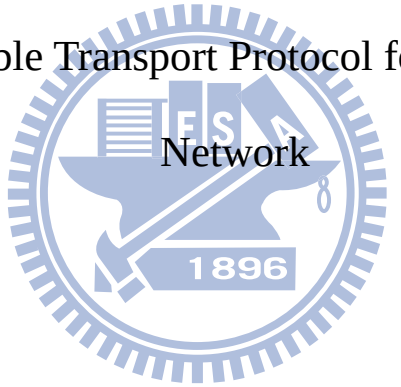
國 立 交 通 大 學

網路工程研究所

碩 士 論 文

適用於無線個人區域網路之可靠的傳輸協定

WPAN-ATP: A Reliable Transport Protocol for Wireless Personal Area Network



研 究 生：李勝焜

指導教授：邵家健　博士

中 華 民 國 九 十 八 年 十 月

適用於無線個人區域網路之可靠的傳輸協定

WPAN-ATP: A Reliable Transport Protocol for Wireless Personal Area Network

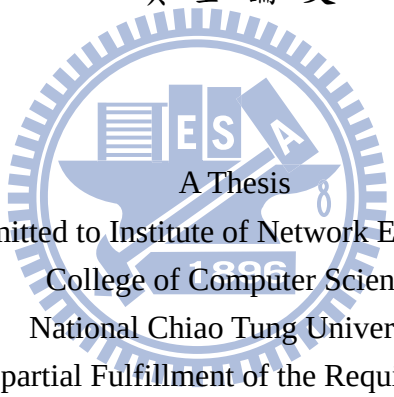
研 究 生：李勝焜　　Student：Sheng-Kun Li

指導教授：邵家健　　Advisor：Kar-Kin Zao

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文



A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

October 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年十月

適用於無線個人區域網路之可靠的傳輸協定

學生：李勝焜　　指導教授：邵家健 博士

國立交通大學網路工程研究所　碩士班

摘要

醫學上對多頻道寬頻生理訊號(例如：EEG)的型態有明確定義，若要在無線個人區域網路(WPAN)中傳送此類生理訊號，必須確保資料的順序性、正確性和完整性。無線個人區域網路中的裝置多半計算能力較低且講求省電，所以要盡量降低傳輸的成本。由於有在單一裝置上開啟多個連線的需求，所以需要透過傳輸層來處理個別連線的傳輸。本論文中，參考現有的 ATP 和 TCP，設計一個適用於無線個人區域網路之可靠的傳輸協定：WPAN-ATP。它具備以下特性：(1) 串流(streaming)和順序(ordering)的傳輸：使用封包序號並做排序動作；(2)封包遺失回復(packet loss resilient)：使用 SACK 且提供資料重傳機制，以確保資料完整性；(3) 有效率使用頻寬(bandwidth-efficient)：利用基於速率的流量控制(rate-based flow control)來避免傳送流量超出接收端的負荷，並在此前提下使用最大頻寬；(4)低成本(low cost)傳輸：簡化流量控制方法中的運算，傳輸短暫中斷時，減少不必要的資料傳送。最後，在無線感測裝置 SunSPOT 上實作 WPAN-ATP，接著在實際環境中進行資料傳輸實驗，並分析協定的行為。

WPAN-ATP: A Reliable Transport Protocol for Wireless Personal Area Network

Student: Sheng-Kun Li Advisor: Dr. John Kar-Kin Zao

Institute of Network Engineering

National Chiao Tung University

ABSTRACT

In medicine, the patterns of multi-channel broadband biomedical signals (e.g. EEG) are clearly defined. When we want to transmit such biomedical signals in Wireless Personal Area Network (WPAN), we have to ensure data order, correctness and integrity. For resource-constrained devices, we need a low cost transmission. In addition, we need a transport layer protocol to handle multiple sessions on one device. In this thesis, we present WPAN-ATP, a reliable transport protocol for WPAN. It has the following characteristics: (1) streaming and ordering transmission: data packet ordering by sequence numbers; (2) packet loss-resilient: SACKs and data retransmission mechanism to ensure data integrity; (3) bandwidth-efficient: rate-based flow control for traffic overloaded avoidance and high utility of bandwidth; (4) low cost transmission: simplify computation and reduce unnecessary data transmission. Finally, we implement WPAN-ATP on wireless motes, SunSPOT, and then run some data transmission experiments in the real environment and analyze behaviors of the protocol.

致 謝

首先，我要感謝指導教授邵家健博士。您的指引和建議幫助我完成這篇論文。在這兩年多的研究歲月中，我從您身上學到如何思考，如何做好事情，如何表達自己的意見。

接著感謝口試委員黃育綸教授和曹孝櫟教授。你們的提問讓我知道論文的缺失，你們的建議讓我學到不少。

我要感謝實驗室的夥伴們。嘉錡、星閔和我都是從交大資工系直升交大網工所，雖然在大學時不是很熟，但是在研究生涯的最後階段，三個人已建立起革命情感，彼此互相打氣，互相幫助。博政、梅瑛、彥霖陪伴我度過兩年時光，不論在研究上、課業上、還是生活上，大家都能互相幫助，能和你們一起度過這段時間，我真的很開心。

我要感謝實驗室的學長們。國晉、芳伯、輔國、哲民幾位學長，在你們的照顧下，我逐漸適應研究生活。我不會忘記大家一起打籃球、一起上大聯盟的日子。特別是哲民學長，常在我的個板上為我加油打氣。我還要感謝育志學長，在大學專題其間給我不少幫助，即使畢業，當我在電資 807 遇到困難時，也趕來學校協助我解決問題。

我要感謝實驗室的學弟妹。子晉、鈞凱、志明、俊瑋、嘉妤，有你們的陪伴，我的生活更多采多姿。尤其要感謝子晉，在口試之前替我跑行政流程，幫我印論文初稿，口試當天幫忙拿水果，讓我能專心做最後衝刺。學弟妹們加油！

我要感謝系辦小姐陳小翠。雖然替我辦理口試及畢業程序是您的工作，但是我的延遲以及疏失為您帶來不少困擾，我覺得不好意思。您辛苦了。

最後要感謝我的家人。爺爺、奶奶、爸爸、媽媽，沒有你們的栽培就沒有現在的我，你們對我的付出和照顧實在太多；弟弟、妹妹們，沒有你們的陪伴我會孤單，你們為我帶來快樂。真的非常感謝我的家人。

感謝所有陪伴我走過這段歲月的人。

目 錄

圖目錄

表目錄

第 1 章　綜論

1.1　問題陳述

健康監測朝無線化發展，在無線個人區域網路(Wireless Personal Area Network)中傳送多頻道寬頻生理訊號(例如：腦電波)的需求提高。醫學上，對於多頻道寬頻生理訊號的振幅和頻率有明確的定義，若傳輸過程中出現資料遺失或錯誤，會使訊號失真，甚至不被接受。所以傳送生理訊號必須確保資料的順序性、正確性和完整性。

無線個人區域網路中最常見星狀網路拓撲(star topology)，且多為單次跳躍(single-hop)。通常星狀拓撲中會有一個基站(base station)負責接收數個無線感測裝置傳來的資料，而且無線感測裝置上的應用程式可能會有分開傳送不同生理訊號的需求，所以會開啟兩個以上的連線(session)。由於媒介存取控制層(MAC layer)無法分辨單一裝置上的不同連線，所以需要靠傳輸層處理。此外，無線感測裝置多半運算能力低、記憶體容量小、低功率。所以在無線個人區域網路中傳送生理訊號，除了要確保每個連線中資料的順序性、正確性、完整性之外，還要確保各連線的流量不會超過基站的處理速度，並且盡量降低資料傳輸對無線感測裝置造成的負擔。

現行可靠的傳輸協定並不能滿足這樣的需求，TCP 在無線網路環境中有不少缺點[1]，許多改良自 TCP 的傳輸協定或新的協定比較複雜，超出我們的需求。所以我們需要適用於無線個人區域網路之可靠的傳輸協定。

1.2　研究方法

針對在 WPAN 中傳送多頻道寬頻生理訊號的需求，我們提出以下構想：中介層(middle layer)建構在傳輸層之上，負責將數組取樣數據分頻道打散重排，並提供斷線儲存與補傳機制。資料打散重排之目的在降低單一封包遺失對整組訊號的影響；斷線儲存與補傳機制用來抵抗連線中斷的衝擊。傳輸層是中介層的基礎，中介層的機制皆建立在可靠的

傳輸層之上。本研究著眼在傳輸層，提供適用於 WPAN 之可靠的串流傳輸。

本研究參考 ATP (Ad-hoc Transport Protocol) 的設計概念，提出一個適用於低速率無線個人區域網路(Low Rate WPAN，LR-WPAN)之可靠的傳輸協定：WPAN-ATP (A Reliable Transport Protocol for WPAN)，中文名稱為「無線個人區域網路可靠傳輸協定」。

WPAN-ATP 具備以下特性：(1)串流(streaming)和順序(ordering)的傳輸：主要用來處理封包遺失導致的次序錯亂問題。資料封包(data packet)有封包序號(sequence number)，並在接收端根據封包序號做排序。(2)封包遺失回復(packet loss resilient)：此機制用來確保接收端可以收到完整的資料。接收端定時回送 SACK (Selective ACK) 告知傳送端已收到的封包序號，傳送端根據此資訊判斷是否要啟動重傳機制。使用 SACK 之目的在減少接收端的傳送次數，並降低 ACKs 佔用的頻寬。(3)有效率使用頻寬(bandwidth-efficient)：基於速率的流量控制(rate-based flow control)，根據傳送端的傳送延遲和接收端的處理速度調整封包傳送速率(transmission rate)，目的在避免傳送流量超出接收端的處理速度。以速率為基礎，能夠更準確反應雙方處理速度和網路狀況，可以更有效使用頻寬。(4)低成本(low cost)傳輸：在流量控制方法上多避免除法運算；在傳輸短暫中斷時，暫停資料和 SACK 的傳送，進行網路偵測，以減少不必要的傳送。

最後，將 WPAN-ATP 實作在無線感測裝置 SunSPOT 上，並遵循 J2ME 定義的通用連接框架(Generic Connection Framework，GCF)，以方便應用程式使用 WPAN-ATP。接著，在實際環境中做資料傳送測試，錄下數據，分析此傳輸協定的行為。

1.3 論文大綱

接下來的論文分為四章。第二章為背景知識，先簡單說明 ATP 的設計概念，再介紹昇陽公司(Sun Microsystems)開發的無線感測裝置 SunSPOT。第三章為本論文的重點，說明 WPAN-ATP 的設計概念、連線管理、資料傳送與接收、SACK 和重傳機制、基於速率的流量控制。第四章討論實作議題，簡單說明幾個實作在 SunSPOT 上的重點，並從實驗數據分析 WPAN-ATP 的行為。第五章講本研究的貢獻以及未來可行的研究方向。

第 2 章　背景知識

2.1　ATP: Ad-hoc Transport Protocol

ATP[2]是針對無線隨意網路(wireless ad-hoc network)設計的傳輸協定，用來解決 TCP 不適用於無線隨意網路的問題。以下說明 ATP 的設計概念：

(1) 分層合作 (Layer Coordination)

ATP 使用下層提供的資訊和來自其他節點的資訊來完成各項機制。使用其他節點的資訊可以做到：(a)根據速率回饋(rate feedback)決定傳送速率的初始值；(b)根據速率回饋做擁塞偵測(congestion detection)、擁塞預防(congestion avoidance)、擁塞控制(congestion control)；(c)路徑失效通知(path failure notification)。

(2) 基於速率的傳輸 (Rate Based Transmissions)

以速率來控制傳輸，可以改善 window-based 傳輸的缺點：(a)避免傳輸叢集(burstiness)；(b)可以捨棄根據 ACKs 調整 congestion window 大小的傳送速率控制方式，做到擁塞控制(congestion control)和可靠度(reliability)分離。此外，傳送開始時，傳送端根據速率回饋決定起始速率，這步驟稱為 Fast Start，目的在改善 TCP 中 Slow Start 可能導致頻寬使用率偏低的問題。

(3) 擁塞控制和可靠度分離

針對擁塞控制，傳送路徑上的中間節點(intermediate nodes)提供延遲(delay)資訊給接收端，接收端據此決定一個速率回送給傳送端，傳送端據此調整傳送速率。延遲資訊透過資料封包傳遞，接收端的建議速率透過回饋封包(feedback)傳遞。針對可靠度，接收端定時傳送 SACKs 回報過去一段時間的串流漏洞，即遺失的封包，而傳送端根據此資訊判斷是否要進行封包重傳。重傳封包的優先權高於正常傳送封包。

(4) 擁塞控制 (Congestion Control)和流量控制(Flow Control)

資料封包經過傳送路徑上的節點時，計算延遲時間，若新的延遲值大於封包上的舊

延遲值，更新該欄位。接收端從資料封包得到的延遲值會是最大值，表示這條路徑的瓶頸。

中間節點的延遲計算如下：

$$\begin{cases} Q_t = \alpha * Q_t + (1-\alpha) * Q_{sample} \\ T_t = \alpha * T_t + (1-\alpha) * T_{sample} \end{cases}$$

其中，Q_t 為佇列延遲(queuing delay)的指數移動平均(exponential moving average，EMA)，T_t 為傳送延遲(transmission delay)的指數移動平均，Q_{sample} 和 T_{sample} 為最新的測量值。單一節點的總延遲 $D = Q_t + T_t$，若此 D 大於資料封包上的 rate feedback，將此值填入封包 rate feedback 欄位 D。

接收端計算 D 的指數移動平均 $D_{avg} = \beta * D_{avg} + (1-\beta) * D$，並比較 D_{avg} 對應的速率和應用程式讀取速率，取速率較小者的延遲值回送給傳送端，這個比較的目的在流量控制。

傳送端從 rate feedback 封包的延遲值算出建議速率 R，進行傳送速率 S 的調整。分為三個模式：Increase(提高)、Decrease(降低)、Maintain(維持)。

Increase：當 $S < R - \phi * S$，調整 $S = S + (R-S) / k$

Decrease：當 $S > R$，調整 $S = R$

Maintain：上列之外的情況，維持原 S

2.2 SunSPOT

SunSPOT 全名是 Sun Small Programmable Object Technology，這是昇陽公司(Sun Microsystems)開發的無線感測裝置，上面有 Java 虛擬機器(virtual machine)，軟體在 J2ME (Java 2 Micro Edition)的環境中開發。以下簡單介紹 SunSPOT 的軟硬體和網路程式開發，細節可以看 SunSPOT 的網站[3]和相關文件[4]。

硬體可分為三部分：電池、eSPOT Main Board、eDEMO Board。eSPOT Main Board 是 SunSPOT 的主體，上面有 180 MHz 32 bit ARM920T core 處理器，記憶體為 512KB RAM 和 4MB Flash。還有電力管理電路(power management circuit)提供省電機制，CC2420

提供 2.4GHz IEEE 802.15.4 radio，USB 介面接口方便和其他裝置連接。eDEMO Board 接在 eSPOT Main Board 之上，提供感測功能。上面有八個 LED 燈、兩個開關，以及溫度、光度、三維加速度的感測器，還有數位和類比的輸出輸入腳位(Input/Output pins)可以外接其他裝置，像是感測器、馬達等。

SunSPOT 上沒有像 TinyOS 那樣的作業系統，而是直接跑一個輕量型的 J2ME 虛擬機器：Squawk。它是一個軟性即時系統(soft real time system)。使用 compact bytecode instruction set，此指令集可以降低 J2ME class 檔案大小，並簡化垃圾回收(garbage collection)。應用程式使用 Java 編寫，在電腦主機上編譯成 class 並包裝成 suite 後再透過 USB 佈署(deploy)到 SunSPOT 上，其中 suite 是 class 的集合。

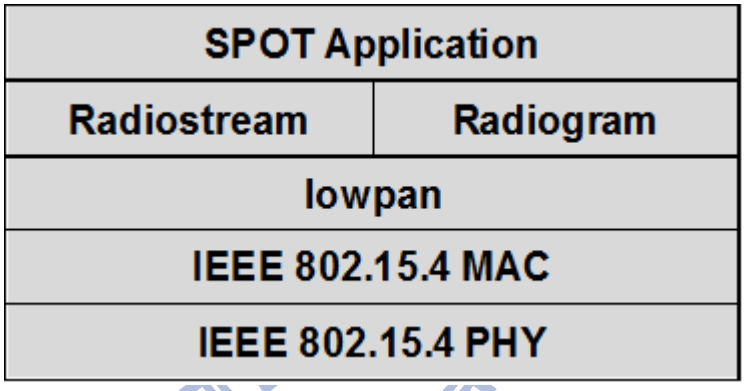


圖 1：SunSPOT 的網路協定堆疊

SunSPOT 的網路協定堆疊(network protocol stack)如圖 1 所示，下兩層使用 IEEE 802.15.4 標準，網路層為 lowpan，路由協定為 AODV，傳輸層有兩個傳輸協定，Radiostream 提供可靠的串流傳輸。Radiostream 是一個很簡單的傳輸協定，使用 ACK 確保資料送達，沒有擁塞控制(congestion control)和流量控制(flow control)機制。

為了方便網路程式開發，SunSPOT 支援 J2ME 通用連接框架(Generic Connection Framework，GCF)[5]。GCF 最基本的定義建構在 CLDC (Connected Limited Device Configuration) 之上，為介面階層(interface hierarchy)架構，每個介面皆可擴充(extends)。新定義的連線型態可以從基本介面擴充並提供實作。此外，GCF 使用統一資源定位器(Uniform Resource Locator，URL)來開啟並區分連線，例如：某 SPOT 要使用 radiostream 開啟連向另一個 SPOT 的連線，URL 為 ”radiostream://MAC_ADDRESS:PORT”。

第 3 章　WPAN-ATP: 無線個人區域隨意網路傳輸協定

3.1　設計概念

無線個人區域網路中，常見星狀網路拓撲(star topology)，基站(base station)是流量的集中點，所以需要流量控制以避免基站過度忙碌。由於裝置間距離較短，所以多為單次跳躍(single-hop)傳輸。單次跳躍網路要做到可靠的傳輸(reliable transmission)，在解決錯誤(bit errors)、路由失效(route failure)、擁塞(congestion)的問題上會比較簡單。在使用 IEEE 802.15.4 radio 的低速率無線個人區域網路(LR-WPAN)中，MAC 層會偵測錯誤；路由失效直接來自感測裝置和基站間的路由路線斷掉；擁塞可能是無線傳輸的頻道忙碌(channel busy)或是基站太忙碌。頻道忙碌由 MAC 層的 CSMA/CA 處理，基站忙碌可以用流量控制解決。

