# 在用戶端以 Win32 攔截技術實現差異性服務
# Implement Differentiated Services at client by Win32 API Intercept technique


陳耀堂 Yao-Tang Chen autow@wshlab2.ee.kuas.edu.tw

徐志偉 Zi-Wei Shu qqq1030@kimo.com.tw

黃文祥 Wen-Shyang Hwang wshwang@mail.ee.kuas.edu.tw

國立高雄應用科技大學電機工程系
Department of Electrical Engineering,
National Kaohsuing University of Applied Sciences



## 摘 要

本文運用 Win32 Intercept 技術攔截 Client 端應用程式的 Winsock 函式，重導(Redirect)到 Setsockopt 函式以修改其 IP 封包標頭中的 TOS 欄位，使其資料依不同的優先等級設定差異性服務的傳送；即於用戶端進行封包差異性服務標註(marking)工作，因而網路核心路由器(Core router) 可將封包以 WFQ(Weighted Fair Queueing)機制依不同服務等級轉送。文中將描述實驗的差異性服務平台系統架構，由測試的結果證實本研究的正確性。



**關鍵詞:** 差異性服務、服務類別、標註、WFQ



## Abstract

This paper applies the API intercept technique in Win32 environment to intercept the Winsock function of applications, and then redirect it to the Setsockopt function in order to modify the TOS field in the IP packet header. The modification according to the packet priority will provide the transmission with Differentiated Services (DiffServ). In another word, it is a DiffServ marking task for packets at client. By the task, core routers could forward these packets using the Weighted Fair Queuing (WFQ) mechanism. In this paper, a differentiated services platform is built and measured. From the measurement results, it proves this study correctness.



**Keywords:** DiffServ、TOS、Marking、WFQ


## 一、前 言

近年來由於網際網路應用的蓬勃發展造成網路頻寬不足而延長封包傳輸時間，使得傳統的網路傳輸技術 Best-effort 已無法提供多媒體及即時性(Real-time)程式正常運作，因而有關網路服務品質(Quality of Service, QoS)的研究及技術陸續被提出來。有關研究網際網路服務品質的文獻可謂多得不勝枚舉，在現今 QoS 的研究主要可分成兩大領域:整合型服務(Integrated service, IntServ)與差異性服務(Differentiated service, DiffServ)。

IntServ[1]是針對網路上的每一個資料流(flow)，將所需要的頻寬保留下來，並在路由器上儲存與維護每個資料流的狀態。而由於路由器的記憶體空間有限，並不能夠無限制的記載每個資料流的訊息，一旦網路中的資料流太多，會使路由器的負擔太重，造成 scalability 的問題。為了改善 IntServ 的缺點，DiffServ[2]於近幾年便因運而生，它是針對封包標頭中 TOS(Type of Service)欄位的設定值來區分不同的服務需求，而路由器中只需做資料流分類的工作，不必去維護每一個資料流的狀態，也沒有處理控制協定的負擔。在 DiffServ 的架構中主要由兩個部分所組成，邊界節點(boundary node)和內部節點(interior node)。網路邊界路由器(Edge router)在邊界節點進行網路流量服務等級分類的工作，而網路核心路由器(Core router)則在內部節點，設定相關的序列機制並判斷所接收封包的優先等級做封包轉送的服務。本文中我們在多個使用者主機端利用一種在 Win32 下的攔截技術，攔截各台主機的應用程式，設定不同的 TOS 欄位，使其發送出具有不同服務等級的網路封包，並實際架設平台探討在網路核心路由器(Core router)中所設定之 WFQ 的序列機制對各個不同服務等級流量的影響。

## 二、背景回顧

在 IPv4 封包標頭中含有 TOS 位元組，但甚少使用。差異性服務利用此位元組，將 TOS 重新定義成 DS 欄位(Differentiated Services

