A Balanced Scheduling Algorithm for High-speed ATM Switch

曾學文
台灣科技大學電機系
台北市基隆路四段四十三號
E-mail: tesng1@ms27.hinet.net

梁維真
台灣大學土木系
台北市 106 羅斯福路四段一號

Abstract

對非同步傳輸模式 (Asynchronous transfer mode 簡稱 ATM) 這種快速的交換機 (switch) 而言, Virtual output queue (簡稱 VOQ) 是一個有效率的結構, 因為它提供了高的輸出效能, 並且由於 ATM 主要是用於資料流量傳輸較大的骨幹網路 (backbone) 上, 所以如何對資料作有效率的排序是必要的. 在本篇論文中, 我們提出一個新的排序方法: Balanced scheduling VOQ algorithm (B-VOQ), 利用此演算法在細胞 (cell) 的排序上, 可以減少平均延遲時間 (average delay), 獲得最好的結果.

關鍵字– ATM, switch, VOQ, B-VOQ, cell

1. Introduction

在新一代的高速網路中, ATM 在 backbone 上提供快速的通訊傳輸及完善的服務品質. 同樣的, 隨著網際網路的快速成長, 交換器 (switch) 和路由器 (router) 毫無疑問的也在資訊的溝通以及傳遞上扮演重要的角色. 在 ATM 的網路主要架構即為 switches, 且每一個 switch 上有若干個輸入埠 (input port) 和輸出埠 (output port), 藉由這些通訊管道來傳送固定長度的細胞(cell). 在今天, 門閂式交換器(crossbar switch) 已被使用, 因為它允許多的 cells 在同時被傳遞而且不會導致資料在傳送時產生碰撞. 雖說這些 cells 來自許多不同的輸入管道 (input), 但它們可能會有相同的輸出管道 (output), 於是我們使用佇列 (queuing) 來存放這些要到同樣目的的 cell.

一般而言, 有三種方法被使用於高速的 switch : 1. input queuing, 2. output queuing, 3. VOQ (virtual output queuing). 在 input queuing switch 中, 每一個 input port 維持一個先進先出 (first-in-first-out 簡稱 FIFO) 的 queue 來儲存進入的 cell, 然後再決定每個 cell 的輸出順序. 這個方法最大的優點就是不需要提升 switch 的速度.然而, 由於網路的 cell 流量是不均勻的, 故導致 HOL (head of line) 的阻礙現象, 而限制了 input queuing switch 的最大輸出率 (throughput) 只有 58.6% [1].

在第二個方法中, 同樣的, 在每個 output port 會有一個 queue 來存放想要被傳送的 output cell. 因為在 output queuing switch, 任何欲進入的 cell 都會藉由 switch 交換到輸出的 port, 所以該方法可以達到 100% throughput. 而此方法的缺點是與 input queuing 相較之下, output queuing switch 需要有 N 倍的處理速度來傳遞資訊. 例如 : 對一個 N*N switch 而言, 我們需要將 output queuing switch 加速 N 倍才能達到需求, 但是此方法的成本過於昂貴.

最後一種方法是 VOQ, 在一個 N*N switch VOQ 中, 每個獨立的 output 都有其相對的 input 並維持一個分割為 N 個 FIFO 的 queues 作為緩衝器, 如 Figure 1. 利用此方法不僅解決了 HOL blocking 的問題, 而且 switch 不需額外的提升速度亦可達到最高的效能. 現在 VOQ 已被證明出可增加 input queuing switch 的效能達 100%.

許多藉由或應用 VOQ 的 switch 的排序演算法已被提出 [2],[4],[5]. 這些演算法提供了不同的實用功能, 其中以分散式排序法(distributed scheduling algorithm)最為重要. 在這個方法中, N*N switch 被模擬為 N 個 inputs 和 outputs, 每個部分包含 N 個節點 (node). 利用上述的模型, 我們希望算出一個最好配對 (matching) 細胞的方法, 以便滿足 (1)資料被傳送時能夠達到機率較小的碰撞 (2)資料可以同時被服務. 在過去的文獻中有四個主要的演算法已被提出, 分別是 PIM[3], RRM[5], iSlip[6], T-RRM[7]. 這些演算法得服務方式都是由三種階段 (phase) 的順序來完成, 分別是 : 要求(request), 允許(grant), 接受(accept), 如 Figure 2 所示.