不過考慮到兩個裝置之間建立多個連線(multi-session)的情況，每一個連線有各自的資料傳輸，然而在 MAC 層無法分辨同一個裝置上的不同連線，所以需要靠傳輸層協定個別處理每個連線的資料排序、遺失封包重傳以及傳輸流量控制。

考慮到無線個人區域網路中常見的裝置多是低功率、低計算能力、低記憶體容量，所以要降低協定中的計算複雜度，並減少不必要的傳輸。

綜合以上論述、ATP 的設計概念以及 TCP 的連線管理方法[6][7]，WPAN-ATP 有以下設計：

(1) 為了有效處理連線，使用類似 TCP 的連線管理方法。每一個連線有自己的生命週期，分為起始階段、建立階段、關閉階段。起始階段同 TCP 做三方交握(three way handshake)，最大的差別在於接收端會將起始速率建議值連同 SYN|ACK 送給傳送端。

(2) 由於 WPAN 中的資料流向多為感測裝置傳往基站，所以將資料傳送和資料接收分開處理。連線型態為 Client 者可以做資料傳送，連線型態為 Server 者可以做資料接收，傳送和接收由各自的狀態機(state machine)控制。

(3) 為了做到串流(streaming)並確保資料順序(ordering)，傳送端會在每個資料封包填入封包序號(sequence number)，從 0 開始，每次加 1。接收端根據封包序號排列資料順序。在單次跳躍傳輸中，封包失序只會發生在封包遺失時。

(4) 為了確保資料完整性，設計封包遺失回復(packet loss resilient)機制。接收端收到資料封包後不立刻回送 ACK，而是每隔一段時間回送 SACK (Selective ACK)；傳送端根據 SACK 得知對方確實收到的封包序號，若有封包遺失，啟動重傳機制。重傳封包的優先權高於一般傳送封包，這可以縮短接收端等待排序的時間。使用 SACK 的目的在減少接收端的傳送次數，讓接收端可以更專注在接收動作；此外，可以降低回送封包佔用的頻寬，降低頻道忙碌的機率。

(5) 為了避免傳送流量過高造成接收端處理速度跟不上，設計基於速率的流量控制(rate-based flow control)機制。接收端根據應用程式的回應時間(application response latency)和 queue 的長度(表示累積的封包數量)，決定一個建議速率(suggest rate)，並跟著 SACK 回送給傳送端。其中，接收端應用程式回應時間反映應用程式處理速度。由於接收端定時傳送 SACK，加上降低回送封包數量的考量，所以流量控制和封包遺失回復機制都透過 SACK 傳遞資訊。傳送端在每次傳送時測量傳輸延遲(Tx latency)，此測量值反映傳送端傳輸層之下各層的處理速度以及 radio 傳輸的速度。傳送端收到 SACK 後，根據傳輸延遲和接收端的建議速率來調整傳送速率(transmission rate)。

(6) 基於速率的流量控制還可以達到有效使用頻寬之目的。在資料傳送開始時，接收端會透過 SYN|ACK 將起始速率建議值送給傳送端；當中斷的傳輸再度恢復時，接收端會透過 PRB|ACK 將建議速率送給傳送端。這個做法能使傳送端開始進行資料傳輸時，很快達到合適的傳送速率，更有效率使用頻寬。

(7) WPAN 中常見的裝置多半計算能力有限，而且講求低耗電，所以要減少不必要的運算以及不必要的傳輸。為了簡化運算，流量控制演算法中涉及除法處，大多以 2^k 為除數，在實作上可以轉換為移位(shift)。傳輸方面，若傳送端一段時間沒有收到接收端的回應，像是沒收到 SACK，或是偵測到路由失效(route failure)，則進入 DETECT 狀態。此時，停止資料傳送，改定時傳送探測封包 PRB；若收到接收端的回應，像是收到 ACK，則恢復正常資料傳送。若接收端一段時間沒收到資料封包，或是偵測到路由失效，則進入 IDLE 狀態。此時，停止傳送 SACK，改被動回應傳送端；若收到資料封包，恢復正常接收。

3.2 協定概述

WPAN-ATP 為適用於無線個人區域網路之可靠的傳輸協定，屬於 OSI 網路堆疊中傳輸層(transport layer)的協定。它具備以下特性：串流(streaming)和順序(ordering)的傳輸、封包遺失回復能力(packet loss resilient)、高效率使用頻寬(bandwidth efficient)、低成本(low cost)傳輸。分別以封包序號和排序機制、SACK 和重傳機制、基於速率的流量控制(rate based flow control)、傳輸中斷的網路偵測狀態來達到上述特性。

WPAN-ATP 有三個狀態機(state machine)負責協定的運作：連線狀態機(Connection State Machine)負責連線管理，傳送狀態機(Tx State Machine)負責資料傳送，接收狀態機(Rx State Machine)負責資料接收。

每一個連線皆有自己的生命週期，分為起始階段、建立階段、關閉階段。資料傳輸在建立階段中進行。整個資料傳輸過程包含資料傳送(data transmission)和資料接收(data reception)。資料傳送有三個模式：正常傳送(Normal Tx)、遺失重傳(ReTx loss)、全部重傳(ReTx all)，重傳有較高的優先權，傳送快慢以傳送速率(Tx rate)為上限。資料接收時會檢查封包序號，目的在資料排序和遺失偵測，順序對的封包內容會暫存在 RxQueue 中，等待應用程式讀取。圖 2 和圖 3 分別為資料傳送和資料接收的概念圖，實作細節不一定要和圖相同。

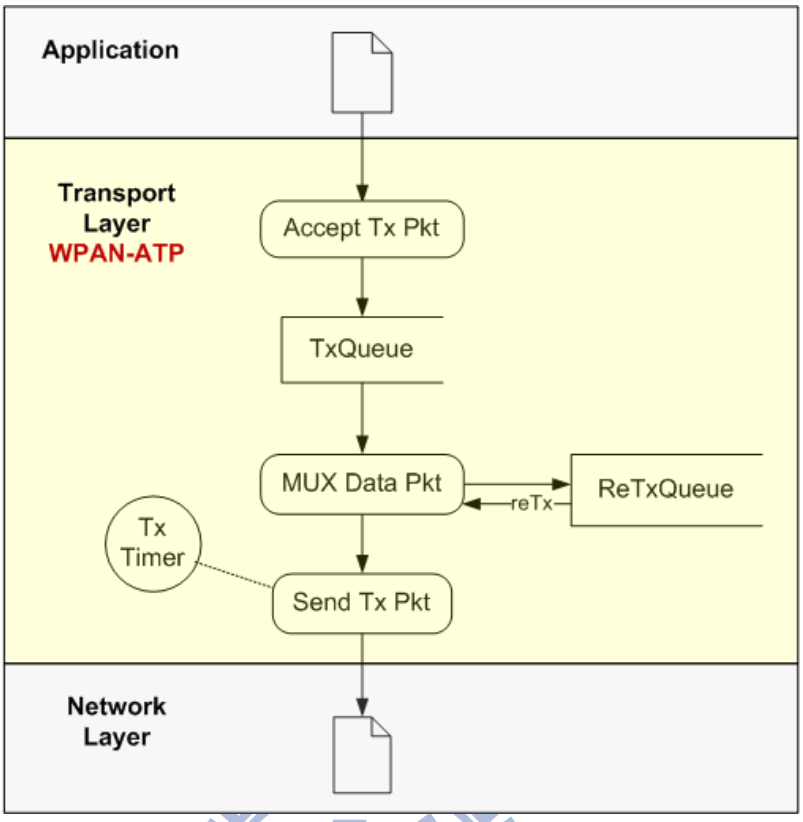


圖 2：資料傳送概念圖

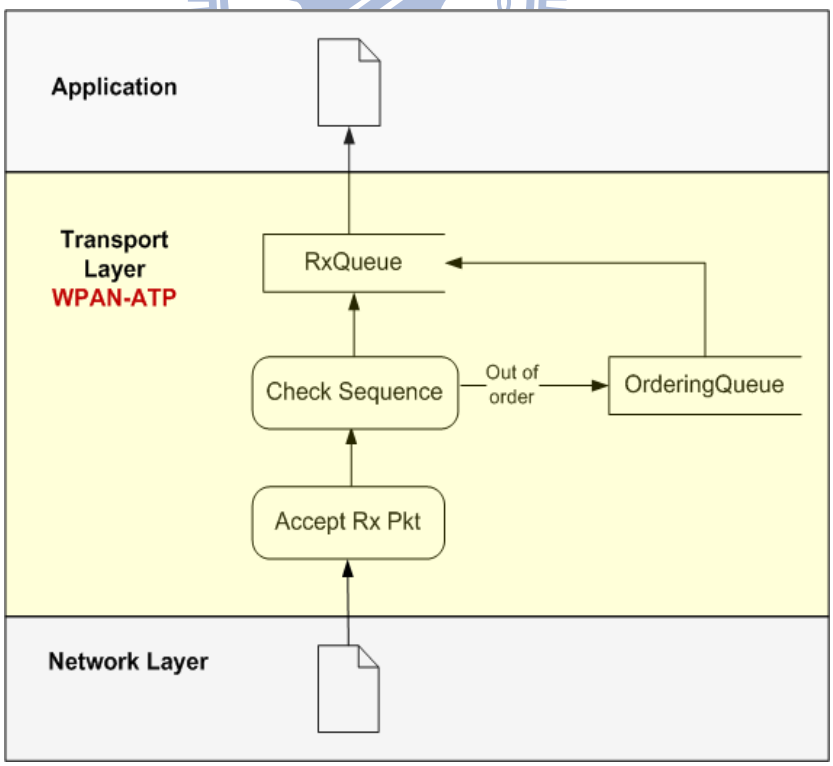


圖 3：資料接收概念圖

根據不同功能，封包型態分為資料封包(DATA)、SACK 封包、訊息封包(INFO)，封包格式如圖 4 所示。第一個 byte 是來源埠(source port)，第二個 byte 是目的埠(destination port)，第三個 byte 是封包型態，將對應的位元設定成 1 來表示。資料封包的 SeqNum 欄位存放封包序號，範圍在 0 到 65535。SACK 封包的 Rate 欄位存放接收端的建議速率，SLen 欄位是 SACK Blocks 的長度，即 block 的個數，在 3.6 有詳細的說明。訊息封包中的 SYN|ACK、SYN、FIN、ACK 用在連線的起始和關閉；當傳輸短暫中斷，PRB 和 PRB|ACK 用來進行網路狀態偵測。

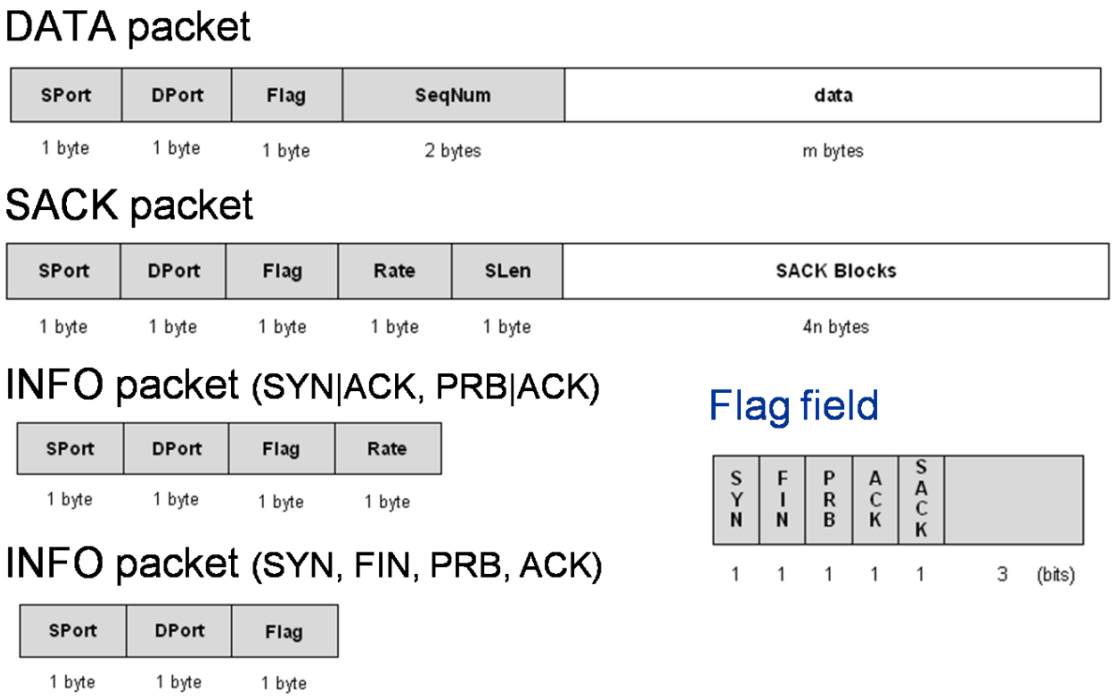


圖 4：封包格式

3.3 連線管理 (Connection Management)

如圖 5 所示，本協定的連線管理機制類似 TCP[6][7]，每個連線有其生命週期，CLOSED 同時為起始狀態和關閉狀態。這裡將連線狀態分為三階段：起始階段、建立階段、關閉階段。

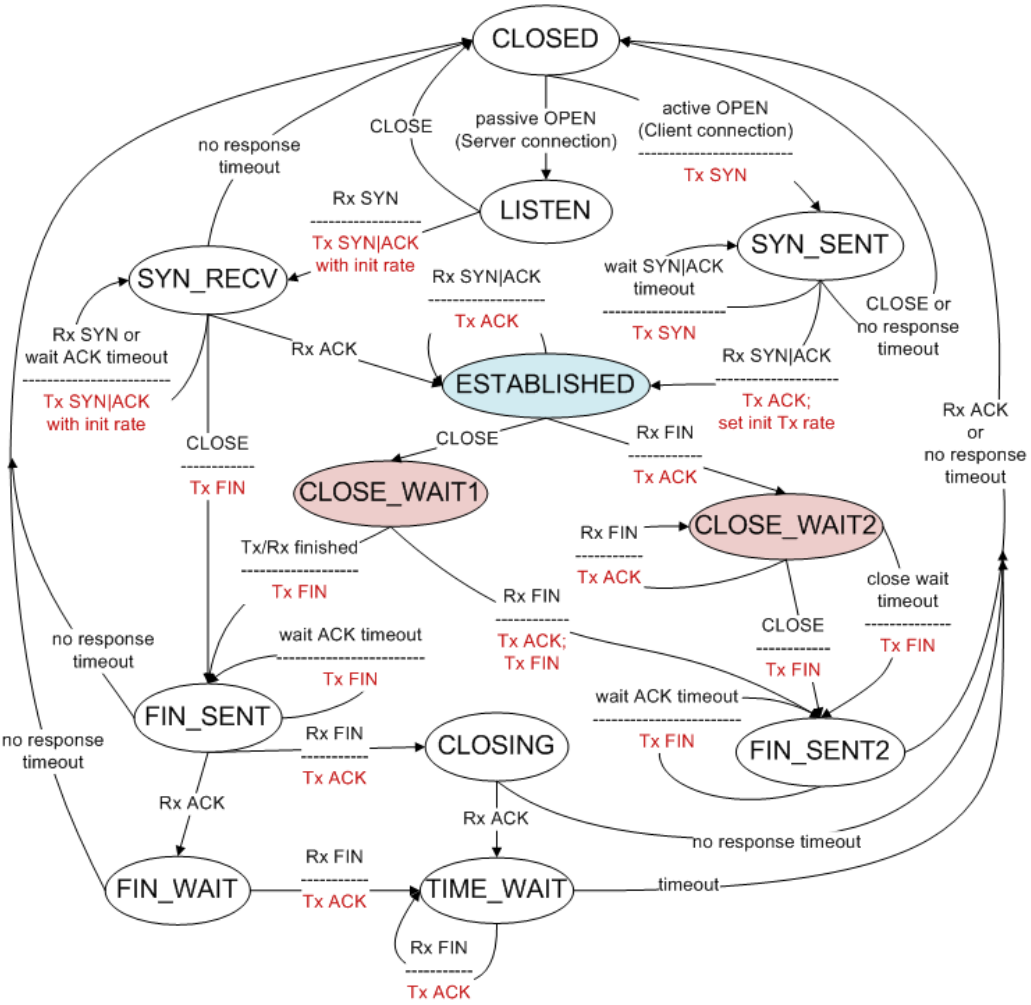


圖 5：連線狀態圖 (Connection State Diagram)

起始階段包含 CLOSED、LISTEN、SYN_RECV、SYN_SENT 四個狀態，和 TCP 一樣做三方交握(three way handshake)，訊息交換如圖 6 所示。傳送端連線型態為 Client，會主動要求連線，接收端連線型態為 Server，被動等待連線要求。此外，接收端會將起始速率建議值連同 SYN|ACK 送給傳送端。因為無線網路環境較容易出現封包錯誤或遺失，所以三方交握進行過程中若發生訊息封包遺失，會重送新的訊息封包，直到逾時(timeout)才放棄。

建立階段包含 ESTABLISHED、CLOSE_WAIT1、CLOSE_WAIT2 三個狀態，資料傳送和接收只在此階段中進行。當應用程式主動要求關閉連線，進入 CLOSE_WAIT1 狀態，等待傳送接收結束後才進入關閉階段。若對方先關閉連線，我方會收到 FIN，進入

CLOSE_WAIT2 狀態，等待傳送接收結束並且應用程式關閉連線後，才進入關閉階段。

關閉階段包含 FIN_SENT、FIN_WAIT、CLOSING、TIME_WAIT、FIN_SENT2，和 TCP 一樣做連線關閉動作，訊息交換如圖 7 所示。過程中若發生訊息封包遺失，會重送新的訊息封包，直到逾時才直接進入 CLOSED 狀態。

