國立台灣大學資訊管理研究所

碩士論文

Department of Information Management

College of Management

National Taiwan University

Master Thesis

無線網狀網路TCP應用之排程設計及效能評估

Preserving Good TCP Transmission Properties in Wireless Networks

黃慈霖

Tzu-Lin Huang

指導教授：孫雅麗　博士

Advisor: Yeali Sun, Ph.D.

中華民國 九十七 年 七 月

July, 2008

目錄

圖目錄

表目錄

謝詞

這篇論文的完成需要感謝很多人。首先我要感謝指導我論文的孫雅麗老師以及口試委員：陳孟彰老師、林永松老師、蔡志宏老師。孫老師花了很多心力指導和修改我的論文，感謝孫老師從專題以來三年多的照顧和指導。感謝諸位口試委員給我的意見與指導，使我的論文更加完善。另外要感謝莉萍學姊，每個禮拜花很多時間跟我討論我的論文進度，給我很多非常有幫助的意見。感謝實驗室的各位學長姊、同學、學弟妹：育群學長、振維學長、世傑學長、舜文學長、力銘學長、士軒學長，以及去年畢業的元傑學長、鼎盛學長、雅華學姊，皓文、許君、昌桓、雲喬、昕彥、意婷，感謝這三年多來的支持與陪伴，讓我順利的度過專題以及碩士班的時光。最後我要感謝我的父母和姊姊，他們對我無止盡的關心、付出和支持，讓我無後顧之憂的念完碩士。

黃慈霖　謹識

民國九十七年七月

中文摘要

作者：黃慈霖

指導教授：孫雅麗 博士

論文題目：無線網狀網路 TCP 應用之排程設計及效能評估

TCP 在有線網路中有三項良好的傳輸特性：一、TCP 藉由調整 congestion window 來調節傳輸速度，使 TCP 可以有效利用網路的可用頻寬(available bandwidth)。二、對於 congestion 引發的 packet loss，TCP 可以很快速地重傳並恢復傳輸。三、流經相同 bottleneck link 的 TCP flows，可以達到非常公平的 throughput 分配。但在 CSMA/CA-based 的無線網狀網路上，packet loss 引發 TCP congestion control 機制使 TCP 降低傳輸速度而讓 TCP 無法有很好的 throughput。Channel access contention、hidden terminal 問題以及 wireless link error 都會造成 packet loss。

在本文中，我們使用 spatial TDMA scheduling scheme 解決 channel access contention 以及 hidden terminal 問題，並加上 link-layer retransmission scheme 在 local link 處理 wireless link error。我們進行模擬實驗，討論並分析使用這些 scheme 下無線網狀網路 TCP 效能。希望藉由這些 scheme 建造一個無線網路環境，使 TCP 不經更改也能保留 TCP 的良好特性，一如 TCP 在有線網路上一般順暢地運作。實驗結果顯示，考量 traffic load 的 spatial TDMA schedule (schedule B)可以讓 TCP 在無線網狀網路上保持很好的 throughput 和 fairness 並且盡可能地使用所有的可用頻寬。而對於 wireless link error 造成的 packet loss 會破壞 TCP throughput 和 fairness，實驗結果也證明使用 link-layer retransmission scheme 可以保存 TCP 快速復原 packet loss 的特性，使 TCP 在 error-prone 的無線網狀網路上也能有好的效能表現。

關鍵詞：spatial TDMA scheduling、TCP、無線網狀網路、Local Retransmission

Abstract

Name: Tzu-Lin Huang

Advisor: Yeali Sun

Preserving Good TCP Transmission Properties in Wireless Networks

In wired networks, there are several good properties for TCP transmission: exploiting available bandwidth as much as possible, providing fairness between different flows that share the same bottleneck link, and fast recovery from packet loss due to congestion. But in CSMA/CA-based wireless mesh networks, packet loss due to channel access contention, hidden terminal, and wireless link error will trigger the TCP congestion control scheme to slow down the data sending rate, thus results in low throughput performance.

In this thesis, we take spatial TDMA scheme to tackle channel access contention and hidden terminal problem in wireless mesh networks. And we also take link-layer retransmission scheme to sustain fast recovery properties. Under the proposed schemes, we analyze TCP performance in wireless mesh network in terms of the good TCP transmission properties. We want to construct a wireless environment so that one can preserve good TCP transmission properties to achieve good overall throughput performance in wireless mesh networks. Simulation results show that in a chain topology, spatial TDMA schedule (schedule B), which considering traffic demand, sustains good TCP fairness and exploits available bandwidth as much as possible. And link-layer retransmission scheme preserve properties of fast recovery of lost packets on error-prone wireless mesh networks.

Keywords: spatial TDMA scheduling, TCP, wireless mesh networks, local retransmission

1 序論

1.1. 背景介紹

1.1.1. TCP

Transmission Control Protocol (TCP)[2] ，是目前最被廣泛使用的 transport layer 通訊協定。TCP 提供 end-to-end 可靠(reliable)的傳輸，可確保上層交付下來的資料不會被遺失，因此眾多網路應用都以 TCP 做爲 transport layer 的通訊協定，例如 Web browsing、E-mail、FTP…等。

TCP 使用 window-based 的壅塞控制(Congestion control)演算法[2] [3] [4] ，可以探測網路的狀況並減輕 end systems 和 network router 的壅塞情形。在有線網路上，TCP 運作地非常順暢。TCP 傳送端使用一個 window 來紀錄已傳送出去但尚未被確認正確收到(unacknowledged)的資料，同時 TCP 傳送端會不斷地加大其 window size 以探測(probe)是否可使用更多的可用頻寬(available bandwidth)，直到發生 congestion 或是到達 TCP 接收端所能接收的最大 window size 爲止。TCP 的 window 變化可分爲兩個 phase - slow start phase 及 congestion avoidance。在新的 TCP connection 建立之後，TCP 處在 slow start phase，此時 TCP 傳送端在每個 round-trip time 以指數的方式倍增其 sending window，直到發生 packet loss 或到達 TCP 接收端所能接收的最大 window size 爲止。packet loss 代表 congestion 發生，TCP 傳送端將其 window 減半，進入 congestion avoidance phase，此時 sending window 在每個 round-trip time 以線性的方式成長，若再度發生 packet loss 則 sending window 則再減半。TCP 藉由兩個 event 來偵測 packet loss 的發生：timeout 和 three duplicate ack。發生 timeout 代表有嚴重的 congestion 發生，TCP 傳送端

會馬上將 congestion window 降至最低以減輕 congestion。TCP 使用 cumulative ACK，意即當 TCP 接收端收到 sequence number x 的 ACK 時，代表 x 之前的封包都已正確送達。當 TCP 接收端收到非正確順序的資料封包時，會回傳截至目前最後收到的封包所相對應的 acknowledgement，因此當一連串 unacknowledged 的封包中的某個封包遺失時，TCP 傳送端就會收到數個 duplicate ACK，若收到 3 個以上的 duplicate ack 即代表某個封包遺失，進而進行 fast retransmission 來復原遺失的封包。這個演算法使 TCP 可以很快地探測網路上可使用的最大頻寬，卻又不致於傳送過量的資料到網路上造成壅塞，而當 congestion 發生時又可以很快地發現並復原 packet loss。

當多條 round-trip time 不同的 TCP flow 共同經過同一條 bottleneck link，不會達到相同數量 throughput，其 throughput 可根據 TCP friendly equation [11] 推估：

$$\text{throughput} = \frac{c}{RTT\sqrt{p}} \qquad \text{Eq. 1}$$

p 爲該 TCP flow 的封包被遺失的機率，RTT 爲該 TCP flow 的 round-trip time，c 爲一固定的常數[11] 。根據這個 equation，當兩個 TCP flows，flow *i* 和 flow *j* 流經同一個 bottleneck link 而且有同樣的封包遺失機率 p，則這兩條 TCP flows 的 throughput 比例應該跟其 RTT 成反比。

$$\frac{\text{throughput}_i}{\text{throughput}_j} = \frac{RTT_j}{RTT_i} \qquad \text{Eq. 2}$$

這是由於 TCP 的 Additive Increase Mulplicative Decrease scheme 的緣故，TCP 每經過一個 RTT(收到 ACK)就調整其 congestion window，RTT 較小的 flow 其 congestion window 亦成長得較快，因此得到較高的 throughput。

總結來說，TCP 有三項好的傳輸特性使之在有線網路上運作地很順暢：

一，TCP 的 AIMD scheme 可以調節 congestion window 使 TCP 盡可能的利用可用頻寬。

二，TCP 的 AIMD 機制可以探測網路上的壅塞狀況並很快地復原 packet loss。

三，流經相同 bottleneck link 的 TCP flows 之間可以保持很好的 throughput fairness。

1.1.2. 無線網狀網路

無線網狀網路(Wireless mesh network)[1] 是近年受到很多注目和被廣泛討論的一種網路架構，它由許多 wireless mesh node 構成，使用者可以利用無線裝置連接上 wireless mesh node 而進入 Internet。每個 wireless mesh node 不一定都可以直接連接到 Internet，因此 wireless mesh node 會將資料經由無線的方式傳到具有 gateway 功能的 wireless mesh node 再傳至 Internet 的。無線網狀網路跟一般的 wireless LAN 不同之處即在於此，wireless mesh node 的 traffic flow 是經過多站(multi-hop)的無線傳輸，不同於一般單站(single-hop)傳輸的 wireless LAN traffic。因此 wireless mesh network 具有延伸 Wireless LAN 傳輸範圍的功能和佈建成本便宜的優點。一個可能的 wireless mesh network topology 如圖 1-1，mesh router 具有 access point 的能力讓使用者裝置連接上來，同時也幫忙 relay 其它 mesh router 傳來的 traffic，因此不必到處佈建「直接連接 Internet 的 access point」，只要佈建無線的 mesh router。在大部份的狀況下，佈建有線的 access point 成本較高，建置 wireless mesh network 是較便宜且較具吸引力的選擇。

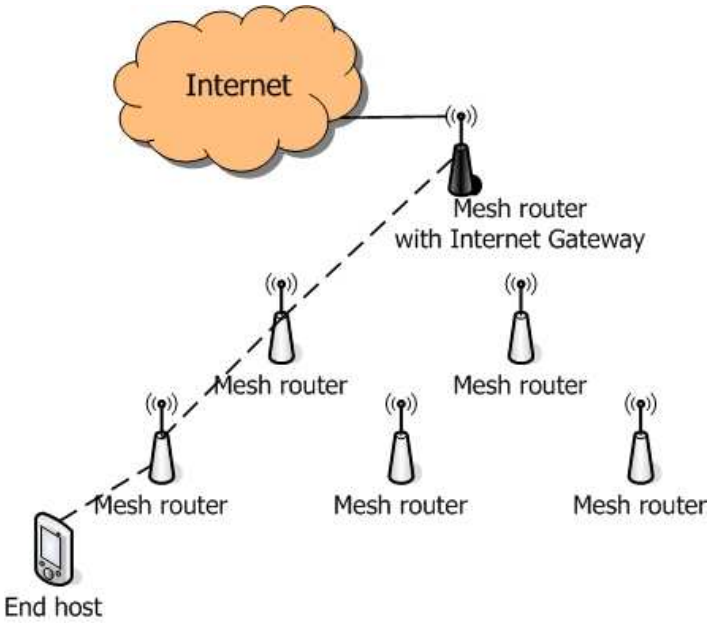


圖 1-1 Wireless mesh network 示意圖

在 Wireless mesh network 之中，相鄰的 wireless node 共享 wireless resource，因此有 channel interference 的問題以及 hidden terminal/exposed terminal 的問題。

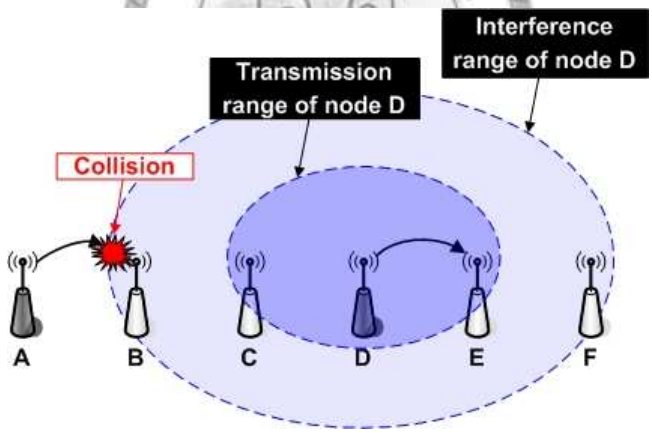


圖 1-2 Hidden terminal 問題

Hidden terminal 問題可以用圖 1-2 來說明，假設圖上所有 node 使用相同的 wireless channel，每個 node 的傳輸範圍與干擾範圍都相同，且干擾範圍約爲傳輸範圍的兩倍。node D 正在傳送 data 給 node E，由於 node A 無法偵測到 node D 正在傳送 data，誤以爲 channel 爲 idle 狀態，因此傳送 data 給 node B，但由於 node B 處於 node D 的干擾範圍之中且 node D 正在發送訊息，因此 node A 傳給 B 的 data 無法正常地被接收。Node D 爲 node A 的 hidden terminal。

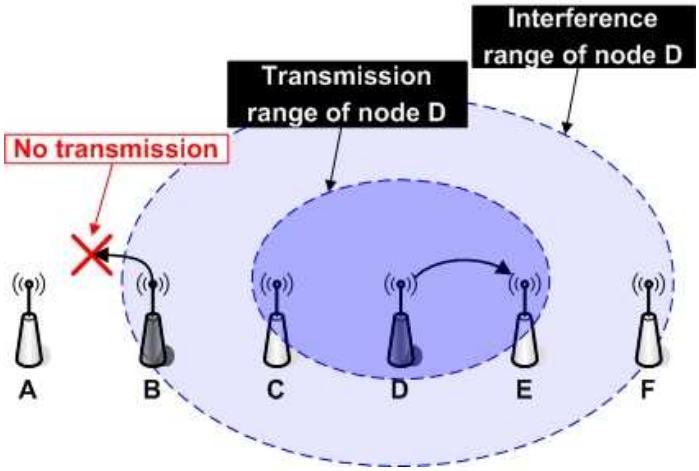


圖 1-3 Exposed terminal 問題

Exposed terminal 問題如圖 1-3 所示。Node D 正在傳送 data 給 node E，由於 node B 位處於 node D 的干擾範圍內，於是 node B 偵測到有其它 node 在使用 channel，因此原本要傳送 data 給 A 的動作被延遲了，然而事實上，node B 可以跟 node D 同時進行傳輸且不會互相干擾到對方的接收端。Hidden terminal 問題產生 collision，使 channel resource 被浪費掉；exposed terminal 問題使原本可以同時進行的傳輸被延遲，沒有有效地利用 channel resource。這兩個問題都造成 wireless mesh network 的資料傳輸更加困難。

1.1.3. TCP Bandwidth Requirement

Wireless mesh network 一些特有的性質會影響 TCP 的 throughput。第一，無線網路是一個共享的傳輸媒介，相鄰的 links 會互相干擾，造成 link 與 link 彼此互相競爭 channel access (MAC layer contention)。此外，還有前述的 hidden terminal/exposed terminal 問題。第二，TCP 是 reliable 的 transport protocol，必須依賴 ACK 的回傳才能順利運作，但是 wireless medium 是 link 之間共享的，且一般而言，TCP 連線中的 data flow 和 ACK flow 會走向相同的路徑，造成同一條 TCP flow 的 data link 和 ACK 相互競爭共有的 wireless medium。因此 data link 和 ACK link 的頻寬分配也是影響 TCP 效能的關鍵。我們可以用 normalized

bandwidth ratio k 來描述 TCP data packet 和 ACK packet 頻寬需求[12] ：

$$k = \frac{\text{data packet channel bandwidth}}{\text{ACK packet channel bandwidth}} \times \frac{\text{ACK packet length}}{\text{data packet length}} \times \frac{1}{d} \quad \text{Eq. 3}$$

其中 d 爲 delay-ACK factor。假設 data packet length 爲固定值，則最佳的頻寬分配是在 k=1 時，此時 data packet 和 ACK packet 得到的頻寬比等於它們的封包長度比。

在後面的章節中，我們的 TDMA scheduling scheme 會同時考慮這兩項特質，並分析使用此 scheduling scheme 的 TCP performance。此外，wireless link 具有容易發生 error 的特質，因此具有很高的 packet loss probability，我們也會藉由實驗分析討論 packet loss 對 TCP performance 的影響，並尋找解決方法。

1.2. 研究動機

由於 wireless mesh network 逐漸成爲熱門的網路架構以及眾多網路應用使用 TCP 做爲 transport layer 通訊協定，如何在 wireless mesh network 上提供良好的 TCP 效能變成一項重要的議題。

1.1.3 小節討論了 TCP 運行在 wireless mesh network 上遇到的問題。以往在 TCP over wireless mesh networks 上的研究[5] [6] [7] [8] [9] ，大部份都是考慮在單一 interface、CSMA/CA-based MAC 的 stationary wireless mesh network 上討論 TCP 的效能以及提升 TCP throughput，著重在處理 TCP 跟 IEEE 802.11 MAC 或 CSMA/CA-based MAC 之間的交互作用以及 hidden terminal/exposed terminal 的議題。使用 CSMA/CA-based 的 MAC，有 channel access contention 的問題，鄰近的 wireless nodes 互相爭搶有限的 wireless resource，非常容易發生 collision 而浪費 wireless resource。若使用 TDMA-based MAC，可以避免因 channel contention 造成的 collision，妥善地使用可用頻寬並提供上層頻寬保證的服務。

1.3. 研究目的

本篇論文想要回答一個問題：假如使用 spatial TDMA 進行排程設計，我們可以保留多少 TCP transmission 在有線網路上的良好特性，使之能在 wireless mesh network 上提供好的 overall throughput 表現？我們考慮 TCP 傳輸的頻寬不對稱特性，使用 spatial TDMA 進行排程設計，在 chain topology 的 multi-hop wireless network 上進行實驗觀察 TCP 的效能，希望能藉由 spatial TDMA 排程設計及其它方法盡可能地保存 TCP 在有線網路上的良好特性(如 1.1.1 所述)。

1.4. 論文架構

本篇論文之後的章節組織如下：第二章整理有關 TCP performance over wireless mesh network 的文獻；第三章，我們將介紹兩種 TDMA scheduling scheme – equal-shared 的 schedule A 和 demand-based 的 schedule B，並進行模擬實驗，討論其在 chain topology 的 wireless mesh network 和理想的無線環境下對 TCP 效能的影響。第四章，我們將討論當使用這兩種 scheduling scheme 的 TCP flows 運行在狀況不佳的無線環境之中(會因爲 wireless link error 發生 packet loss)的效能表現。第五章，我們使用 link-layer retransmission scheme 來改善 TCP 因 packet loss 而下降的效能，並討論其優缺。第六章是總結我們的研究以及未來的研究方向。

2. 文獻探討

在文獻上，改進 wireless mesh network 上 TCP 效能的方法主要分三類：從 MAC layer 著手、從 transport layer 著手、以及其它方式。

2.1. MAC Layer Approach

2.1.1. Distributed Link Random Early Drop

Distributed Link Random Early Drop (簡稱 LRED)由 Z. Fu 等人提出[5] ，這篇 paper 主要在討論 wireless mesh network 上 TCP 的 throughput 以及封包遺失的成因。在這篇 paper 中，作者們觀察到 TCP 在 wireless mesh network 上的 average window size 通常都比 optimal window size 來得大，因此 TCP 常常送過量的 data 到網路中，然後又因嚴重的 MAC-layer contention 造成封包遺失而迅速降低 window，在這樣反覆升升降降的過程中使網路的 utilization 沒有最佳化，造成 TCP throughput 不佳。LRED 的主要理念是根據 wireless mesh router 觀測到的 MAC layer queue length 調整 wireless link 的 dropping probability，這樣的作法類似有線網路的 router 的 Random Early Drop 方法。當 buffer queue 到達一定高度時 (但尚未 overflow)，LRED 讓封包以一固定機率遺失，藉此通知上層的 TCP「網路已經趨近 overload」，使 TCP 降低傳送速度，同時也可以避免過多的封包送到 wireless channel 上產生過度的 MAC-layer contention。

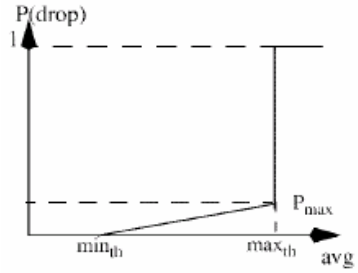


圖 2-1 LRED 機制[5]