1. Request phase : 每一個 input 廣播它們要求傳送的資料到 output.
2. Grant phase : 每一個 output 從上述接收到的 requests 中, 獨立的挑選一個要求, 然後傳送 grant 回該 input.
3. Accept phase : 而後, 每一個 input 挑選一個 grant 作為接受. 當一個 input 接受一個由 output 傳來 grant, 該 input 必須傳送一個 cell 至 output.

在眾多的演算法中, 通常只有不同於如何在 output 中從眾多的 request 挑選一個作為 grant 和如何在 input 中選擇 grant 當作 accept. 在 PIM 排序演算法中, grant 和 accept 的挑選都是決定於亂數產生的機率. 而 RRM 和 iSlip 的演算法中, grant 和 accept 的選取是藉由輪循式 (round-robin) 的選擇順序. 在 RRM, 每一個 output 根據固定 round-robin 的順序通知下一個出現的 request, 然後移動 round-robin 的指標到另一個被通知的 request, 同理, 每個 input 亦是採用同樣的方式來接受 grant. 然而, 在 iSlip, 對每一個 output 而言, round-robin 的指標並不會改變, 除非該相關的 grant 已被接受, 才會循環到下一個. 簡而言之, 在 RRM, output 不等 input 傳來的 accept 訊號即 round-robin 移動到下一個位置. 相反的, 在 iSlip, output 會一直等到 input 回覆 accept 才 round-robin 到下一個點. 在上述的演算法中, 只有 iSlip 可達到 100%的 throughput.

由於 PIM, RRM, iSlip 都沒有考慮到 queue 本身的狀態. 例如 : 可能某一 port 內 queue 的長度很長 (即想傳送到該目的地的 cells 很多), 但因為 round-robin 的指標需要繞一圈才能再度到同一位置且一單位時間內只能傳送一個 cell (即一次服務一個人), 這樣會導致延遲 (delay) 過長. 因此 T-RRM 被提出於在 VOQ 排序 cells 的長度. 在 T-RRM 演算法中, 依舊使用 round-robin 來決定 cell 交換的順序, 也同樣利用 request, grant, accept 三種 phases 作為傳輸, 且與 iSlip 相同, 都可達到 100% throughput. T-RRM 主要的想法是 : 如果一個 queue 有過多的 cells 以致於超過門檻 (threshold) 限制時, 這時在該 VOQ 大於 threshold port 中的 cell 會先被傳送到目的地. 而我們所提出的 B-VOQ 在 traffic load 變大之時, 可能所有 VOQ 中的 cell 長度皆以超過 threshold, 除了考量是否大於 threshold 外, 在其中選擇長度最長的先服務, 以達到每個 VOQ 長度的平

衡.

本篇論文的組成架構如下，在第二章節我們將會介紹 B-VOQ 演算法並且假設了一個簡單的例子來描述 T-RRM 是如何運作. 在第三章節我們將模擬結果分析來比較文獻中所提的演算法與B-VOQ演算法的效能差異. 第四章是我們的結論.

2. B-VOQ Algorithm

就像 RRM, iSlip, T-RRM, B-VOQ 也使用 round-robin 來決定 cell 的排序，且同樣基於三個 phase 作為傳遞方式. 在描述 B-VOQ 之前，我們先對 marked port 下定義. 如果有一個 input 被挑選為藉由 crossbar switch 傳送 cell 到某一 output，且該相關 VOQ 長度大於 threshold，我們把該 input 和 output 標示起來稱為 marked port. 當 marked port 內的 cells 數目被服務為小於或等於 threshold 時，我們再把該 input 和 output 取消標示,使其變為 unmarked port.

B-VOQ 的主要構想是 ：如果 cell 量來的很多時，相對的導致許多 VOQ 超過 threshold，我們把這些相關的 input 和 output 標示為 marked port，且將該 VOQ 收集起來，並計算這些 VOQ 內的 cells 數量. 擁有長度最長的 VOQ 會在該 slot time 先被傳送. 待下一 slot time，再重新判斷每一 VOQ 內的 queue 長度是否超過 threshold，選擇超過 threshold 且在 VOQ 中長度最長的先處理. 如果此時產生擁有相同 cell 個數且都超過 threshold 的 VOQ，這時即採取 round-robin 的方式作為傳遞 cell 的優先順序. 假設 cell 數較少，以致於此時並無大於 threshold 的 VOQ，則利用一般 round-robin 的方式對 cell 做遞減. 該方法最大的優點在於可以平衡 VOQ 內 cell 的長度. 當 traffic load 很大時，採用 T-RRM 演算法會產生 queue 的不平衡，導致有些 VOQ 內的 cell 數目過多，使得某些超過 queue 容量的 cell 被拋棄(drop)，欲傳送的資料因而遺失.

