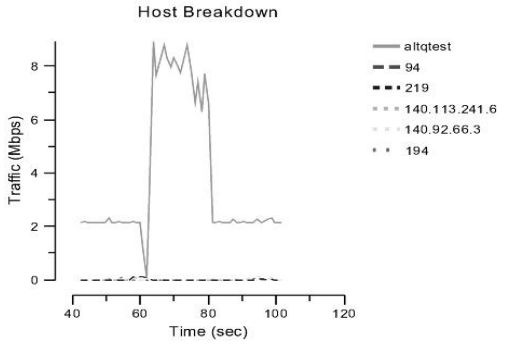


圖 12　TBF TEST 2 (TBF avpkt=500)

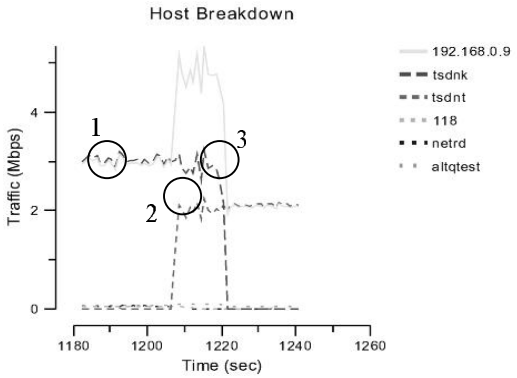
由測試的結果發現 TBF 並不會因為平均封包大小有所改變而影響其精準度。但因為 TBF 僅能提供頻寬限制，因此接下來的實驗都會在 CBQ 的平均封包大小等於 500 的情況下操作。

5. 頻寬侵占行為測試

因為 CBQ 是一種樹狀結構的 Queueing，因此同一層的 Class 的頻寬不一定只有一種，可以有很多不同的 Class 存在。而在 CBQ 中之 bounded 的參數，目的在保護每一個 Class 有不可被侵占的頻寬，所以這部份的測試主要是想知道當不同頻寬的 Class 並存時，是否會侵占其他已釋放資源之 Class 的頻寬。測試的方式是將頻寬分成兩個 Class(Class1、Class2)，頻寬限制分別為 2Mbps 和 3Mbps，而 Class1 等 Class2 快結束時才進行(在此測試是等 Class2 進行到 80%)，如此可較易看出有無侵占的行為，其參數部份及結果如表 5 及圖 13。

表 5　CBQ 測試 bounded 參數設定

avpkt (平均封包大小)	整體頻寬	Class1 下傳檔案大小	rate (Class1)	Class2 下傳檔案大小	rate (Class2)
500 (byte)	10Mbps	34MB	2Mbps	18MB	3Mbps



1 → Class2 進行傳輸　　2 → Class1 等 Class2 進行到 80%時開始傳輸　　3 → Class2 傳輸完畢

圖 13　CBQ TEST 7 (CBQ avpkt=500)

從圖 13 中我們可以發現當 Class2 傳輸完畢時，並未影響到 Class1 傳輸的速率，可知 bounded 參數發揮了作用，可有效防止 Class 間頻寬侵占的問題。

6. 多點同時傳送之 CBQ 功能測試

這部份的測試是希望能看出當頻寬限制在某種程度大小的情況下，檔案的多點傳輸是否會正常的平均分攤現有的頻寬。測試的過程如下：

(一)用一台 FTP 的 Client 端向兩台 FTP Server 同時各抓一個 38MB 大小的檔案。

(二)在傳輸到一半時以 CBQ 的方式將總頻寬限制在 3MB，並限定不可超過 3MB。

(三)當傳輸在限制頻寬下傳輸一段時間後再將限制取消，看是否會正常回復為原有頻寬。

所得結果如圖 14 及圖 15 所示：

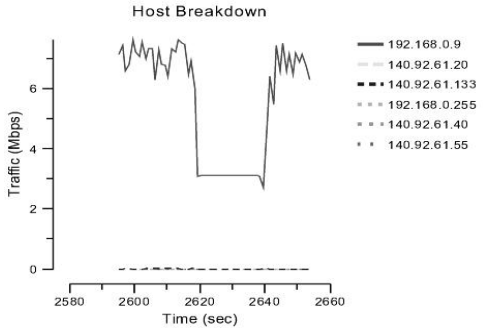


圖 14　CBQ TEST 8 (CBQ avpkt=500)

