連線允諾控制中利用平行演算法求最小延遲

黃蓮池[1]、王三元[2]、郭家旭[3]
義守大學{電機系[1]、資工系[2]}、虎尾科技大學資工系[3]
E-mail: {lain[1], sywang[2]}@isu.edu.tw, kuoch@nfu.edu.tw[3]

摘要

本文提出一個簡單，但卻很有效率的演算法，用在連線允諾控制上。連線允諾控制是未來多媒體網際網路提供服務品質的第一道控制機制。服務品質使用服務曲線來描述是一個有效的方式，因而 Sariowan 提出了 SCED 的排程機制用來保證連線要求的服務曲線。SCED 也是屬於 EDF，而 Firoiu 針對 EDF 的連線允諾控制提出了一種低複雜度的演算法。這種演算法複雜度雖低，但因計算過程是使用序列進行方式，所以比較耗時。因此本文將提出一種演算法，利用平行計算的方式來加速最低延遲的計算。平行演算法不僅保有序列演算法之低複雜度，且可在較短的時間內完成連線允諾控制。除了提出平行演算法外，本文亦修正 Firoiu 文中的一些錯誤。

關鍵詞：服務品質、連線允諾控制、片段線性函數、到達程序外封、序列演算法、平行演算法。

Abstract

This paper proposed a simple but efficient algorithm, which is applied to the connection admission controls. Connection admission controls are the first gate of control mechanisms for provision of quality of service in the future multimedia Internet. Utilizing a service curve to describe the quality of service is an effective method. Therefore, Sariowan proposed the SCED scheduling mechanism to guarantee the service curves required by connections. The SCED is also a kind of EDF scheduling and Firoiu proposed an algorithm with low complexity for the connection admission control of EDF. Although the complexity of the algorithm is low, it is time-consuming because it processes the calculation in series. Therefore, this paper proposed a new algorithm that accelerates the calculation of the lowest delay by a method of parallel calculation. The parallel algorithm keeps the low complexity of the series algorithm and completes the connection admission control in a shorter time. Except proposing the parallel algorithm, this paper also modified some mistakes in the Firoiu's paper.

Keywords: Quality of service, connection admission control, piecewise linear function, envelope of arrival process, series algorithm, parallel algorithm.

1. 研究動機與背景

未來的網際網路將是一個多媒體的網際網路，多媒體訊務的一個特色是具有各式各樣不同的服務品質(Quality of Service, QoS)要求。這對於未來多媒體網際網路將是一個挑戰。網際網路的資源不可能是無限多的，尤其是頻寬容量(capacity)，所以網際網路想要提供服務品質就必須具備一些控制機制。

網際網路工程工作小組(Internet Engineering Task Force, IETF)為了滿足各種網路應用之服務品質而發展出兩種網路架構，也就是整合性服務(integrated services, IntServ) [2]及差異性服務(differentiated services, DiffServ) [1]。整合性服務是以符合某些服務品質需求(如語音、檔案傳送和電子郵件發送)所設計的規範。其中也定義了如何利用 RSVP (Resource Reservation Protocol) [3]來進行網路資源的預留。訊務進入網路之後，為了達到服務品質的要求，必須經過管制(Policing)、分類(Classification)、排程(Scheduling)等處理，來提供連線不同的服務等級，即針對每一連線來分配網路資源。雖然網路資源能精確的分配，但由於必須記錄相對連線的參數，所有路由器跟主機皆須對每一連線保留一份連線狀態，導致核心路由器負荷過大，使得服務品質降低。因此整合性網路無法提供有效的規模彈性(scalability)，尤其是當連線暴增時。而差異性服務並不是針對每一個連線提供不同之服務品質，而是將所有的服務品質分為數種類別，歸屬於同一種類別之連線則能得到相同的轉送待遇(forwarding treatment) [15]，因此差異性服務可解決規模彈性不佳的問題及降低其複雜度。

在所有與服務品質相關的控制機制中，連線允諾控制(connection admission control, CAC)將是第一道防線，而連線允諾控制與網路中節點的排程機制有關。為了提供服務品質，Sariowan [14]提出一種排程機制，稱為 SCED (Service Curve based Earliest Deadline first policy)。SCED 也是一種 EDF (Earliest Deadline First)排程法則，只是 SCED 可以保證提供連線要求的服務曲線(service curve)，以達到連線服務品質之要求。Pyun [13]也是探討一種 SCED，只是他所探討的服務曲線是一種比較單純的服務曲線。服務曲線也是網路計算法(network calculus)的重要因子。服務曲線的觀念是一種廣效工具，使得在系統效能及設計參數(如延遲、緩衝區空間、頻寬分配)的計算上更加簡易且有效率。服務