關於 B-VOQ 的 pseudo-code 描述如下:

```
/* i : number of input ports;  j : number of
output ports;  cell_num : the number of cell in
VOQ ;  TMport : how many ports own the
same cell numbers which is the greatest one
*/

for(all i) request(i ){ /* Request phase */
if(marked_input[i]= = Yes){ /* is it a marked
input ? */
        sort;/* calculate the cell numbers of
marked input*/
        find max cell number; /* find max cell
number in VOQ*/
         if(TMport>=2)
         {
          Torr_priority[j]=(i+1); /* select a request
over threshold based on the round-robin of
output j */
        }
       accept(i,j);/* use the same matching in this
time slot */
        if(cell_num[i,j]<=threshold)
           marked_input[i] =No;/* unmark the
input */ }
else for(all j){
if(cell_num[i,j]>0){ /*queued cells for output
j ?*/
        request(i,j);/* request to output j */
}

for(all j) grant(j){ /* Grant phase */
if(num_request[j]>0){ /*received any request ?*/
        i=orr_select(j); /* select a request based on
the round-robin of output j */
        grant_request(i,j); /*grant request j back to
input i */
        }
}
for(all i) accept(i){ /* accept phase */
if(num_grant[i] >0){ /* received any request? */
        j=irr_select(i); /*select a request based on
round-robin of input i? */
        accept_grant(i,j); /* accept the grant j */
        irr_priority[i]=(j+1); /* set the highest
priority of round-robin of input i higher than
granted output */
        orr_priority[j]=(i+1); /* set the highest
priority of round-robin of output j higher than
granted input */
        if(cell_num[i,j]>threshold){ /* queued
cells more than threshold ? */
        marked_input[i]I=Yes; /* mark input I as
marked input */
        }
```

```
    }
}
```

三種 phase 傳遞方式敘述如下 :

1. Request : unmarked input 傳送 request 到 output.
2. Grant : 當一 unmarked output 收到任何 request, 它會根據 round-robin 的方式來傳送通知訊號.
3. Accept : 如果一個 input 接收到 grant, 該 input 會傳送 cell 到 output. 更進一步的說, 如果 input queue 有超過 threshold 的 cells 在排隊等待傳送時, 該 input queue 和 output queue 將會被標示成為 marked port, 直到 VOQ 內的 cell 數小於或等於 threshold 才恢復成為 unmarked port.

Figure 3b 呈現 B-VOQ 如何運作在一個 threshold 設為 3 的 4*4switch 的範例.我們利用一個二維陣列 L 來表示 input 內 VOQ 的 cell 個數. 坐標 (x, y) 分別表示 input 和 output. 在第一個 slot time, L = {{4,5,4,3},{5,4,5,3},{6,6,7,3},{7,4,6,2}},即 input 1, input 2, input 3, input 4 傳送到 output 1 的 cell 個數分別為 4, 5, 6, 7 (個); 傳送到 output2 的 cell 個數分別為 5, 4, 6, 4 (個), output 3, output 4 的 cell 個數如上所述. 經過 cell 個數的計算後, 在 output 1 得到 cell 個數大於 threshold 有 input 1, input 2, input 3, input 4 分別為 4, 5, 6, 7 (個), 根據擁有最多 cell 個數的 port 會先被服務, 所以先挑選 input 4. 接下來 output 2 一樣去判斷是否超過 threshold, 會得到 input 1, input 2, input 3, input 4 皆超過 threshold, 經過計算後, 選擇其中 cell 個數最多的 input 3 作為服務對象. 同樣的對 output 3 而言 input 1, input 2, input 3, input 4VOQ 中的 cell 個數亦超過 threshold, 所以一樣選擇 cell 個數最多的 input 3 先服務, 最後 output 4 經過判斷都沒有超過 threshold 故以 round-robin 的方式處理, 所以先選 input 1. 由上述, 得知 (1,4), (3,2), (3,3), (4,1) 內的 cell 數在第一個 slot time 會分別減 1. 在第二個 slot time 時, L = {{4, 5, 4, 2}{5, 4, 5, 3},{6, 5, 6, 3},{6, 4, 6, 2}}, 這時再經計算後 input 1, input 2, input 3, input 4 傳送到 output 1 的 cell 個數分別為 4, 5, 6, 6 (個), 由於 cell 個數皆超過 threshold, 且 input 3 和 input 4 的 cell 個數皆為 6, 所以使用 round-robin 的方式選擇 input 3 來服務. 在第二個 slot time, (1,2), (2,4), (3,1), (3,3) 內的 cell 數會分別減 1. 此後在每個 slot time, cell 的遞減方式皆以此類推. 由 Figure 3a 使用 T-RRM 和 Figure 3b 使用 B-VOQ 的方式比較後發現 Figure 3b 在 VOQ 的 cell 個數會比 Figure 3a 的平衡.故可了解我們所提的方法有較好的效能.