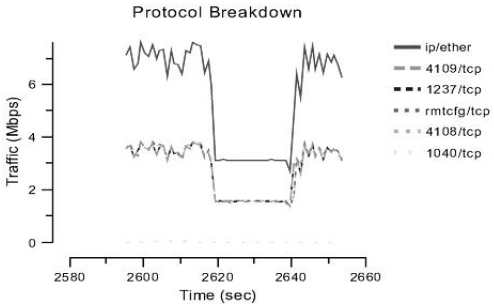


圖 15　CBQ TEST 9 (CBQ avpkt=500)

在圖 14 和圖 15 的比對中可清楚的看出不論總頻寬是原來的 10Mbps 或是限制後的 3Mbps，兩傳輸速率一直保持非常好的平分資源的角色。

7. 保留頻寬效能測試

不論是在 RSVP 或 DiffServ 中最重要的都是研究要如何保證所預留頻寬之優先權要比任何其它的訊流都要高，因為唯有如此才可給使用者有保障的速率，以遵循 SLA 中所簽定的內容。所以在這一部份主要的目標就是要測試在大量的流量中加入單一的頻寬限制可否達到頻寬預留的效果，其測試程序如下(測試環境請參考前圖 2)：

(一)Client A 向 FTP Server 1 下載一個 38Mbit 的檔案；另外開四個視窗向 FTP Server 2 分別下載 25Mbit、25Mbit、17Mbit、68Mbit 檔案。

(二)在傳輸到一半時以 CBQ 的方式預留 3Mbps 的頻寬給向 FTP Server 1 下載檔案的服務(Client A)。

(三)當傳輸在預留頻寬的服務下傳輸一段時間後將預留的服務取消，看是否可釋放出原來預留的頻寬。

結果如圖 16 及圖 17 所示：

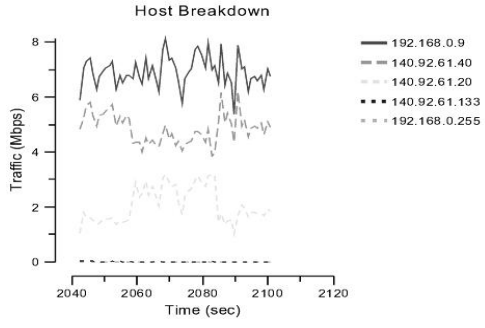


圖 16　CBQ TEST 10 (CBQ avpkt=500)

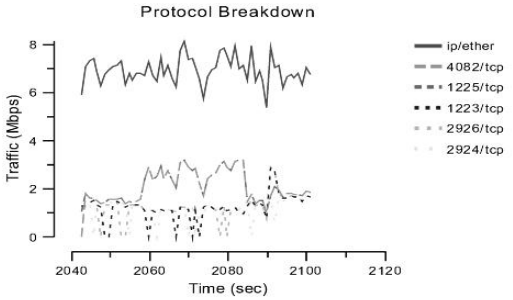


圖 17　CBQ TEST 11 (CBQ avpkt=500)

在圖 16 中，Client A 原本是和其餘的四個服務(FTP Server 2)平分其頻寬資源，因此在 1.5Mbps 的速率震盪(相當於 8Mbps/5 的值)，當 CBQ 為 Client A 預留 3Mbps 的同時，Client A 即立即享有特殊的服務-獨立的 3Mbps 頻寬，其頻寬不會和別人共享，也因此另外的四個服務變成共享 7Mbps 的頻寬。等 CBQ 取消後 Client A 的特殊服務隨即消失，又和其餘的四項服務成為平等式的服務。但因為網路環境複雜，因此無法維持 CBQ 所控制的同一速率，但下圖(圖 18)可看出其使用預留頻寬與未使用預留頻寬之速率的相對關係(也就是 Client A 向 FTP Server 1 抓檔案時所佔總可用頻寬的百分比)：