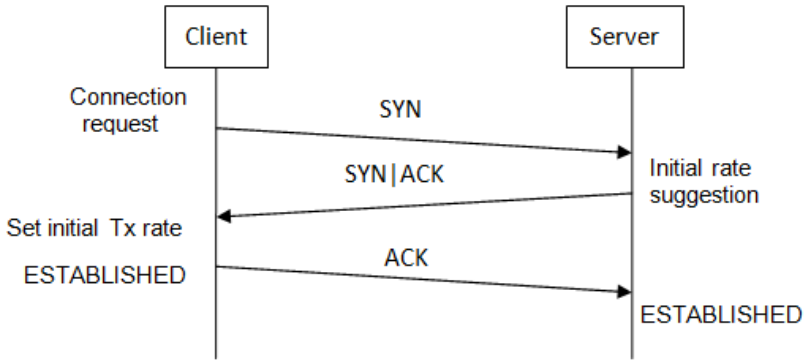


圖 6：起始階段訊息流(INIT message flow)

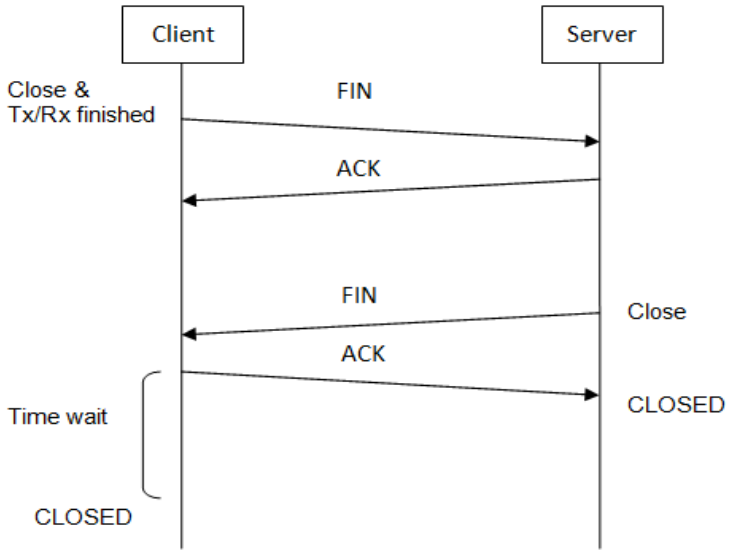


圖 7：關閉階段訊息流(CLOSE message flow)

3.4 資料傳送 (Data Transmission)

應用程式要求傳送資料時，傳輸層會將資料包裝成資料封包，並根據傳送速率進行封包傳送。所有已送出尚未 ACK 的封包都暫存在佇列(queue)中，每次收到 SACK 時，會將對方確實收到的封包從 queue 中刪除，若發生封包遺失，則改變傳輸模式，做重傳

動作。SACK 和重傳機制在 3.6 有詳細說明。

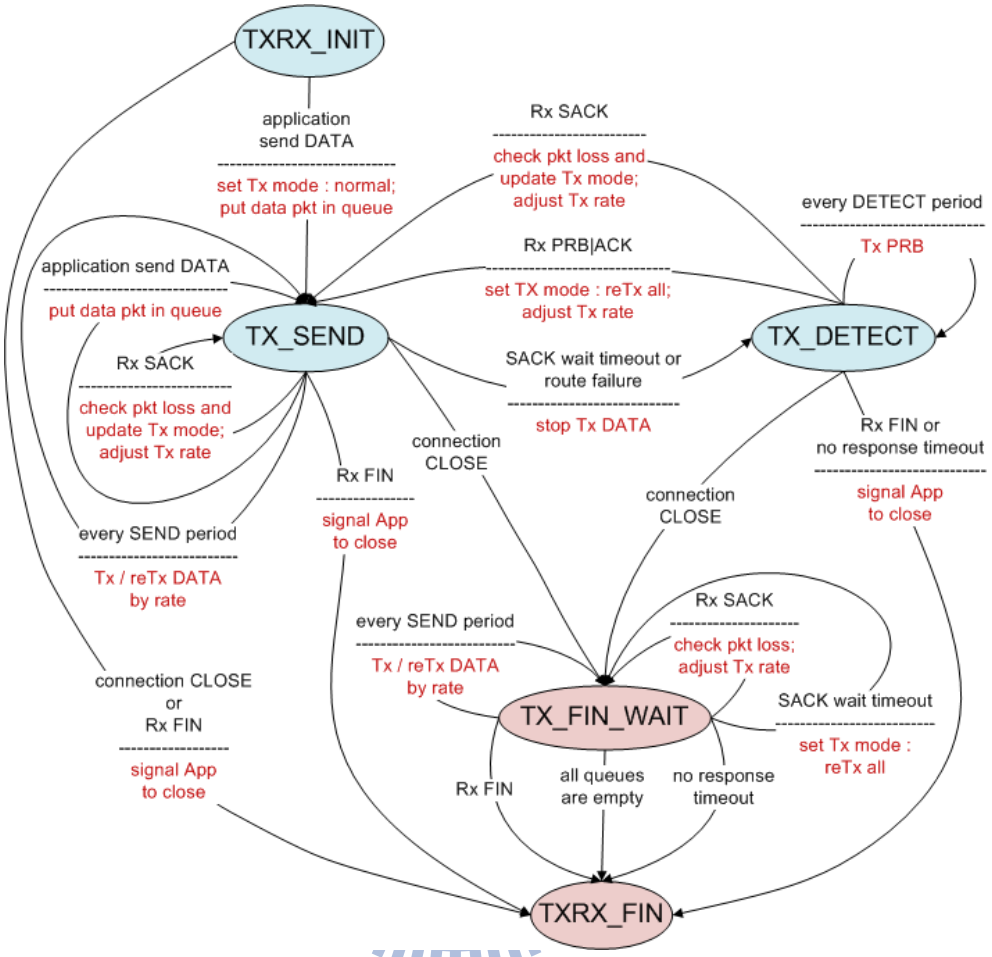


圖 8：傳送狀態圖 (Tx State Diagram)

如圖 8 所示，傳送行為由傳送狀態機(Tx State Machine)控制。當應用程式第一次要求傳送資料時，傳送狀態由 TXRX_INIT 進入 TX_SEND，一般情況下，傳送動作在此狀態下運作，圖 9 為資料傳送示意圖。若傳送端沒有收到 SACK 且等待逾時(timeout)，或是偵測到路由失效(route failure)，傳送狀態由 TX_SEND 進入 TX_DETECT。此時，停止資料傳送，每隔探測週期(detect period)傳送一個探測封包 PRB，探測週期要比傳送週期長，探測訊息傳遞如圖 10 所示。若收到接收端回應的 PRB|ACK 或 SACK，回到 TX_SEND 狀態，繼續資料傳送；若在此狀態超過一段長時間，則認定為連線中斷，進入終止狀態 TXRX_FIN。

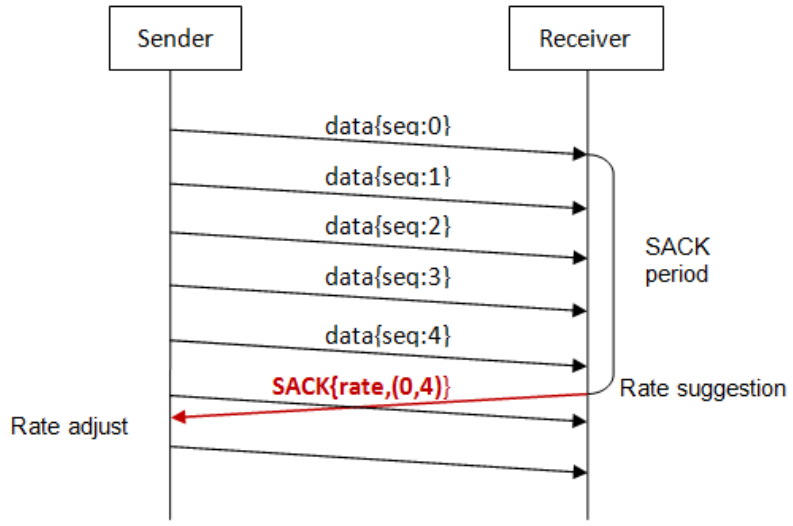


圖 9：資料傳輸示意圖

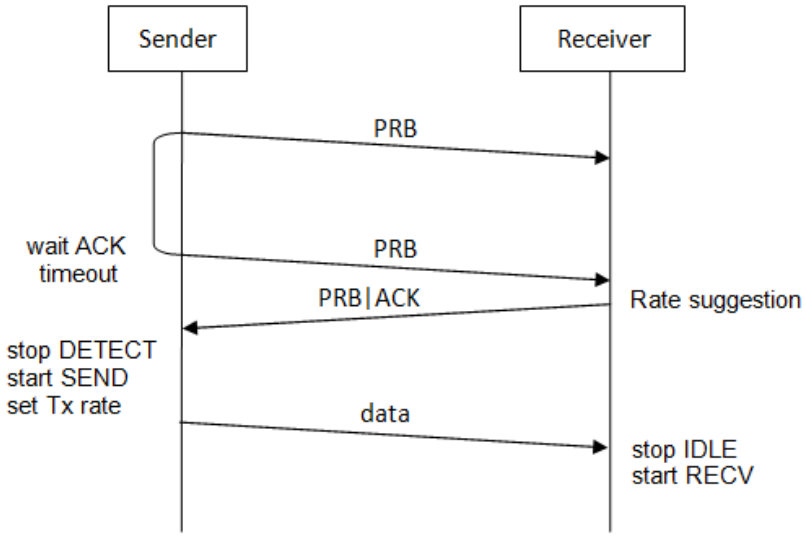


圖 10：網路探測訊息流

傳送速率的調整時機在收到 SACK 的時候，此外，當傳送狀態從 TX_DETECT 回到 TX_SEND 時，會根據 PRB|ACK 中的建議速率來調整傳送速率。傳送速率的調整方法在 3.7 有詳細說明。

資料傳送模式可分為正常傳送(Normal Tx)、遺失重傳(ReTx loss)、全部重傳(ReTx all)。其中，全部重傳模式啟動的時機在傳送狀態由 TX_DETECT 回到 TX_SEND 的時候，目的在於傳送端必須主動補上短暫傳輸中斷期間(狀態停留在 TX_DETECT 期間)未成功送出的資料封包。

當應用程式要求關閉連線，若傳送狀態在 TX_SEND 或 TX_DETECT，則進入 TX_FIN_WAIT 狀態。此時，不再接受應用程式的傳送要求，若 queue 中還有封包，繼續做資料傳送，直到 queue 完全清空為止。若因為連線中斷而無法將資料送出，持續做重傳動作，直到逾時(timeout)才放棄，進入終止狀態 TXRX_FIN。

3.5 資料接收 (Data Reception)

傳輸層收到封包後，會根據封包型態做不同的處理。新進資料封包的處理如下：比較封包序號和預期封包序號(expected sequence number)。若封包序號等於預期封包序號，將該資料封包的內容(payload)存放在 RxQueue 中等待應用程式讀取；若封包序號在預期封包序號之後，將該資料封包暫存，等待排序；若封包序號在預期封包序號之前，表示這是重複的封包，直接丟棄，不處理。預期封包序號的公式為：

Expected SeqNum = (RxQueue 中最後一個資料的 SeqNum + 1) mod 65536

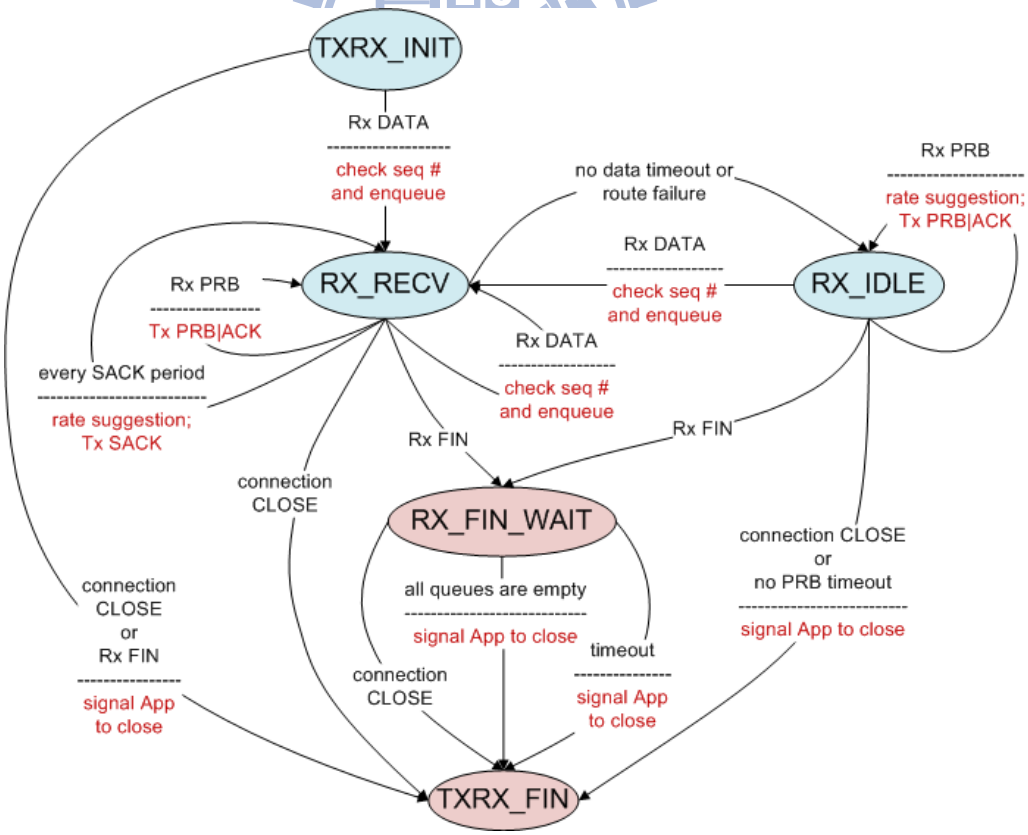


圖 11：接收狀態圖 (Rx State Diagram)

如圖 11 所示，接收行為由接收狀態機(Rx State Machine)控制。當接收端收到第一個資料封包時，接收狀態從 TXRX_INIT 進入 RX_RECV，一般情況下，接收動作在此狀態下運作。接收端每隔 SACK 週期(SACK period)送一個 SACK 給傳輸端，內容包括要 ACK 的封包序號，即確定收到的封包序號，以及用於流量控制的建議速率，圖 9 為示意圖。若接收端一段時間沒有收到資料封包，接收狀態由 RX_RECV 進入 RX_IDLE，此時，停止傳送 SACK，改為被動回應傳送端。若收到 PRB，回送 PRB|ACK，其中包含建議速率；若收到資料封包，接收狀態回到 RX_RECV，恢復資料接收以及 SACK 機制。上述動作的訊息流如圖 10 所示。

當收到 FIN，表示對方先關閉連線，若接收狀態在 RX_RECV 或 RX_IDLE，則進入 RX_FIN_WAIT 狀態。此時，停止 SACK 機制，等待 queue 中的資料消化完畢後，進入終止狀態 TXRX_FIN。若經過一段時間 queue 的長度仍未降低，則放棄等待，進入終止狀態 TXRX_FIN。

3.6 SACK 和重傳機制

封包遺失回復(packet loss resilient)能力仰賴 SACK 和重傳機制的配合，接收端定時送 SACK 回報已收到的封包序號，傳送端根據此資訊判斷是否發生封包遺失，若有遺失，啟動遺失重傳(reTx loss)模式。

這裡講的「遺失」意思是傳輸層該收到卻沒有收到的情況，有可能是封包在傳送的路徑上被丟棄，或是因封包出現錯誤導致接收端 MAC 層無法辨認。

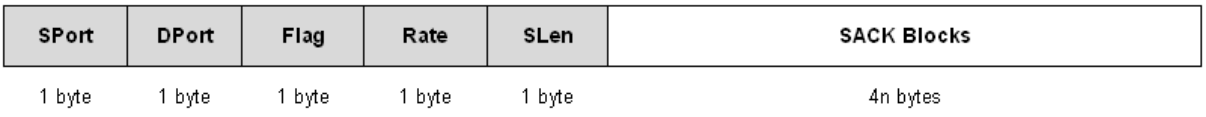


圖 12 : SACK 封包格式

SACK 封包格式見圖 12，參考 TCP Selective ACK [8] 的設計，SACK Blocks 存放

要 ACK 的封包序號，每一個 SACK block 代表一段連續的封包序號，以 (left seq , right seq) 表示，left seq 是該 block 中排列最前面的封包序號，right seq 是該 block 中排列最後的封包序號。因為本協定用 2 bytes 表示封包序號，所以一個 SACK block 的長度為 4 bytes。SACK 封包中的 SLen 欄位是 SACK block 個數，根據此值可以判斷是否有封包遺失，像是 SLen 大於 1 的情況，表示封包序號不連續，可看出此 SACK 週期內有遺失封包。

在 3.4 已說明傳送端如何做資料傳送，這裡著重在資料重傳。傳送端收到 SACK 後，先檢查 SACK Blocks，並把接收端已收到的資料封包從 queue 中刪除，剩下已送出尚未 ACK 的資料封包中，有可能是此 SACK 週期之後才送出的封包，也可能是在此 SACK 週期內遺失的封包。為了正確辨識，每次資料傳送時紀錄最後送出的封包序號(last out seq)，每次檢查 SACK Blocks 時紀錄最後一個 block 的 right seq，即最後一個 ACK 的封包序號(last ACK seq)。

遺失重傳(reTx loss)時，傳送 queue 中封包序號在 last ACK seq 以前的資料封包。全部重傳(reTx all)時，傳送 queue 中封包序號在 last out seq 以前的資料封包。當該重傳的封包已全部送出，或是 queue 為空，傳送模式回到正常傳送(normal Tx)。重傳的優先權高於正常傳送，在符合傳送速率的限制下，完成重傳後會接著傳送新的資料封包。

3.7 基於速率的流量控制 (Rate Based Flow Control)

傳送端和接收端合作完成流量控制。接收端根據應用程式回應延遲(application response latency)和 queue 的長度決定建議速率(suggest rate)，並透過 SACK 送給傳送端。傳送端根據下層的傳輸延遲(Tx latency of lower layers)和來自 SACK 的建議速率來調整傳送速率(transmission rate)。以下分別說明接收端和傳送端的動作。