LRED 的作法可以用圖 2-1 來說明。橫軸爲 MAC-layer 的 average queue length，當 queue length 超過 minimum threshold 時，啓動 LRED 讓封包以一定的機率遺失，封包遺失的機率隨 queue length 成正比。這個方法的好處是讓 wireless mesh network 在 overload 之前就降低傳輸速度，避面進一步的惡化 MAC-layer contention 的狀況。

2.2. Transport Layer Approach

文獻上也有一些改善方式從 transport layer 著手，修改 TCP 的 congestion control 機制使得 TCP 能適應 wireless mesh network 的環境，以提供更好的 throughput，以下逐一介紹。

2.2.1. Dynamic Adaptive Acknowledgement Strategy

R. Oliveira 和 T. Braun[6] [7] 提出 dynamic adaptive acknowledgement strategy 以提升 TCP 在 wireless mesh networks 上的 throughput。這兩位作者認爲 TCP 在 wireless mesh network 上最大的問題在於太多「不必要的重傳」以及 data 和 ACK 封包之間互相競爭 wireless channel resource。他們把 RFC 2581[2] 中提出的 delayed ACK 機制延伸，接收端延遲 ACK 的發送以減少 ACK traffic，並根據

channel condition 猜測傳送端的 congestion window 以動態調整 delayed ACK 的數量。此法利用 TCP 的 cumulative ACK 的特性，sequence number 爲 *i* 的 ACK 可以 acknowledge 前面所有 sequence number ≤ *i* 的 data packet，以減少 ACK traffic 量。

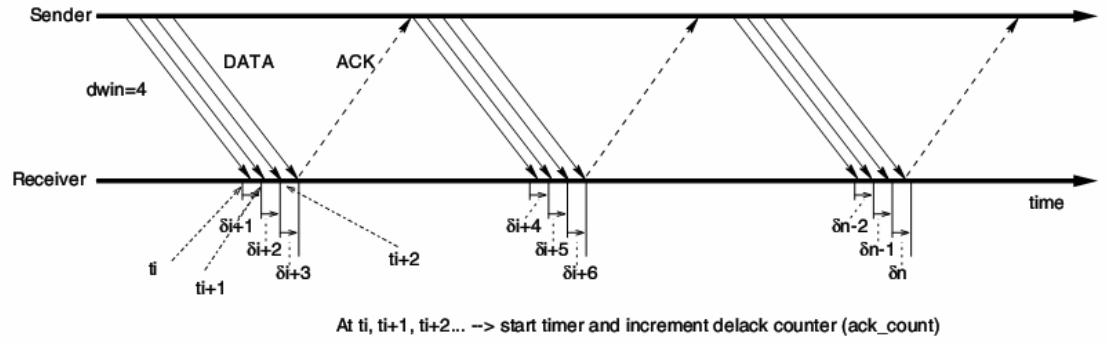


圖 2-2 TCP-DAA[6] [7]

Dynamic adaptive acknowledgement strategy 的方法可以用圖 2-2 來描述，在傳輸順暢的狀況下，接收端一直延遲 4 個 ACK(對於每 4 個連續的 data 封包回傳一個 ACK 封包)，4 爲 delay ACK factor。當封包遺失時再調降 delay ACK factor。

2.2.2. Adaptive Pacing

S. ElRakabawy 等人[8] [9] 提出 TCP adaptive pacing 的機制。Adaptive pacing 的構想可用圖 2-3 來說明。在簡單的 chain topology 上，假設傳輸範圍爲 250 公尺，干擾範圍爲 550 公尺，wireless node 沿著一直線擺放且相鄰 200 公尺。在這樣的 topology 上，連續的 4 個 hop 之內同時只能有一個 node 進行傳輸，否則就會發生 collision。以圖 2-3 爲例，node D 正在傳輸，因爲其干擾範圍涵蓋 node B、C、E、F，因此這些 node 都不能接收其它 node 的傳輸。以此類推，在長度爲 h hops 的 chain topology 上，最多只能有 h/4 個 node 在傳輸，也就是讓網路上同時最多只有 h/4 個 data packet in flight。因此 TCP with adaptive pacing 的主要構想就是讓連續的 data 封包彼此間隔 4 個 hop 以上。作者計算 data packet 通過 4 個 hop 所

需的時間，稱爲 FHD（four hop propagation delay），讓傳送端在傳送封包時，使每個連續封包之間的時間差都至少大於 FHD，如此可以達到減少 channel interference 以提升 TCP throughput 的目的。

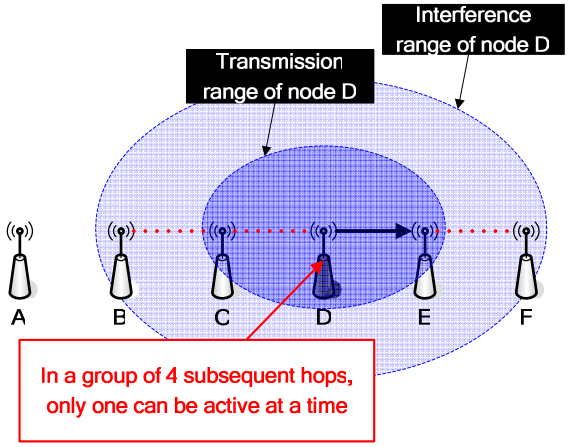


圖 2-3 Spatial reuse

Adaptive pacing 的主要理念來自於對 chain topology 的觀察，因此這個方法在其它 network topology 上表現不佳，同時 adaptive pacing 將 TCP 的傳送速度限制住，使 network utilization 沒有被最大化。

2.3. Other Approach

有學者提出新的 transport layer 通訊協定，這類通訊協定多半爲了 operability 而建置在 TCP 之下，使所有使用 wireless mesh network 的裝置不用更改通訊協定，只要在 mesh router 上裝置新的通訊協定即可。

K. Tan 等人[10] 針對 wireless mesh network 的特性提出新的 congestion control protocol，稱作 Explicit Wireless Congestion Control Protocol (EWCCP)。這個 protocol 設置在 TCP 之下 IP 之上，並在 data 封包上加入自訂的 header 以提供 mesh router 觀察到的 congestion 狀態。EWCCP 以 queue length 作爲判斷 congestion 的依據，相鄰的 mesh router 互相傳遞封包通知對方自己的 queue length，然後每個 mesh router 再根據自己以及鄰居的 queue length 計算出合理的 sending rate，分

配到每條 transport layer 的 flow 上，並將此值填入相對應的 flow 的封包 header 中，以告知該 flow 的 sender 調整速度。這樣的作法並不侷限於某些特殊型態的 network topology，但可能的缺點是 mesh router 互相之間要傳遞 queue length information，消耗掉有限的 wireless resource，此外新增的 header 也帶來額外的 overhead。

2.4. Improving TCP performance on error-prone wireless link

Wireless link 的一項特質是容易發生 link errors 而造成 packet loss，如同第一章討論的，TCP 分辨不出 packet loss 是因爲 congestion(queue buffer 滿了)還是 wireless link errors，因此 TCP 不當地降低 congestion window 造成 throughput 下降。文獻上有許多文章討論如何改善因 wireless link errors 而造成 TCP 效能下降的問題，主要可分成三類：TCP snoop、link-layer retransmission 及 split-connection。

2.4.1. TCP snoop

Balakrishnan 等人[15] [16] 提出 TCP Snoop protocol，幫助改善 TCP 在 wireless link 上的 performance。TCP Snoop 基本上也是一種 local 的 retransmission scheme，但它不依賴 link-layer 傳送 ACK，而是在 link-layer 加上 Snoop module 以 monitor TCP flow 的狀態。Snoop module 會將經過 wireless station 但尚未被 acknowledged 的 TCP data packet 暫存。當 Snoop module 發現第二個 duplicate ACK 的時候，立即在 wireless link 上進行 retransmission，並將之後的 duplicate ACK 丟棄以避免 TCP 發生 fast retransmission。TCP Snoop 的問題在於 wireless station

必須要維護所有流經其上的 TCP flows 的 state，並且要 cache 這些 TCP flows 的 unacknowledged data packet。

2.4.2. Local link-layer retransmission scheme

Eckhardt 等人[14] 討論 local link-layer retransmission 對 TCP performance over error-prone wireless link 的改善。他們認爲對付 local wireless link error 最佳的方式是 local 的 error control，因此進行 emulation 實驗討論如何設計理想的 link-layer retransmission 及 error control scheme 使 TCP 正常運作。他們發現，如果要讓 TCP 維持 steady state，link-layer retransmission scheme 必須避免發生 packet reordering 而造成 TCP 的 fast retransmit，此外 retransmission 的速度必須要夠快以避免 TCP 發生 timeout。

2.4.3. Split-connection

另外一種改善 error-prone wirelesss link 上 TCP 效能的方法是 split-connection，如 Bakre 等人提出的 I-TCP[17] 。Split-connection 的作法將 TCP 傳送端到接收端的 TCP 連線分爲二段連線(有線和無線)，第一段(有線)由有線傳送端到 AP，第二段(無線)則由 AP 到無線的接收端，因此在無線這段 TCP 連線所遭遇的任何 packet loss 都不會影響原來的有線 TCP 傳送端。Split-connection 的缺點在於它破壞 TCP end-to-end semantics，當有線段的傳送端送出某個封包後，該封包尚未真正地被傳送到無線的接收端之前，傳送端就可能已經收到中間點 AP 所回傳該封包的 ACK，因此 TCP 傳送端對可用頻寬以及網路狀態的探測可能是錯的。

2.5. 討論

前述文獻討論的方式皆使用 CSMA/CA-based 的 MAC 作爲 medium access protocol，CSMA/CA 最大的缺點是讓 wireless node 以隨機的方式競爭 channel，由於現有的 wireless interface 不能在傳送資料的同時偵測 collision 的發生，因此發生 collision 的代價很高，而 CSMA/CA 發生 collision 之後的 backoff 也帶來很大的 overhead，channel resource 無法有效地被利用，最終無法得到良好的 TCP throughput。Hidden terminal 問題也會增加 collision 發生的機率。我們認爲理想的方式應是使用 centralized spatial TDMA (time division multiple access) scheduling scheme，妥善並公平地分配 channel resource，使 TCP 的 throughput 提升。TDMA 可以避免 packet collision，同時可以利用 spatial reuse 特性解決 channel interference 的問題。

以往討論改善 TCP performance over wireless links 的文獻，多半是聚焦在 single-hop 的 wireless network，這個問題在 multi-hop 的 wireless mesh network 比較複雜，因爲每條 TCP flows 的 path hop 長短不一，wireless link errors 對於長短不同的 TCP flows 可能會有不同程度的傷害。我們使用 link-layer retransmission 來處理 multi-hop wireless network 上 wireless link error，希望能保持 TCP 的良好傳輸特性提升 throughput。

3. TDMA 排程及 TCP 效能分析

我們將在這一章討論如何分配頻寬使 TCP 順暢流動(*k*=1 model)，並依此提出兩種 TDMA scheduling 演算法，並進行模擬實驗討論其 TCP 效能。

3.1. k=1 Model

TCP 是 feedback-based 的傳輸協定，TCP 傳送端依賴 ACK 確認封包是否已送達並將之從 buffer 中清掉，同時 TCP ACK flow 也影響 congestion window 的變化，因此 TCP throughput 同時受到 data packet flows 和 ACK packet flows 的影響。TCP data flow 和 ACK flow 如果走相同的路徑，則如何將此路徑的頻寬分配給 TCP data 和 ACK flow 以使得 TCP throughput 最大化是一個問題。

我們使用 normalized bandwidth ratio *k* [12] 來描述 TCP data flow 和 ACK flow 的頻寬需求。

$$k = \frac{\text{data packet channel bandwidth}}{\text{ACK packet channel bandwidth}} \times \frac{\text{ACK packet length}}{\text{data packet length}} \times \frac{1}{d} = \frac{C_{data}}{C_{ack}} \times \frac{L_{ack}}{L_{data}} \times \frac{1}{d} \quad \text{Eq. 4}$$

其中 *d* 爲 delay-ack factor，代表 TCP 接收端每收到 *d* 個 data packet 回傳 1 個 ACK。*k*=1 時是 TCP 運作得最順暢，此時 data packet 和 ACK packet 得到的頻寬比正好等於 packet length 的比，換句話說，在 *k*=1 的情況下，我們每傳送出一個 TCP packet 就會剛好對應到回傳一個 ACK packet。當 *k*>1 時，ACK packets 所獲得的頻寬不足，因此接收端以較慢的速率傳送 ack，這樣會造成傳送端的 congestion window 成長速度緩慢，進而造成 TCP throughput 表現不佳。當 k<1 時，data packet 所獲得的頻寬不足，傳送端被迫以較慢的速度傳送 data，而 TCP

ACK packet 的傳輸端看接收端收到多少的 TCP data packet，在這樣的狀況中，TCP ACK flow 受到 data flow 的影響也以較低的速度傳送，因此 ACK flow 並沒有完全地使用到它所被分配的頻寬。因此，我們稍後在進行 centralized TDMA scheduling 分配 timeslot 時將以在每段 wireless link 以及接收端都達到 normalized bandwidth ratio k=1 爲目標，以使 TCP 能以最順暢的方式運作。

TDMA scheduling 將 channel access time 切成一個一個 timeslot 分配給 wireless network 上的 node，每個 wireless node 只能在它所分配到的 timeslot 之中傳送封包(無論 data 封包或 ACK 封包)。當所有 wireless node 獲得足夠傳送資料的 timeslots，則這些 timeslots 集合成一個 frame。定義 N_{frame} 爲一個 frame 中的 timeslot 數量，如果能使 N_{frame} 最小化，也就是讓所有 wireless node 在最短的時間內都獲得足夠傳送資料的 timeslots，即可最大化 throughput。

因此，給定一個固定的 network topology，我們的 TDMA scheduling 希望達成兩個目標：

1 在每個 hop 上達到 normalized bandwidth ratio $k = 1$，

$$k = \frac{\text{data packet channel bandwidth}}{\text{ACK packet channel bandwidth}} \times \frac{\text{ACK packet length}}{\text{data packet length}} \times \frac{1}{d} \qquad \text{Eq. 5}$$

，使 TCP flow 順暢地流動。

2 一個 frame 所包含的 timeslot 數(N_{frame})爲最小，如此可以最小化一個傳輸循環所需的時間，以達到最大 throughput。

Wireless mesh network 的組成中含有一個以上具有 gateway 能力的 mesh node，大部份的 traffic flow 皆流向此 gateway node 或從 gateway node 流出以連接 Internet。我們的 system model 先考慮 chain topology 的狀況，如圖 3-1：

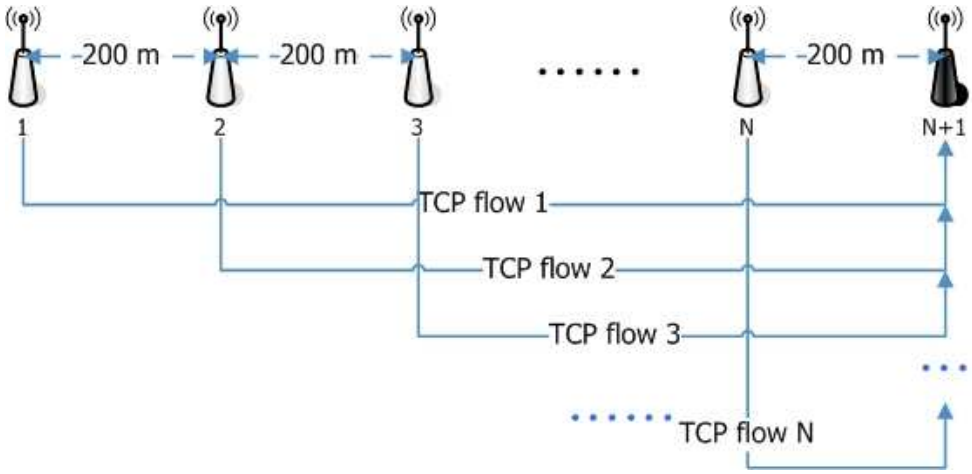


圖 3-1 N-hop chain network

在這個 N-hop chain 上每個 wireless node 都有一 TCP flow 流向連接 Internet 的 node (N+1)。

我們定義一個 bidirectional graph G=(V,E)，其中的 V 爲節點的集合，表示 wireless mesh network 中的 wireless nodes，E 爲 edge 的集合，v_i 指向 v_j 表示 node *j* 在 node *i* 的 transmission range 之內。接著，爲了將 wireless mesh network 中各個 wireless link 的干擾關係表達出來，我們將 G 轉換成 contention graph G'=(V',E')， 其中 V'爲 vertex 的集合，表示 G 上的 directional wireless link，E 以 E'爲 edge 集合，表示 directional wireless link 的干擾關係。這邊以一個 four-node 的 chain topology 爲例，如圖 3-2：

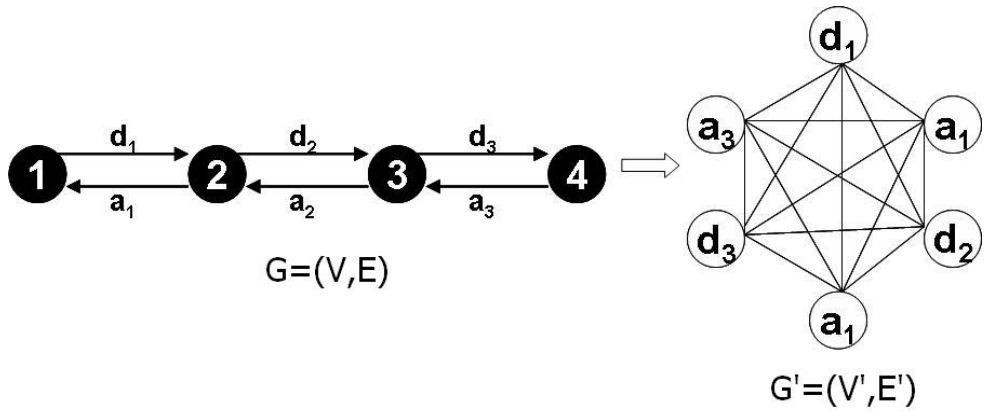
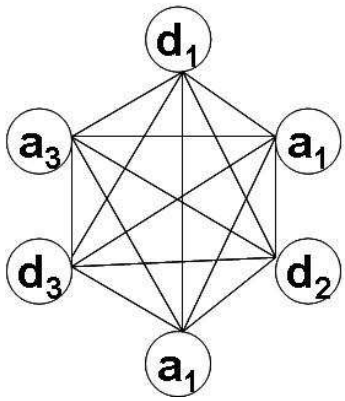


圖 3-2 Contention graph

左邊爲 network topology graph，其中每個 node 間隔 200 公尺，因此 d_1 會跟圖上所有 wireless link 互相干擾，因此在右邊的 contention graph 中，vertex d_1 跟所有 vertex 都有連線，表示互相干擾。

爲了達成前述的 *k*=1 條件，每個 wireless link 所分配到的 timeslot 數量應該要符合其 TCP data flow 或 ACK flow 的頻寬需求。我們將 contention graph 上的每個 vertex 給定一個權重值如圖 3-3，此權重值即代表該 wireless link 的頻寬需求。



$$w(d_1)=\sum_{\forall f_i\in D_1}\frac{L_{d,f_i}}{L_{a,f_i}},\quad w(a_1)=\sum_{\forall f_i\in D_1}\frac{L_{a,f_i}}{L_{a,f_i}}=1,$$

$$w(d_2)=\sum_{\forall f_i\in D_2}\frac{L_{d,f_i}}{L_{a,f_i}},\quad w(a_2)=\sum_{\forall f_i\in D_2}\frac{L_{a,f_i}}{L_{a,f_i}}=1,$$

$$w(d_3)=\sum_{\forall f_i\in D_3}\frac{L_{d,f_i}}{L_{a,f_i}},\quad w(a_2)=\sum_{\forall f_i\in D_3}\frac{L_{a,f_i}}{L_{a,f_i}}=1$$

圖 3-3 Weight assignment

圖 3-3 以 four-node chain topology 爲例，圖中 $w(d_i)$表示 link d_i 分配到的權重值，的 $L_{d,fi}$ 表示 TCP flow *i* 的 data packet 長度，$L_{a,fi}$ 表示 flow *i* 的 ACK packet 長度，D_j 表示所有流經 link d_j 的 flow 所形成的集合。TDMA scheduling 演算法及按照此 weighted contention graph 進行 timeslot 的分配及排程。