曲線被廣泛討論使用是受到 Cruz 的論文[7]之影響，但在之前，Parekh [11, 12]即提出了服務曲線的觀念。Cruz [7]所使用的服務曲線與 Parekh 所使用的意義不同，以因果而論，Parekh 的服務曲線是一種果，而 Cruz 的是一種因。網路計算法這個術詞隱約見於 Cruz 的文章[5, 6]，真正開始使用這個術語則是 Boudec [9]，Boudec [9, 10]利用迴旋(convolution)使網路計算法變簡易，清大張正尚教授更將這種方式發揮到極致[4]。

吾人認為服務曲線將是描繪服務品質的一種有效方式，由於服務品質的重要，能保證服務曲線的排程機制將是未來多媒體網路重要的一種控制機制。[13, 14]兩篇論文都有提及在連線允諾控制方面可以使用 Firoiu [8]所提出的方法來降低複雜度。[8]所探討的問題是：連線具有片段線性(piecewise linear)函數之到達程序(arrival process)外封(envelope)，在要求某一最大延遲的情形下，網路是否可以接受此一新的連線。Firoiu 所使用的演算法，在此吾人稱之為序列法，因求最小延遲時需先依序求出兩個值，再求其最小值。而在本文，吾人將提出一種新的方法，稱為平行法，此法是利用平行方式求出兩個值，再取其最小值。這樣方式並未增加複雜度，且可以更快速求出最小延遲。

本論文的結構如下。第一節簡介服務品質、服務曲線、網路計算法、SCED 及連線允諾控制之關係與重要性。第二節則簡單復習序列法，並指出該文部分錯誤。第三節介紹吾人提出的平行法，並計算其複雜度。最後一節則是結論。

2. 序列法之修改

表一 符號表

$A(t)$: 新連線之到達程序外封。
$F(t)$: 可用函數(availability function)。
$\|E\|$: 表集合 E 所含元素之個數。
$\{E\}_i$: 表集合 E 中的第 i 個元素。
F_{-i}:當 f 為片段線性函數時,表第 i 個線段之反函數。
Λ_f: f 所有上凹(concave)點的集合。
V_f: f 所有下凹(convex)點的集合，若'(0+) > 0，則原點歸屬為下凹點。
W_f: f 所有轉折(含下凹及上凹)點的集合。
m: $F(t)$下凹點的數目，即 $m = \|V_F\|$。
v_i: $F(t)$的第 i 個下凹點，$1 \le i \le \|V_F\| + 1$，即 $v_i = \{V_F\}_i$，其中 $v_1 = 0$，且令 $v_{m+1} = \infty$。
n: $A(t)$轉折點的數目，即 $n = \|W_A\|$。

在[8]中所使用的前提是：網路為固定服務速率，連線之到達程序外封為片段線性函數[8, 定義一]，考慮 EDF 之排程。[8]最後提出之演算法[8, Fig. 7]，是以序列方式先求 m_x，再求 m_z，故在此稱為序列法，本節將摘要序列法，並對[8]中的一些錯誤予以修正。首先定義符號於表一，並將序列法使用比較接近程式語言的寫法描寫在圖一，其中使用這篇文章符號取代原文的符號。

因假設所有到達程序外封為片段線性函數，所以網路可用函數亦為片段線性函數，當一個具到達程序外封 A 之新連線向網路請求服務時，序列法主要依據以下三個式子[8, (15)-(16)]來計算網路所能提供的最低延遲 $d_{\min}$，若 $d_{\min}$ 小於等於新連線要求的最大延遲，則此連線可以建立。

$$d_{\min} = \max(m_x, m_z) \tag{1}$$

$$m_x = \max\{u - A_{-1}(F(u)) : u \in V_F\} \tag{2}$$

$$m_z = \max\{F^{-1}(A(a)) - a : a \in \Lambda_A, \; A(a) \in (F(u_{a,\text{left}}), F(u_{a,\text{right}}))\} \tag{3}$$

其中[8, (13)-(14)]

$$u_{a,\text{left}} = \max\{x : x \le m_x + a, x \in \{v_i : 1 \le i \le m\}\} \tag{4}$$

$$u_{a,\text{right}} = \min\{x : x > m_x + a, \; x \in \{v_i : 2 \le i \le m+1\}\} \tag{5}$$