◆ 接收端

接收端流量控制的方法如表 1 所示。在連線起始階段(INIT)，起始速率(initial rate)可以由應用程式設定；若無設定，使用預設的起始速率，值同 RxQueue 的容量。接收端

將起始速率建議值透過 SYN|ACK 送給傳送端。

T_{app} : application response latency [應用程式回應延遲]
T_{appA} : exponential moving average of T_{app} [T_{app} 的指數移動平均]
ER_{app} : equivalent rate of T_{appA} [T_{appA} 對應的速率]
R_{Rx} : data packet receive rate [資料封包接收速率]
R_s : suggest rate [建議速率]
R_{init} : initial rate [起始速率]
L_{RQ} : length of RxQueue (amount of pkts)
[RxQueue 的長度，表示累積封包數量]
S_{RQ} : total size of RxQueue [RxQueue 的容量]

[INIT]
1 Assign the value of S_{RQ} to default R_{init} : default $R_{init} \leftarrow S_{RQ}$
2 If application set R_{init} , $R_s \leftarrow R_{init}$
3 else $R_s \leftarrow$ default R_{init}
4 Get the projected latency T_{appA} from R_s as the initial value of EMA

[ESTABLISH]
Every application response :
5 Compute EMA : $T_{appA} \leftarrow \alpha T_{appA} + (1-\alpha) T_{app}$ // $\alpha = 7/8 = 0.875$

Every SACK :
6 Get the equivalent rate ER_{app} from T_{appA}
7 case $L_{RQ} > \beta S_{RQ}$: // $\beta = 1/2 = 0.5$
8 Rate Down : $R_s \leftarrow \min(\delta R_{Rx}, ER_{app})$ // $\delta = 1/2 = 0.5$
9 case $L_{RQ} < \gamma S_{RQ}$: // $\gamma = 1/4 = 0.25$
10 Normal : $R_s \leftarrow ER_{app}$
11 case $\gamma S_{RQ} \leqq L_{RQ} \leqq \beta S_{RQ}$:
12 still Rate Down : $R_s \leftarrow \min(maintain\ R_s, ER_{app})$
13 still Normal : $R_s \leftarrow ER_{app}$

表 1：接收端流量控制方法 (虛擬程式碼，pseudo code)

在建立階段(ESTABLISH)，每次應用程式讀取資料時測量回應延遲，並計算其指數移動平均(exponential moving average，EMA)。這裡的回應延遲(response latency)指的是

應用程式從取得資料開始到下次讀取 RxQueue 的時間間隔。此數值反映應用程式處理資料的速度。

每隔 SACK 週期做速率建議(rate suggestion)，可分為正常模式(Normal)和速率調降模式(Rate Down)，預設為正常模式。方法如表 1 的 6-13 行和圖 13 所示。

正常模式中，建議速率直接從應用程式回應延遲求得，可視為應用程式的回應速度。速率調降模式在 RxQueue 的長度超過自身容量的 1/2 時啟動，把速率降低為資料封包接收速率的 1/2，速率減半的用途相當於緊急剎車，避免 queue 成長太快。若下次做速率估計時 queue 的長度仍過半，再降低速率，直到 queue 的長度不超過自身容量的一半才不再調降。在速率調降模式中，取調降速率和應用程式回應延遲對應速率中較低者為建議速率。當 RxQueue 的長度小於自身長度的 1/4 時，回到正常模式。

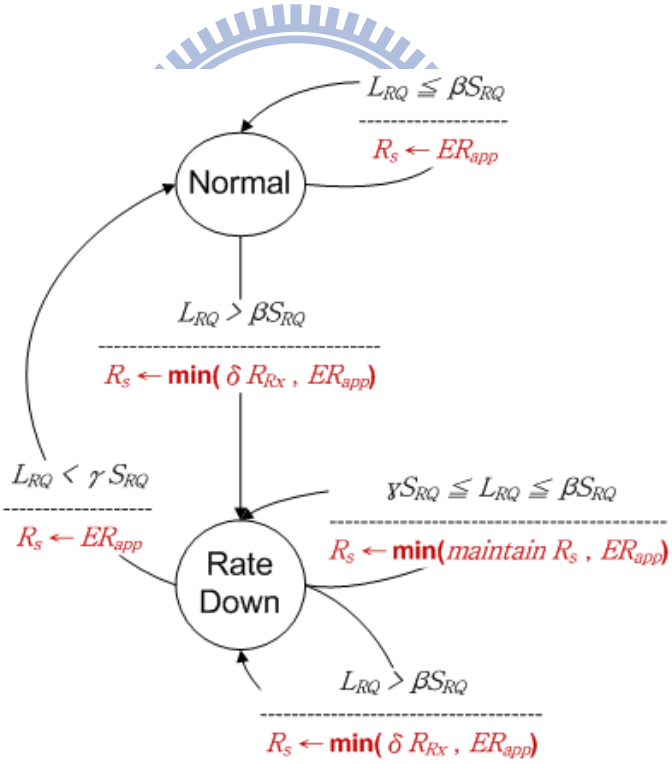


圖 13：速率建議(Rate Suggestion)狀態圖

◆ 傳送端

傳送端流量控制的方法如表 2 所示。在連線起始階段(INIT)，傳送端送出 SYN 時測

量傳輸延遲(Tx latency)。收到 SYN|ACK 後，取建議速率和傳輸延遲的對應速率中較小者為起始傳送速率。

T_{tl} : transmission latency of lower layer [下層的傳輸延遲]
T_{tlA} : exponential moving average of T_{tl} [T_{tl}的指數移動平均]
ER_{tl} : equivalent rate of T_{tlA} [T_{tlA}對應的速率]
R_{Tx} : data packet transmission rate [資料封包傳送速率]
R_s : suggest rate [建議速率]
R_{init} : initial rate [起始速率]
R_b : base rate in flow control [流量控制方法中的基準速率]
R_m : margin rate in flow control [流量控制方法中的邊界速率]

[INIT]
1 Assign the Tx latency of SYN T_{tl} to T_{tlA} as the initial value of EMA : $T_{tlA} \leftarrow T_{tl}$
2 Get the equivalent rate R_{init} from T_{tlA}
3 Get R_s from SYN|ACK
4 $R_{Tx} \leftarrow \min(R_{init}, R_s)$

[ESTABLISH]
Every Tx :
5 Compute EMA : $T_{tlA} \leftarrow \alpha T_{tlA} + (1-\alpha)T_{tl}$ // $\alpha = 7/8 = 0.875$

When receive SACK :
6 Get R_s from SACK
7 Get the equivalent rate ER_{tl} from T_{tlA}
8 $R_b \leftarrow \min(R_s, ER_{tl})$
9 $R_m \leftarrow \delta R_{Tx}$ // $\delta = 1/16 = 0.0625$
10 case $R_{Tx} > R_b + R_m$:
11 Decrease : $R_{Tx} \leftarrow R_b$
12 case $R_{Tx} < R_b - R_m$:
13 Increase : $R_{Tx} \leftarrow R_{Tx} + K(R_b - R_{Tx})$ // $K = 1/2 = 0.5$
14 case $R_b - R_m \leqq R_{Tx} \leqq R_b + R_m$:
15 No Change : maintain R_{Tx}

表 2：傳送端流量控制方法 (虛擬程式碼，pseudo code)

在建立階段(ESTABLISH)，每次資料傳送時測量下層的傳輸延遲，並計算其指數移動平均(EMA)。這裡的傳輸延遲(Tx latency)指的是從傳輸層將資料封包送到下層開始，到確定封包已送出為止的時間間隔。此數值反映傳送端傳輸層之下各層的處理速度以及 radio 傳輸的速度。

當收到 SACK 時，取建議速率和傳輸延遲的對應速率中較小者為基準速率，接著進行傳送速率調整。分為三個模式：提高模式(Increase)、降低模式(Decrease)、不變模式(No Change)。配合表 2，可以簡單表示三個傳送速率調整模式：

Decrease (降低模式)：當 $R_{Tx} > R_b + R_m$ ，調整 $R_{Tx} = R_b$

Increase (提高模式)：當 $R_{Tx} < R_b - R_m$ ，調整 $R_{Tx} = R_{Tx} + K(R_b - R_{Tx})$

No Change (不變模式)：當 $R_b - R_m \leq R_{Tx} \leq R_b + R_m$ ，不調整

如表 2 中的 10-11 行所示，若傳送速率大於基準速率加邊界速率，進入降低模式(Decrease)，以基準速率為新的傳送速率；如 12-13 行所示，若傳送速率小於基準速率減邊界速率，進入提高模式(Increase)，增加傳送速率；若傳送速率介在兩者之間(表 2 的 14-15 行)，為不變模式(No Change)，維持原速率。其中，邊界速率為 1/16 倍傳送速率，使用邊界速率的目的在於減少速率波動的影響，維持傳送速率的穩定。

除了資料傳輸過程中的流量控制外，當資料傳輸從短暫中斷恢復正常時，更新傳送速率。在 3.4 和 3.5 有提到資料傳送的 DETECT 狀態和資料接收的 IDLE 狀態，在這樣的狀態下，傳送端定時送出 PRB 來探測網路狀況。若傳輸路徑恢復暢通，接收端會收到 PRB，接著執行速率建議程序，方法同表 1 中 6-15 行，建議速率透過 PRB|ACK 送給傳送端；傳送端收到 PRB|ACK 後，進行傳送速率更新，方法同表 2 中 1-4 行，但第 1 行的 Tx latency 是 PRB 的傳送延遲。

第 4 章　實作與分析

4.1　在 SunSPOT 上實作 WPAN-ATP

如圖 14 所示，WPAN-ATP 建構在 SunSPOT 網路協定堆疊(network protocol stack)的 lowpan 層之上，和 SunSPOT 提供的 Radiostream 在同一層。

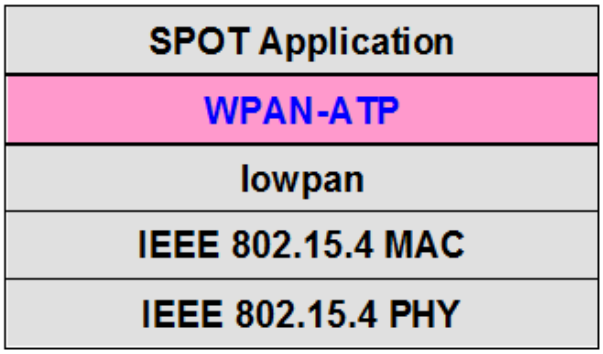


圖 14：WPAN-ATP 在 SunSPOT 的網路協定堆疊

有關實作，這裡不在細節著墨太多，只列出以下幾個重點：

(1) 協定的運作以連線狀態機(Connection State Machine)、傳送狀態機(Tx State Machine)、接收狀態機(Rx State Machine)為基礎，加上 SACK 和重傳機制、流量控制方法。圖 15 為 WPAN-ATP 實作示意圖，簡要說明如下：資料傳送時，TxQueue 暫存要送出的資料封包，OutputHandlet 負責進行資料封包的傳送和重傳，ReTxQueue 暫存已送出尚未 ACK 的資料封包。資料接收時，先檢查新進封包的型態，不同的型態有不同的處理。資料封包要經過封包序號檢查程序，不符合次序的封包被暫存在 OrderingQueue 中等待排序，符合次序的封包之內容(payload)暫存在 RxQueue 中等待應用程式讀取。傳送週期和探測週期由 Tx Timer 控制，SACK 週期由 SACK Timer 控制，計時等待時間以及逾時(timeout)通知由 Wait Timer 負責。

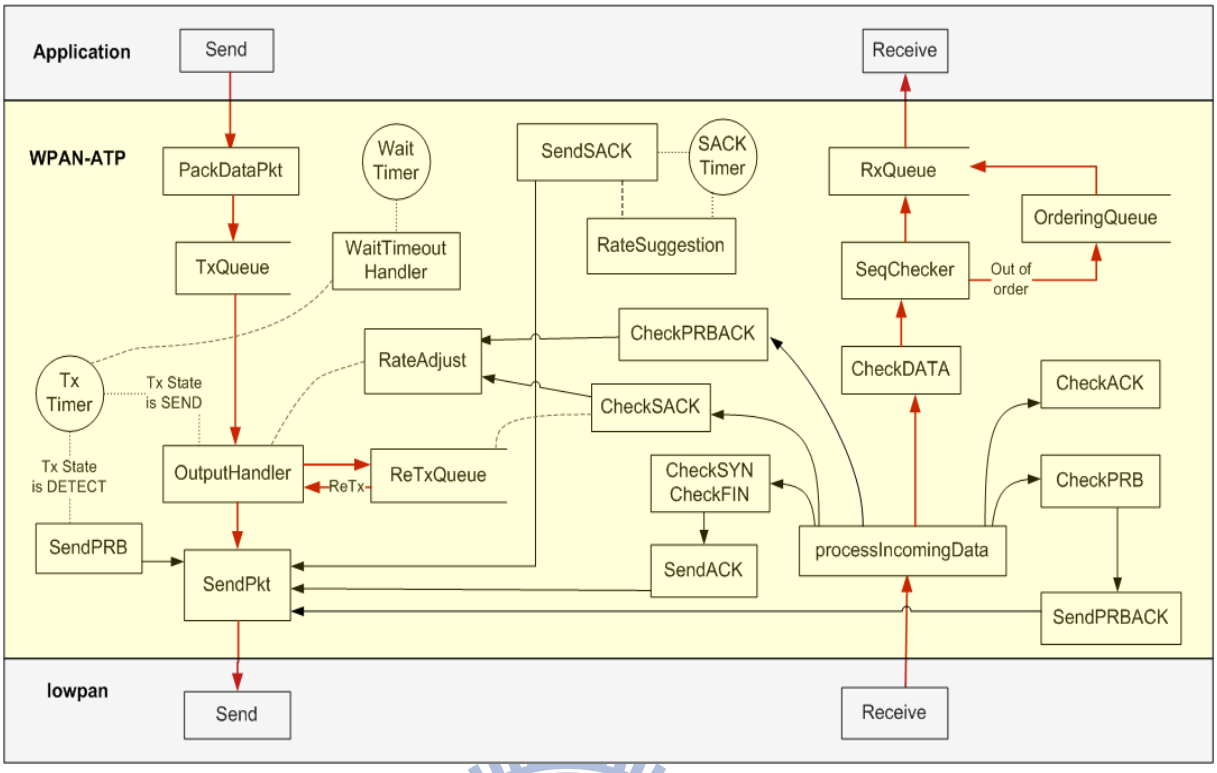


圖 15：WPAN-ATP 實作示意圖

(2) 協定中幾個重要的參數如下表：

佇列容量 (單位：pkts)		週期 (單位：ms)	
TxQueue	32	SEND_PERIOD (傳送週期)	1
ReTxQueue	128	SACK_PERIOD (SACK 週期)	250
RxQueue	32	DETECT_PERIOD (探測週期)	250
OrderingQueue	128	等待時間 (單位：ms)	
		SACK_WAIT_TIME	2000

表 3：實作中設定的參數

(3) 圖 16 是 SunSPOT radio 封包示意圖，radio 封包長度為 128 bytes，IEEE 802.15.4 MAC frame 最大長度為 127 bytes，lowpan 封包最大長度為 102 bytes，WPAN-ATP 資料封包的最大長度為 79 bytes，其中，封包表頭長度為 5 bytes。

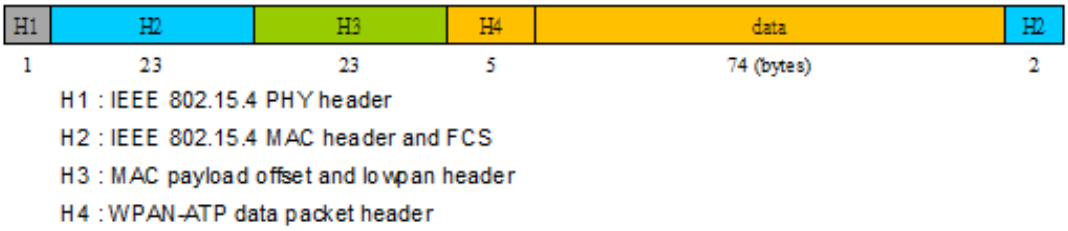


圖 16 : SunSPOT radio packet

(4) 根據上面的封包格式，圖 12 中的 SACK Blocks 可容納的 block 個數之最大值為 Floor(74/4) = 18。長度為 4 × 18 = 72 bytes。

```
1   WPANATPConnectionImpl server_con, client_con;
2   WPANATPConnectionID server_cid, client_cid;
3   DataOutputStream dos;
4   DataInputStream dis;
5
6   // server
7   server_con = (WPANATPConnectionImpl) Connector.open("wpanatp://:" + PORT);
8   server_cid = server_con.accept();
9   dis = server_con.openDataRxStream(server_cid);
10  try {
11      dis.read();
12  } catch(NoRouteException e) {
13  } catch(ShutdownException e) { }
14
15  // client
16  client_con = (WPANATPConnectionImpl) Connector.open("wpanatp://" +
    Dest_Address + ":" + SERVER_PORT);
17  client_cid = client_con.request();
18  dos = client_con.openDataTxStream(client_cid);
19  try {
20      dos.write(data);
21  } catch(NoRouteException e) {
22  } catch(ShutdownException e) { }
```

表 4：使用 WPAN-ATP 的應用程式範例

(5) WPAN-ATP 的開發環境為 J2ME，遵守 J2ME Generic Connection Framework (GCF)，所以應用程式可以透過通用的方法使用 WPAN-ATP。如表 4 的第 7 行和第 13 行所示，使用 Connector 來開啟連線。

(6) 如表 4 所示，Server Connection 透過呼叫 accept() 開始被動等待連線(圖 5 的 passive OPEN)，並使用 DataInputStream 接收資料；Client Connection 透過呼叫 request() 開始主動要求連線(圖 5 的 active OPEN)，並使用 DataOutputStream 傳送資料。

(7) 每個連線使用連線 ID (表 4 中的 WPANATPConnectionID) 來做辨識，連線 ID 存放來源地址(source address)、來源埠(source port)、目的地址(destination address)、目的埠(destination port)、連線型態(Server/Client)等資訊。