3.2. TDMA Scheduling 演算法

3.2.1. Schedule A – Equal Share on each wireless link

最簡單最直覺的 scheduling 方式是讓每段 wireless link 有相同分量的 timeslot，但遵循 *k*=1 的原則，意即在每個 frame 之中的每段 wireless link 所得 timeslot 數剛好可以傳送一個 TCP data packet 和一個 TCP ACK packet，如此我們可以讓 TCP flow 最順暢地運作。

Schedule A 演算法的 pseudo code 如圖 3-4(第一階段)及圖 3-5 圖(第二階

段)。演算法第一階段先將所有 weighted contention graph 上的 vertices 區分成數個 scheduling groups，在同一個 scheduling group 中的所有 vertices(代表 wireless link)同時傳輸不會互相干擾。演算法第二階段則根據每個 scheduling group 中的 vertices 所得到的權重，分配 timeslots 給此 scheduling group。在同一個 scheduling group 中的 wireless link 會被排在同一時間內傳送，但這些 wireless link 的權重大小不一，意即所需的 transmission time 大小不同，因此一個 scheduling group 所分配到的 timeslot 決定於其中權重最大的 wireless link：

$$w(S_g) = \max\{w(u) \mid u \in S_g\};$$ Eq. 6

其中 S_g 爲一 scheduling group，u 爲 S_g 中的一個 wireless link。在演算法的第一階段進行區分 scheduling group 的動作時，我們使用 greedy approach，希望盡可能包含最多的 non-conflicting link 在同一個 scheduling group，以達到 spatial reuse 的效果並最小化 frame 的長度。

```
/* step.1  partition vertices of G' into non - conflicting scheduling groups S_g */
U = the set of vertices of weighted contention graph G';
g = 0; /* initial group number index */
while U ≠ φ {
    g++;
    select the vertex v with the largest weight from U;
    U = U - {v};
    S_g = S_g + {v};
    W = U;
    while W ≠ φ {
        if vertex with the largest weight w in W does not conflict with any vertex in S_g {
            S_g = S_g + {w};
            U = U - {w};
        }
        else
            W = W - {w};
    }
    w(S_g) = max{w(u) | u ∈ S_g}; /* compute the maximum timeslots for each scheduling group */
}
```

圖 3-4 Schedule A 演算法第一階段

將 scheduling group 分好之後，接下來以演算法的第二階段就是進行 timeslot

assignment。每個 timeslot 的長度爲傳送一個 TCP ACK packet 所需時間。

```
/* step.2  do the slot assignment sequentially by randomly selecting the secheduling group */
S = {S_g | g ∈ 1...k}; /* there are k scheduling groups */
i = 1; /* initial timeslot number index */
while S ≠ φ {
    randomly select a scheduling group S_j from S;
    next_group_step = w(S_j);
    while S_j ≠ φ {
        randomly select a wireless link v from S_j, where v represents a sender - receiver pair (s, r);
        mark the sender's state vector from i to w(v) as sending;
        mark the receiver's state vector from i to w(v) as receiving;
        S_j = S_j − {v};
    }
    i = i + next_group_step;
    S = S − {S_j};
}
```

圖 3-5 Schedule A 演算法第二階段

利用這個演算法，我們可以給定 four-node chain topology 的 wireless mesh network 產生如圖 3-6 的 scheduling。因爲 four-node chain topology 中的所有 wireless link 都會互相干擾，因此每個 scheduling group 只包含一個 wireless link，所以在分配 timeslot 時，沒有任何可以達成 spatial reuse 的機會。

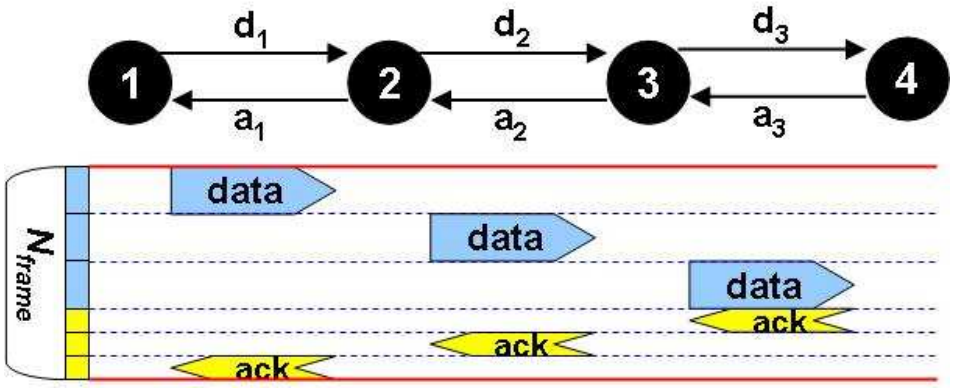


圖 3-6 Schedule A of four-node topology

在長度較長的 chain topology 中，如 seven-node chain topology，可以產生如圖 3-7 的 schedule。其中幾組互不干擾的 links (如 link d_1 和 link d_5) 被分配到同一個 scheduling group 以達到 spatial reuse 的效果。

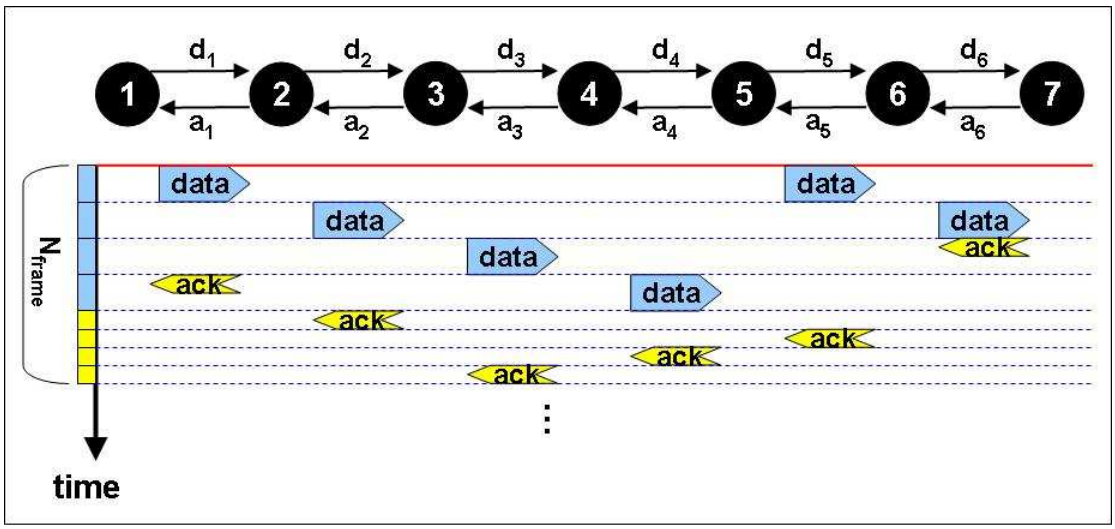


圖 3-7 Schedule A of seven-node topology

3.2.2. Schedule B – Demand-Based

我們考慮的 network topology 中，每個 wireless mesh node 都有一條 TCP flow 流向 gateway node (在 N-hop chain 中就是 node N+1)，因此連接 gateway node 的 wireless links 是整個網路中 loading 最重的部份(bottleneck link)，在前一小節中的 schedule A 並沒有考慮 loading 對 throughput 的影響。如果我們將經過 link d_i 的 TCP flows 數量定義爲 link d_i 的 load，並在進行 scheduling 時分配 timeslot 的步驟中將 wireless link 的 loading 一併考慮，可能可以提升 TCP throughput。我們稱這個按照 traffic demand 的 scheduling 方式爲 schedule B。提出的 schedule B 的理由可以用圖 3-8 來說明：

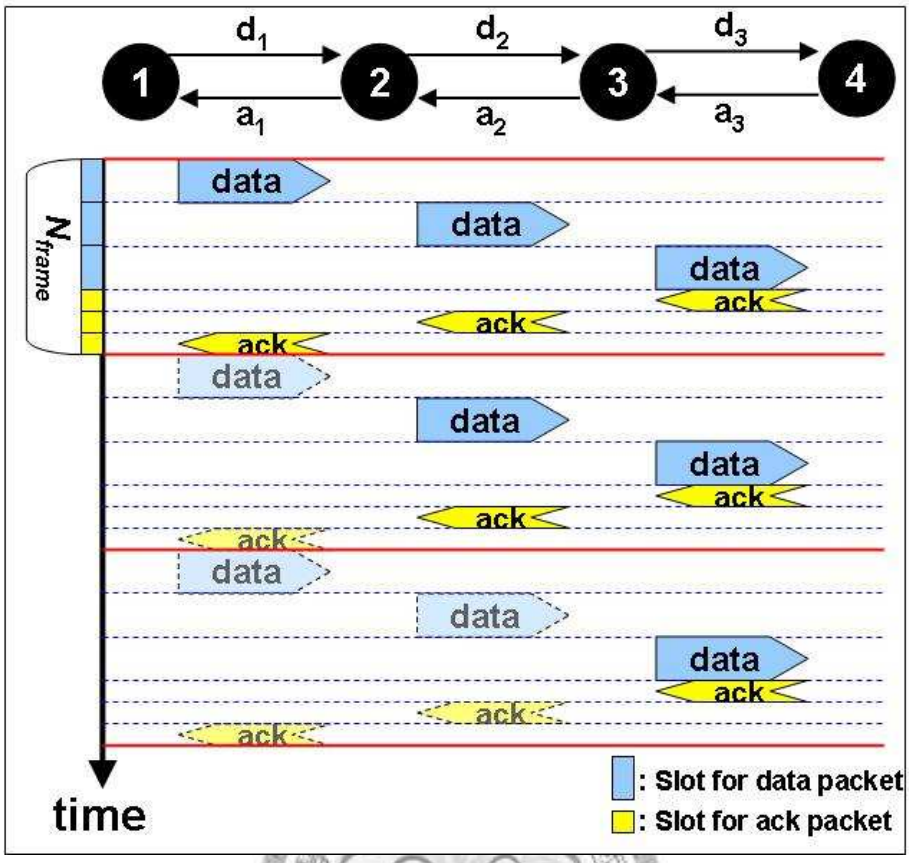


圖 3-8 Transmission cycle time of schedule A

圖 3-8 中列出了 schedule A 的 3 個 frame。由於有 3 條 TCP flow 經過 bottleneck link (link d_3 及 link a_3)，至少需要 3 個 frames 才能讓所有 TCP flows 有機會完成一輪的傳輸 (對一條 TCP flow 來說，完成一輪傳輸指 TCP 傳送端傳送一個 TCP data packet 到 TCP 接收端且 TCP 接收端對此 data packet 回覆一個 ACK packet 到 TCP 傳送端)。我們定義 N_{u_data} 爲傳送一個 TCP data packet 所需的 timeslot 數，N_{u_ack} 爲傳送一個 TCP ACK packet 所需的 timeslot 數，Schedule A 的 3 個 frame 所需的時間爲 $9N_{u\_data}+9N_{u\_ack}$。圖上所繪的虛線 timeslot，雖然可以讓 node 1 和 node 2 傳送 data(或 ACK) packet，但受限於 bottleneck link 的 timeslot 數目，node 1 和 node 2 也只是暫時把 packet 送至 node 3 的 buffer 裡暫存。如果能依照每個 wireless link 的 loading 分配 timeslot，可以在更短的時間內完成一輪的傳輸以提升 throughput。

Schedule B 的演算法跟 schedule A 不同之處在於設定 contention graph 的權重以及分 scheduling group 的演算法。首先是要知道 network 中每個 wireless link 的

TCP traffic 需求數量以決定權重，因此我們以 TCP traffic matrix 來表示 network 中的 TCP traffic 需求，如圖 3-9：

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \dots \\ N-1 \\ N \\ \\ \end{array}\begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & \dots & (N-1) & N \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ \dots & \dots & \dots & 0 & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \begin{array}{cccccc} d_1 & d_2 & d_3 & \dots & d_{N-1} & d_N \end{array} \\ \begin{array}{cccccc} a_1 & a_2 & a_3 & \dots & a_{N-1} & a_N \end{array} \end{array}$$

圖 3-9 TCP traffic matrix

在 TCP traffic matrix 中，橫列代表此 TCP flow 的傳送端(起始點)，直行代表此 TCP flow 經過的 node，因此位處(1,2)的 1 代表有 1 條 TCP flow 由 node 1 發出經過 node 2，反之若爲 0 則代表沒有 TCP flow 經過此 node。計算每個 wireless link 的 load 時，要把此 TCP flow matrix 的關係對應到 wireless link 上，例如第 0 行只有(1,2)爲 1，意即只有 1 條傳送端爲 node 1 的 TCP flow 經過 d_1，因此 link d_1 的 load 爲 1，$F(d_1)=1$，在 traffic matrix 上即是把 link d_1 相對應的直行的數字相加。因爲只有一條 TCP flow 經過 d_1，所以 a_1 也只有一條 TCP flow 的 ACK flow 經過，所以 $F(a_1) = 1$。以 four-node chain topology 爲例，經過計算 TCP flow matrix 上的 load 之後，分配 contention graph 權重如圖 3-10：

$$w'(d_1) = TD(d_1) \times \sum_{\forall f_i \in D_1} \frac{L_{data,f_i}}{L_{ack,f_i}}, \quad w'(a_1) = TD(d_1) \times \sum_{\forall f_i \in D_1} \frac{L_{ack,f_i}}{L_{ack,f_i}} = F(d_1),$$

$$w'(d_2) = TD(d_2) \times \sum_{\forall f_i \in D_2} \frac{L_{data,f_i}}{L_{ack,f_i}}, \quad w'(a_2) = TD(d_2) \times \sum_{\forall f_i \in D_2} \frac{L_{ack,f_i}}{L_{ack,f_i}} = F(d_2),$$

$$w'(d_3) = TD(d_3) \times \sum_{\forall f_i \in D_3} \frac{L_{data,f_i}}{L_{ack,f_i}}, \quad w'(a_3) = TD(d_3) \times \sum_{\forall f_i \in D_3} \frac{L_{ack,f_i}}{L_{ack,f_i}} = F(d_3)$$

圖 3-10 Weight assignment of Schedule B

爲了和 Schedule A 中的權重有所區別，schedule B 的權重以 *w'*表示。Schedule

B 演算法中按照 *w'*分配 scheduling group 的 pseudo code 如圖 3-11：

```
/* step.1 partition vertices of G' into non-conflictin g scheduling groups S_g */
U = the set of vertices of weighted contention graph G';
g = 0; /* initial group number index */
w''(v)= w'(v) ∀v ∈ U;
TD'(v) = TD(v) ∀v ∈ U;
while U ≠ φ {
    g++;
    select the vertex v with the largest weight from U;
    w''(v) = [ (TD'(v) - 1) / TD'(v) ] × w''(v);
    TD'(v) = TD'(v) - 1;
    if (w''(v) == 0) {
        U = U - {v};
        S_g = S_g + {v};
    }
    W = U - {v};
    while W ≠ φ {
        if vertex with the largest weight w in W does not conflict with any vertex in S_g {
            w''(w) = [ (TD(w) - 1) / TD(w) ] × w''(w);
            TD'(v) = TD'(v) - 1;
            if (w''(w) == 0) {
                S_g = S_g + {w};
                U = U - {w};
            }
        W = W - {w};
    }
    w(S_g) = max{ w'(u)/TD(u) | u ∈ S_g }; /* compute the maximum timeslots for each scheduling group */
}
```

圖 3-11 Schedule B 演算法第一階段

由於 wireless link *v* 的權重已經乘上經過 *v* 的 TCP flow 數量，因此在分配 scheduling group 時，我們希望從 vertex set 中取 vertex 出來放到 scheduling group 的次數能按照 load 的數量取出，如此每個 scheduling group 的 timeslot 長度剛好能容納一個 TCP data packet 或 TCP ACK packet。分配 timeslot 的演算法則和 schedule A 的部份相同，如前面圖 3-5。

使用 schedule B 演算法對 four-node chain 和 seven-node chain topology 排出的

schedule 如圖 3-12 和圖 3-13：

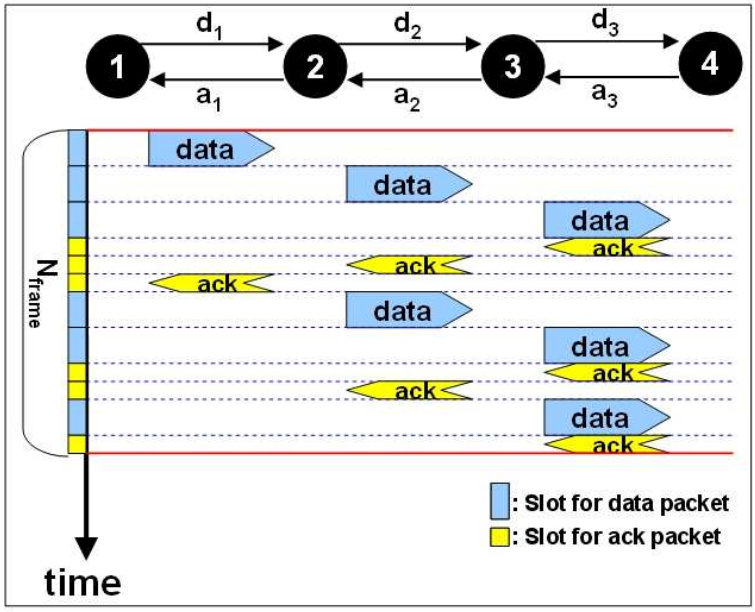


圖 3-12 Schedule B of four-node topology

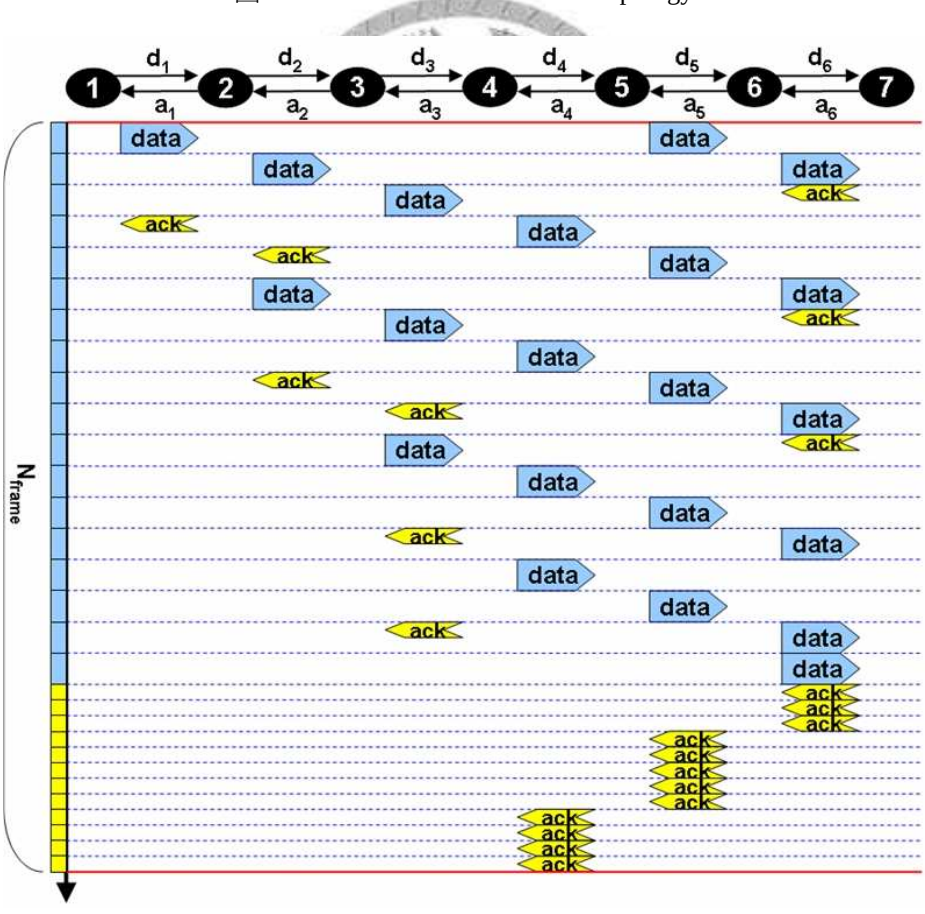


圖 3-13 Schedule B of seven-node topology

3.2.3. 討論

我們將 schedule A 和 schedule B 的比較總結如表 3-1：

表 3-1 schedule A 和 schedule B 的比較

	Cycle time(所有 flow 完成 1 個 round-trip 傳輸所需的時間)		可能的 bottleneck
	Four-node chain	Seven-node chain	
Schedule A	$9N_{u\_data}+9N_{u\_ack}$	$24N_{u\_data}+24N_{u\_ack}$	最終所有 TCP flow 的傳輸速度會被 bottleneck link 限制住, 因此分配給其它 link 的 timeslots 可能造成浪費
Schedule B	$6N_{u\_data}+6N_{u\_ack}$	$18N_{u\_data}+12N_{u\_ack}$	

