第六章 DLBP 演算法和 DLB 演算法複雜度分析之比較

我們在本論文中提到三種演算法：TSLB 演算法、DLB 演算法以及 DLBP 演算法。三者之間的關係爲：DLB 演算法乃是應用了 TSLB 演算法來尋找路徑，並增加了使資料流重新路由的功能，以改善了 TSLB 演算法會使網路資源不合理分配的缺失，提高了網路資源利用率。DLBP 演算法乃是將 DLB 演算法加以延伸，加入了優先權的觀念，使其能應用於具有優先權的資料流傳送，且達到公平、合理、高網路資源利用率目的。由以上的敘述及我們前面幾章的介紹，我們不難發現此三者複雜度的關係爲 DLBP 演算法高於 DLB 演算法；DLB 演算法高於 TSLB 演算法。以下各節我們將分別分析此三種演算法的複雜度。

6.1 TSLB 演算法的複雜度分析

我們在第三章中提過在 MNS 模擬器已有與 TSLB 演算法功能一樣的模組，但其演算法，與 TSLB 演算法有些不同，有點類似 Dijkstra 的最短路徑演算法，而其所求到的路徑 FC 值滿足新資料流的頻寬需求。因爲本篇論文是以 MNS 模擬器爲基礎的，因此在本篇所有的模擬，只要需要用到 TSLB 演算法時，皆使用在 MNS 模擬器已經具有，而功能和 TSLB 演算法相同的模組，而不另外撰寫 TSLB 模組。因此我們此時對 TSLB 演算法的複雜度分析，也是針對 MNS 模擬器內功能和 TSLB 演算法相同的模組來分析。因爲其程式很繁瑣，因此我們不打算在此深入描述，而只報告其分析結果。此演算法其複雜度爲 $O(n^2)$，其中 n 爲節點的數目。因此當網路節點很多時，其複雜度將大爲提高。