3. Simulation result

為了證明 B-VOQ 演算法和其他演算法相比較有明顯改善的 average delay, 我們將 B-VOQ 模擬在一個 16*16 switch. 由於 PIM 和 RRM 在先前的文獻已經被證明出無法達到 100%的 throughput, 所以在 simulation 中, 我們將只比較 iSlip, T-RRM 與 B-VOQ 的 delay. 在所有的模擬中, 我們假設 input 和 output 的連結速度都是一樣的, 且根據 Bermoulli traffic, input 流量對每一個 input port 接收以及傳送到 output 的 cell 是均勻分布的. 根據這個典型的網路交通流量模擬, 我們將執行 50000 個 time slots 來比較在不同 traffic load 的 delay 與各個演算法的關係.

經過證明, 已得知只有 iSlip 和 T-RRM 可達到 100%throughput, 且根據文獻[7],且在 iSlip 與 T-RRM 相較下, T-RRM 會獲得較低

的 average delay, 因此, 我們將比較 B-VOQ 分別對 iSlip 與 T-RRM 的 delay_improvement.

delay_improvement (iSlip vs B-VOQ) = $(delay_{iSlip} - delay_{B\text{-}VOQ})/delay_{iSlip}$.

delay_improvement (T-RRM vs B-VOQ) = $(delay_{T\text{-}RRM} - delay_{B\text{-}VOQ})/delay_{T\text{-}RRM}$.

我們先模擬 iSlip, T-RRM, B-VOQ 在不同的 traffic load 及 threshold 下的 average delay. 結果如 Figure 4-6 所示.

Figure 4 顯示出 threshold 設為 2 時, 當 traffic load 小於 0.5, iSlip 與 T-RRM, B-VOQ 的 delay 沒有顯著的差異. 當 traffic load 越來越大時 iSlip 與 T-RRM 的差距會在 0.6 慢慢分開,在 0.7 時最為明顯, 但這二個方法與 B-VOQ 的 delay 差距將會隨著 traffic load 變大而越來越明顯. 當 traffic load 為 1.0 時, 該現象最為明顯.

由 Figure 5 顯示出 threshold 設為 3 時, 當 traffic load 小於 0.5, iSlip 與 T-RRM, B-VOQ 的 average delay 幾乎沒有差異. 同樣地, 當 traffic load 越來越大時 iSlip 與 T-RRM 的差距會在 0.7 慢慢分開, 而在 0.9 時又漸漸不明顯. 但這二個方法與 B-VOQ 的 delay 差距將會隨著 traffic load 變大而越來越明顯. 當 traffic load 為 1.0 時, 該差距最大. 因此, 我們的演算法與先前文獻中所提的方法相比較下, 在 average delay 上會有較好的表現.

在 Figure 6 顯示出當 threshold 設為 4 時, 依舊可以發現 B-VOQ 有較好的 average delay. 因此, 由 Figures 4-6 中可發現不管 T-RRM 的 threshold 設為多少,所不同的只在於和 iSlip 的 average delay 在多少 traffic load 時會有明顯的分別.當 traffic load 增加時 B-VOQ 和上述兩者相比較皆會有較佳的 average delay.