(8) 應用程式可以接收到 WPAN-ATP 發出的事件，像是路由失效(route failure)或是連線關閉通知(close signal)這類的事件。如表 4 所示，路由失效時，應用程式可以收到 NoRouteException；連線要關閉時，會收到 ShutdownException。

4.2 WPAN-ATP 行為分析

在 SunSPOT 上實作 WPAN-ATP 後，接著進行實驗來觀察協定的行為。觀察重點有二：基於速率的流量控制(rate-based flow control)、封包遺失回復(packet loss resilient)。

在實際環境中，使用 SunSPOT 進行傳送接收實驗，網路拓撲由一個感測裝置和一個基站構成，進行單一連線傳輸。兩裝置距離在一公尺以內，電池電量充足，所以距離和電力因素可以忽略。

✧ 第一部分實驗：觀察「基於速率的流量控制」。

傳送端每次送出的資料長度為 74 bytes，可以填滿 WPAN-ATP 資料封包。

傳送端應用程式的傳送時間間隔固定，接收端應用程式設定三個不同的回應延遲。

- 實驗 1-1：傳送時間間隔設為 25 ms，接收回應延遲設為 8 ms。
- 實驗 1-2：傳送時間間隔設為 25 ms，接收回應延遲設為 25 ms。

■ 實驗 1-3：傳送時間間隔設為 25 ms，接收回應延遲設為 50 ms。

觀察重點：

1. 傳送端/接收端在延遲(delay/latency)和 queue 長度之變化與關係。
2. 接收端如何根據應用程式回應延遲(application response latency)和 RxQueue 的長度來決定建議速率(suggest rate)。
3. 傳送端如何根據 lowpan 傳輸延遲(lowpan Tx latency)和接收端建議速率(receiver suggest rate)來調整傳送速率(transmission rate)。
4. 傳送速率和傳送端傳輸量(throughput)的關係。

圖示說明：

1. 以圖 17 中的 application response latency (unit: ms) (EMA) [left axis] 為例。此為應用程式回應延遲的指數移動平均(EMA)，以毫秒(ms)為單位，使用左縱軸。其他以此類推。
2. 圖的橫軸以秒(sec)為單位。理論上，每一個 SACK 週期間隔 250 ms (實際上，每一個 SACK 的間隔會有誤差，只能達到平均間隔為 250 ms)，所以每一秒內包含 4 個 SACK 週期。除了傳輸量(throughput)的資料每秒取樣一次之外，其他資料皆每 SACK 週期取樣一次。

本實驗中定義：

Rx propagation delay：

從資料封包進入傳輸層開始，到資料存入 RxQueue 為止的時間間隔。此數值反映 WPAN-ATP 處理接收封包的速度。

Tx propagation delay：

從應用程式傳送資料開始，到資料封包下送到 lowpan 層為止的時間間隔。此數值反映 WPAN-ATP 處理傳送封包的速度，以及 TxQueue 的 queuing delay。

lowpan Tx latency：

從資料封包下送到 lowpan 層開始，到 lowpan 回應傳送成功為止的時間間隔，

此測量值就是 3.7 流量控制方法中的傳輸延遲。

✧ 第二部分實驗：觀察「封包遺失回復」。

因為不容易做到實際訊號干擾，所以使用程式隨機丟掉封包來模擬封包遺失。這裡的封包遺失是雙向的，也就是資料封包和 SACK 都有同樣的遺失率。

傳送端應用程式的傳送時間間隔設為 25 ms，接收端應用程式的回應延遲設為 8 ms。表示接收端應用程式的處理速度高於傳送端應用程式的傳送速度。

■ 實驗 2-1：封包遺失率設為 0.20

觀察重點：

1. 從傳送端送出的封包序號(sequence number)看到資料封包重傳。
2. 傳送端的資料封包重傳和接收端的資料封包排序。
3. 傳送端 queue 的長度變化，尤其是 ReTxQueue 的封包累積。
4. 接收端 queue 的長度變化，尤其是 OrderingQueue 的封包累積。以及封包序號和 OrderingQueue 長度的變化與關係。

圖示說明：

圖的橫軸以秒(sec)為單位。除了送出的封包序號每次傳送取樣一次之外，其他皆每 SACK 週期取樣一次。

本實驗中定義：

last Rx sequence number：

每次取樣時，到該時間點為止，最後接收到的封包序號。

last sequence number in RxQueue：

每次取樣時，到該時間點為止，RxQueue 中最後一個封包序號。在此序號以前的封包已排好順序。

(實驗1-1) 傳送時間間隔：25 ms，接收回應延遲：8 ms。

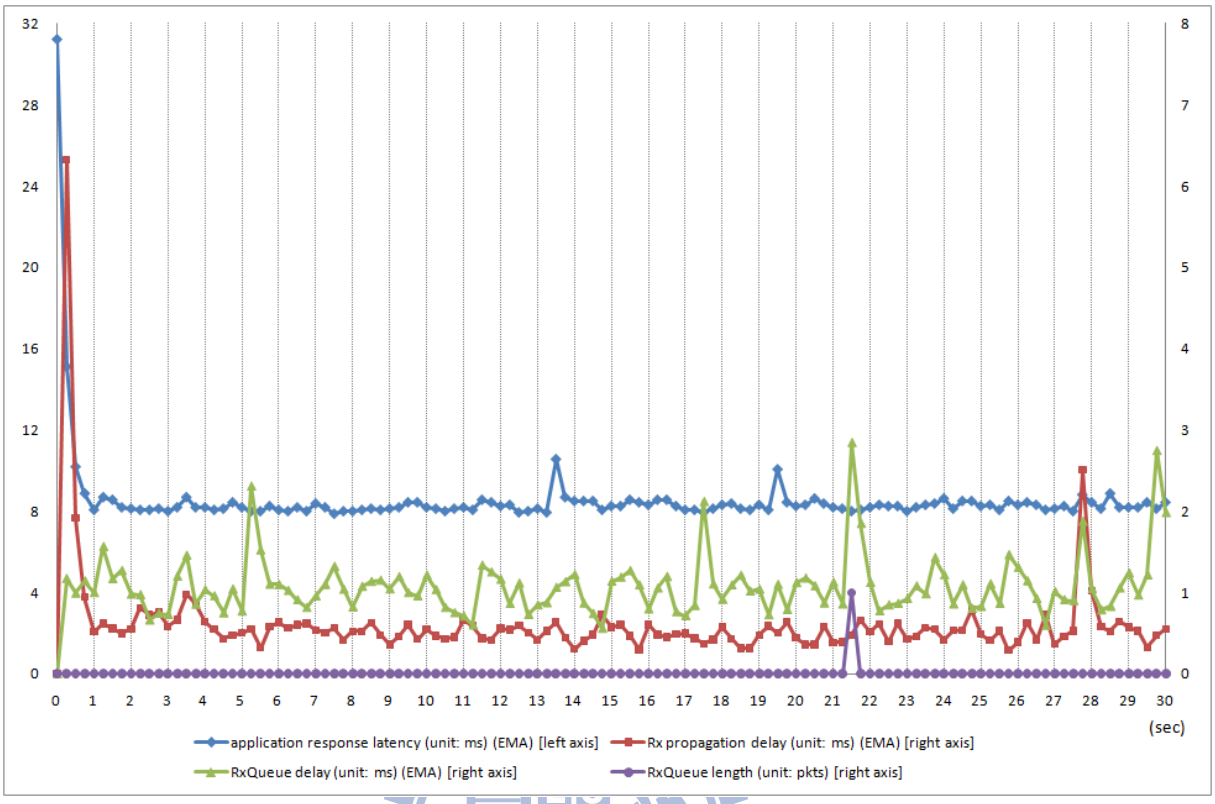


圖 17 : [實驗 1-1] 接收端的延遲(latency/delay)和佇列(queue)

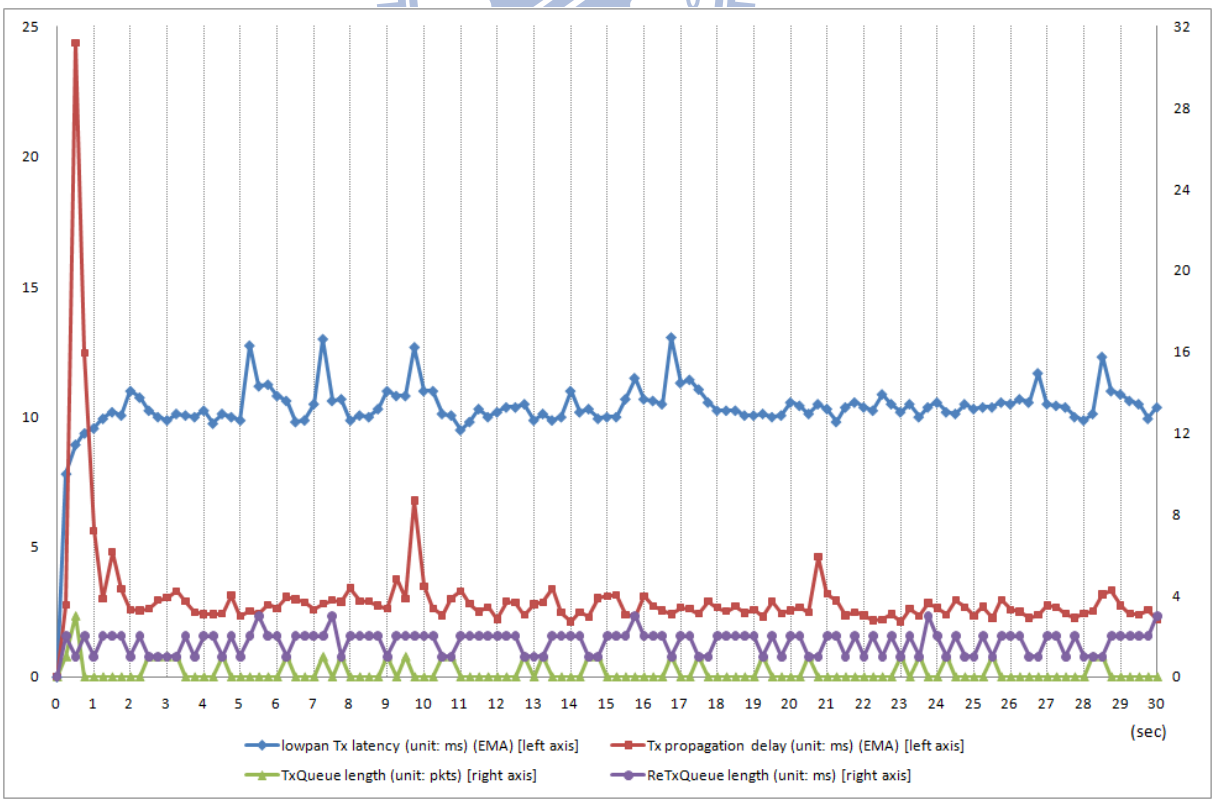


圖 18 : [實驗 1-1] 傳送端的延遲(latency/delay)和佇列(queue)

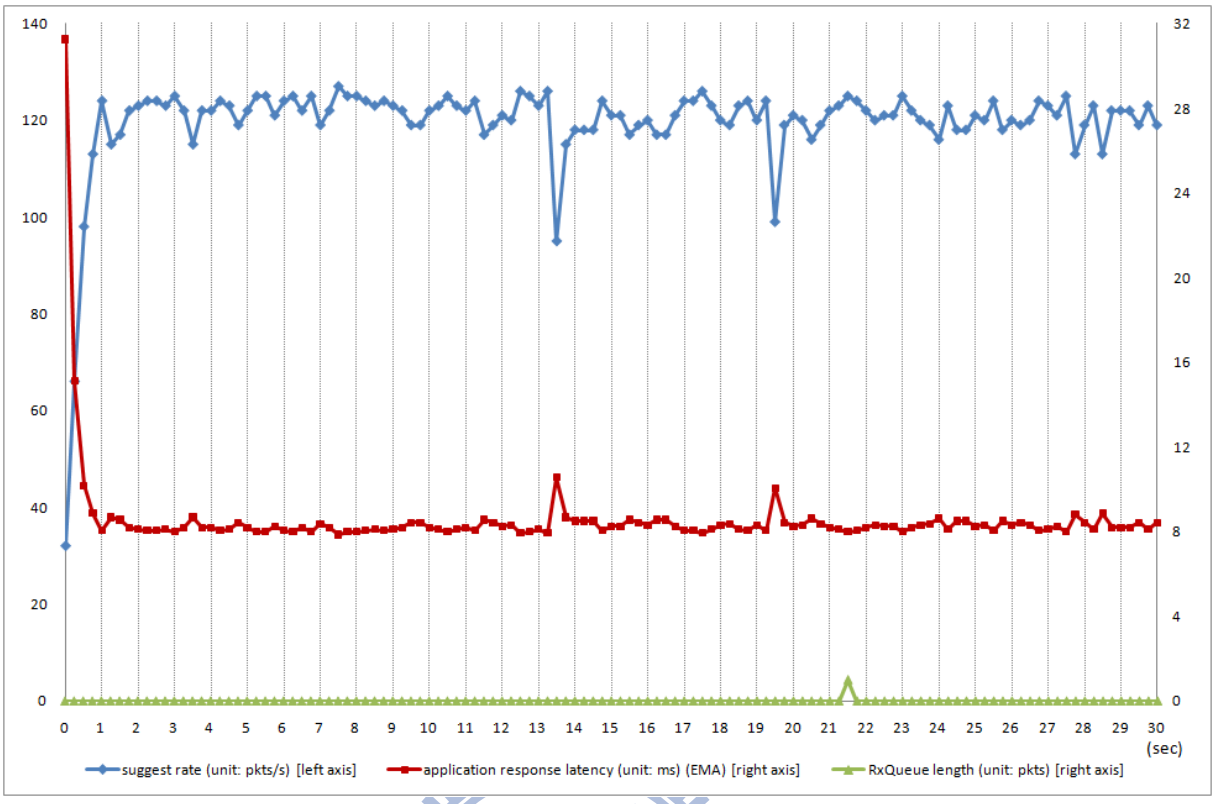


圖 19 : [實驗 1-1] 建議速率(suggest rate)的決定

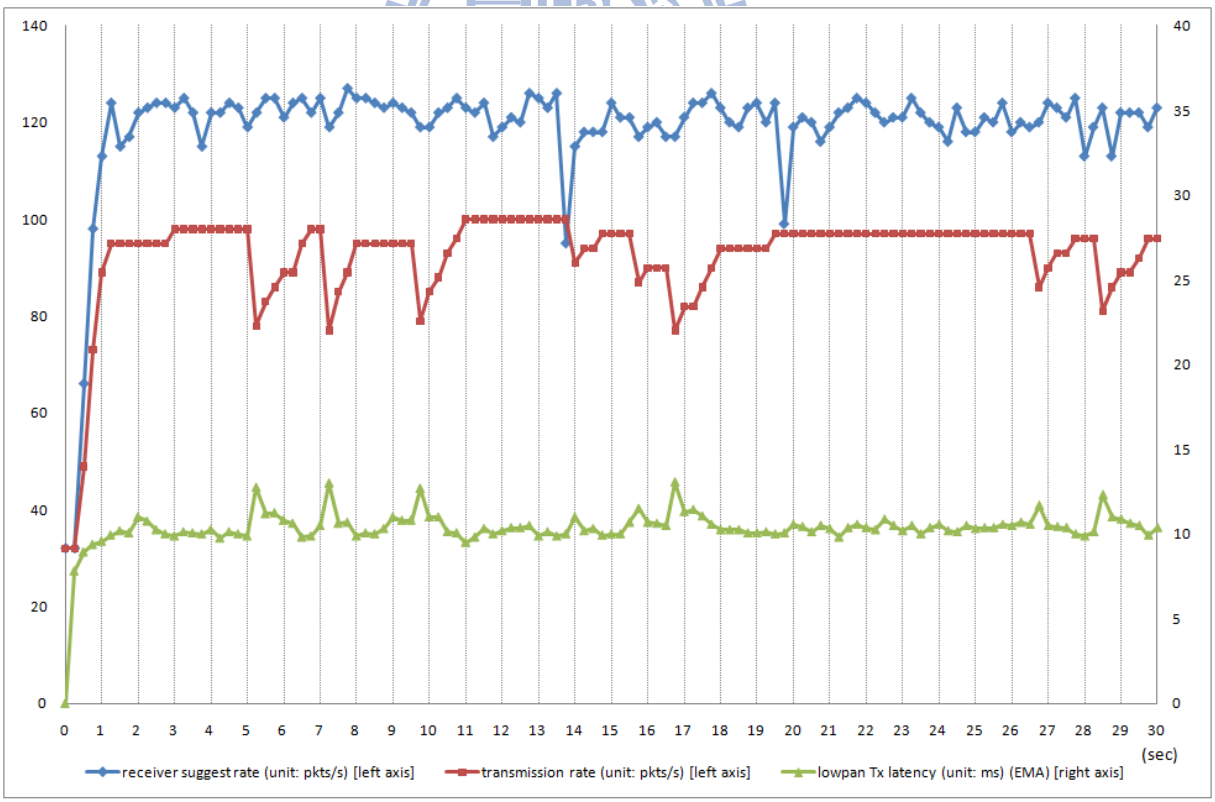


圖 20 : [實驗 1-1] 傳送速率(transmission rate)的調整

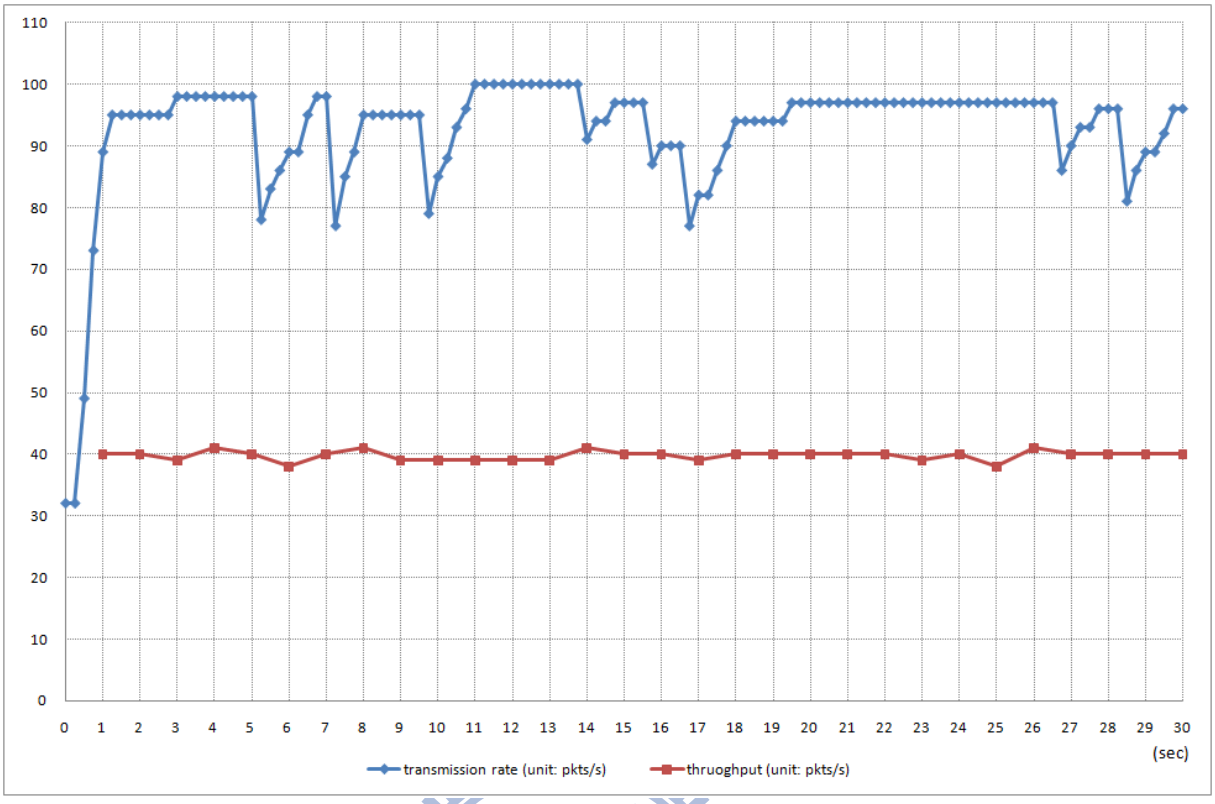


圖 21 : [實驗 1-1] 傳送端的傳輸量(throughput)和傳送速率

分析：

(1) 從圖 17 可見，由於接收端的處理速度夠快，所以延遲情況變化不大，而且 queue 的長度在大部分的時間為0。開頭的高起是起始速率所致，起始速率預設值為 32 pkts/s，對應的延遲值為 1000 / 32 = 31.25。