Schedule A 平均的將 timeslots 分配至所有 wireless links 上，以 round-robin 的方式服務每條 TCP flow，可能的缺點在於所有 TCP flow 會被 bottleneck link 限制住，而分配給其它 wireless link 的 timeslots 可能沒有被有效利用，使得 link utilization 過低。Schedule B 讓每個 wireless link 所分配到的 timeslots 數量和 traffic demand 成正比，而且可以在較短的時間內讓所有 TCP flow 完成一個 round-trip 的傳輸，使 TCP throughput 提升。

3.3. 實驗結果

我們在 QualNet simulator 4.0 上進行模擬實驗分析比較 TCP 在兩種 TDMA scheduling scheme – schedule A 和 schedule B 的效能。實驗環境包含 four-node 及 seven-node chain topology 的 wireless mesh networks，其中除了 gateway node 之外的每個 wireless node 都與 gateway 建立一條 TCP flow，如圖 3-14 和圖 3-15。

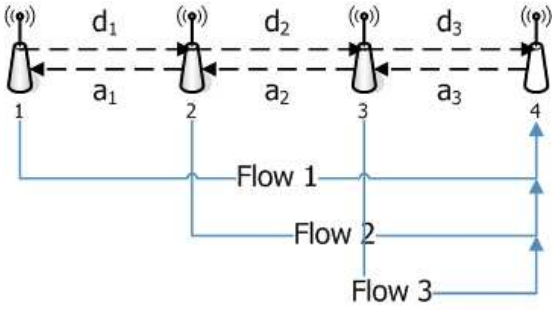


圖 3-14 four-node topology

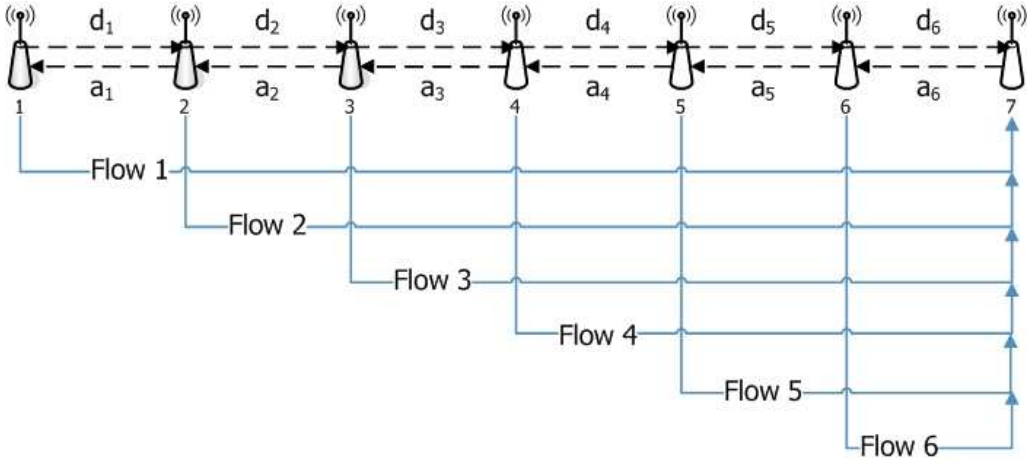


圖 3-15 seven-node topology

我們實驗在理想的 wireless 環境中進行(沒有 wireless link error 造成 packet loss)，physical layer protocol 使用 IEEE 802.11b，data rate 固定在 11Mbps，TCP segment size 爲 1460 bytes，每組實驗情境都進行 150 秒，FTP 傳輸從第 5 秒開始至第 145 秒結束。我們觀察的效能標的主要有 overall 的 TCP throughput 及各條 TCP flow 的 individual throughput、throughput 的 fairness index[13] 、TCP congestion window 隨時間變化的情形、以及 smoothed round-trip time。我們觀察這些效能標的以瞭解 TCP 在使用 spatial TDMA scheduling 的 wireless mesh network 上是否能保有第一章描述 TCP 在有線網路上的良好特性。

3.3.1. Throughput Performance

Four-node chain topology 的 overall TCP throughput 如圖 3-16，individual TCP

throughput 如圖 3-17。

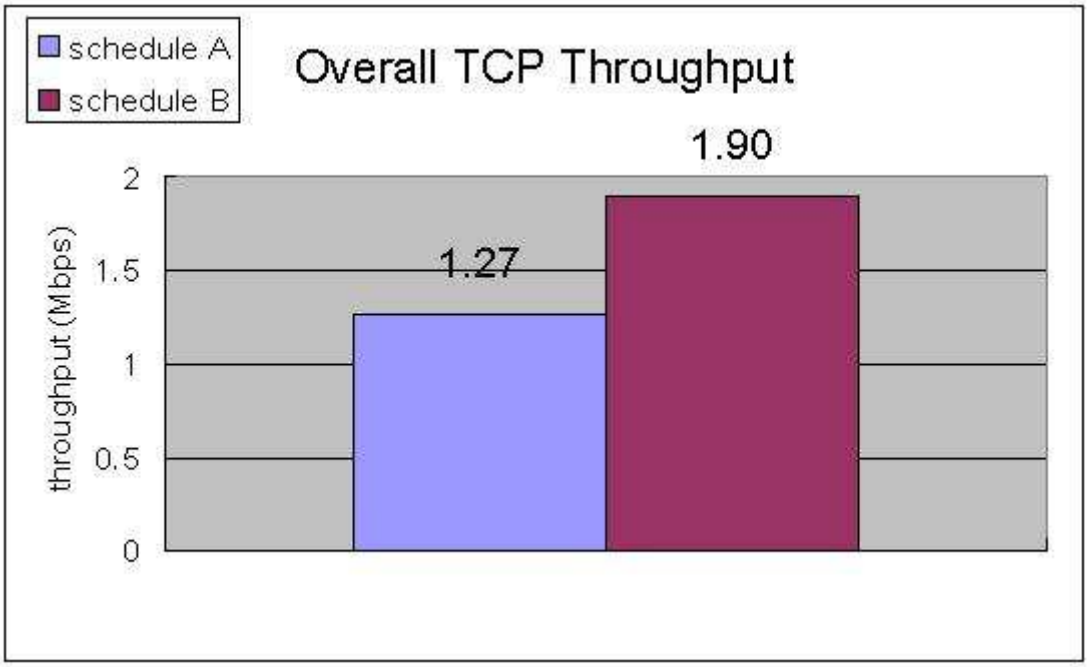


圖 3-16 overall TCP throughput of four-node topology

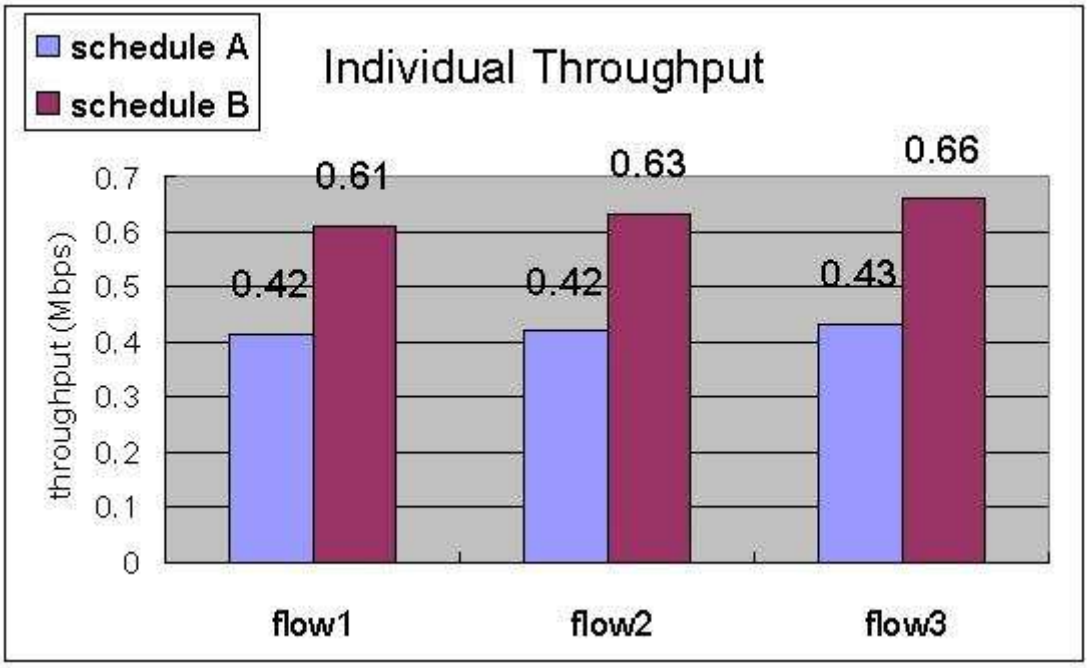


圖 3-17 individual TCP throughput of four-node topology

Schedule B 的 overall throughput 和其中各 TCP flow 的 individual throughput 都比 Schedule A 高 50%。在 Four-node chain topology 中 Schedule B 中只要花費 $6N_{u\_data}+6N_{u\_ack}$ 的時間就可以讓所有 TCP flow 完成 1 個 round-trip 的傳輸，但在 Schedule A 中需要 $9N_{u\_data}+9N_{u\_ack}$ 的時間，爲 schedule B 所費時間的 1.5 倍，因此 schedule B 有約 50%的 throughput gain，符合我們的預期。

圖 3-18 和圖 3-19 分別爲 seven-node chain topology 的 overall throughput 和 individual throughput。

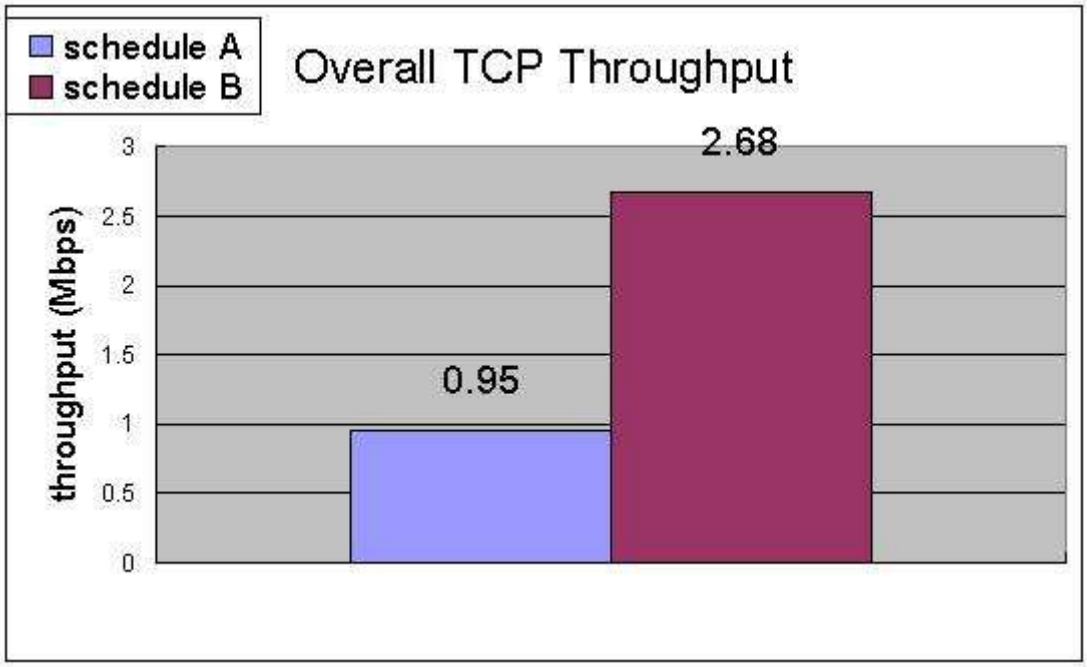


圖 3-18 overall TCP throughput of seven-node topology

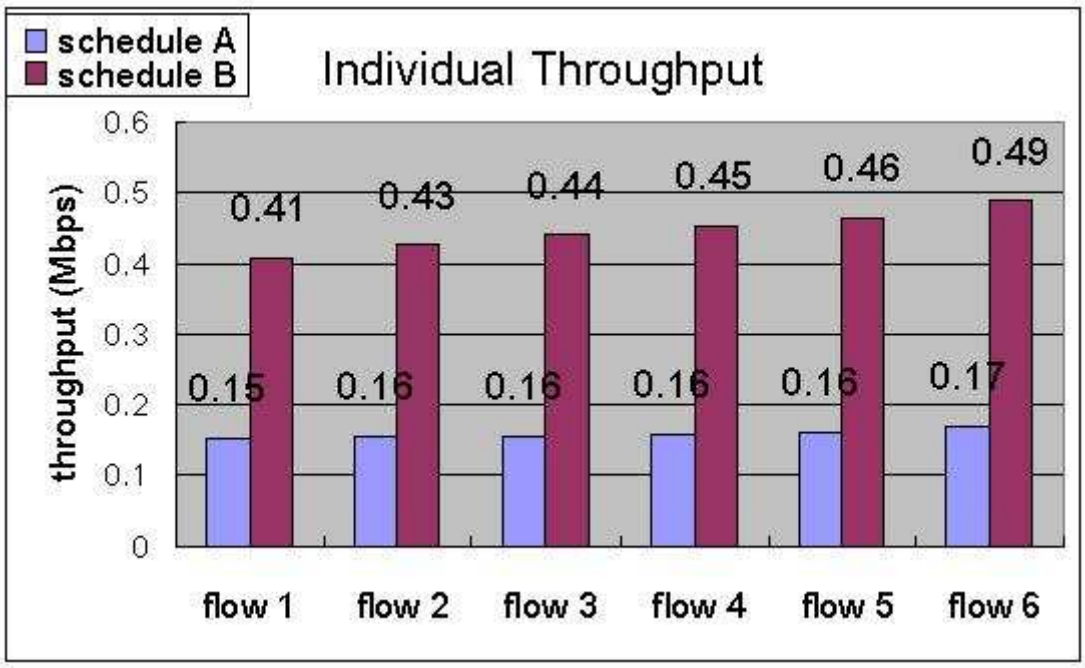


圖 3-19 individual TCP throughput of seven-node topology

在 Seven-node 的情境中，schedule B 的 throughput 約爲 schedule A 的 280%。Schedule A 讓所有 flows 完成一個 round-trip 傳輸需時 $24N_{u\_data}+24N_{u\_ack}$， schedule B 爲 $18N_{u\_data}+12N_{u\_ack}$，在模擬環境設定中，1 個 N_{u_data} 的傳輸時間爲 1.298 ms，1 個 N_{u_ack} 的傳輸時間爲 0.236 ms，因此 schedule A 的時間約爲 schedule B 的 1.41

倍，但實驗結果 schedule B 對 schedule A 的 throughput gain 多達 180%，比我們預期的還要高。在 seven-node chain topology 中，TDMA scheduling 可以充份利用 spatial reuse 的特性，將不互相干擾的 wireless link 安排在同一時間進行傳送。依照 traffic demand 分配 timeslots 的 Schedule B 在可以 spatial reuse 的環境中表現更突出。

衡量 Individual throughput 的 fairness 程度，我們可以使用 Jain's fairness index[13] ：

$$\text{fairness index} = \frac{\left(\sum x_i\right)^2}{\left(n \cdot \sum x_i^2\right)} \qquad \text{Eq. 7}$$

x_i 爲 flow i 所得的 throughput，n 爲 flows 的數量。Fairness index 的值介於 1/n 到 1 之間，愈接近 1 代表 throughput 分配愈均勻愈公平，反之則愈不公平。我們將兩種 schedule 方法在兩種 chain topology 下的 throughput fairness 整理在表 3-2 中，實驗結果顯示這 2 種 TDMA scheduling 都可以得到非常好的 fairness，TCP 的 throughput fairness 特性(如 1.1.1 第二點所述)得以保持。

表 3-2 fairness index

Fairness Index		
	Four-node chain topology	Seven-node chain topology
Schedule A	0.9996	0.9988
Schedule B	0.9990	0.9965

3.3.2. TCP Congestion Window 變化情形

我們將實驗中觀察到的 TCP congestion window 隨時間變化畫成圖，圖 3-20 和圖 3-21 分別爲 four-node chain topology 中 schedule A 和 schedule B 的 TCP congestion window 變化情形，圖 3-22 和圖 3-23 爲 seven-node chain topology 的 TCP congestion window 變化情形。

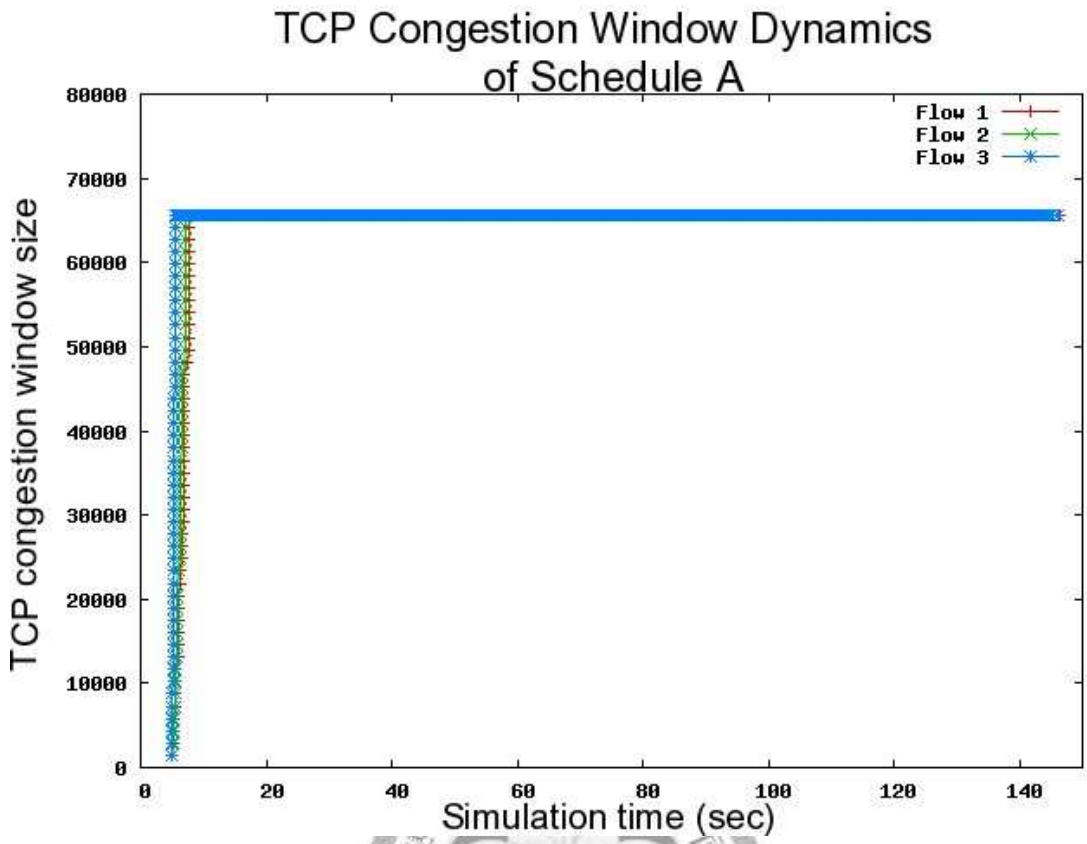


圖 3-20 TCP congestion window 變化情形(schedule A, four-node)