$v_{m+1} \equiv \infty$及 F^{-1} 為 F 之虛反函數(pseudo inverse function)，因為 F 不一定是一對一函數，所以 F^{-1} 是一個集合，表示成[8, (6)]

$$F^{-1}(y) = \{x : F(x) = y\} \tag{6}$$

我們在此修改了(4)及(5)集合內 x 可能的範圍，就 $u_{a,\text{left}}$ 而言，其值不可能低於 0 及高於最大的下凹點，即不可能低於 v_1 及高於 v_m。同理，就 $u_{a,\text{right}}$ 而言，其值不可能低於 v_2 及高於 v_{m+1}。利用式子(1)-(3)，我們以程式可以實作的寫法改寫序列法於圖一中。其基本原則是定位出 F 與 A 接觸點以便求出 A 之位移量，即 $d_{\min}$。此種寫法將也應用在下一節平行法。為了低複雜度的緣故，$F(V_F)$是由小到大排序後之集合，且其對應的時間集合表為 T_F。及原來未排序時之位置索引集合以 I 表示。定義集合Θ，其中元素為三項組(triplet)，即

$$\{\Theta\}_i = (\{I\}_i, \{W_F\}_i, \{F(V_F)\}_i) \tag{7}$$

Θ不只儲存 $F(t)$的函數值，且會儲存對應的 t 值及表示在 F 曲線上第幾個下凹點的索引。例如，最小的下凹點是 F 曲線上第五個下凹點，其座標為(30, 123)，則$\{\Theta\}_1 = (5, 30, 123)$，即每個項目是儲下凹點的原來位置索引及座標。

圖一在求 m_x 時，i 從 2 開始，因為 $i = 1$ 時為 F 之原點，不可能是 F 與 A 之接觸點。j 則是用以定位接觸點所在位置是屬於 A 的第幾個線段。求 m_z 時，i 從 1 開始，因 A 之上凹點可以與 F 原點後之直線形成接觸點。k 表 A 之上凹點之序號，從 1 開始。a 為 k 對應上凹點之橫軸座標。l 則是定位接觸

點所在位置是屬於 F 此上凹區域的第幾個線段。得知 l，便可知接觸點在 F 的第 $j+l-1$ 段，則可直接使用反函數求得接觸點在 F 之橫軸座標。

```
mx =0;
for i = 2 to m //Move the pointer of Θ
{   //Move the pointer of WA
    j = max{k: A(ak) < {F(VF)}i};
    mx = max(mx, {TF}i −  A_j^{-1}({F(VF)}i) );
}

mz =0;
k = 1;
a = {ΛA}k;
for i = 1 to m //Move the pointer of Θ
{   j = {I}i;
    E = ΛF ∩ [vj, vj+1);
    do while (F(vj+1) ≤ A(a) < F(vj+1))
    {   if (vj ≤ mx + a < vj+1)
        //Move the pointer of E
        l = max{p: F({E}p) < A(a)}
        mz = max(mz, F−(j+l−1)(A(a)) − a);
        k++; //Move the pointer of ΛA
        a = {ΛA}k;
    }
}
dmin = max(mx, mz);
```

圖一 序列演算法

在[8]中有幾個嚴重的符號錯誤，其主要的錯誤來源是第 562 頁右欄第六行文字：It is easy to show that only the *concave* points of F and *convex* points of A_f^* impose constraints on the position of A_f^*. 正確來說真正影響整個演算法的應是 F 的下凹點及 A_f^* 上凹點才是。這也使得之後所有式子及演算法相關符號的錯誤，幾乎所有應為 X_F^{cx}、$X_{A_f^*}^{cv}$ 及 $\hat{X}_F^{cx}$ 皆分別誤值為 X_F^{cv}、$X_{A_f^*}^{cx}$ 及 $\hat{X}^{cv}$，例如在[8, (11)-(16)]。在本文的演算法及式子中修正了這些錯誤，本文所使用的符號分別為 V_F 及 Λ_A 取代 X_F^{cx} 及 $X_{A_f^*}^{cv}$，使用 V 及 Λ 分別表示下凹及上凹比使用 cx 及 cv 來得清楚。另外我們未使用集合 $\hat{X}_F^{cx}$，因這個集合並非絕對需要，本文使用 v_j 代表 F 第 j 個下凹點。