由 Figure 7 說明 B-VOQ 對 iSlip 的 delay_improvement, 我們發現 threshold 設為 2, 3, 4, 且 traffic load 分別為 0.6, 0.7, 0.8 時, 在 delay_improvement 將產生 99.89%, 95.21%, 55.05% 的最佳改善率.

Figure 8 顯示 B-VOQ 對 T-RRM 的 delay_improvement, 我們亦可發現 traffic load 為 0.7 時將擁有最好的 delay_improvement, 此外, 在 traffic load 為 0.8 且 threshold 設在 4 時, 亦有較佳的改善率.

4. Conclusion

在這篇論文中, 我們提出一種簡單的方法 B-VOQ 用來減少 cell 在 input queue 中所需的延遲時間, 在 B-VOQ 中, 每一組搭配好的 input 和 output 每經過一個 slot time 便會重新計算每一 VOQ 的 cell 的個數, 如果 cell 個數大於 threshold, 則該 input 和 output 將會被標示, 且這些 VOQ 內擁有最多的 cell 個數將會先被服務, 否則即用 round-robin 的傳輸方式. B-VOQ 的模擬結果顯示出當 traffic load 大於 0.6 時, 相較於 T-RRM, 它可以減少 16-68%的 average delay.

REFERENCES

[1]. Karol, M.; Hluchyj, M.; and Morgan, S.; "Input output queuing on a space division switch," IEEE Trans. Communications, vol.35, no.12,pp.1347-1356, 1988.

[2]. Ajmone Marsan, M.G.; Bianco, A.; and Leonardi E.;" RPA: A Simple, Efficient, and Flexible Policy for Input buffered switch,"

IEEE Communication Letters, l93-:83-86, May 1997.

[3]. Andeson, T.; Owicki, S.; Saxe, J.; and Thacker, C.; "High Speed Switch Scheduling for Local Area Network," ACM Trans. on Computer System. pp.319-352. November 1993.

[4]. McKeown, N; Anantharam, V; and Walrand, J; " Achieving 100% Throughput in an Input-Queued Switch," Proc. of IEEE '96, San Francisco, March 1996.

[5]. McKeown, N;" Switching Algorithm for Input-Queued Cell Switches," PhD Thesis, University of California at Berkeley, 1995.

[6]. N. McKeown, "The iSlip Scheduling Algorithm for Input-queues Switches," IEEE Trans. on Networking, Vol. 7, No.2, pp. 188-201, April 1999.

[7]. Wenzhe Cui, Hanseok Ko and Sunshin,"A Threshold Based scheduling Algorithm for Input Queue Switch," Dept. of Electronic Engeneering, Korea University, Seoul, Korea, ICOIN 15, 2000.