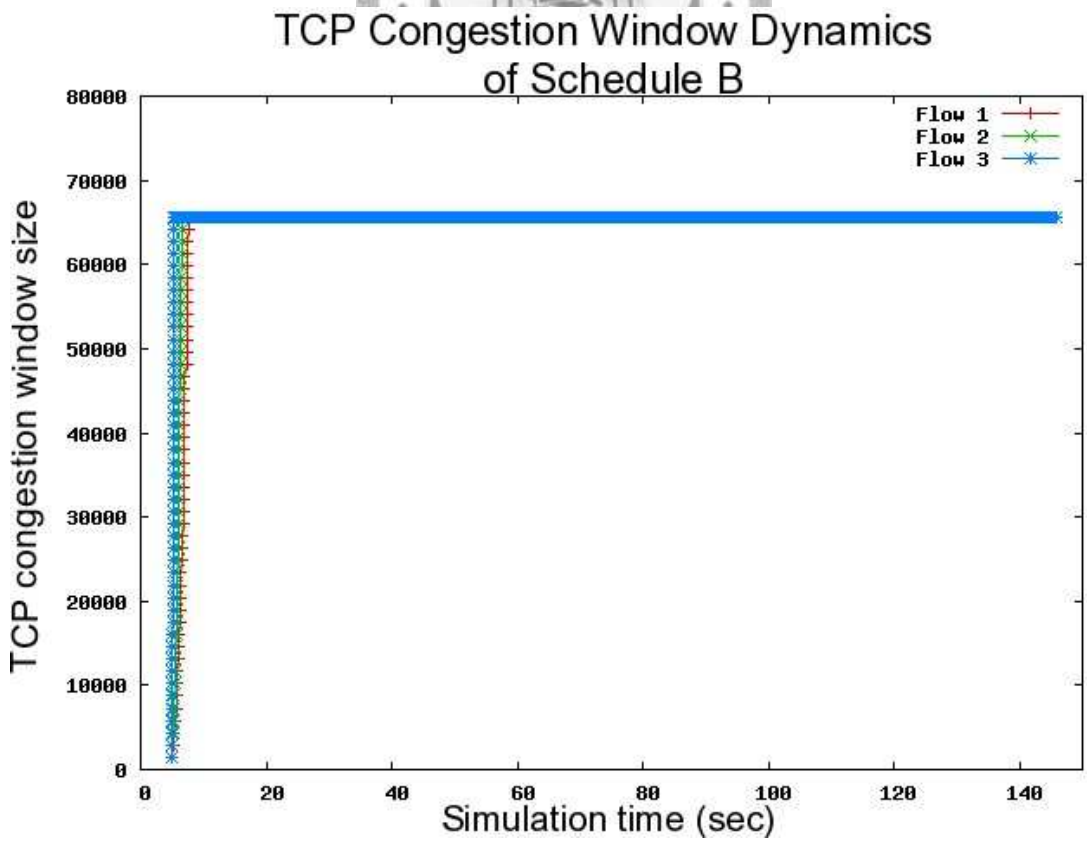


圖 3-21 TCP congestion window 變化情形(schedule B, four-node)

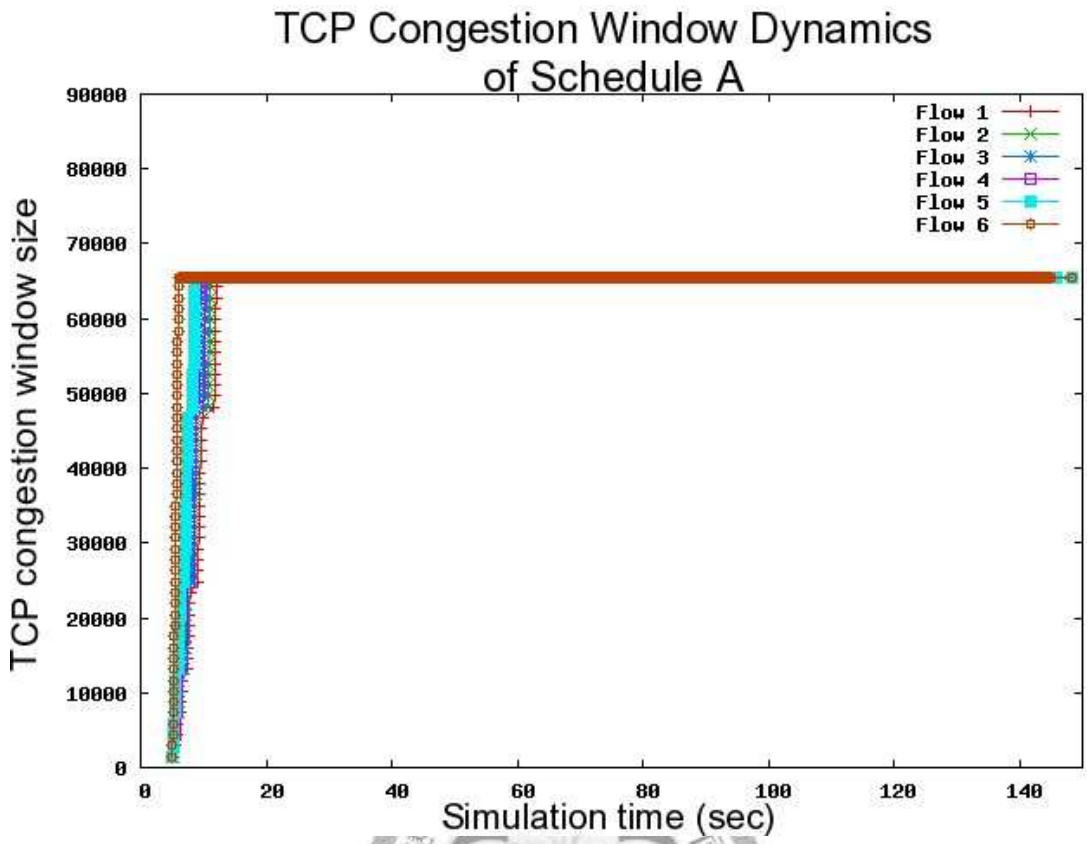


圖 3-22 TCP congestion window 變化情形(schedule A, seven-node)

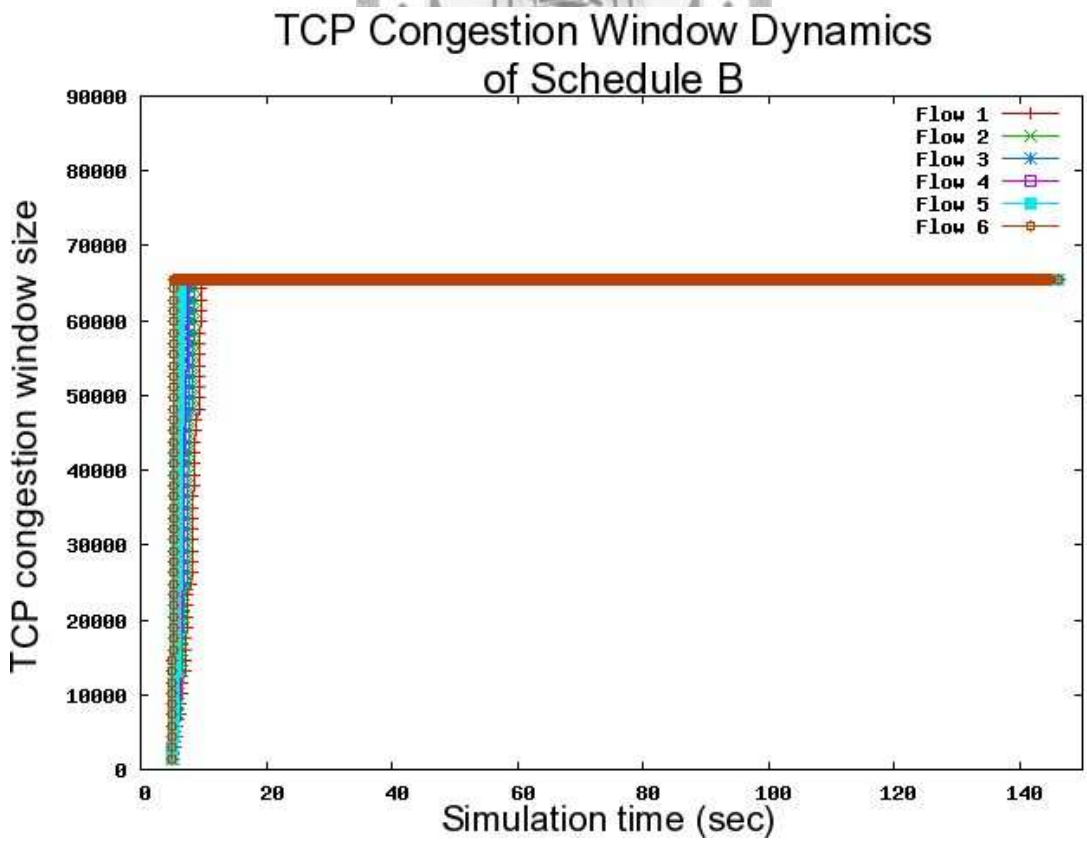


圖 3-23 TCP congestion window 變化情形(schedule B, seven-node)

由以上四張圖可以看出，無論是 schedule A 或 schedule B，所有 TCP flow 的 congestion window 都可以很快的成長到最大值 65535 Bytes 並且維持在此處直到模擬結束，也就是所有 TCP flow 都可以很快地到達"steady state"。但 long-hop flow (如 four-node 的 flow 1 和 seven-node 的 flow 1, flow 2…等等)需要比 short-hop flow 晚一些才到達 steady state。

3.3.3. TCP smoothed round-trip time

提出 schedule B 的理由之一是因爲 schedule B 可以在比較短的時間內讓所有 TCP flow 完成一個 round-trip 的傳輸，因此，我們預期 schedule B 所有 TCP flows 的 round-trip time 應該要小於 schedule A。表 3-3 列出 four-node chain topology 中，模擬結束時所有 TCP flow 的 smoothed round-trip time，表 3-4 爲 seven-node chain topology 的 smoothed round-trip time。

表 3-3 srtt 比較(four-node)

Smoothed round-trip time – four-node chain topology			
	Flow 1	Flow 2	Flow 3
Schedule A	1.25 sec	1.22 sec	1.28 sec
Schedule B	0.88 sec	0.84 sec	0.84 sec

表 3-4 srtt 比較(seven-node)

Smoothed round-trip time – seven-node chain topology						
	Flow 1	Flow 2	Flow 3	Flow 4	Flow 5	Flow 6
Schedule A	3.25 sec	3.06 sec	3.19 sec	3.28 sec	3.19 sec	3.23 sec
Schedule B	1.30 sec	1.23 sec	1.20 sec	1.19 sec	1.17 sec	1.14 sec

在 four-node chain topology 中，Schedule A 的 smoothed round-trip time 平均比 schedule B 多 50%。在長度更長的 seven-node chain topology，schedule A 的 smoothed round-trip time 平均比 schedule B 多 165%。一如我們的預期，使用 schedule B 的 TCP flow 可以更快地完成 round-trip 的傳輸，減少 round-trip delay。我們進一步的觀察 round-trip time 的組成。我們可以計算 queueing delay - srtt ratio

如下：

$$\text{queueing delay - srtt ratio} = \frac{\text{packet average time in queue}}{\text{srtt}} \qquad \text{Eq. 8}$$

表 3-5 和表 3-6 列出 four-node chain topology 和 seven-node chain topology 各個 TCP flows 的 queueing delay – srtt ratio。所有的 TCP flows 的 queueing delay – srtt ratio 都超過 86%以上，代表 queueing delay 佔 srtt 86%以上的時間，意即 TCP data 封包在無線網狀網路上傳送時，86%的時間都是排在 queue 裡面等待。

表 3-5 queueing delay-srtt 比值(four-node)

Queueing delay - srtt ratio – four-node chain topology			
	Flow 1	Flow 2	Flow 3
Schedule A	93.56%	93.18%	96.16%
Schedule B	90.93%	95.75%	91.00%

表 3-6 queueing delay-srtt 比值(seven-node)

Queueing delay - srtt ratio – seven-node chain topology						
	Flow 1	Flow 2	Flow 3	Flow 4	Flow 5	Flow 6
Schedule A	96.33%	96.13%	96.52%	95.53%	96.17%	95.04%
Schedule B	86.90%	88.89%	90.75%	90.47%	91.16%	94.33%

3.3.4. Link utilization

我們認爲 schedule A 有其缺點，由於 schedule A 將 timeslots 平均分配給所有 wireless link，但在其上運行的 TCP 效能可能會被 bottleneck link 的 capacity 限制住。離 gateway 較遠的 node 雖然可以很快地送出封包，但也終究會因靠近 gateway (所有 TCP flows 匯集之處)的 bottleneck link capacity 不夠而將這些封包暫存在 intermediate node，而使得 TCP flows 不能夠順暢的流動。由於 TCP 是 feed-back based 的 protocol，會自動探測可用的 bandwidth 並調整傳送端的 sending rate，在 schedule A 中最終的結果可能導致離 gateway 較遠的 wireless link 沒有被有效利用。

我們可以觀察各個 wireless link 的 utilization 證明這點。表 3-7 和表 3-8 將

four-node chain topology 和 seven-node chain topology 的 link utilization 列出。編號較小的是離 gateway 較遠的 wireless link，traffic demand 比較少，在 schedule A 中，數據顯示這些 link 的 utilization 都相當低。但在 schedule B 中，timeslots 的分配是按照 traffic demand 輕重程度來排，因此所有 link 都可以保持很高的 utilization。實驗證明 Schedule B 保存 TCP 的良好特性之一：讓 TCP flows 盡可能地使用所有的可用頻寬。

表 3-7 link utilization (four-node)

Link utilization – four-node chain topology			
	Link d_1	Link d_2	Link d_3
Schedule A	32.62%	65.74%	99.98%
Schedule B	96.08%	97.92%	99.92%

表 3-8 link utilization (seven-node)

Link utilization – seven-node chain topology						
	Link d_1	Link d_2	Link d_3	Link d_4	Link d_5	Link d_6
Schedule A	16.01%	32.15%	48.29%	64.78%	81.30%	99.02%
Schedule B	88.62%	90.60%	92.51%	94.28%	95.49%	97.39%

3.4. 討論

我們在這一章討論使 TCP 順暢流動的 k=1 model，並依此提出兩種 TDMA scheduling scheme：一個是平均分配 timeslots 給所有 wireless link 的 schedule A，另外一個是按照 traffic demand 分配 timeslots 的 schedule B。實驗結果的各項數據整理在下面表 3-9 和表 3-10 中。

表 3-9 各項效能比較(four-node)

Four-node topology						
	Cycle time	Overall throughput	Fairness index	Bottleneck	Link utilization	
Schedule A	$9N_{u\_data}+9N_{u\_ack}$	1.27 Mbps	0.9996	最終所有 flow 的 data rate 會被 link d3 跟 a3 限制住, 因此分配給 d1,d2,a1,a2 的 slots 可能造成浪費	Link d_1	32.62%
					Link d_2	65.74%
					Link d_3	99.98%
Schedule B	$6N_{u\_data}+6N_{u\_ack}$	1.90 Mbps	0.9990		Link d_1	96.08%
					Link d_2	97.92%
					Link d_3	99.92%

實驗數據證明根據 traffic demand 分配 timeslots 的 schedule B 比 schedule A 提供 TCP 更好的 overall TCP throughput，主要原因是在於 schedule A 分配一樣多的頻寬給每個 hop，使得 cycle time 變長，per-link 可以獲得的頻寬變少，per-flow 可以使用的頻寬變少，所以影響到 overall throughput。schedule B 根據 per-hop 的 traffic demand 讓 end-to-end 的傳輸一輪所需的 cycle time 比 schedule A 小，這點可以從 3.3.3 的 TCP flows 的 round-trip time 數據上看出，schedule B 的所有 TCP flows 的 round-trip time 都比 schedule A 的 round-trip time 小。另外，Schedule A 的 timeslots 不考慮 per-hop 的 traffic demand，因此讓 TCP flow 卡在 bottleneck link，這樣會導致其它 wireless link 的 utilization 過低而造成 timeslots 的浪費，schedule B 改善了這個情況，這點在 3.3.4 的數據中可以觀察到。

表 3-10 各項效能比較(seven-node)

Seven-node topology						
	Cycle time	Overall throughput	Fairness index	Bottleneck	Link utilization	
Schedule A	$24N_{u\_data}+24N_{u\_ack}$	0.95 Mbps	0.9988	最終所有 flow 的 data rate 會被 link d6 跟 a6 限制住, 因此分配給其它 link 的 slots 可能造成浪費	Link d_1	16.01%
					Link d_2	32.15%
					Link d_3	48.29%
					Link d_4	64.78%
					Link d_5	81.30%
					Link d_6	99.02%
Schedule B	$18N_{u\_data}+12N_{u\_ack}$	2.68 Mbps	0.9965		Link d_1	88.62%
					Link d_2	90.60%
					Link d_3	92.51%
					Link d_4	94.28%
					Link d_5	95.49%
					Link d_6	97.39%

在 individual throughput fairness 方面來看，兩種 scheduling 方式都可以得到很好的 fairness 結果。這點可以從 3.3.2 的 congestion window 的變化情形來說明。兩種 scheduling 的 congestion window 變化情形極爲類似，所有 TCP flow 都在很短的時間內將 congestion window 成長到並且維持在最大值，進入 steady state，因此在大部份時間內，所有 TCP flow 都是以最大的 window size 傳輸資料，這造成所有 TCP flows 送出大量的資料到網路上，讓 intermediate node 的 queue size

維持在很高的狀態，因此也使得 TCP flows 的 rtt 很大。Short-hop flow (編號較大的 flow)的 window size 成長速度稍快，因此 throughput 稍微大於 long-hop flow。但長期來說，因爲所有 TCP flow 在絕大部份時間內都以最大 window size 傳輸資料，而且在同一種 scheduling 中的 per-flow round-trip time 差異不大，因此 per-flow 的 throughput 差異不明顯，可以有很好的 throughput fairness。

總結來說，schedule B 使 TCP 有較好的效能表現，schedule B 同時保留兩項 TCP 在有線網路上的良好特性：使 TCP 盡可能的利用可用頻寬，以及保持 throughput fairness。在之後的章節中，我們將只使用 schedule B 進行效能評估。

4. Wireless link errors 引發的 packet loss 對 TCP 效能的影響

使用 TDMA-based scheduling scheme 可以避免無線網狀網路因爲 channel interference 和 channel contention 造成的 collision，我們稱此類 packet loss 爲 MAC layer packet loss，因爲這些 packet loss 是由於 MAC layer protocol 機制的本質所造成。而在現實的環境中，wireless link 非常容易發生 error 而造成 packet loss。Timeout 和 three duplicate ACK 是 TCP 判斷 packet loss 的指標。在有線網路上，輕微的 congestion 可能造成一個 window 內的一兩個封包遺失，因此有 three duplicate ACK 的狀況發生，嚴重的 congestion 可能讓一連串送出的 data 封包都被遺失(因爲 buffer 滿了)而發生 timeout，而 TCP 將所有 packet loss 都視爲網路發生 congestion 的徵兆，發生 packet loss 時，TCP congestion control 機制會調整 congestion window 甚至是 timeout value，進而影響 TCP 的 throughput。但在 wireless link 上，packet loss 是偶然發生，但這些 packet loss 不代表網路發生 congestion，此時 packet loss 引發 TCP congestion control 機制調降 window size 對 TCP throughput 可能造成傷害。本章將著重 TCP 良好特性的第二點 (參考 1.1.1)，討論因 wireless link error 而造成的 random packet loss 對使用 TDMA scheduling 的 TCP performance 將有何影響？

4.1. Packet loss at the bottleneck link

我們考慮只有在 traffic load 最重的 bottleneck link 發生 packet loss 的情境，如圖 4-1，所有 TCP flows 都會流經此處。我們定義一個參數：packet loss probability on single wireless link probability p，讓 bottleneck link 以 p 的機率隨機地遺失封包。我們觀察各種不同的 TCP flow 在遭遇 packet loss 時的效能有何變化。