(2) 從圖 18 和圖 21 可見，由於傳送速率(transmission rate)高於應用程式的傳送速度，所以 TxQueue 長度不會大幅成長。ReTxQueue 長度維持在 2 上下是因為 queue 中有已送出尚未 ACK 的資料封包，若有封包遺失的情況，ReTxQueue 長度會較顯著成長。

(3) 根據 3.7 中的接收端流量控制方法，根據應用程式回應延遲(application response latency)和 RxQueue 的長度來決定建議速率(suggest rate)。從圖 19 中可見，在接收端處理速度較快的情況下，queue 的長度大多為 0，建議速率直接由應用程式回應延遲主導，即表 1 流量控制中的正常模式(Normal)。

(4) 根據 3.7 中的傳送端流量控制方法，根據下層的傳輸延遲(Tx latency)和接收端的建議

速率(receiver suggest rate)來調整傳送速率。從圖 20 可見，接收端建議速率高於 lowpan 傳輸延遲(lowpan Tx latency)的對應速率，傳送速率取二者中較小者，也就是傳送速率由 lowpan 傳輸延遲主導。這表示傳送速率的瓶頸在傳送端。

(5) 從圖 21 中可見，傳送速率表示資料傳送的最高速度限制，當傳送端應用程式傳送的頻率不高時，實際的傳輸量(throughput)會低於傳送速率。

(實驗1-2) 傳送時間間隔：25 ms，接收回應延遲：25 ms。

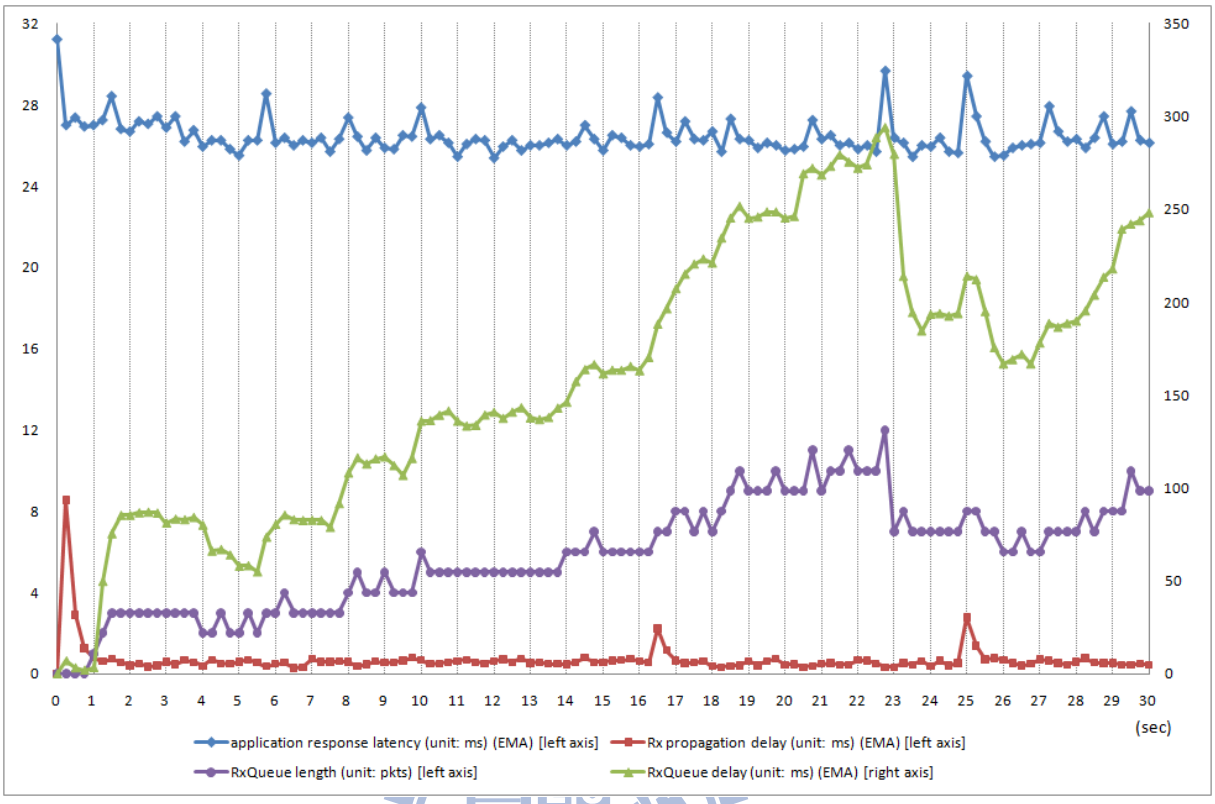


圖 22 : [實驗 1-2] 接收端的延遲(latency/delay)和佇列(queue)

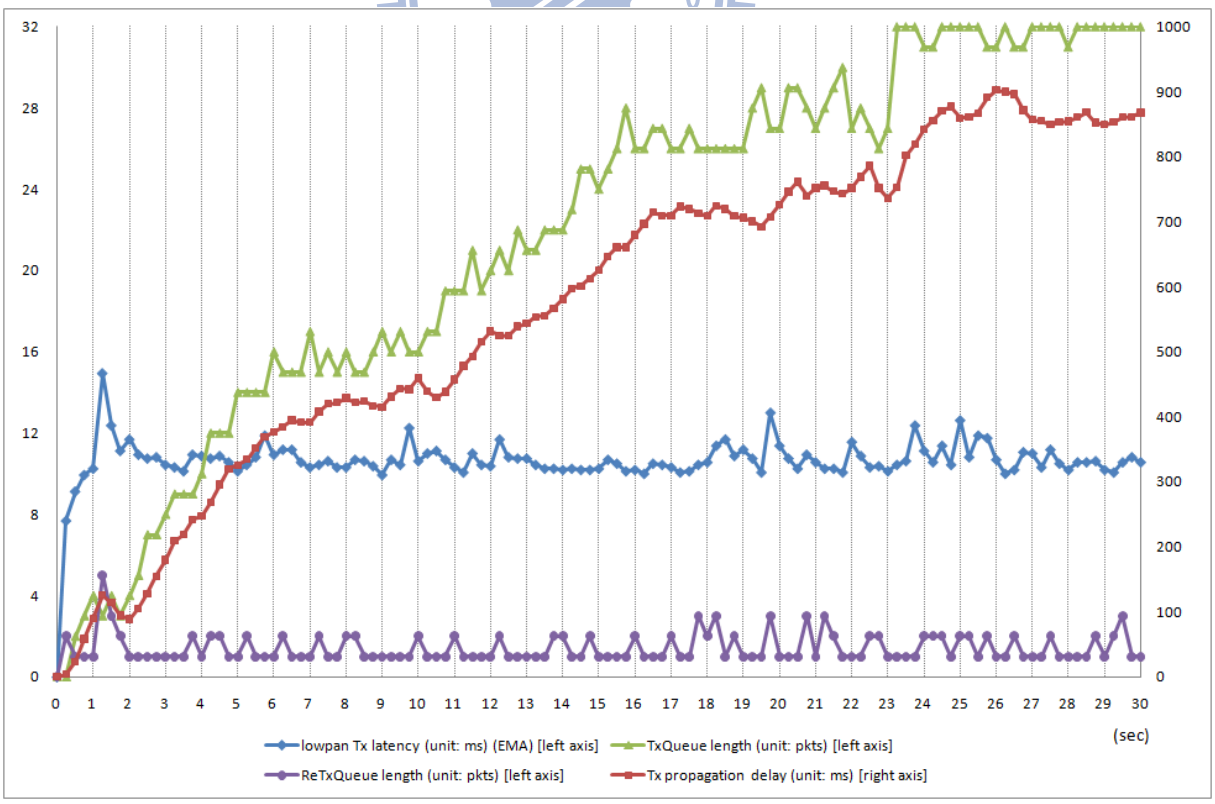


圖 23 : [實驗 1-2] 傳送端的延遲(latency/delay)和佇列(queue)

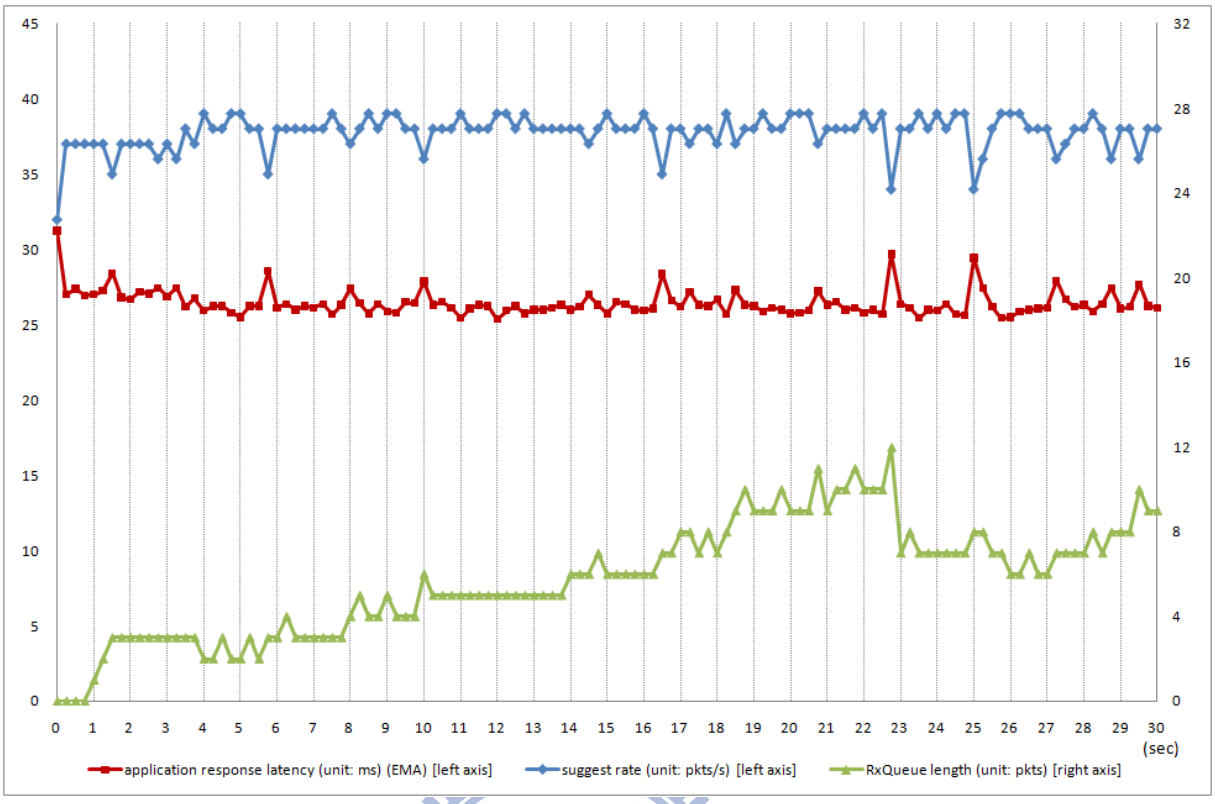


圖 24 : [實驗 1-2] 建議速率(suggest rate)的決定

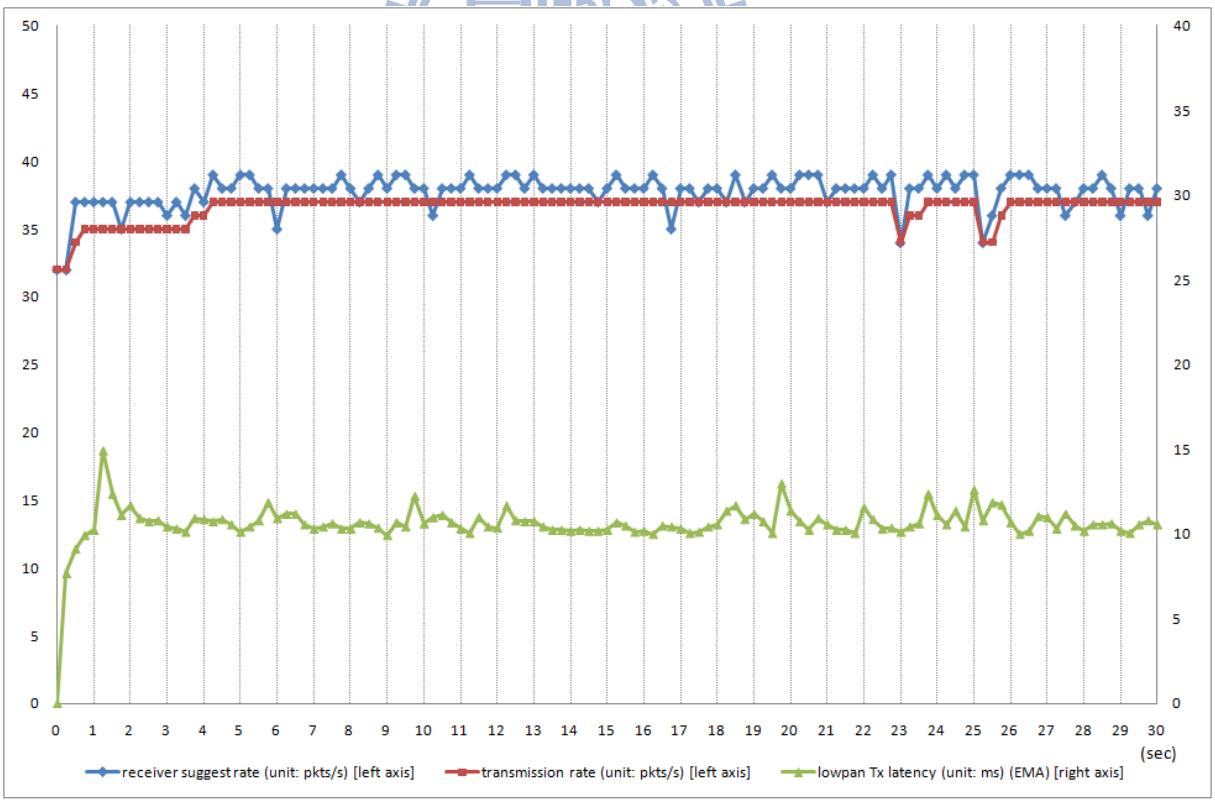


圖 25 : [實驗 1-2] 傳送速率(transmission rate)的調整

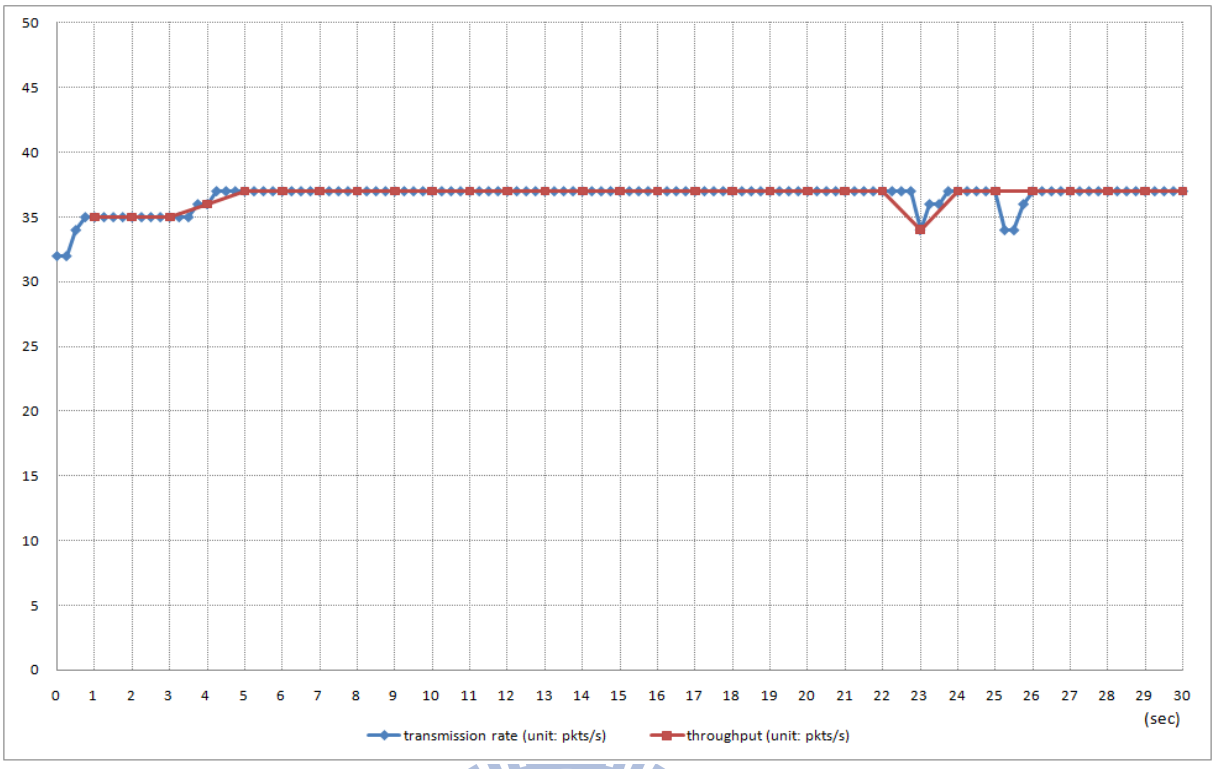


圖 26 : [實驗 1-2] 傳送端的傳輸量(throughput)和傳送速率

分析：

(1) 從圖 22 可見，RxQueue 有慢慢累積封包的現象，表示接收端應用程式的處理速度稍小於資料封包到達的速率。RxQueue delay 會隨著 queue 長度增加而變長。