6.2 DLB 演算法的複雜度分析

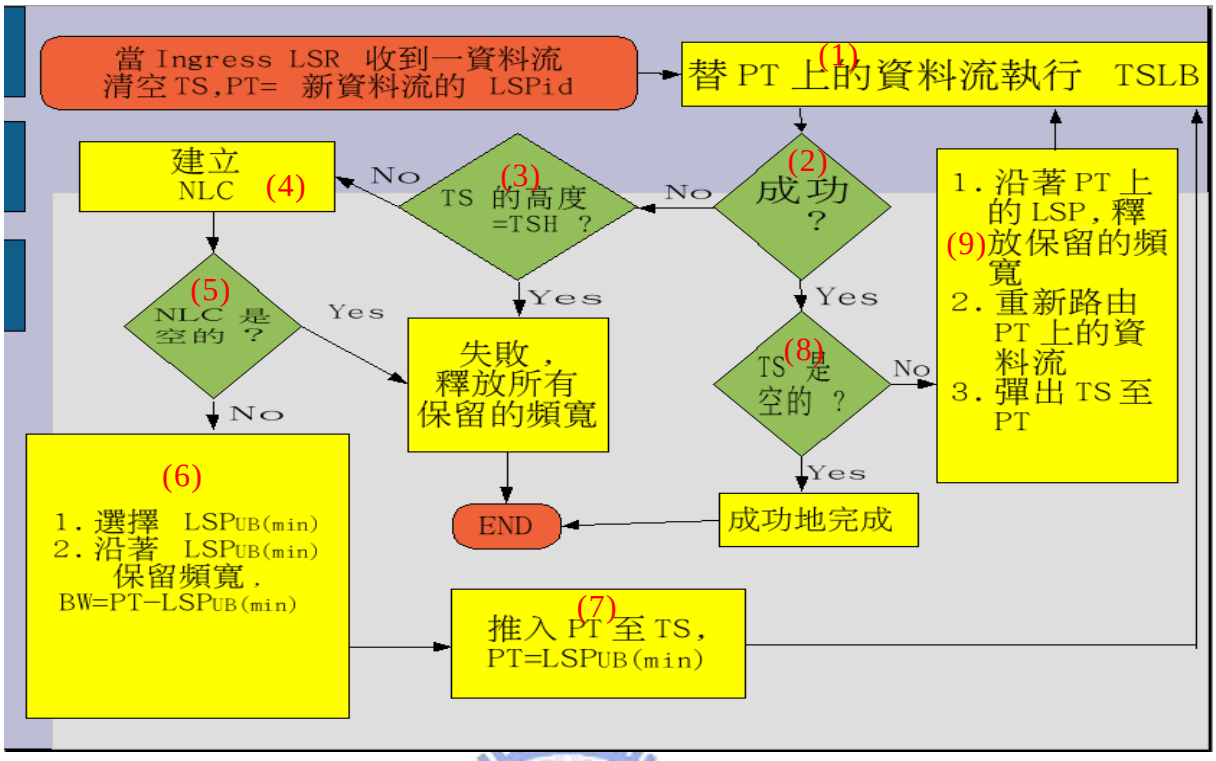

圖 67 DLB 演算法流程圖

我們對照上圖 DLB 演算法流程圖及編號來分析 DLB 演算法的複雜度。假設 Ingress LSR 與 Egress LSR 之間，現存在有 K 條 CR-LSP，在最壞的情況爲：所有現存的 CR-LSP 之 UB 值都小於新的資料流頻寬需求，而且我們重覆如上圖編號(1)到(7)，重覆 K 次後，使用 TSLB 演算法才替 pending-Traffic 上的那一個資料流找尋到路徑，建立 CR-LSP。之後，開始重新路由的工作，我們重覆流程圖上編號(1)-(2)-(8)-(9)-(1)，重覆 K 次後，才替新資料流找到路徑，傳送新資料流成功。

我們統計所有所需時間：我們觀察整個流程，主要花費的時間在於執行 TSLB 演算法的執行時間，以及建立 CR-LSP 時間和釋放 CR-LSP 時間。在此最壞的情況下，共花費了 2K+1 次的 TSLB 演算法的執行時間，以及 K+1 次建立 CR-LSP 時間和 K 次釋放 CR-LSP 時間。其中 TSLB 演算法的執行時間取決於網路節點數目，節點數目越多，所需執行時間越長；而建立 CR-LSP 時間和釋放 CR-LSP 時間，取決於 Ingress LSR 至 Egress LSR 之距離，因爲建立 CR-LSP 時，必須傳送 label request message 從 Ingress LSR 至 Egress LSR 及傳送 label mapping message 從 Egress LSR 至 Ingress LSR；釋放 CR-LSP 時，必須傳送 release message 從 Ingress LSR 至 Egress LSR。

表 1　DLB 演算法複雜度分析之舉例

	網路上已存在的 CR-LSP 之 LSPid 值	各 CR-LSP 之 UB	各 CR-LSP 之 CB
1	1100	9 Mbps	10 Mbps
2	1200	8 Mbps	9 Mbps
3	1300	7 Mbps	8 Mbps
4	1400	6 Mbps	7 Mbps
5	1500	5 Mbps	6 Mbps
6	1600	4 Mbps	5 Mbps
7	1700	3 Mbps	4 Mbps

表 1 即爲此情況之一例，以供參考。假設目前 Ingress LSR 收到一個新資料流，其頻寬需求爲 10Mbps。此資料流利用 TSLB 演算法找不到路徑，而目前 Ingress LSR 與 Egress LSR 之間之現存在 CR-LSP 有 7 條如表 1，此 7 條 CR-LSP 之 UB 值都小於新的資料流的頻寬需求，而第 7 條 UB=3Mbps 之 CR-LSP 可以找到路徑，重新路由至其它路徑，此例總共執行了 2K+1=14+1=15 次 TSLB 演算法，以及 K+1=8 次建立 CR-LSP 時間和 7 次釋放 CR-LSP 時間，來完成整個演算法，成功地替新資料流建立 CR-LSP。

如果執行時間不是太長，DLB 演算法對網路資源的利用和傳輸品質的保證是一個很好的演算法。爲了避免執行時間太長，影響傳輸品質，我們在 DLB 演算法中加入設定允許重新路由的最高次數 TSH 參數，在新資料流尋找路徑的過程中，若發現重新路由的次數會超過 TSH 值，則即放棄尋找路徑，以確保網路對資料流的傳輸品質。如果傳送新資料流，建立 CR-LSP 失敗的頻率太高，則表示網路頻寬已明顯不足，應建設擴充網路，以增加網路頻寬。