Field, DS Field);並且定義了路由器中封包轉送的基本規則，稱為 PHB(Per-Hop Behavior)。在 DiffServ 的架構下欲進行網路流量分類的工作，必須要在 Edge router 上或使用者應用程式端設定 DS Field 中的 DSCP 欄位[3]，如圖一所示。一旦在差異性服務網域(DS domain)的入口處(ingress node)決定了每一個封包的服務等級，在 Core router 中就可依照其服務等級來決定各個封包的 PHB。


圖一、DS field structure

在 DSCP 欄位中，主要是設定前三個位元的 IP Precedence 欄位，以決定不同封包的服務等級(Class of Service)，如表一所示。IP Precedence 可分為八個等級，隨著編號增加等級越高。在尚未設定 IP Precedence 之前，其預設值為 routin，且通常最後兩個等級(internet 和 network)是保留給內部的網路使用[4]。

表一、IP Precedence Values

| Number | Name |
|---|---|
| 0 | Routin |
| 1 | Priority |
| 2 | Immediate |
| 3 | Flash |
| 4 | Flash-override |
| 5 | Critical |
| 6 | Internet |
| 7 | Network |

## 三、Win32 攔截技術

Win32 攔截程式[5]的設計目的為不需修改被攔截應用程式的原始碼，並且必須將攔截程式碼插入到被攔截應用程式的行程範疇中，因此表示攔截程式必須是一個動態連結函式庫(DLL)。此技術的基本觀念首先就是把已完成的攔截程式 DLL 檔載入到欲攔截應用程式的位址空間中，使原應用程式在執行時其行程中函式的呼叫動作之前，先一步取得控制權並重導(Redirect)呼叫動作到使用者插入的函式，記錄或更改所需要的資訊之後，再將行程控制權歸還給原應用程式。攔截程式不僅可以記錄可執行檔或動態連結函式庫(DLL)呼叫了那一個 Win32 函式，而且可顯示被呼叫的函式名稱，並能記錄 API 函式的參數及其傳回值。綜合以上所述，一個 Win32 攔截程式必須符合下列幾個條件:

- 對於一個已知的 Win32 行程(process)，攔截程式必須建立一個組態檔(configuration file)，以紀錄該行程針對某一組 DLLs 所發出的函式呼叫。
- 使用者能夠經由組態檔擴充欲攔截之 DLLs 內的函式名單。
- 使用者可以在組態檔中指定某個函式的參數，使其和函式名稱一併被記錄下來。
- 可以記錄函式的傳回值。
- 能夠在 Win32 平台上執行。
- 不須修改被攔截程式之原始碼或可執行檔。
- 允許使用者指定一個攔截對象，執行並產生一個攔截記錄(ASCII 文字檔)。當被指定程式的行程結束後，使用者可以從任何一個文字編輯器中看到那份攔截記錄。

在 Win32 系統中，當執行時期的程式有須要時，會去動態連結作業系統中的共享函式庫到其行程中執行[6]。而在一個行程中對同一個函式做呼叫時，它的每一個呼叫動作都會把控制權轉移到某記憶體位置處的一個 DWORD，然後跳到 DWORD 所指之處，即程式所連結之真正位址。DWORD 函式位址可以從 Import Address Table (IAT)中獲得，如圖二所示，而 IAT 通常置於可執行檔的 idata section 中。在可執行檔所連結的每一個動態連結函式庫(DLL)都有一個對應的 DWORD 陣列，內含此 DLL 輸入函式(Import Function)之位址，當作業系統中的 Win32 載入器把一個可執行檔載入到記憶體中執行，它會以適當的位址填寫 DWORD 陣列。故欲達到攔截的目的只需要在可執行檔的 Imports section 中找到 Import Address Table，改寫其內容，使其中的 DWORD 導向攔截程式碼即可。其主要步驟如下所示:

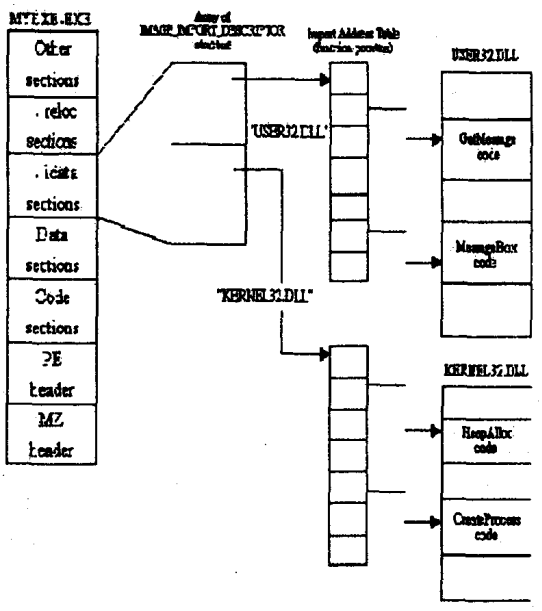
步驟一：建立一個組態檔，內容包括每一個欲攔截函式的名稱、所屬 DLL 名稱、及函式參數相關資訊。攔截程式會針對組態檔中每一個函式產生一小段碼 stub，內含程式碼和資料，並將 stub 位址加入一個 stub 指標陣列中。

步驟二：由於攔截程式必須是一個動態連結函式庫(DLL)，在理想情況下，應該要求被攔截應用程式的行程(或稱為目標行程)去呼叫 LoadLibrary 載入攔截程式碼，但因為不能改變目標行程中的函式，故須另外撰寫一個程式載入器，

載入攔截程式碼至目標行程中執行。載入器的主要動作首先是將目標行程凍結(設定一個中斷點)，修改其中的記憶體和暫存器使之能夠呼叫 LoadLibrary。設定好記憶體及暫存器之後，將該行程解凍並開始執行，此時目標行程就會去呼叫 LoadLibrary，使作業系統被迫載入攔截程式碼至其位址空間中。最後等 *LoadLibrary 回返* 之後，再次將目標行程凍結，使記憶體和暫存器恢復原狀。

步驟三：在被攔截的程式啟動時會到 IAT 中取其所須要相關函式的位址，此時攔截程式會去檢查在步驟一所建立的 stub 陣列，搜尋是否有那一個 stub 的第一個 DWORD 內容與剛才所取得的位址相同。如果有的話，則攔截程式就以 stub 第一個指令的位址寫入目標行程的 IAT 中適當的 DWORD，使目標行程的呼叫動作導向 stub 函式位址。如此即完成 Win32 函式重新導向的過程。

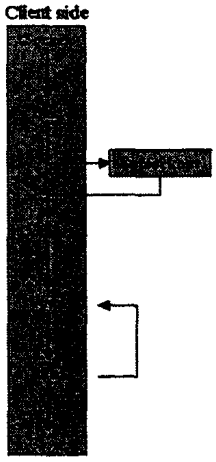
步驟四：在目標行程的輸入函式被重新導向至 stub 函式位址時，攔截程式會把原先的函式位址存放到堆疊之中，並記錄函式的相關參數及傳回值，等到 stub 函式執行完畢，再將置於堆疊之中的函式位址取出並歸還控制權。




圖二、執行檔案的 .idata section 包含了 importfunction address

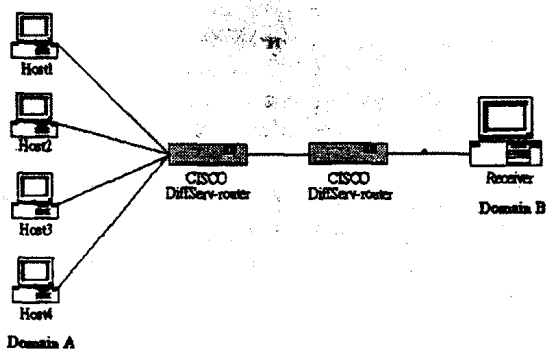
## 四、系統架構

本文實驗平台是架設在 Windows98 上，利用 Win32 Intercept 技術攔截 Client 端的 FTP(File Transfer Protocol)程式，在其行程要去呼叫 WinSock API 時，攔截 Socket 相關函式(我們選擇攔截 Bind 函式)並記錄其資訊(包括函式所屬之 DLL 名稱、函式名稱以及函式相關資料)至堆疊中，然後將呼叫動作導向我們所加入之 Setsockopt 函式[7]，改變 DSCP 中的欄位，當攔截程式執行完畢，就將控制權還給原應用程式處理程序，如圖三所示。