(2) 從圖 23 可見，Tx propagation delay 曲線和 TxQueue length 曲線有些相似，表示造成 Tx propagation delay 增加的主因是 TxQueue 的佇列延遲(queuing delay)。

(3) 從圖 24 可見，當 RxQueue 的長度在自身容量的一半以下時，建議速率(suggest rate)由應用程式回應延遲(application response latency)主導。

(4) 從圖 25 可見，接收端建議速率低於 lowpan 傳輸延遲(lowpan Tx latency)的對應速率，所以傳送速率(transmission rate)由接收端建議速率主導。這表示資料傳輸的瓶頸在接收端。此外，傳送速率相較於建議速率來得平穩，表示表 2 中的邊界速率(margin rate)發揮作用。

(5) 從圖 26 可見，當傳送和接收的速度接近(傳送稍快於接收)時，實際傳輸量(throughput)受傳送速率限制，和圖 21 比較，可以看到 throughput 略低。

(實驗1-3) 傳送時間間隔：25 ms，接收回應延遲：50 ms。

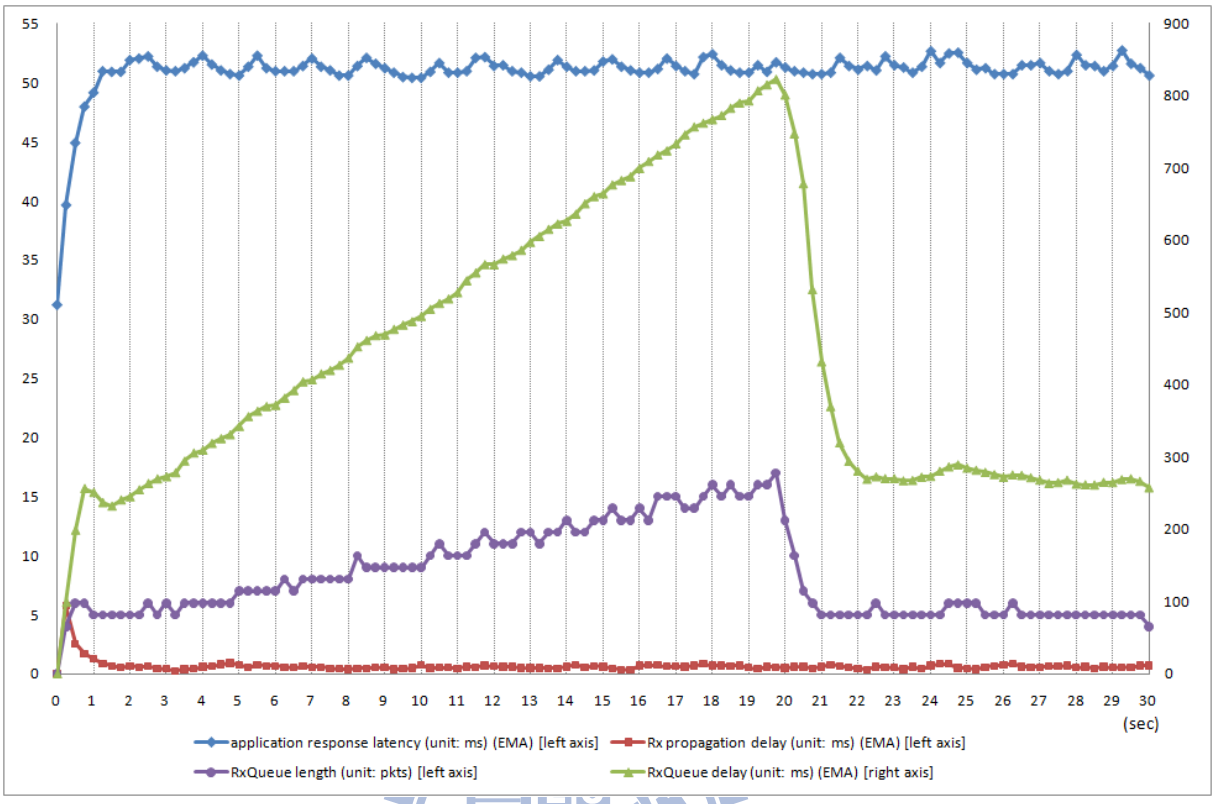


圖 27 : [實驗 1-3] 接收端的延遲(latency/delay)和佇列(queue)

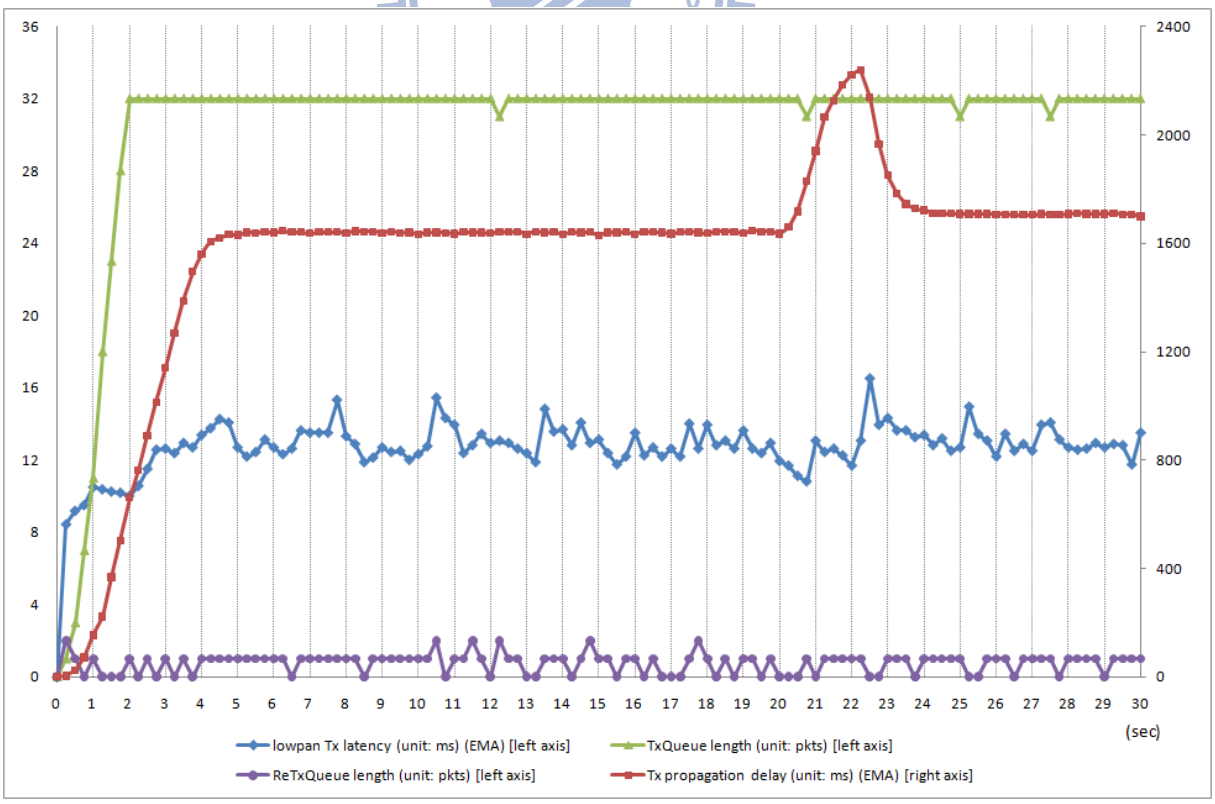


圖 28 : [實驗 1-3] 傳送端的延遲(latency/delay)和佇列(queue)

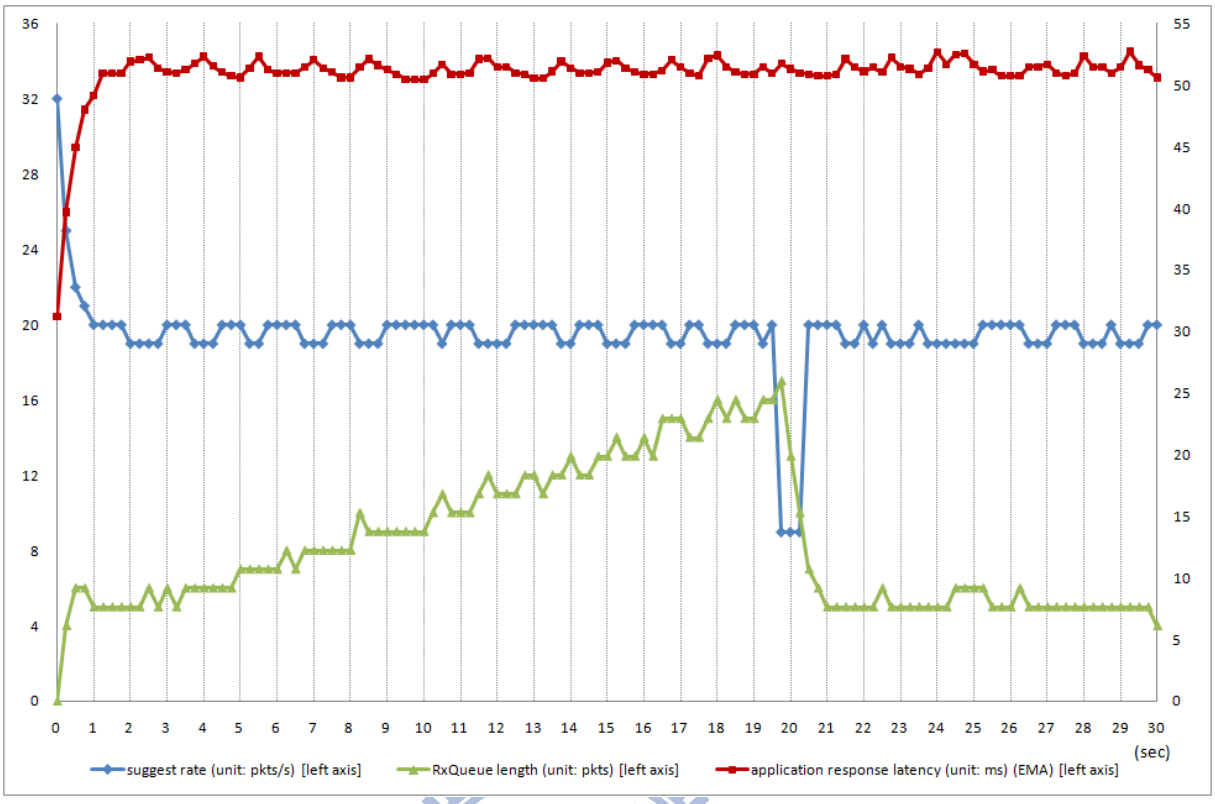


圖 29 : [實驗 1-3] 建議速率(suggest rate)的決定

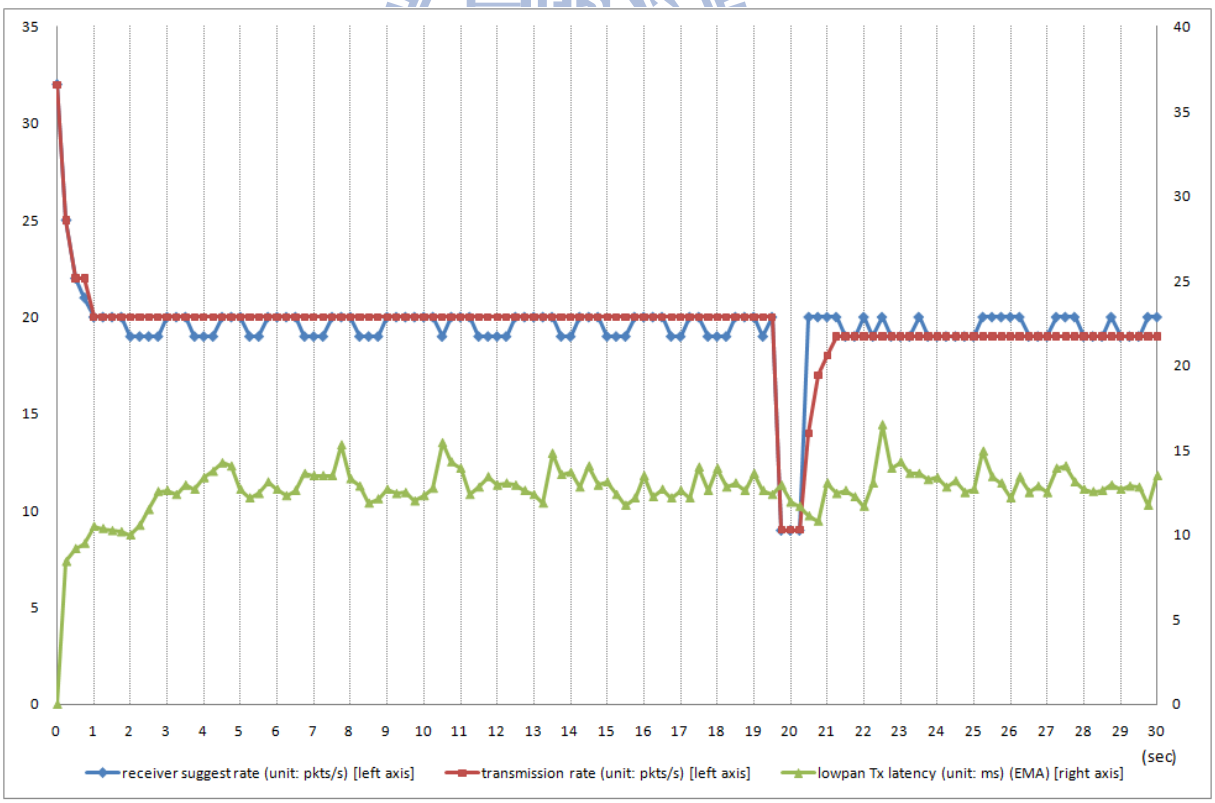


圖 30 : [實驗 1-3] 傳送速率(transmission rate)的調整

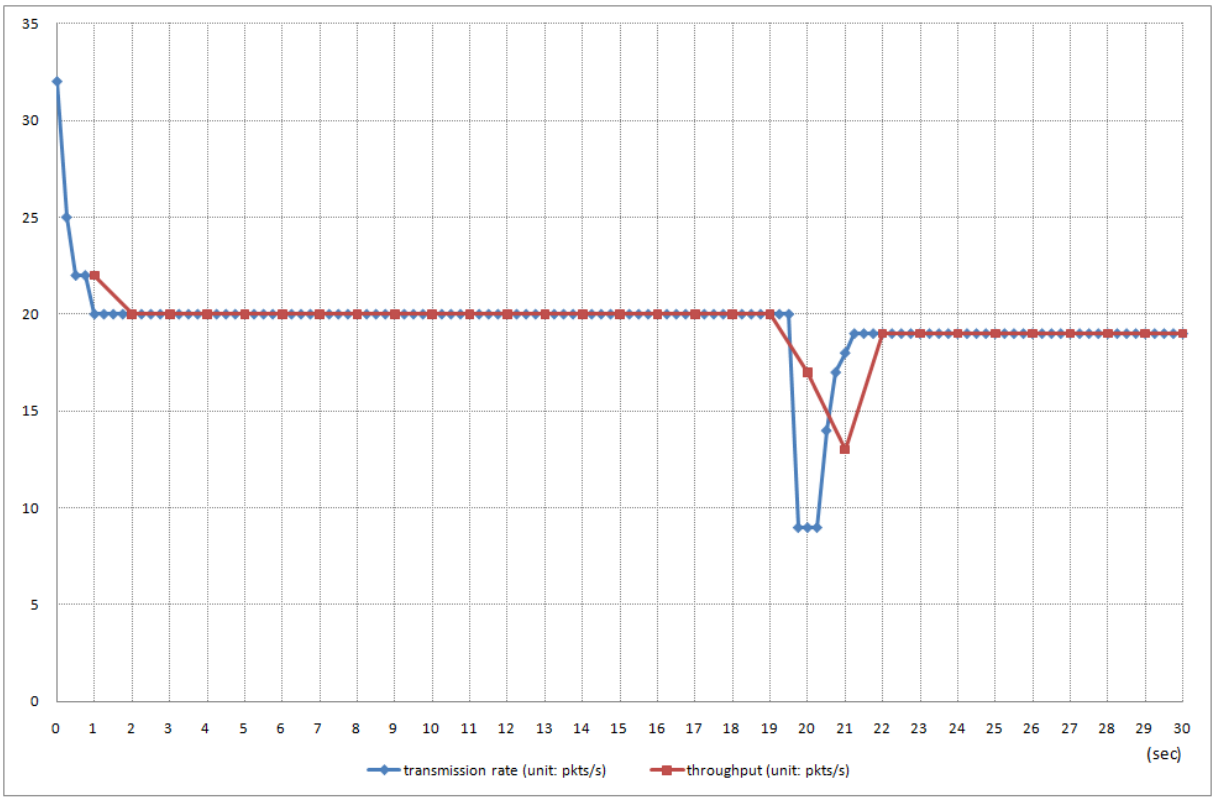


圖 31 : [實驗 1-3] 傳送端的傳輸量(throughput)和傳送速率

分析：

(1) 從圖 27 可見，RxQueue 有累積封包的現象，表示接收端應用程式的處理速度小於資料封包到達的速率。RxQueue delay 會隨著 queue 長度增加或減少而變長或縮短。圖中，RxQueue 長度陡降是因為流量控制發揮作用(見(6)之說明)。