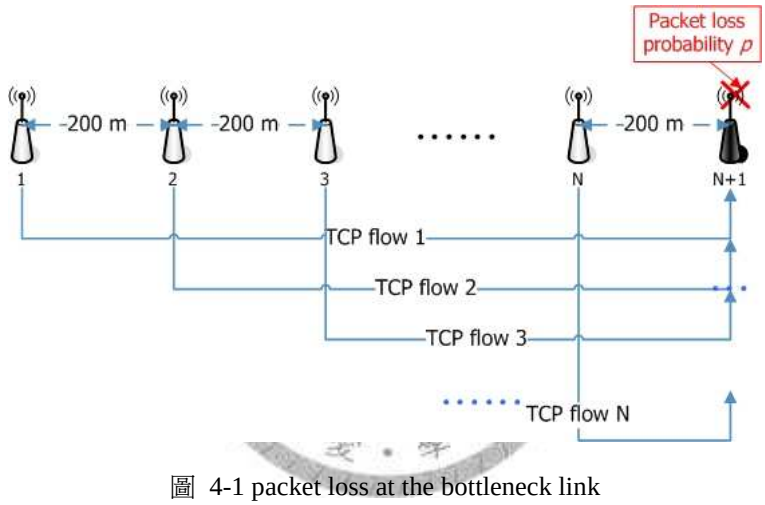


圖 4-1 packet loss at the bottleneck link

4.1.1. Throughput performance

表 4-1 和表 4-2 是沒有 packet loss (即 p=0%)和 p=5% packet loss 的 overall TCP throughput 比較。在 p=5%時，TCP 的 throughput 下降，也無法保持 throughput fairness。

表 4-1 有無 packet loss 之效能比較(four-node)

Four-node topology					
	p=0%		*p*=5% at the bottleneck link		
	Overall throughput	Fairness index	Overall throughput	Throughput degradation ratio	Fairness index
Schedule B	1.90 Mbps	0.9990	1.50 Mbps	78.93%	0.9287

表 4-2 有無 packet loss 之效能比較(seven-node)

Seven-node topology					
	p=0%		*p*=5% at the bottleneck link		
	Overall throughput	Fairness index	Overall throughput	Throughput degradation ratio	Fairness index
Schedule B	2.68 Mbps	0.9965	2.32 Mbps	86.66%	0.9099

我們觀察 per-flow 的 individual throughput，如圖 4-2 和圖 4-3。Long-hop flows 的 throughput 衰退程度比 short-hop flows 嚴重。對於 long-hop low 來說，packet loss 發生在離 TCP 傳送端最遠的 wireless link，TCP 的 end-to-end retransmission 機制讓 long-hop flow 要花比較久的時間(相對於 short-hop flow)才發現 packet loss，因此也要花比較長的時間復原。但 short-hop flow 在發生 packet loss 時，throughput 居然不降反升，這是因爲 long-hop flow throughput 下降之後空出來的可用頻寬被反應速度比較快(距離發生 packet loss 的 link 比較近)的 short-hop 使用掉。

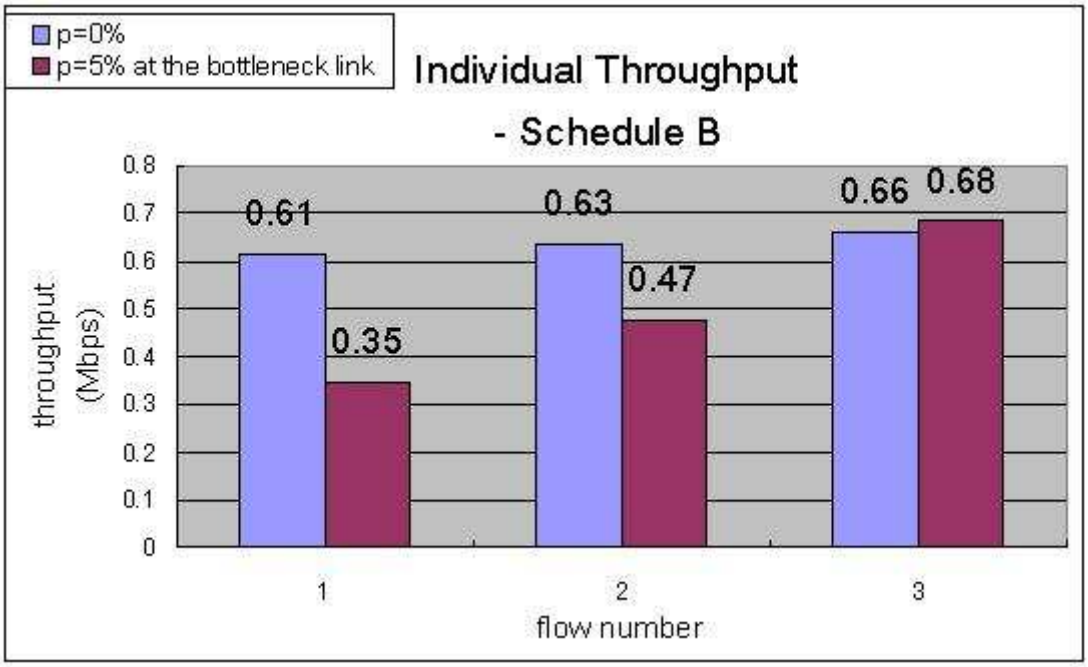


圖 4-2 individual TCP throughput under *p*=5% (four-node)

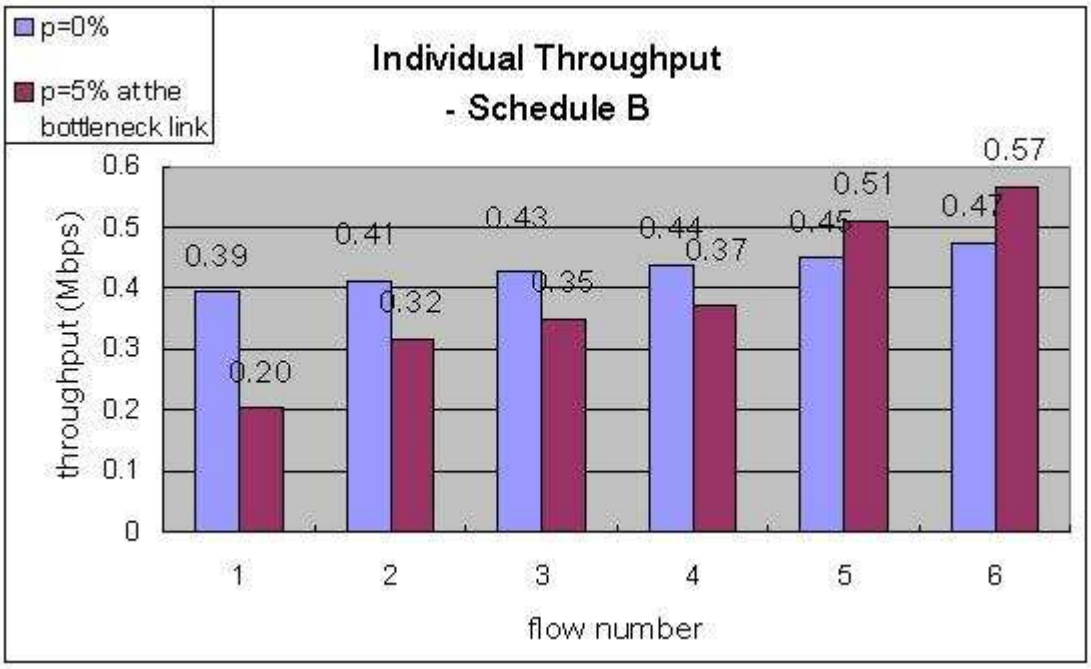


圖 4-3 individual TCP throughput under *p*=5% (seven-node)

圖 4-4 和圖 4-5 爲 flow 1 (long-hop flow)和 flow 6 (short-hop flow)的 TCP sequence number 隨時間變化的情形。兩張圖的 x 軸時間尺度是一樣的，在圖上連續 sequence number 中間的斷層表示 TCP 遭遇 packet loss 發生 timeout，斷層越大表示反應和恢復速度越慢。flow 1 (圖 4-4)上的斷層比起 flow 6 (圖 4-5)的斷層大，說明 long-hop flow 的確需要花較多的時間復原 lost packet。

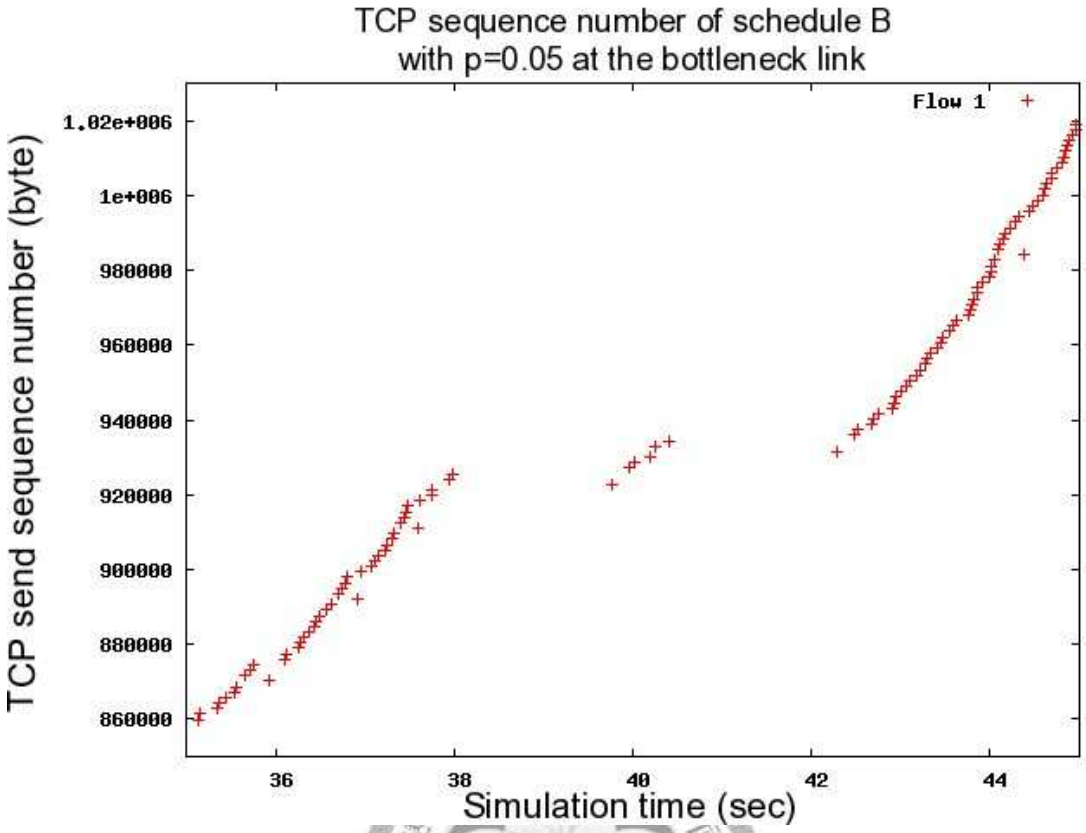


圖 4-4 TCP sequence number of flow 1 under *p*=5% (seven-node)

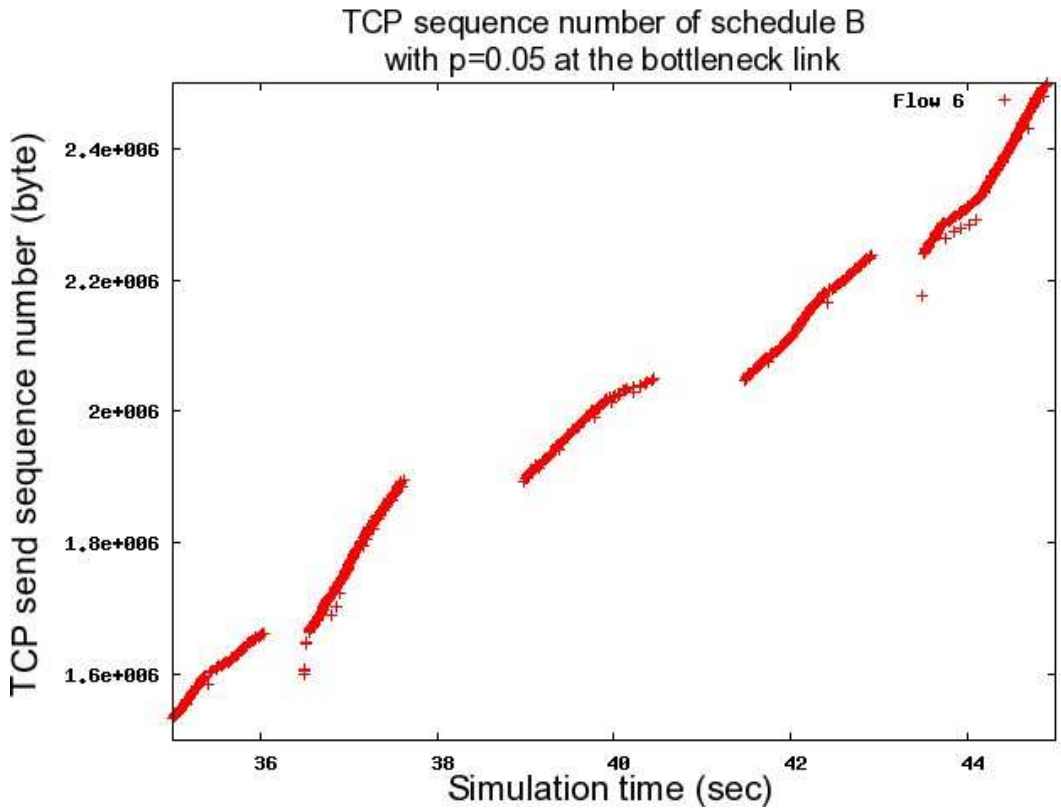


圖 4-5 TCP sequence number of flow6 under *p*=5% (seven-node)

4.1.2. TCP congestion window 變化情形

我們將 p=5%的TCP congestion window隨時間變化的情形整理在圖 4-6和圖 4-7。

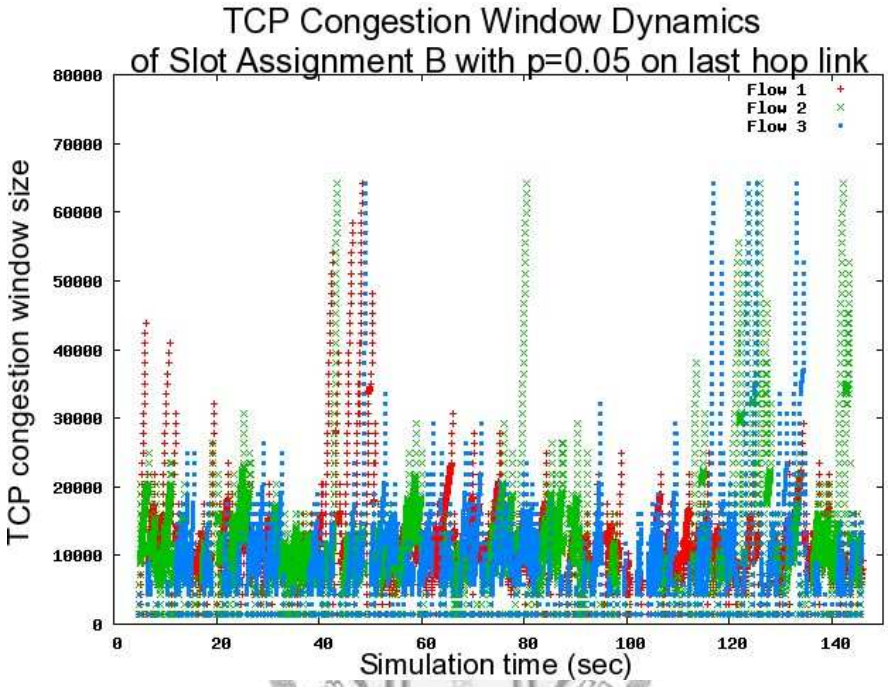


圖 4-6 p=5%時 TCP congestion window 變化情形(four-node)

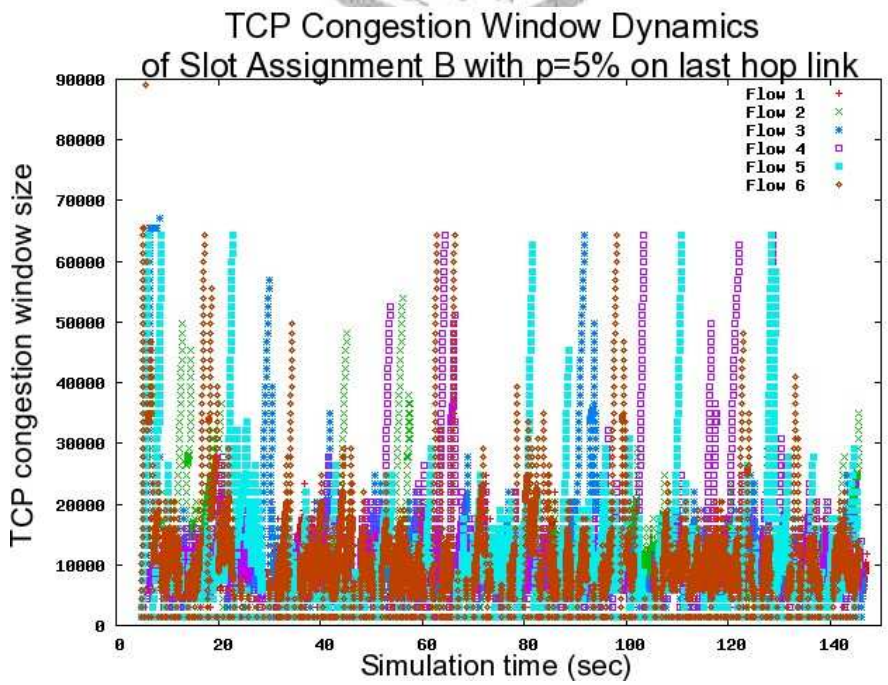


圖 4-7 p=5%時 TCP congestion window 變化情形(seven-node)

我們比對 3.3.2 之中 *p*=0%的數據(圖 3-21 和圖 3-23)發現，*p*=5%時，TCP 的 congestion window 不斷地上下震盪，因爲 TCP 的 congestion control 演算法在發生 packet loss 時會調降 congestion window size，但隨著及時收到的 TCP ACK 又會逐漸調升其 congestion window size。由此可知 TCP 將 wireless link error 所造成的 packet loss 當成是因 network congested 造成的 packet loss，因此而不恰當地調降 congestion window size 使 throughput 下降。實際上這些 packet loss 是隨機地發生，調降 congestion window 並沒有辦法避免 packet loss 的發生。

4.1.3. TCP smoothed round-trip time

表 4-3 表 4-4 和列出在 *p*=5%時 TCP 模擬結束時的 smoothed round-trip time。跟表 3-3 表 3-4 和比較起來，*p*=5%時的 round-trip time 都小很多。因爲 packet loss 導致 TCP 時時調降的 congestion window 的緣故，TCP 沒辦法持續以最大的 window size 送出資料，因此 data 並不像 *p*=0%時大量排在 queue 中而增加 queueing delay。

表 4-3 srtt under *p*=5% (four-node)

Smoothed round-trip time – four-node chain topology			
	Flow 1	Flow 2	Flow 3
Schedule B	0.2031 sec	0.1250 sec	0.1094 sec

表 4-4 srtt under *p*=5% (seven-node)

Smoothed round-trip time – seven-node chain topology						
	Flow 1	Flow 2	Flow 3	Flow 4	Flow 5	Flow 6
Schedule B	0.2188 sec	0.2500 sec	0.1406 sec	0.1875 sec	0.1875 sec	0.1563 sec

4.2. 討論

在 Wireless bottleneck link error 造成的 random packet loss，會使得 wireless mesh network 上 path hop 長度不同的 TCP flow 有不同程度的 throughput 衰退，long-hop flow 的 throughput 衰退程度遠比 short-hop flow 嚴重，因此使 throughput fairness 下降。這是因爲在實驗的環境中，僅依靠 TCP end-to-end retransmission 機制處理 random packet loss，而 long-hop flow 需要花較長的時間才能 recover，所以 long-hop flow 的 throughput 衰退得比較嚴重。另外，TCP congestion control 機制受到 random packet loss 的影響，使得 congestion window 不斷上下震盪無法保持穩定，對 throughput 也是一項傷害。

我們在第五章將使用 link-layer retransmission scheme 解決 random packet loss 造成的 TCP 效能衰減。

5. Link-layer 重傳機制及 TCP 效能分析

針對 bottleneck link 發生 packet loss 而造成 TCP 效能下降問題，我們採用 link-layer retransmission scheme 作爲解決方法。在文獻上，有許多文章[14] [15] [16] 討論如何在容易發生 error 的 wireless link 上提升 TCP throughput，不過都是考慮在 single-hop wireless network 的環境。Eckhardt 等人[14] 認爲處理 wireless link errors 的理想方法應該是用 local error control，而不應該讓 TCP 的 end-to-end error control 機制或是其他 end-to-end 的方式處理。End-to-end 的方法有其缺點，end-to-end retransmission 在沿途經過的每段 wireless link 都要花費時間和消耗網路資源。若使用 link-layer local error control，可以迅速地處理 packet loss，而且可以達到 transparent 的效果(上層 protocol 不需要進行修改，而且也不容易發覺 packet loss 的發生)，因此佈署 link-layer local error control 比佈署 end-to-end error control 機制容易。