再者，在[8, Appendix C]中求序列法之複雜度時，在第二段第二行錯置 concave 及 convex 而導致一些錯誤。以下是[8, p.568]的錯誤修正：在第二段的後半求 m_z 時，所有的 X_F^{cv} 及 $X_{A_f^*}^{cx}$ 應分別改為 X_F^{cx} 及 $X_{A_f^*}^{cv}$，同時在複雜度的表示法中 K_1 及 K_2 皆需對換，即該頁倒數第 9 行之 $O(K_2N+K_2) = O(K_2N)$ 及倒數第 3 行之 $O(K_1N)$ 應分別更正為 $O(K_1N+K_1) = O(K_1N)$ 及 $O(K_2N)$。

在此除了修正用字及符號錯誤外，亦修正一個觀念上的錯誤及補強一個觀念。在第二段中指出，F 函數之下(上)凹點是所有連線上(下)凹點的聯集，一般而言是對的，但有些特例則非如此。考慮某一時刻之 F 函數，並在此時允許某一具到達程序外封為 A 的新連線加入服務，此時新的 F 將等於舊的 F 扣除 A。倘若舊的 F 與 A 在 $t = a$ 時皆為上凹點，且 $F(a_+) = \alpha, F(a_-) = \beta, A(a_+) = \gamma, A(a_-) = \theta, \alpha - \gamma > \beta - \theta$，則原 A 之上凹點，在新的 F 仍為上凹點而非下凹點，同理在 A 為下凹點並不一定在新的 F 為上凹點。這些特例是存在的，只是對複雜度的級數並無影響，所以 F 所推出的複雜度仍是正確的。另外補強的部分是，F 的轉折點正確而言是受向右平移後之到達程序外封所左右，而非原到達程序外封所決定，也因向右平移將使得到達程序之外封多一個下凹點。

此外本文也將更精確地表示出，當到達程度外封之線段個數甚至級數有別之時，正確的複雜度應是如何。類似[8, Appendix C]的求法，使用排序過的 $F(V_F)$，至於 W_A，因為 A 為嚴格遞增函數，所以已是排序過的。在求 m_x 時，使用兩個指標分別指向 Θ 及 W_A，依序前進，無需回轉。則求 m_x 之複雜度為 $O(|V_F| + |W_A|) = O(m + n)$。在求 m_z 時，同樣使用兩個指標分別指向 Λ_A 及 Θ，依序前進，無需回轉。這樣得到複雜度 $O(|\Lambda_A| + m)$。亦使用一個指標指向 $E = \Lambda_F \cap [v_j, v_{j+1})$，當找到適當的 v_j 時，指向 E 之指標開始前進，直到所指 F 上凹點為小於且最接近正在檢測的 A 之上凹點，此時表接觸點位於此 F 線段。若下一個合符條件的 a 值仍在 $[v_j, v_{j+1})$，則指標將繼續前進，無需回轉。在最差的情形下，每個 v_j 且 $[v_j, v_{j+1})$ 區間內每個線段都要執行一次，即 F 所有線段都會執行一次，而線段數等於轉折點數，故所需複雜度為 $O(|W_F|)$。最後求 m_z 的複雜度為 $O(|\Lambda_A| + m + |W_F|)$，而整個演算法的複雜度則為 $O(m + n + |\Lambda_A| + m + |W_F|) = O(n + |W_F|)$，即 F 與 A 所有轉折點的和。若所有 N 個到達程序之外封的轉折點數目級數皆為 K，則複雜度即如[8]所求之 $O(KN)$。

3. 平行法之運作與複雜度

利用一個重要的新觀察，我們可以簡化[8]複雜度為 $O(K^2N)$ 之演算法[8, Fig. 5]，使之與序列法具有相同之複雜度等級，且又可利用平行運算的方式，使之在求最小延遲的計算上較為快速省時。

新的演算法是利用平行方式同時進行 m_x 與 m_y [8, (10)-(12)]之計算，故稱為平行法，其中 m_x 之求法與序列法一樣，而 m_y 之求法無需先求出 m_x 即可獨立進行，且因為一個重要的新觀察(見下一段)使得複雜度與序列法相同。