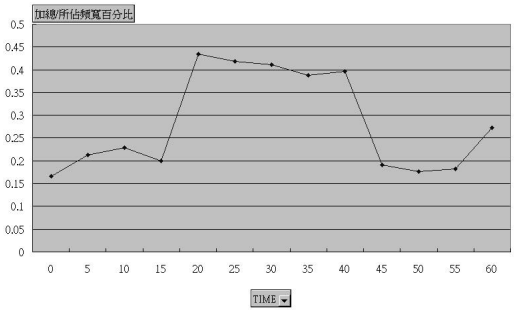


圖 18　Client A 連往 FTP Server 1 所佔頻寬百分比

在此圖中就非常清楚的看到其實 CBQ 在頻寬預留的處理上是以相對的百分比來預留頻寬，就像此實驗中預留 3Mbit 給 Client A 連到 FTP Server 1 的傳輸速率，其實應該是如公式 1，而並非一直都以 3Mbps 的速度在傳輸，而相反的是提供一個相對的速度給 Client A。

傳輸速率所佔頻寬百分比=預留的頻寬/全部可用的頻寬　(1)

8. CBQ Class 樹狀結構測試

CBQ 是目前較為強勢的 Queueing Discipline，其最大的優點便是在於其 Class 與 Class 之間的關係(有父節點與子節點的關係)，可將頻寬以樹狀結構的概念切開。這部份的測試較為複雜，以三種不同的方式進行測試以找出 CBQ 中父節點與其子節點之間存在的規則，分別為

(一)三層式的 Class(如圖 19 所示)：在最下層的 Class 中加入子節點，以測試樹狀結構下其效能及可行性，其測試結果如圖 2-22。

(二)雙層式的 Class(如圖 20 所示)：在單層 Class 中加入兩個父節點，以觀察已存在之父節點中的各父節點之關係。

(三)單層式的 Class(如圖 21 所示)：在實體頻寬下建立多個父節點，測試樹狀結構在實體頻寬中的模式。

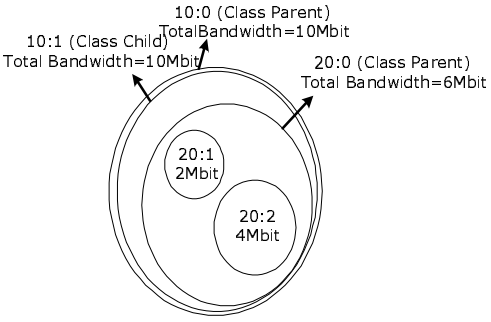


圖 19　三層式的 Class

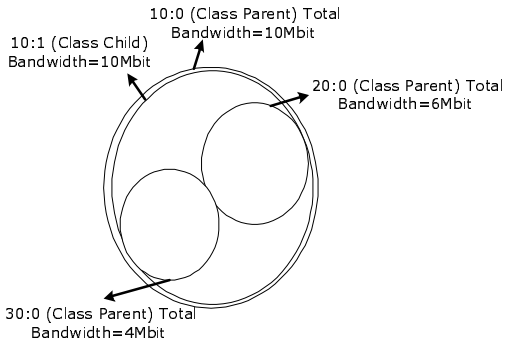


圖 20 雙層式的 Class

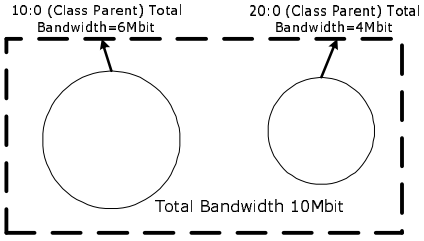


圖 21 單層式的 Class

測試結果發現單層式和雙層式的架構都是不可行的，也就是在 CBQ 實作時的樹狀架構其實是要以子節點來做切割的動作(如圖 22 所示)，父節點在單一的 Class 中僅能是唯一的。