在這一章，我們將加入一個 link-layer retransmission scheme 以幫助 TCP 在 error-prone 的 wireless mesh network 上提升 throughput，並進行模擬實驗觀察並討論其效能。

5.1. Link-layer retransmission scheme

我們使用一個簡單的 link-layer retransmission scheme，運作方式如圖 5-1，每個 wireless node 若收到一個封包，則回覆傳送者一個 link-layer ACK 封包。TCP data 和 TCP ACK 對 link-layer 來說都是一個 link-layer data 封包，因此都要回覆一個 link-layer ACK 以確認封包是否正確被接收。送出 link-layer data 封包之後，

若在下一個 receiving timeslot 沒有收到 link-layer ACK，傳送者就認爲此封包遺失，會立即在下一個 sending timeslot 安排重傳此 link-layer data 封包。不使用 timer 計時以判斷是否 timeout 的原因是由於 TDMA timeslot 的時間非常規律，timeslot index 本身就可以當 timer 作計時使用。如同 IEEE 802.11，我們也設定一個重傳上限次數的參數，重傳次數上限設爲 7 次，若重傳 7 次還沒有成功送達，此封包就會被丟棄。

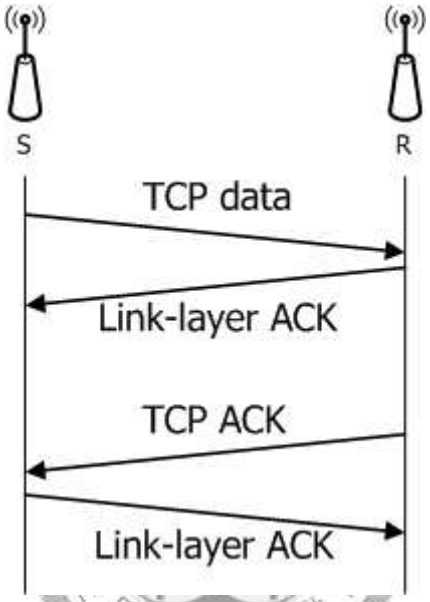


圖 5-1 link-layer retransmission scheme 運作方式示意圖

5.2. 實驗結果

和 4.1 一樣，我們設定 bottleneck link 有 *p* 的機率發生 packet loss，分別在 four-node chain topology 和 seven-node chain topology 上進行實驗，比較使用 link-layer retransmission scheme 和沒有使用 link-layer retransmission scheme 的 TCP 效能表現。

5.2.1. 分析與量測 link-layer retransmission scheme 的 overhead

由於 link-layer retransmission scheme 需要傳輸 link-layer ACK 以確認封包是否送達，會額外消耗可用頻寬帶來 overhead。我們先在 *p*=0%的狀況下，比較有否使用 link-layer retransmission scheme 的 throughput 來量測 link-layer ACK 造成多少 overhead。

每個 TCP data packet 或 TCP ACK packet，對 link-layer 來說都是 link-layer data，因此傳送一個 TCP data packet 或 TCP ACK packet 額外付出的代價是一個 link-layer ACK packet 所消耗的頻寬。要分析 link-layer ACK 造成的 overhead，我們必需要知道 TCP data packet、TCP ACK packet 和 link-layer ACK packet 分別所消耗的頻寬。我們將傳送封包所花費的時間視爲消耗的頻寬，圖 5-2、圖 5-3、圖 5-4 分別爲 TCP data packet、TCP ACK packet 和 link-layer ACK packet 的結構及傳輸所耗費時間。

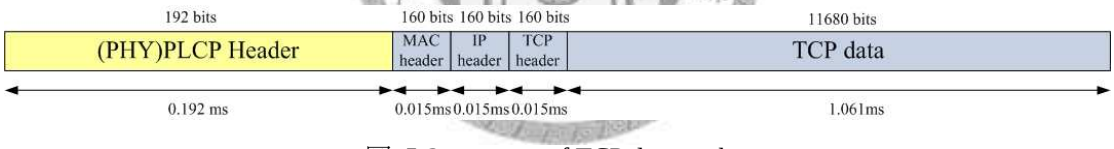


圖 5-2 structure of TCP data packet

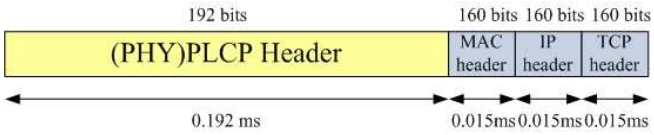


圖 5-3 structure of TCP ACK packet

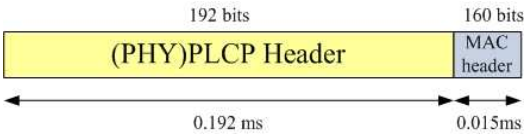


圖 5-4 structure of link-layer ACK packet

在 IEEE 802.11b 中，封包中 physical layer header 的部份是以最低速 1Mbps 傳送，後面 link-layer 以上的部份才用設定的速度(11Mbps)傳送。因此，傳送一個 payload 爲 1460Bytes 的 TCP data packet 需時：

$$\frac{192\text{bits}}{1\times10^{6}\text{Mbps}}+\frac{160\text{bits}+160\text{bits}+160\text{bits}+11680\text{bits}}{11\times10^{6}\text{Mbps}}=1.298\text{ms} \quad \text{Eq. 9}$$

傳送一個 TCP ACK packet 需時：

$$\frac{192\text{bits}}{1\times10^{6}\text{Mbps}}+\frac{160\text{bits}+160\text{bits}+160\text{bits}}{11\times10^{6}\text{Mbps}}=0.237\text{ms} \quad \text{Eq. 10}$$

傳送一個 link-layer ACK packet 需時：

$$\frac{192\text{bits}}{1\times10^{6}\text{Mbps}}+\frac{160\text{bits}+160\text{bits}+160\text{bits}+11680\text{bits}}{11\times10^{6}\text{Mbps}}=0.207\text{ms} \quad \text{Eq. 11}$$

計算出傳送各種封包所需時間之後，我們可以根據前面排出的兩種 schedule，計算出在一個 frame 裡面所傳送的 link-layer ACKs 消耗的頻寬，然後依此計算傳送 link-layer ACK 所產生的 overhead。

在 four-node chain topology 中 schedule B(參考圖 3-12)的 link-layer ACK overhead：

$$\frac{0.207\times12}{1.298\times6+0.237\times6+0.207\times12}=0.212417 \quad \text{Eq. 12}$$

在 seven-node chain topology 中 schedule B(參考圖 3-13)的 link-layer ACK overhead：

$$\frac{0.207\times30}{1.298\times18+0.237\times12+0.207\times30}=0.191560 \quad \text{Eq. 13}$$

我們將計算的結果整理在表 5-1。

表 5-1 overhead due to transmission of link-layer ACK

	Four-node	Seven-node
Schedule B	**21.24%**	**19.16%**

實驗結果如下，表 5-2 爲 four-node chain topology 的結果，表 5-3 爲 seven-node topology 的結果。

表 5-2 比較 *p*=0%時有無使用 link-layer retransmission scheme 之 TCP throughput (four-node)

		Throughput without link-layer retransmission scheme (Mbps)	Throughput with link-layer retransmission scheme (Mbps)	Throughput degradation ratio (%)	Overhead (%)
Schedule B	Flow 1	0.61	0.48	78.84	21.16
	Flow 2	0.63	0.50	78.40	21.60
	Flow 3	0.66	0.52	78.98	21.02
	Overall	**1.90**	**1.50**	**78.74**	**21.26**

模擬實驗的結果和表 5-1 分析的結果相當吻和，使用 link-layer retransmission scheme 因爲傳送 link-layer ACK 所帶來的 overhead 約爲 19%到 21%。在 *p*=0%的情況下就有 19%到 21%的 overhead，在 *p*>0 的情況下，我們預期 overhead 會更大，local retransmission 會消耗頻寬造成 throughput 下降，但我們希望 local retransmission 能幫助 TCP 避免引發不必要的 end-to-end retransmission。

表 5-3 比較 *p*=0%時有無使用 link-layer retransmission scheme 之 TCP throughput (seven-node)

		Throughput without link-layer retransmission scheme (Mbps)	Throughput with link-layer retransmission scheme (Mbps)	Throughput degradation ratio (%)	Overhead (%)
Schedule B	Flow 1	0.41	0.33	79.95	20.05
	Flow 2	0.43	0.34	79.80	20.20
	Flow 3	0.44	0.35	79.59	20.41
	Flow 4	0.46	0.36	79.88	20.12
	Flow 5	0.46	0.38	81.08	18.92
	Flow 6	0.49	0.40	82.67	17.33
	Overall	**2.68**	**2.16**	**80.55**	**19.45**

5.2.2. TCP throughput performance

表 5-4 和表 5-5 比較在 packet loss probability p>0 之下有無使用 link-layer retransmission scheme 對 throughput 和 fairness 的影響。結果顯示，在所有的情境底下，使用 link-layer retransmission scheme 的 throughput 並沒有提升，但是可以維持很好的 throughput fairness。因爲使用 link-layer retransmission scheme 多了 link-layer ACK 造成的 overhead 和進行 retransmission 造成的 overhead，可用頻寬比不使用 link-layer retransmission scheme 來得少，這樣比較有些不公平，讓兩種 throughput 結果在同一基準點上比較。

表 5-4 比較 p=5%時有無 link-layer retransmission scheme 效能(four-node)

Four-node topology					
	p=5% at the bottleneck link		p=5% at the bottleneck link with link-layer retransmission scheme		
	Overall throughput	Fairness index	Overall throughput	Throughput degradation ratio	Fairness index
Schedule B	1.50 Mbps	0.9287	1.42 Mbps	-5.18%	0.9991

表 5-5 比較 p=5%時有無 link-layer retransmission scheme 效能(seven-node)

Seven-node topology					
	p=5% at the bottleneck link		p=5% at the bottleneck link with link-layer retransmission scheme		
	Overall throughput	Fairness index	Overall throughput	Throughput degradation ratio	Fairness index
Schedule B	2.32 Mbps	0.9099	2.05 Mbps	-11.78%	0.9954

因此我們根據 5.2.1 分析的結果，將 p=5%沒有使用 link-layer retransmission scheme 的 throughput 進行 normalization。方法如下：

$$\text{normalized throughput} = \text{throughput} \times (1 - \text{overhead}) \quad \text{Eq. 14}$$

我們將 p=5%不使用 link-layer retransmission scheme 的 TCP throughput 乘上表 5-1 的 overhead，再將結果和使用 link-layer retransmission scheme 的 throughput 比較。經過 normalization 之後的結果如表 5-6 和表 5-7 所示。

表 5-6 比較 *p*=5%時有無使用 link-layer retransmission scheme 之 normalized throughput (four-node)

Four-node topology					
	p=5% at the bottleneck link (normalized)		*p*=5% at the bottleneck link with link-layer retransmission scheme		
	Overall throughput	Fairness index	Overall throughput	Throughput degradation ratio	Fairness index
Schedule B	1.18 Mbps	0.9287	1.42 Mbps	20.66%	0.9991

表 5-7 比較 *p*=5%時有無使用 link-layer retransmission scheme 之 normalized throughput (seveeen-node)

Seven-node topology					
	p=5% at the bottleneck link (normalized)		*p*=5% at the bottleneck link with link-layer retransmission scheme		
	Overall throughput	Fairness index	Overall throughput	Throughput degradation ratio	Fairness index
Schedule B	1.87 Mbps	0.9099	2.05 Mbps	9.34%	0.9954

經過 normalization 之後再比較使用 link-layer retransmission scheme 和不使用 link-layer retransmission scheme 的 throughput performance 可以發現，packet loss probability *p*=5%時使用 link-layer retransmission scheme，可以有效改善 throughput 和 fairness。我們再進一步觀察 individual throughput 的分配，如 four-node chain topology 的圖 5-5 和 seven-node topology 的圖 5-6。

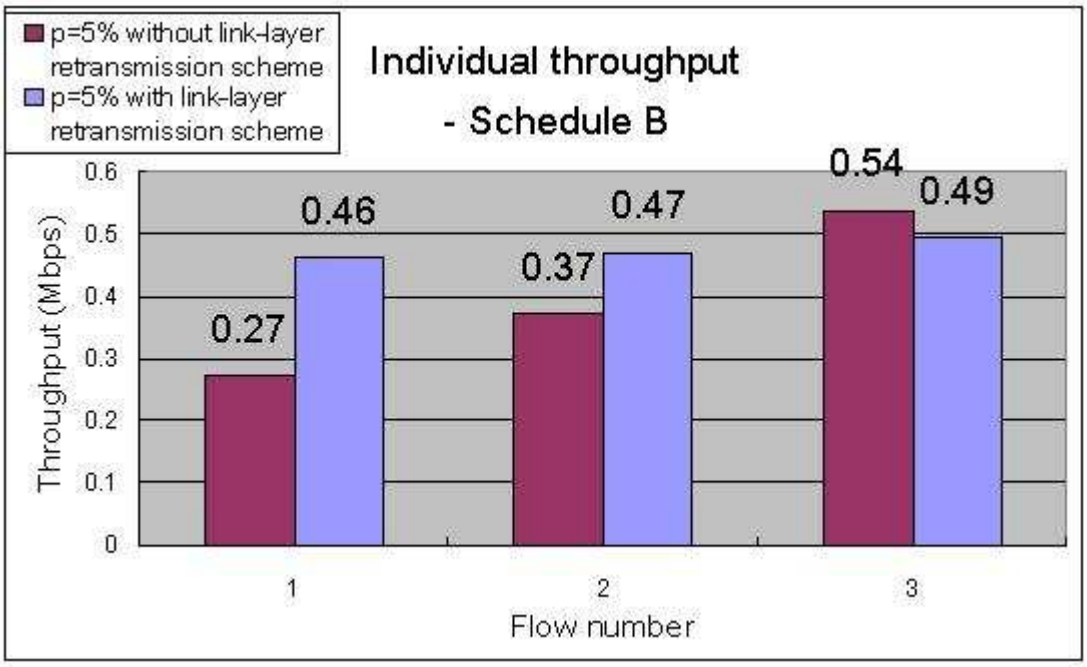


圖 5-5 individual TCP throughput under *p*=5% with link-layer retransmission (four-node)

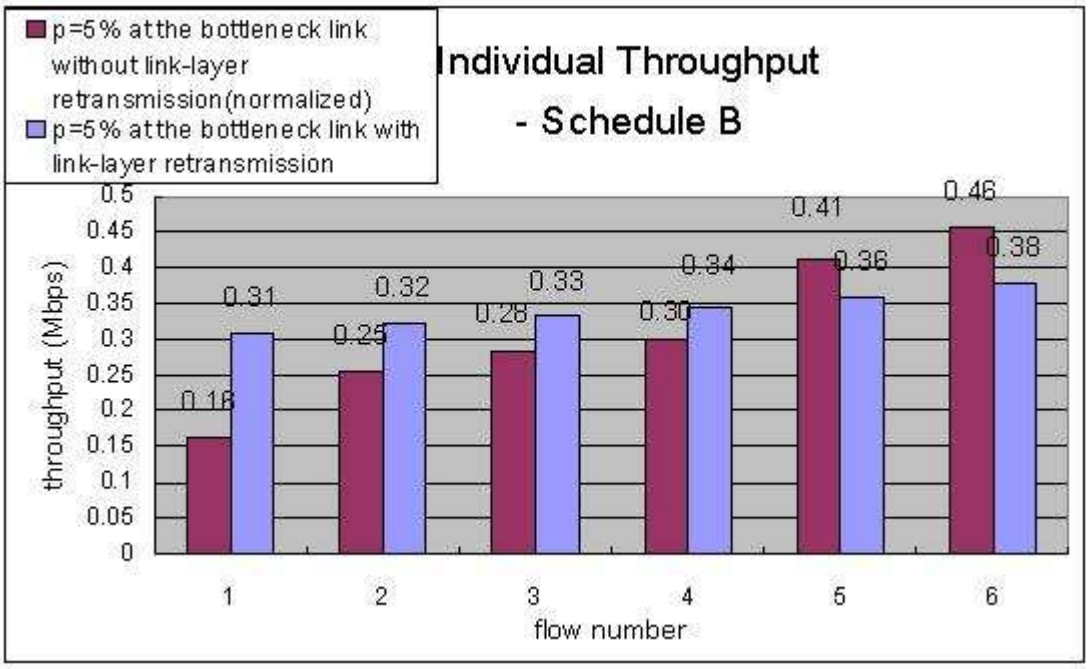


圖 5-6 individual TCP throughput under *p*=5% with link-layer retransmission (seven-node)

由這兩張圖可以發現一個現象，使用 link-layer retransmission scheme 對 long-hop flow 的 throughput 有很明顯的改善，但反而傷害 short-hop flow 的 throughput。這是因爲對 short-hop flow 來說，因爲反應時間較短，TCP end-to-end 的 retransmission scheme 就足以讓它們很迅速地進行 packet loss recovery，因此加上 link-layer retransmission scheme 反而增加 short-hop flow 不必要的負擔，使可用頻寬減少。Link-layer retransmission scheme 對 long-hop flow 處理 packet loss 的

幫助比較大。

我們將 packet loss probability *p* 變化，觀察有無使用 link-layer retransmission scheme 的 TCP throughput，結果如圖 5-7。

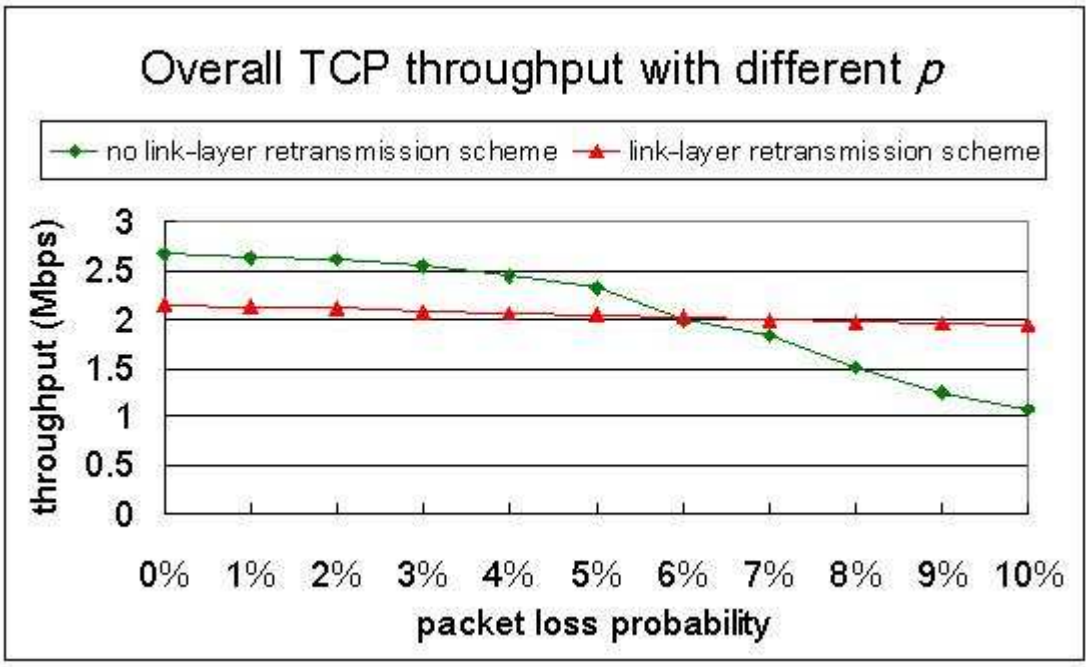


圖 5-7 不同 packet loss probability 時 TCP throughput 之變化

圖 5-7 中不使用 link-layer retransmission scheme 的 throughput 是未經 normalization。我們可以觀察到一個趨勢，隨著 *p* 增加，不使用 link-layer retransmission 的 TCP throughput 衰退愈趨嚴重；但使用 link-layer retransmission scheme 的 TCP throughput 衰退並不明顯。因爲使用 link-layer retransmission scheme 增加 overhead(花費頻寬傳輸 link-layer ACK 和 local retransmission)，它的好處只有在 wireless link 狀況非常糟糕、packet loss 很嚴重時才會顯現出來。在 packet loss 狀況輕微時，end-to-end 的機制就足以保持 TCP 快速復原 packet loss 的特性；但是當 packet loss 漸趨嚴重時，需要 local link-layer retransmission scheme 幫助 TCP 保持快速復原 packet loss 的特性。