現在描述此重要之新觀察。m_y 的求法是利用到達程序之外封 A 之上凹點與可用函數 F 接觸，並使得到達程序外封全部在可用函數 F 之下方。因為到

達程序之外封 A 為嚴格遞增函數，假設 A 右移 m_y 後，其上凹點與 F 之接觸點在 $t = \tau$，對於所有 $t > \tau$ 將必須是 $F(t) > F(\tau)$。若非如此，假設在 $t = \gamma > \tau$ 時，$F(\gamma) < F(\tau)$，則因 $F(\tau) = A(\tau + m_y) < A(\gamma + m_y)$，使得 $F(\gamma) < A(\gamma + m_y)$，即到達程序外封在可用函數之上方。由於這個觀察，吾人得知接觸點一定在 F 某個下凹點 P 之後的線段，且接觸點的高度不得高於 P 右側所有的下凹點，或說要低於右側最低下凹點且此下凹點高度必定高於下凹點 P 之高度。一言以蔽之，接觸點之高度將介於某一下凹點之高度與其右側最低下凹點高度之間。

```
m_x =0;
for i = 2 to m //Move the pointer of Θ
{   //Move the pointer of W_A
    j = max{k: A(a_k) < {F(V_F)}_i};
    m_x = max(m_x, {T_F}_i − A_{−j}({F(V_F)}_i));
}

m_y =0;
q = 1;
a = {Λ_A}_q;
i =0;
v′ = 0;
do while (v′ ≤ max{F(V_F)})
{   k = min{j: {I}_j > i}; //Move the pointer of Θ
    v″ = {F(V_F)}_k;
    E = Λ_F ∩ [v′, v″);
    do while (a ≤ max{Λ_A} and A(a) < v″)
    {   q++; //Move the pointer of Λ_A
        a = {Λ_A}_q;
        //Move the pointer of E
        l= max{p: {E}_p < A(a)};
        m_y = max(m_y, F_{−(i+l−1)}(A(a))− a);
    }
    i = k;
    v' = v";
}

d_min = max(m_x, m_y);
```

圖二 平行演算法

平行法的運作程序描繪在圖二。圖二求 m_x 是與圖一相同的。求 m_y 則類似於序列法在求 m_z，最大不同在於無需先求出 m_x 即可直接求 m_y。同樣使用兩個指標分別指向 Λ_A 及 Θ，依序前進，無需回轉。平行法使用 while 迴圈，而非序列法中的 for 迴圈，因為並非所有 F 的下凹點都需要檢測，從 F 第一個下凹點(即原點，也是排序後的第一個下凹點)開始，每次搜尋下一個需要檢測之下凹點時，必定要符合前述觀察所得，即需在此次下凹點之右側，且要為右側所有下凹點之最低者。因為 $F(V_F)$ 已由小到大排序過，假設原下凹點所對應的三項組為(i, γ, v')，當指向 Θ 之指標依序前進時，假設指標所指的三項組為(p, t, v)，此時只需注意是否新的下凹點是否在原下凹點右側即可，即 p 是否大於 i 即可。所以一般不需要檢測到 F 所有的下凹點。確定兩下凹點的值後，接著需確定 A 的哪個上凹點可能是接觸點，所以指向 Λ_A 的指標依序前進，指到的每個上凹點的值與 F 所確定出兩下凹點值間的上凹點集合 E 做比較，以定位出接觸點可能所在之線段，若定位出接觸點所在位置是屬於 F 此上凹區域的第 l 個線段，則便可知接觸點在 F 的第 $i + l - 1$ 個線段，最後可直接使用反函數求得接觸點在 F 之橫軸座標。

若 F 與 A 的接觸點在某一個下凹點 P 之後，平行法無需如同序列法檢測 P 與其緊接的下凹點之間所有線段(以橫軸之觀點)，而只需檢測 P 與 P 之後最小的下凹點之間的線段(以縱軸之觀點)即可，即圖二中 v'與 v''之間。因在 v'與 v''之間的上凹點是漸增排序，A 也是漸增排序，所以在尋找接觸點時指向上凹點集合 E 的指標亦無需回轉，所以仍是低複雜度。現在計算平行法的複雜度，在求 m_x 時與序列法相同，所以仍為 $O(m + n)$。求 m_y 時，以最差的情形來看，兩個分別指向 Θ 及 Λ_A 的指標，因無需回轉，所以最差的複雜度為 $O(|\Lambda_A| + m)$。而確定某一 F 的下凹點後，要確定接觸點所在的線段，最差的情形就是在此下凹點至下一個下凹點前所有的線段皆檢測，而最差情況是每一個下凹點皆需要如此檢測，所以最差情形將是 $O(|W_F|)$。因此求 m_y 之複雜度為 $O(|\Lambda_A| + m + |W_F|)$，整體之複雜度為 $O(n + |W_F|)$。結果在最差的情形下複雜度是與序列法相同。