圖三、利用攔截程式加入 Setsockopt 函式

圖四為本文的實驗系統架構圖，此系統是由兩部 Cisco 路由器分成兩個網域(Domain): Domain A 及 Domain B。在 Domain A 中有四台 Client 端的電腦，經由 Win32 攔截程式更改 DSCP 欄位，將資料封包在 Client 端做標註的工作。在 Cisco 路由器中，則是設定 WFQ(Weight Fair Queueing)的序列機制，WFQ 為 Cisco 路由器所支援的佇列方式(queueing method)之其中一種，主要是用來改善路由器上預設的 FIFO(First In First Out)佇列方式先到先服務的缺點。它是一種動態排程的機制，具有 IP Precedence-aware 的功能，使封包根據在 Edge router 或使用者應用程式端所標註的等級做轉送的服務。最後在 Domain B 中有一台 Receiver 電腦做資料接收的工作。




圖四、實驗平台

## 五、測試結果探討

為了要證實 Win32 攔截技術可以達成封包分類(packet classification)的目的以及 WFQ 是否具有 IP Precedence-aware 的功能，因此在本文中做了三個實驗，如下所示:

<實驗一>

在 Client 端的四台主機分別用 Win32 攔截技術將網路流量分成四種不同的等級，並在 Core router 設定 WFQ 的序列機制，使網路封包依優先權等級作差異性服務的傳送。在 WFQ 的機制中，可以偵測到經由 Edge router 或使用者應用程式端所標註的封包等級(Precedence)，並依照封包服務優先等級做頻寬分配(bandwidth allocation)的工作。在差異性服務網域(DS domain)中，各個來源端資料流所設定之 IP Precedence 服務等級(0~7)的總合為 $S_{marked}$。

$$S_{marked} = \sum_{i=0}^{n} [(P_i + 1) \times F_i] \qquad (1)$$

如公式(1)所示，n 表示在差異性服務中所支援之八種服務等級(如表一所示，即 n = 8)，其中 $P_0$~$P_7$ 表示所標註之 IP Precedence 的等級，分別為等級 0 到 7，而 $F_i$ 表示每一種服務等級資料流(flow)的數量。因此欲計算每一個資料流所分配到的頻寬 BEF(Bandwidth for Each Flow)可依照底下的公式得到:

$$BEF = \frac{Bandwidth}{S_{marked}} \times (P_{marked} + 1) \qquad (2)$$

Bandwidth 為網路提供的頻寬，在此的單位為 Kbytes/sec，而 $P_{marked}$ 為各別來源端資料流之 IP Precedence 的等級。

在此實驗中四台主機所送出資料流的服務等級分別為 Routin(等級 0), Priority(等級 1), Immediate(等級 2), Flash(等級 3)，所以其 $S_{marked}$= 10，在頻寬設定為 240Kbytes/sec 的前題下，各個服務等級的資料流經由公式(2)所應獲得的頻寬分別為:

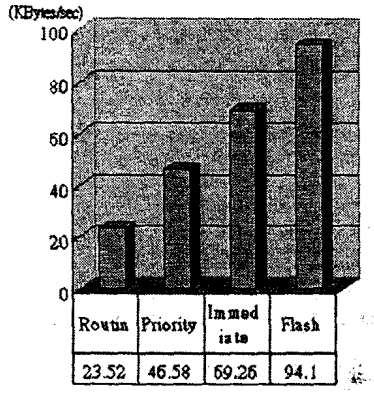
$BEF_1 = 24\,KBytes/\sec$ ..... 等級 0

$BEF_2 = 48\,KBytes/\sec$ ..... 等級 1

$BEF_3 = 72\,KBytes/\sec$ ..... 等級 2

$BEF_4 = 96\,KBytes/\sec$ ..... 等級 3

實際架設平台後的測試結果如圖五所示。由圖中可看出實際測試結果與上述計算結果並沒有太大的出入，即也證實了 WFQ 有 IP Precedence-aware 的功能。