6.3 DLBP 演算法的複雜度分析

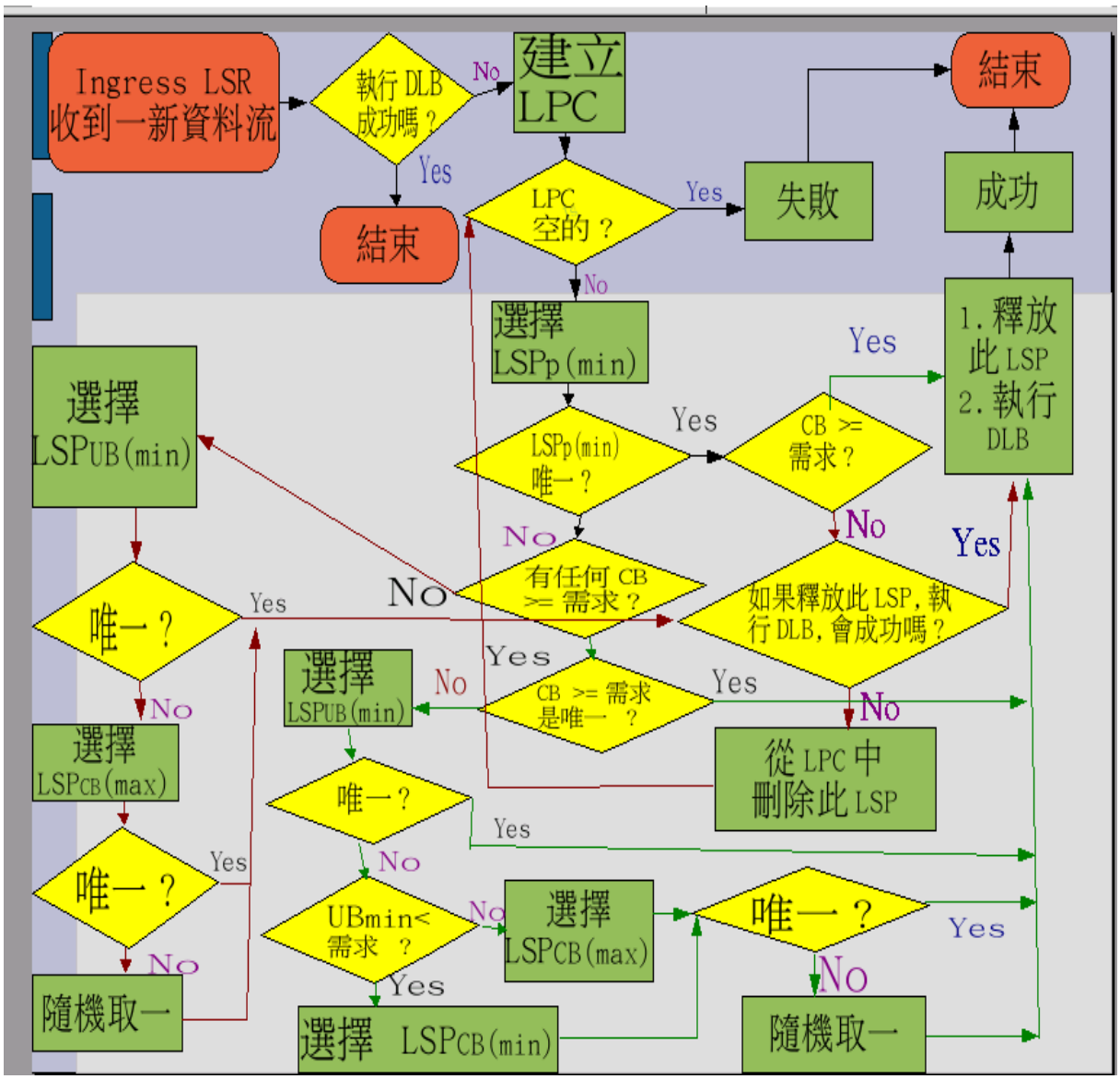

圖 68 DLBP 演算法之流程圖

我們對照圖 68 DLBP 演算法流程圖來分析 DLBP 演算法的複雜度。當新資料流利用 DLB 演算法找不到路徑時，假設目前 Ingress LSR 與 Egress LSR 之間，現存在有 K 條 CR-LSP，在最壞的情況爲：所有現存的 CR-LSP 之優先權都小於新的資料流的優先權（即每一 CR-LSP 都在 LPC 集合內），可分成三種情況分析：

1. 情況一：

LPC 中只有一條 CR-LSP，其 CB 值大於或等於新資料流的頻寬需求，且其

優先權在此 K 條 CR-LSP 中最高。在行使 K-1 次 DLB 演算法皆失敗後，才因第 K 個 CR-LSP 之 CB 值大於或等於新資料流的頻寬需求，因而釋放此 CR-LSP，而使新資料流找到路徑。

在整個過程中，包括剛開始新資料流利用 DLB 演算法找不到路徑，總共有行使 K 次 DLB 演算法失敗，在此情況下，每執行一次 DLB 演算法失敗，即使用了二次 TSLB 演算法，再加上最後一次新資料流利用 TSLB 演算法找到路徑，因此總共執行了 2K+1 次 TSLB 演算法。除了執行 2K+1 次 TSLB 演算法所需之時間外，還需要一次釋放 CR-LSP 時間及一次建立 CR-LSP 時間。