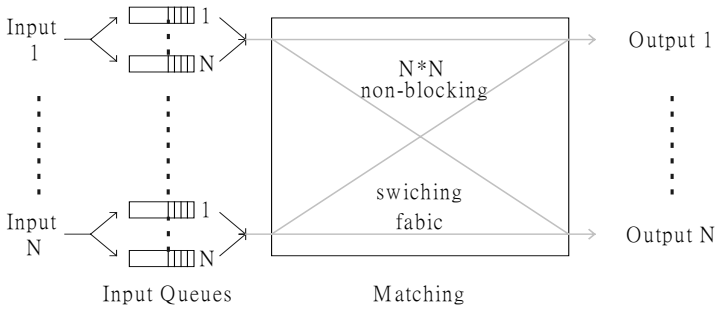

Figure 1. VOQ Switch Architecture

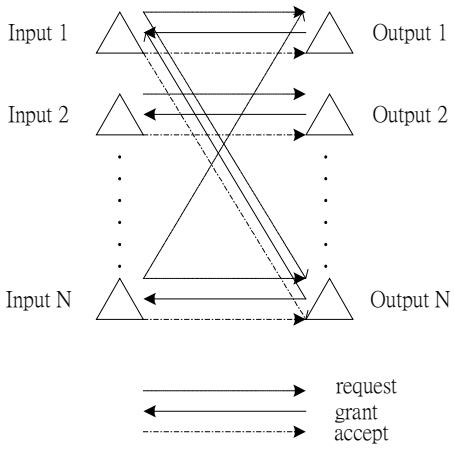

Figure 2. Distributed Scheduling Algorithm

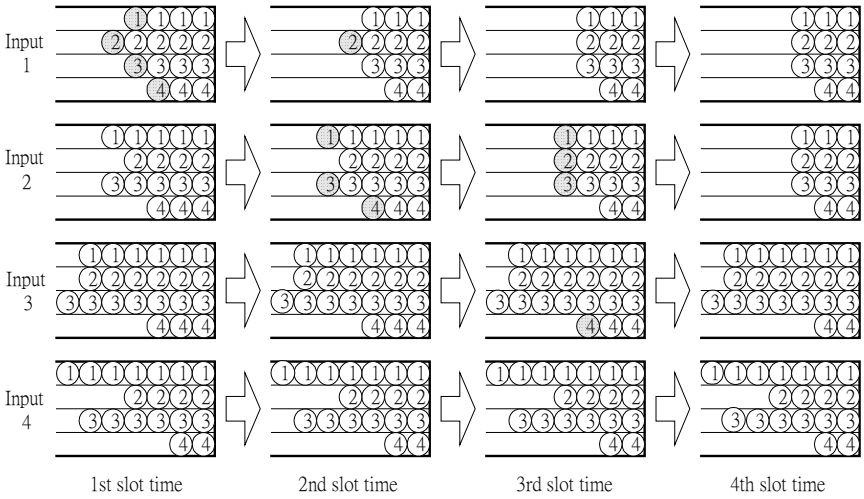

Figure 3a. T-RRM for 4*4 Switch

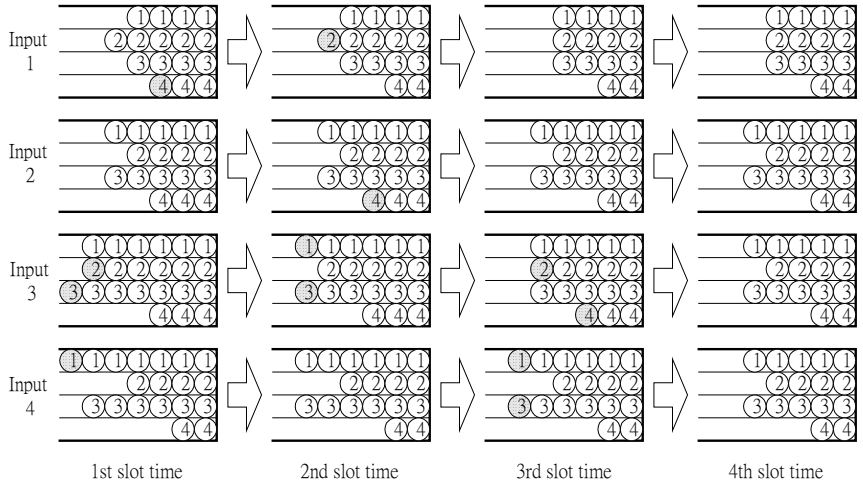

Figure 3b. B-VOQ for 4*4 Switch

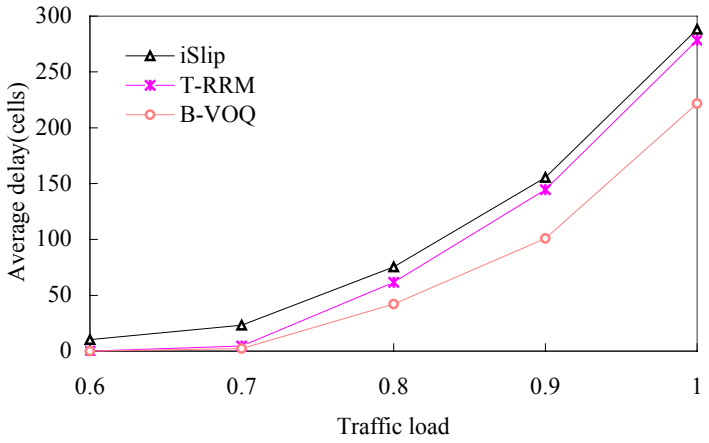

Figure 4. Threshold 設爲 2 時.