圖五、 在 DS domain 之 Client 端的四台主機利用 Win32 攔截技術將送出資料流的服務等級分為四類，並在 Core router 中設定 WFQ 機制

<實驗二>

同實驗一，但在 Client 端的四台主機所送出資料流的服務等級更改為兩台 Routin(等級 0)，兩台 Priority(等級 1)，所以其 $S_{marked}$= 6，因此各個服務等級的資料流經由公式(2)所應獲得的頻寬分別為:

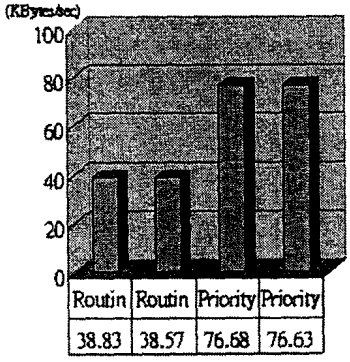
$BEF_1 = 40\,KBytes/\sec$ ..... 等級 0

$BEF_2 = 40\,KBytes/\sec$ ..... 等級 0

$BEF_3 = 80\,KBytes/\sec$ ..... 等級 1

$BEF_4 = 80\,KBytes/\sec$ ..... 等級 1

實際架設平台後的測試結果如圖六所示。

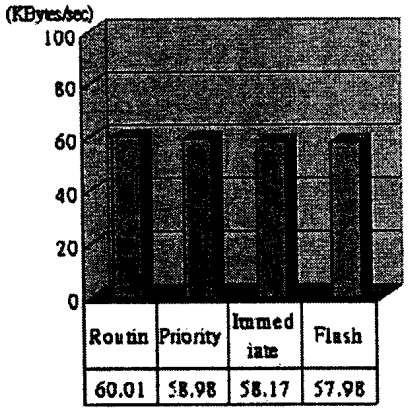



圖六、 在 DS domain 之 Client 端的四台主機利用 Win32 攔截技術將送出資料流的服務等級分為兩類，並在 Core router 中設定 WFQ 機制

<實驗三>

同實驗一，但是在 DS domain 中之 Core router 沒有設定 WFQ 的機制而僅有內部預設的 FIFO 機制。由於 FIFO 並沒有支援 IP Precedence-aware 的功能，所以在圖七的實驗結果中可看出各個不同服務等級的資料流所

分配到的頻寬大約皆為全部頻寬的四分之一，即 60Kbytes/sec。




圖七、在 DS domain 之 Client 端的四台主機利用 Win32 攔截技術將送出資料流的服務等級分為四類，但在 Core router 中沒有設定 WFQ 機制

## 六、結論

本文中描述了一種在 DS domain 之使用者應用程式端運作的 Win32 攔截技術，將來源資料流分類為不同的服務等級，並在 Core router 內設定支援 IP Precedence 的 WFQ 序列機制，使資料流依照不同的 IP Precedence 做不同的轉送服務。最後談及實驗的量測平台及測試的結果，結果證實了 Win32 攔截技術及 WFQ 的可行性。

## 七、參考文獻


[1] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture:an overview," IETF RFC 1633, July 1994.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998.

[3] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, December 1998.

[4] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, USC/Information Sciences Institute, September 1981.

[5] Matt Pietrek 原著 侯俊傑 譯, "Win95 系統程式設計大奧秘",旗標出版有限公司。

[6] Leland L. Beck, "System Software", Second Edition.

[7] Dave Robers 原著 王遠帆 編譯, "深入 Internet with WinSock 設計",松格資訊有限公司。