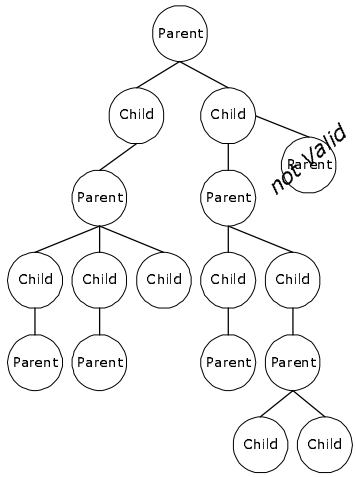


圖 22　CBQ Class 限制

並且在實體頻寬的切割時也必須要注意到的就是如果一開始以 CBQ 綁住多大的頻寬，接下來 CBQ 所能使用的頻寬就已限定在此頻寬大小內，無法再擴大。

另外在三層式的 Class 測試中，的確達成了頻寬預留的效果，証明其樹狀的可行性(如圖 23 所示)。但是在測試的過程中觀察到一個現象，在圖 23 中 A 點到 B 點之間的區段，發生 bounded 參數失效的情況發生，也就是雖然頻寬的總預留 6Mbps(參考前圖 19)的限制依然存在，但各自應限制的 2Mbps 和 4Mbps 失效，變成兩服務共享 6Mbps，因此兩者都是以 3Mbps 的速率做傳輸；而過 B 點流量之所以有限制存在是參數重新設定的結果，並非 CBQ 自身的還原。C 點到 D 點是將 CBQ 去除時會自動恢復頻寬的無限制狀態，因此兩服務平分近 8Mbps 的速率。

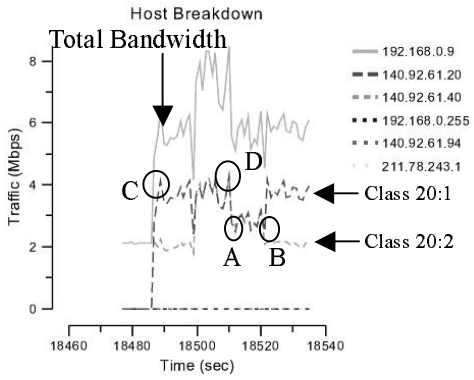


圖 23　CBQ TEST 12 (CBQ avpkt=500)

9. TOS Byte 轉為 DSCP 之格式轉換測試

在 DiffServ 的架構中最重要的就是如何將 IPv4 中的 TOS Byte 改為 IPv6 的 DSCP 格式，如此在 DiffServ 的環境中才可由 Bandwidth Broker 將已標記(mark)過的訊流(flow)交給不同的服務類別，而達到分類效果。在實驗的過程用簡單的 FTP 訊流在經過具有 Ingress Edge Router 或 Leaf Router 功能的 Linux NAT Server 時，觀看其封包中 TOS Byte 的變化。一般封包在未經處理時的值預設為 0，但經由 iproute2 軟體的處理後已被更改為實驗中所設定的 0x28(0010 1000)，因此 DS Field 的格式轉換是可行而且是合理的。

因為不論是在 IPv4 或 IPv6 中的 DS Field 都同樣是 8 bit(為此 IETF 特地將 IPv6 中的 flow label 欄位由 24 bit 縮減為 20 bit，將剩下的 4 bit 挪給 class 欄位使用，因為原本所定義的 priority 只有 4 bit)。在實驗過程中另發現一個有趣的現象，就是一般使用者所使用的 ping 指令，竟也支援 DSCP 的格式，竟可將 IPv4 封包中的 TOS Byte 改為 0x88，這是連在 TOS Byte 都不支援的格式，但 ping 卻可以做到。