表 2 即爲此情況之一例，以供參考。假設目前 Ingress LSR 收到一個新資料流，其頻寬需求爲 10Mbps，設定優先權爲 4。此資料流利用 DLB 演算法找不到路徑，而目前 Ingress LSR 與 Egress LSR 之間之現存在 CR-LSP 有 7 條如表 2，此 7 條 CR-LSP 之持有優先權都小於新的資料流的設定優先權（即每一 CR-LSP 都在 LPC 集合內），在此例中 LSPid=1100 之保留頻寬將被新資料流所侵佔。此例總共執行了 2K+1=2*7+1=15 次 TSLB 演算法，以及一次釋放 CR-LSP 時間及一次建立 CR-LSP 時間。

表 2　DLBP 演算法複雜度分析情況一之舉例

網路上已存在的 CR-LSP 之 LSPid 值	各 CR-LSP 之 UB 值	各 CR-LSP 之 CB 值	各 CR-LSP 之持有優先權
1100	9Mbps	10 Mbps	5
1200	3Mbps	8 Mbps	6
1300	3Mbps	7 Mbps	6
1400	3Mbps	6 Mbps	6
1500	3Mbps	5 Mbps	6
1600	3Mbps	4 Mbps	6
1700	3Mbps	3 Mbps	6

2. 情況二：

網路狀況有如 DLB 演算法最壞的情況，Ingress LSR 收到一個新資料流，執行 DLB 演算法找不到路徑，在此 K 條 CR-LSP 中，選擇釋放 UB 值最低的那一條 CR-LSP 後，即可利用重新路由 CR-LSP 的方式，替新資料流找到路徑。在此情況下，總共執行了 4K 次 TSLB 演算法。除了執行 4K 次 TSLB 演算法所需之時間外，還需要 K 次釋放 CR-LSP 時間及 K 次建立 CR-LSP 時間。其所花費時間與 DLB 演算法最壞情況相比，重新路由的次數相近，而執行 TSLB 次數約多一倍，在此情況下 DLBP 演算法複雜度最高。

表 3 即爲此情況之一例，以供參考。假設目前 Ingress LSR 收到一個新資料流，其頻寬需求爲 10Mbps，設定優先權爲 4。此資料流利用 DLB 演算法找不到路徑，而目前 Ingress LSR 與 Egress LSR 之間之現存在 CR-LSP 有 7 條如表 2，此 7 條 CR-LSP 之持有優先權都小於新的資料流的設定優先權（即每一 CR-LSP 都在 LPC 集合內），在此例中 LSPid=1700 之保留頻寬將被新資料流所侵佔。此例總共執行了 4K+1=4*7+1=29 次 TSLB 演算法，以及 k=7 次釋放 CR-LSP 時間及 k=7 次建立 CR-LSP 時間。

表 3　DLBP 演算法複雜度分析情況二之舉例

網路上已存在的 CR-LSP 之 LSPid 值	各 CR-LSP 之 UB 值	各 CR-LSP 之 CB 值	各 CR-LSP 之持有優先權
1100	9Mbps	10 Mbps	6
1200	8Mbps	9 Mbps	6
1300	7Mbps	8 Mbps	6
1400	6Mbps	7 Mbps	6
1500	5Mbps	6 Mbps	6
1600	4Mbps	5 Mbps	6
1700	3Mbps	4 Mbps	6

3. 其它：情況一及情況二是屬於比較特殊的狀況，因此我們提出來特別加以分析，而其它未提到情況乃介於情況一和情況二之間，而其複雜度亦介於二者之間。

6.4 結論

我們將 DLB 演算法及 DLBP 演算法的複雜度分析結果整理如表 4，我們從表 4 可看出 DLBP 演算法比 DLB 演算法只有稍微複雜一些，但其不但具有 DLB 演算法的所有優點，並且增加了可以處理具有優先權資料流的重要功能，因此我們深信 DLBP 演算法是一個不錯的演算法。

表 4 DLB 演算法及 DLBP 演算法情況（一）和情況（二）的複雜度分析

演算法 / 複雜度	DLB 演算法	DLBP 演算法		
		情況（一）	情況（二）	其它
執行 TSLB 演算法次數	2K+1	2K+1	4K	介於 2K+1 及 4K 之間
建立 CR-LSP 次數	K	1	K	介於 1 及 K 之間
釋放 CR-LSP 次數	K+1	1	K	介於 1 及 K 之間