5.2.3. TCP congestion window 變化情形

p=5%時使用 link-layer retransmission scheme 的 TCP congestion window 隨時間變化的情形如圖 5-8、圖 5-9。比對不使用 link-layer retransmission scheme 的圖(圖 4-6、圖 4-7)之後，我們可以發現，使用 link-layer retransmission scheme 的 TCP congestion window 很快地成長到最大值並維持在最高點，達到 steady state，一如沒有發現 packet loss 的狀況(圖 3-20 圖 3-21、圖 3-22、圖 3-23)。這代表使用 link-layer retransmission scheme 可以讓上層的 TCP 不會發現 packet loss 而引發 retransmission/fast retransmission 的反應而使 congestion window 不斷地震盪。

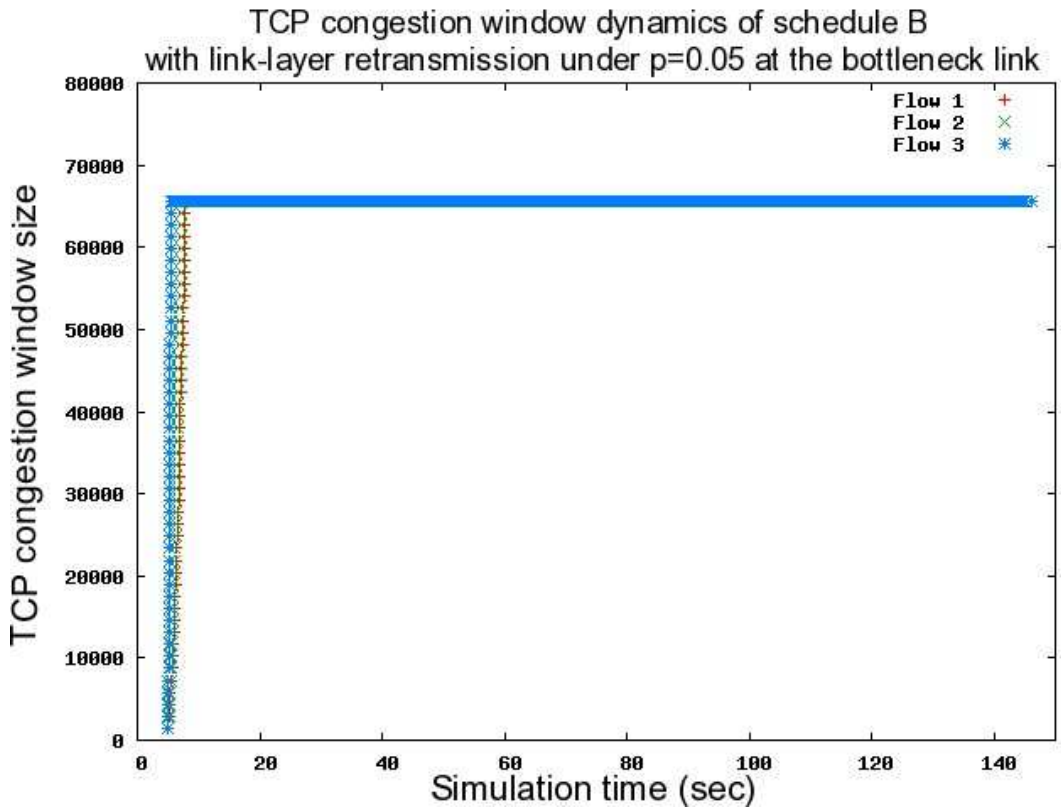


圖 5-8 *p*=5%使用 link-layer retransmission scheme 之 TCP congestion window 變化情形(four-node)

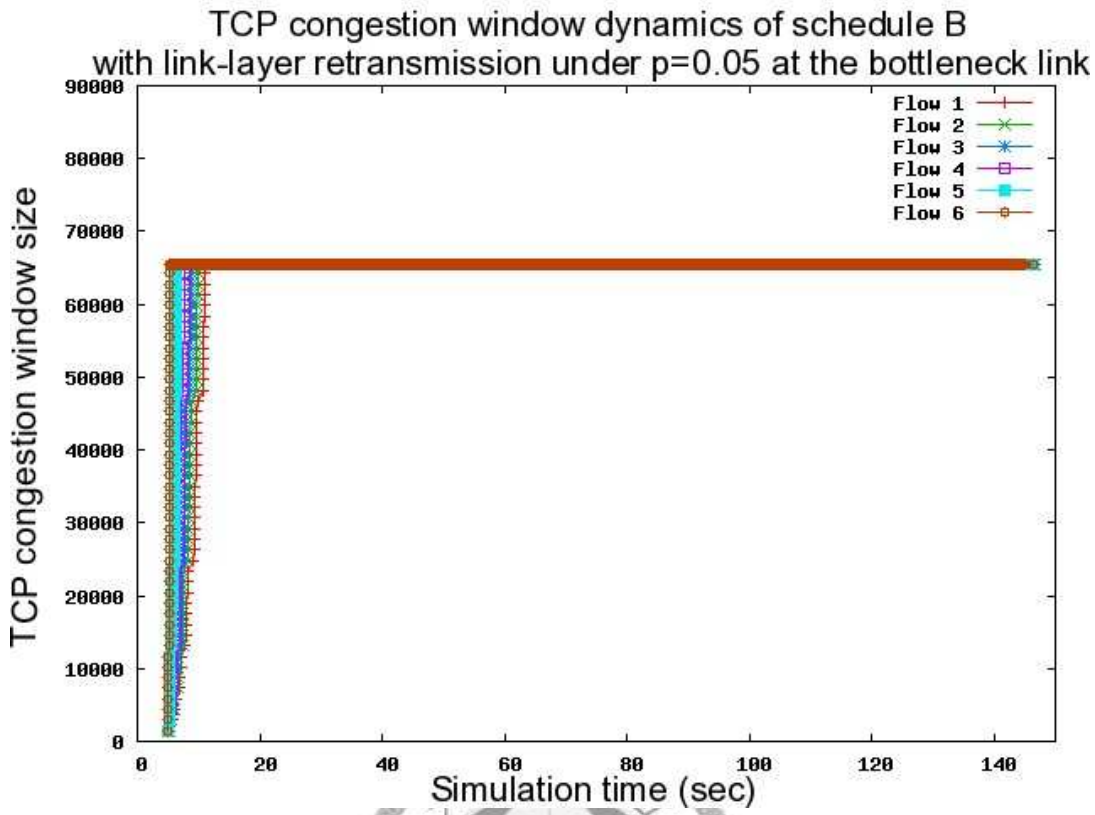


圖 5-9 *p*=5%使用 link-layer retransmission scheme 之 TCP congestion window 變化情形(seven-node)

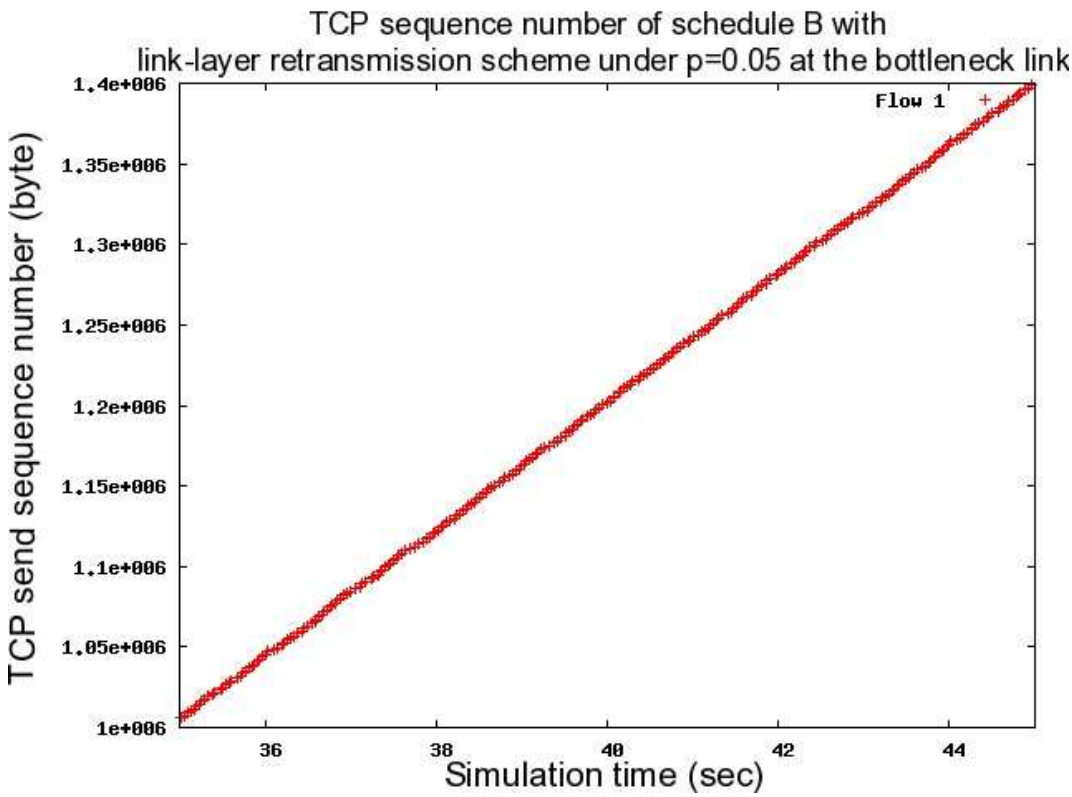


圖 5-10 TCP sequence numberof flow 1 with link-layer retransmission scheme under p=5% (seven-node)

圖 5-10 是使用 link-layer retransmission scheme 之後，接收端收到的 TCP sequence number，時間尺度與圖 4-4 相同，這裡僅列出 flow 1 的結果，其它 flow 也有類似的結果。與圖 4-4 比較可以發現，使用 link-layer retransmission scheme 之後，圖上沒有出現明顯的時間斷層，表示 TCP 不再受到 random packet loss 的影響而發生 timeout，因爲 link-layer retransmission scheme 已經在 local link 將 packet loss 處理好了。因此對 TCP 來說，下層有 link-layer retransmission scheme 的保護，TCP 不會知道 packet loss 發生，使用 link-layer retransmission scheme 可以保持 TCP 快速復原 packet loss 的良好特性。

5.2.4. TCP smoothed round-trip time

表 5-8 比較 *p*=0%時有無使用 link-layer retransmission scheme 之 srtt (four-node)

		TCP srtt under *p*=0% without link-layer retransmission scheme (sec)	TCP srtt under *p*=0% with link-layer retransmission scheme (sec)	Increased ratio (%)
Schedule B	Flow 1	0.9219	1.0625	15.25
	Flow 2	0.8594	1.1094	29.09
	Flow 3	0.8906	1.1719	31.58

我們先列出 *p*=0%時的 smoothed round-trip time(srtt)實驗結果比較，以瞭解使用 link-layer retransmission scheme 會對 srtt 有何影響，再觀察 *p*=5%時的 srtt。表 5-8 是 four-node chain topology、*p*=0%的 srtt 比較，表 5-9 是 seven-node chain topology、*p*=0%的 srtt 比較。在沒有 packet loss (意即沒有任何 retransmission)的狀況下，使用 link-layer retransmission scheme 比不使用約增加 srtt 15%到 30%的長度。

表 5-9 比較 *p*=0%時有無使用 link-layer retransmission scheme 之 srtt (seven-node)

		TCP srtt under *p*=0% without link-layer retransmission scheme (sec)	TCP srtt under *p*=0% with link-layer retransmission scheme (sec)	Increased ratio (%)
Schedule B	Flow 1	1.3281	1.6094	21.18
	Flow 2	1.2656	1.5000	18.52
	Flow 3	1.2344	1.5000	21.52
	Flow 4	1.2344	1.4844	20.25
	Flow 5	1.2188	1.4844	21.79
	Flow 6	1.1563	1.3750	18.92

表 5-10 和表 5-11 列出 *p*=5%的 srtt 數據，有 packet loss 的情況下使用 link-layer retransmission 大幅度地增加 TCP 的 round-trip time，增加幅度從 5 倍到 15 倍。

表 5-10 比較 *p*=5%時有無使用 link-layer retransmission scheme 之 srtt (four-node)

		TCP srtt under *p*=5% without link-layer retransmission scheme (sec)	TCP srtt under *p*=5% with link-layer retransmission scheme (sec)	Increased ratio (%)
Schedule B	Flow 1	0.2031	1.1563	469.23
	Flow 2	0.1250	1.1563	825.00
	Flow 3	0.1094	1.1094	914.29

Link-layer retransmission scheme 在 packet loss 的情況下，讓 srtt 比不使用 link-layer retransmission 大幅增加的原因有二：一，Link-layer retransmission scheme 在傳送每個 TCP data packet 或 ACK packet 之後都要花時間回傳 link-layer ACK，或是花時間進行 local retransmission，因此增加 round-trip time。二，因爲 link-layer retransmission scheme 讓 TCP 完全沒有發覺下層的 packet loss，因此 TCP 很順利很迅速地達到 steady state，因此造成大量 TCP data packet 排隊排在 bottleneck link，增加 queueing delay，這個狀況和 *p*=0%的情形相同。

表 5-11 比較 *p*=5%時有無使用 link-layer retransmission scheme 之 srtt (seven-node)

		TCP srtt under *p*=5% without link-layer retransmission scheme (sec)	TCP srtt under *p*=5% with link-layer retransmission scheme (sec)	Increased ratio (%)
Schedule B	Flow 1	0.2188	1.6250	642.86
	Flow 2	0.2500	1.6094	543.75
	Flow 3	0.1406	1.6094	1044.44
	Flow 4	0.1875	1.5000	700.00
	Flow 5	0.1875	1.5000	700.00
	Flow 6	0.1563	1.5000	860.00

5.3. 討論

在這一章，我們使用 link-layer retransmission scheme 改善 TCP 在 wireless mesh network 上遭遇 random packet loss 而下降的 performance。Link-layer retransmission scheme 的機制中 link-layer 必須傳送 ACK 以確認封包送達，因此會增加 overhead。在我們的環境設定中(參見 3.3 和 5.2.1)，傳送 link-layer ACK 約增加 21%的 overhead。在 *p*=5%的狀況中，僅比較 throughput 的話，因爲 overhead 太高，使用 link-layer retransmission scheme 的 throughput 反而沒有提升。若將 link-layer ACK 所增加的 overhead 排除(將不使用 link-layer retransmission scheme 的 throughput 做 normalization)，則 link-layer retransmission 對 throughput 是有幫助的，尤其是對 long-hop flow 的幫助非常顯著；但對 short-hop flow 來說，TCP 的 retransmission 機制已經可以處理 random packet loss，加上 link-layer retransmission 反而沒有幫助。但整體來說，使用 link-layer retransmission 可以使 throughput fairness 提升。

使用 link-layer retransmission 之後，由於 random packet loss 已經在 link-layer

被處理掉了，TCP 完全不會發覺 packet loss，因此 congestion window 變化情形如同第三章沒有 packet loss 的狀況一樣，所有 TCP flows 都可以很快的達到 steady state。

總結來說，link-layer retransmission scheme 在 packet loss 很嚴重的狀況下，對 throughput 和 fairness 都很有幫助。但是在不嚴重的 packet loss 情形之中，link-layer retransmission scheme 帶來的 overhead 大於對 throughput 改善的程度，因此僅有 fairness 提升。最重要的是，link-layer retransmission scheme 保持了 TCP 一項重要的特性：讓 TCP 能快速的復原 packet loss。

6. 總結與未來展望

在本篇論文裡面，我們考慮 TCP transmission 的特性妥善分配 TCP data 和 TCP ACK 的頻寬，並考慮無線網狀網路的干擾問題提出兩種 spatial TDMA scheduling scheme，一種是讓所有 wireless hop 分配一樣數量的 timeslots 的 schedule A，另一種是按照 traffic load 分配 timeslots 的 schedule B，並進行模擬實驗討論這兩種 scheduling 的 TCP 效能表現。我們希望能藉由 spatial TDMA scheduling 的方式，讓 TCP transmission 在有線網路上的良好特性能在無線網狀網路的環境上被保留。

在 chain topology 上的實驗數據顯示，在沒有 packet loss 發生的狀況下，2 種 TDMA scheduling 都可以讓 TCP flows 很快的到達 steady state，但考量 traffic demand 的 scheduling B 可以使 TCP 更完全更有效地使用可用頻寬並縮短 round-trip time 而改善 TCP throughput。

在第四章我們更進一步考慮 wireless link error 造成的 random packet loss 對上述 TCP performance 的影響。在所有 TCP flows 共同流經的 bottleneck link 上發生的 packet loss 對 long-hop flow 的傷害大於 short-hop flow，因此有不公平的 throughput 分配。我們在第五章提出 link-layer retransmission scheme 來改善因爲 packet loss 造成的效能衰退。Link-layer retransmission scheme 在 packet loss 嚴重的狀況下，有效的改善 overall TCP throughput 和 fairness；但在輕微 packet loss 的狀況下，link-layer retransmission scheme 帶來的 overhead 顯得太高，僅改善 faireness 而對 throughput 沒有明顯幫助。

在本篇論文中，模擬實驗都只有在 chain topology 上進行，後續的工作應是在 topology 較複雜的無線網狀網路上(例如 grid topology)，進行實驗分析使用這兩種 TDMA scheduling scheme 的 TCP 效能表現。

參考文獻

[1] I. F. Akyildiz, X. Wang, W. Wang, “Wireless Mesh Networks: A Survey,” *Computer Networks*, vol. 47, no. 4, pp. 445-487, March 2005.

[2] M. Allman, V. Paxson, W. Stevens, “Transmission Control Protocol,” *RFC 2581*, IETF Network Working Group, April 1999.

[3] S. Floyd, T. Henderson, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” *RFC 2582*, IETF Network Working Group, April 1999.

[4] S. Floyd, T. Henderson, A. Gurtov, “The NewReno Modification to TCP’s Fast Recovery Algorithm,” *RFC 3782*, IETF Network Working Group, April 2004.

[5] Zhenghua Fu, Pertros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang, Mario Gerla, “The Impact of Multihop Wireless Channel on TCP Throughput and Loss,” *IEEE INFOCOM* 2003.

[6] Ruy de Oliveira, Torsten Braun, “A Dynamic Adaptive Acknowledgement Strategy for TCP over Multihop Wireless Networks,” *IEEE INFOCOM* 2005.

[7] Ruy de Oliveira, Torsten Braun, “A Smart TCP Acknowledgment Approach for Multihop Wireless Networks,” *IEEE Transaction on Mobile Computing*, Febuary 2007.

[8] Sherif M. ElRakabawy, Alexander Klemm, Christoph Lindermann, “TCP with Adaptive Pacing for Multihop Wireless Networks,” *ACM MobiHo*c 2005.

[9] Sherif M. ElRakabawy, Alexander Klemm, Christoph Lindermann, “Gateway adaptive pacing for TCP across multihop wireless networks and the Internet,” *ACM Proc. of Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2006.

[10] Kun Tan, Feng Jiang, Qian Zhang, Xuemin Shen, “Congestion Control in

Multihop Wireless Networks,” *IEEE Transaction on Vehicular Technology*, March 2007.

[11] Jitentra Padhye, Victor Firoiu, Don Towsley, Jim Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation, ” *ACM SIGCOMM*, 1998.

[12] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz, “The Effects of Asymmetry on TCP Performance,” *Mobile Netw. Appl*. Vol.4, 1999, pp. 219-241.

[13] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems", *DEC Research Report TR-301*, September 1984.

[14] David A. Eckhardt, Peter Steenkiste, “An Internet-style approach to wireless link errors,” *Wirel. Commun. Mob. Comput.*, Vol.2, no.1, pp. 21-35, December 2001.

[15] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz, “Improving TCP/IP Performance over Wireless Networks,” *ACM Mobicom*, 95

[16] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz, “Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks,” *Wireless Networks*, vol.1, no.4, December 1995

[17] A. Bakre and B. R. Badrinath, “I-TCP: Indirect TCP for Wireless hosts,” in *Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS '95)*, Vancouver, BC, May 1995