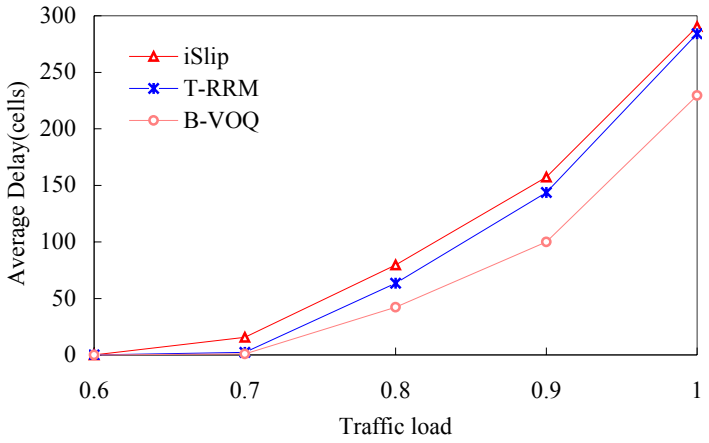

Figure 5. Threshold 設爲 3 時.

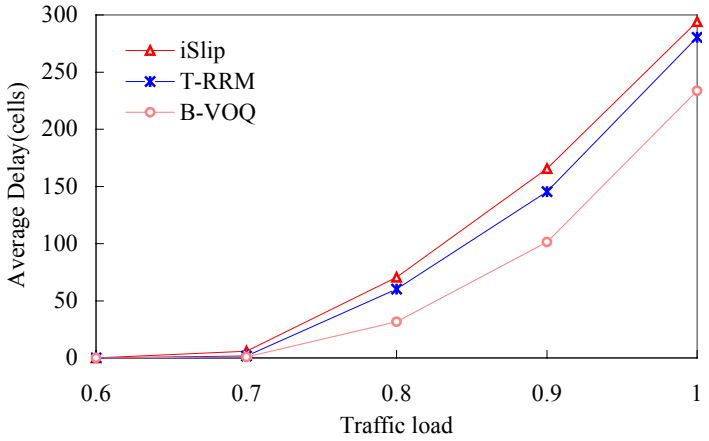

Figure 6. Threshold 設爲 4 時.

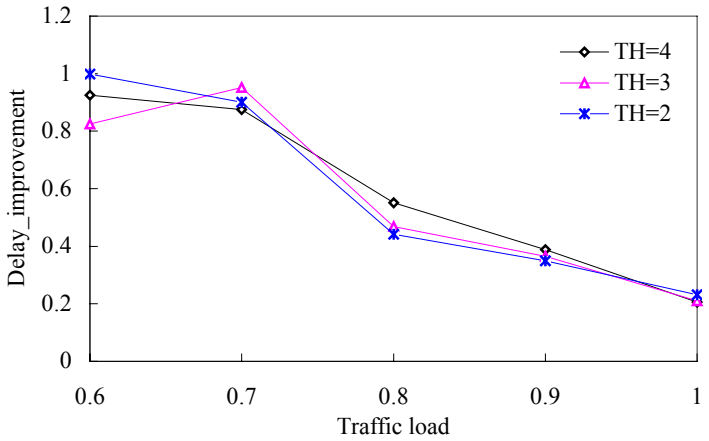

Figure 7. B-VOQ 對 iSlip Delay improvement.

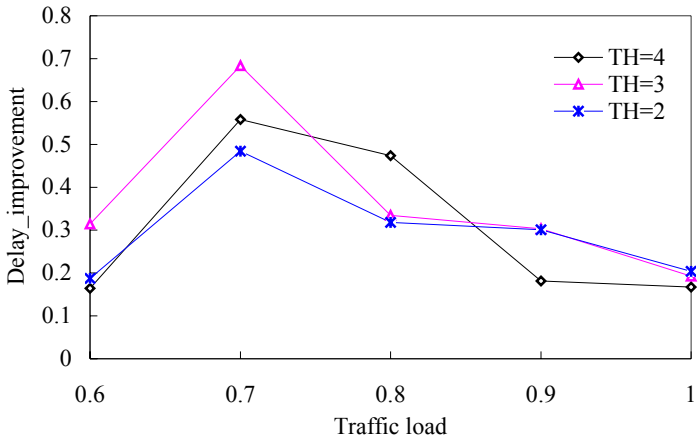

Figure 8. B-VOQ 對 T-RRM Delay improvement.