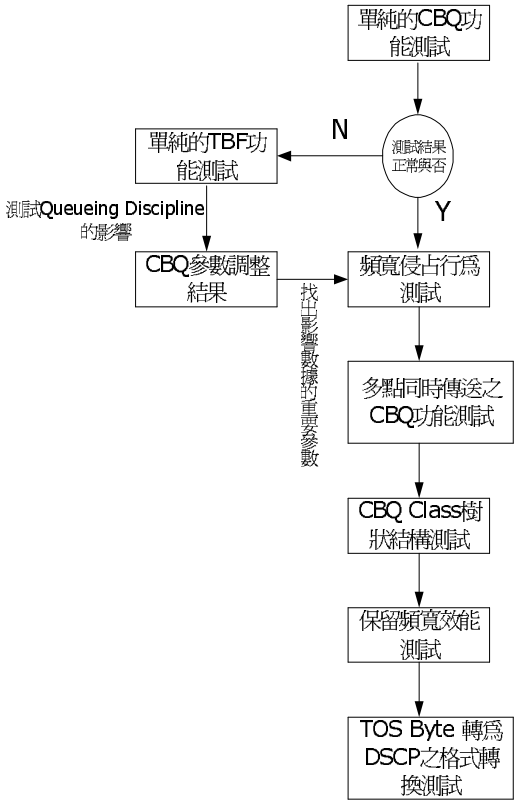


圖 24　測試流程建議圖

四、結論

在整個測試的過程中，令人最滿意的是其限制及管理功能的有效發揮，不論是針對訊流或是群組的分類都已達到目標；但在測試及文獻探討的期間也發現一些值得注意或應改進的地方。

（一）驗證後證實 IP 網路之 QoS 已為可行之服務。

（二）在 QoS 的設定過程中，參數的改變會影響到網路服務的提供者和使用者之間計費的問題。

（三）在收費公平的原則下，針對不同頻寬管理工具都必需要能夠真正做到 bounded、prio(priority)的目標，如此對業者及使用者兩方才有真正公平付費的機會。

（四）在經過一連串驗證後，我們提出一個有效且可測試 QoS 完整性的測試流程圖，如下：

五、參考文獻

1. 梁隆星，嚴劍琴，陳向明，王永鐘，網際網路的服務品質技術[線上資料]，來源：http://www.chttl.com.tw [2001, March 26]。

2. Almesberger, Hadi & Kuznetsov(1999). *Differentiated Services on Linux* [Online]. Available:http://snafu.freedom.org/linux2.2/docs/draft-almesberger-wajhak-diffserv-linux-00.txt [1999, Februry 1].

3. J. Heinanen, F. Baker, W. Weiss & J. Wroclawski(1999). *Assured Forwarding PHB Group*[Online]. Available: http://www.ietf.org/rfc/rfc2597.txt [1999, June 1].

4. Linux Advanced Routing & Traffic Control list [Online]. (1999). Availale: http://mailman.ds9a.nl/mailman/listinfo/lartc [No date].

5. Netherlabs BV, Gregory Maxwell, Remco van Mook, Martijn van Oosterhout, Paul B Schroeder & Jasper Spaans(2001). *Linux 2.4 Advanced Routing HOWTO* [Online]. Available:

http://www.europe.redhat.com/documentation/HOWTO/Adv-Routing-HOWTO.php3 [2001, April 22].

6. Paul F., & Geoff H. (1998). *Quality of Service in the Internet:Fact, Fiction, or Compromise* [Online]. Available: http://power2.nsysu.edu.tw/conference/INET2000/inet98/6e/6e_1.htm [No date].

7. Stef Coene(2001). *QoS test Area* [Online]. Available: http://users.belgacom.net/staf/ [2001, May 10].

8. S. Deering & R. Hinden(1998). *Internet Protocol, Version 6 (IPv6) Specification* [Online]. Available: http://www.ietf.org/rfc/ rfc2460.txt [1998, December 1].

9. Uyless B. (2000). *QOS in Wide Area Networks*. New Jersey: Prentice-Hall.

10. Zheng W. (2001). Internet QoS Architectures and Mechanisms for Quality. London:Academic Press.