在此強調兩點來說明平行法可以比序列法更快求得最小延遲。第一點，序列法得先求出 m_x 才能進行 m_z 之計算，而平行法可以同時進行 m_x 及 m_z 之計算，所以可節省大半的時間。第二點，上述在求複雜度時是以最差的情形來考量，實際上，並非所有 F 的下凹點及線段都需要檢測，要檢測的下凹點，一般而言是不多的，就 F 曲線而言，檢測完某一下凹點後，下一個需要檢測的下凹點是右邊值最小的下凹點，這中間極有可能跳過很多下凹點。另外檢測線段時也並非檢測某一下凹點與其下一個下凹點間所有的線段，而是只檢測介於此下凹點的值到下一個最小下凹點的值之間的線段即可，這也極有可能只佔兩下凹點間線段數目的一小部分。所以整體而言，平行法在最差情形下將與序列法具相同複雜度，但在執行上利用平行處理的方式及略過很多不必要的下凹點及線段，可以節省較多時間而更快速求得最小延遲。

4. 結論

未來的網際網路將是一個多媒體的網際網路，多媒體訊務是需要網路提供各種不同的服務品質，而提供服務品質的第一個控制機制是連線允諾控制。[8]提出一種低複雜度的序列法計算最低延遲用在連線具有片段線性函數之到達程序外封的連線允諾控制。序列法雖具有低的複雜度 $O(n + |W_F|)$，即複雜度的級數是新連線轉折點與可用函數

轉折點之和，但因求最小延遲時需先依序求出兩個值，再取最小值，所以較沒有效率。本文除了修正[8]文中的一些錯誤外，更提出一種新的方法稱為平行法，利用平行方式求出兩個值，再取最小值。這樣方式之複雜度仍為 $O(n + |W_F|)$，且因平行計算使最小延遲之計算可以更有效率更快速。

至於未來尚可進一步探討的主題包括：一、到達程序外封若不符合[8]之定義一所定義之片段線性函數，將如何修正序列法或是平行法以求最小延遲；二、連線允諾控制不只是受到達程序外封之影響，尚需考慮其他因素，當加入其他因素之時，則吾人所提之平行法或[8]之序列法將如何因應。

參考文獻

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF RFC 2475, Dec. 1998.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," IETF RFC 1633, June 1994.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)--Version 1 Functional Specification," IETF RFC 2205, Sep. 1997.

[4] C.S. Chang, "On deterministic traffic regulation and service guarantee: A systematic approach by filtering," IEEE Trans. Inform. Theory, vol. 44, no.3, pp. 1097-1110, May 1998.

[5] R.L. Cruz, "A calculus for network delay, part I: Network elements in isolation," IEEE Trans. Inform. Theory, vol. 37, no. 1, pp. 114-131, Jan. 1991.

[6] R.L. Cruz, "A calculus for network delay, part II: Network analysis," IEEE Trans. Inform. Theory, vol. 37, no. 1, pp. 132-141, Jan. 1991.

[7] R.L. Cruz, "Quality of service guarantees in virtual circuit switched networks," IEEE JSAC, vol. 13, no. 6, pp. 1048-1056, Aug. 1995.

[8] V. Firoiu, J. Kurose, and D. Towsley, "Efficient admission control of piecewise linear traffic envelopes at EDF schedulers," IEEE/ACM Trans. Networking, vol. 6, no. 5, pp. 558-570, Oct. 1998.

[9] J.Y. Le Boudec, "Network calculus made easy," Tech. Rep. EPFL-DI 96/218, 1996. http://lrcwww.epfl.ch/PS_files/d4paper.ps

[10] J.Y. Le Boudec, "Application of network calculus to guaranteed service networks," IEEE Trans. Inform. Theory, vol. 44, no. 3, pp. 1087-1096, May 1998.

[11] 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 Trans. Net., vol. 1, no. 3, pp. 344-357, June 1993.

[12] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," IEEE/ACM Trans. Net., vol. 2, no. 2, pp. 137-150, Apr. 1994.

[13] K. Pyun, and H.K. Lee, "The SCED service discipline with O(1) complexity for deadline calculation," IEICE Trans, Comm., vol. E85-B, no. 5, pp. 1012-1019, May 2002.

[14] H. Sariowan, R.L. Cruz, and G.C. Polyzos, "Scheduling for Quality of Service Guarantees via Service Curves," in Proc. IEEE ICCCN, Las Vegas, Nevada, pp. 512-520, Sep. 20-23, 1995.

[15] Z. Wang, Internet QoS: Architectures and mechanisms for quality of service, Academic Press, 2001.