(2) 從圖 28 可見，TxQueue 的長度快速成長，且在 2 秒後達到滿 queue 的情況，表示傳送速率低於應用程式資料傳送的速度。Tx propagation delay 曲線和 TxQueue length 曲線有些相似，表示造成 Tx propagation delay 增加的主因是 TxQueue 的佇列延遲(queuing delay)。圖中，Tx propagation delay 的高起是因為流量控制所致(見(6)之說明)。

(3) 從圖 29 可見，當 RxQueue 的長度在自身容量的一半以下時，建議速率(suggest rate)由應用程式回應延遲(application response latency)主導；當 RxQueue 的長度超過自身容量的一半時，建議速率調降為接收速率的一半，即進入表 1 中的速率調降模式(Rate

Down)。

(4) 從圖 30 可見，接收端建議速率低於 lowpan 傳輸延遲(lowpan Tx latency)的對應速率，所以傳送速率(transmission rate)由接收端建議速率主導。這表示資料傳輸的瓶頸在接收端。此外，傳送速率相較於建議速率來得平穩，表示表 2 中的邊界速率(margin rate)發揮作用。

(5) 從圖 31 可見，當傳送快於接收時，實際傳輸量(throughput)受傳送速率限制，和圖 21 比較，可以看到 throughput 明顯較低。此外，在 throughput 的低點可以看到傳送速率大幅調降產生的影響。

(6) 圖 29 說明接收端 RxQueue 長度過半觸發速率建議(rate suggestion)過程中的速率調降(Rate Down)，配合圖 30 和圖 31，可以說明傳送端根據建議速率降低傳送速率，讓實際傳輸量(throughput)下降，所以接收端有時間消化 RxQueue 中的資料，queue 的長度下降。接著看圖 28 和圖 31，傳輸速率下降導致資料封包待在 TxQueue 中的時間拉長，反映在 Tx propagation delay 曲線的高起處。

(實驗2-1) 封包遺失率：0.20

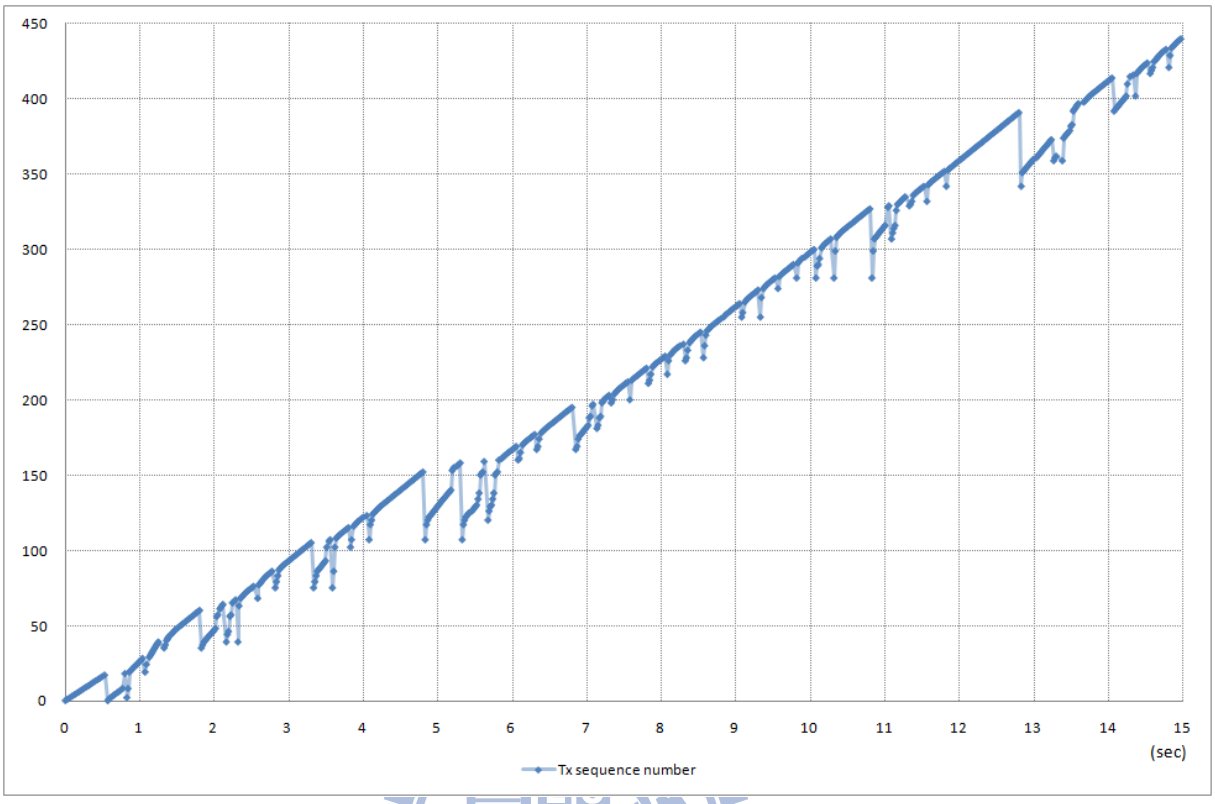


圖 32 : [實驗 2-1] 傳送端資料傳送與重傳

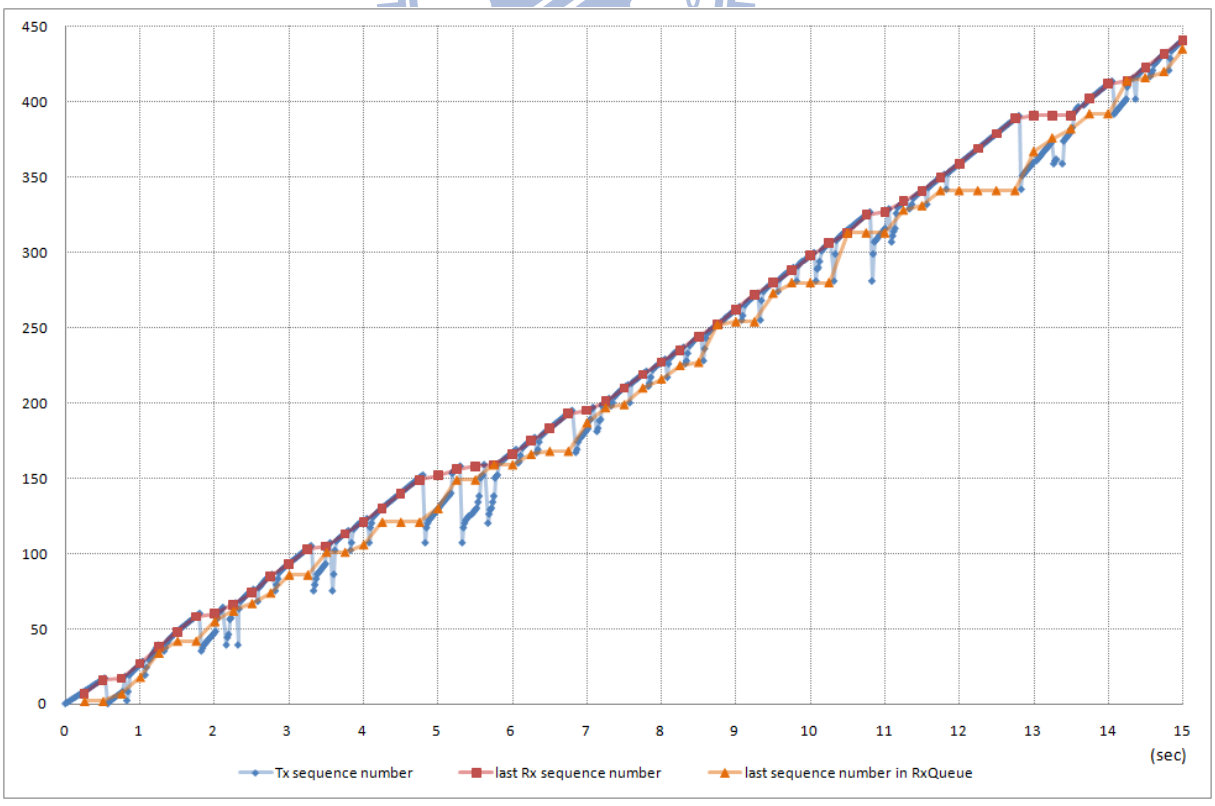


圖 33 : [實驗 2-1] 資料傳送和接收的情況

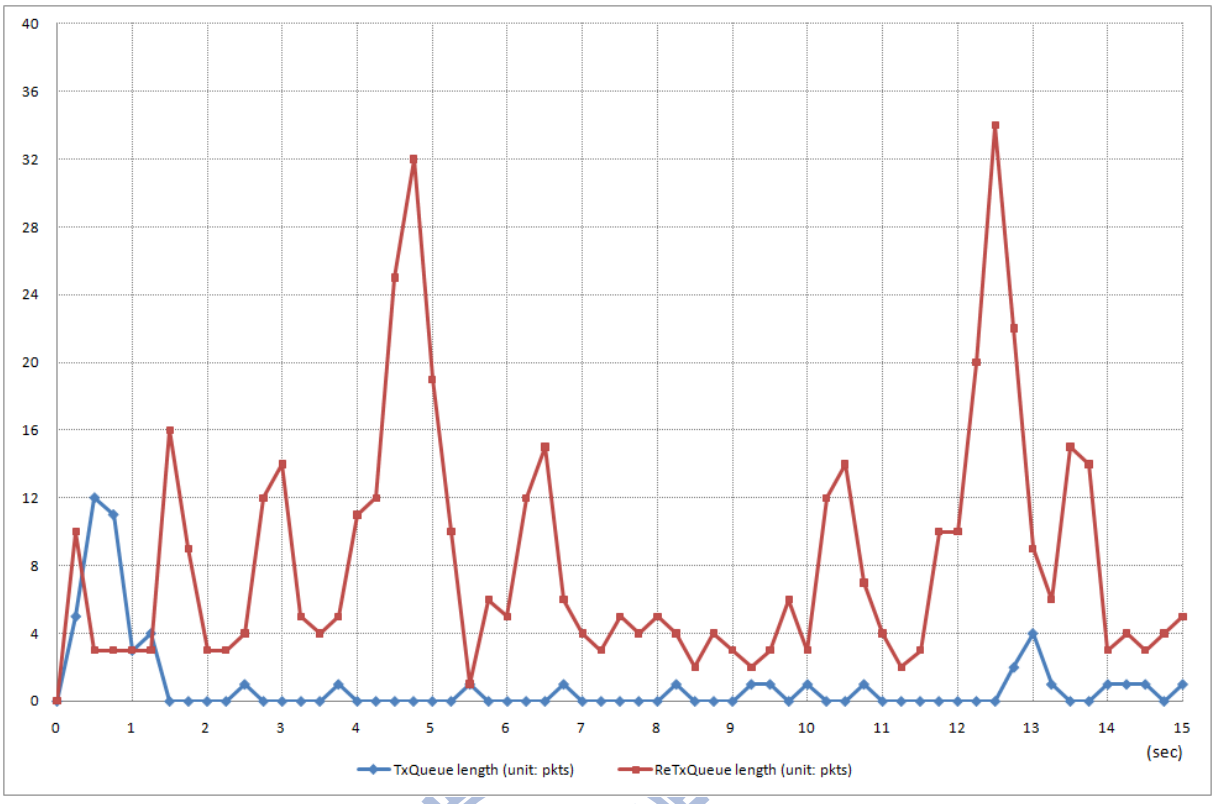


圖 34：[實驗 2-1] 傳送端的佇列(queue)

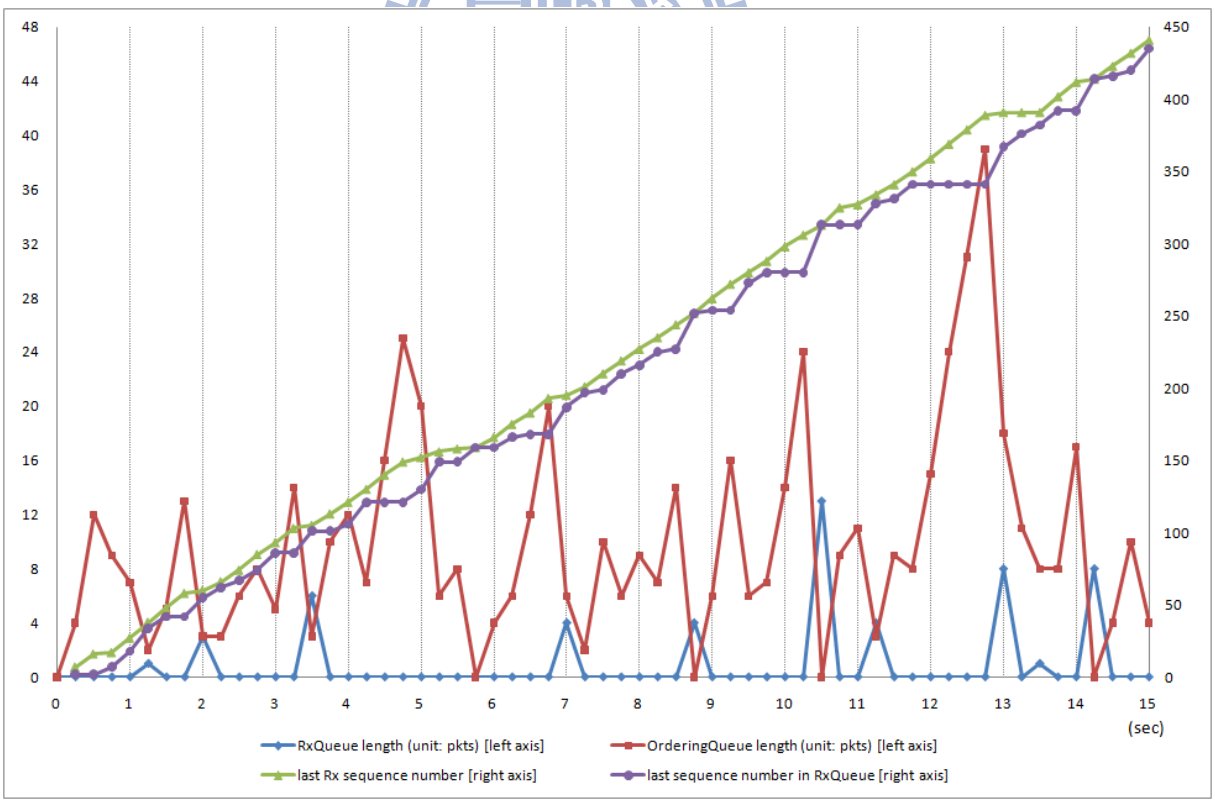


圖 35：[實驗 2-1] 接收端的佇列(queue)

分析：

(1) 從圖 32 可見，傳送端送出的封包序號大致構成一條斜率為正的直線，不過這條線出現許多斷裂，封包序號重複的部分就是資料封包重傳。

(2) 圖 33 中有三條曲線：第一條是傳送端送出的封包序號(sequence number)；第二條是接收端最後收到的封包序號(last Rx sequence number)；第三條是 RxQueue 中最後一個封包序號(last sequence number in RxQueue)。第二條和第三條的距離愈大，表示接收端收到愈多需要等待排序的封包，這是封包遺失所致。此外，從圖中可以發現第一條部分片段在第三條之下，表示重傳的封包序號接收端已收過，也就是出現重複傳送，這是因為 SACK 封包遺失所致。

(3) ReTxQueue 中暫存已送出尚未 ACK 的資料封包，如圖 18 所示，在沒有封包遺失的情況下，ReTxQueue 長度維持在 2 上下。從圖 34 可見，ReTxQueue 的長度大幅增加，原因在於許多資料封包沒有 ACK，表示出現封包遺失。

(4) 從圖 35 可以看到接收端對封包遺失的反應。OrderingQueue 用來暫存次序不對的資料封包以等待排序。在沒有封包遺失的情況下，資料封包到達接收端的順序幾乎和傳送端送出的順序相同，OrderingQueue 的長度大幅增加表示出現封包遺失。觀察 last Rx sequence number 和 last sequence number in RxQueue 這兩條曲線。OrderingQueue 累積時，兩條線的距離較大，表示出現封包遺失，新進的資料封包必須等待排序；OrderingQueue 長度下降時，兩條線的距離接近，表示收到符合順序的資料封包，在 OrderingQueue 中等待排序的封包也依序移到 RxQueue。

第 5 章　結語

5.1　研究成果

本研究的主要貢獻有下列幾點：

- 針對 WPAN 的特性，以及滿足多重連線(multi-session)的需求，設計一個適用於 WPAN 之可靠的傳輸協定：WPAN-ATP。提供串流(streaming)和順序(ordering)的資料傳輸，設計封包遺失回復(packet loss resilient)機制以確保資料完整性，設計基於速率的流量控制(rate-based flow control)以達到有效率使用頻寬(bandwidth-efficient)，提供低成本(low cost)傳輸以減輕無線裝置的負擔。
- 封包遺失回復：使用 SACK 減少接收端回送 ACK 的次數，而且可降低 ACKs 流量佔用的頻寬。重傳機制能夠快速補上遺失的資料封包，或是短暫傳輸中斷期間未成功送出的資料封包。
- 基於速率的流量控制：起始速率可以讓傳送端快速達到合適的傳送速率，提高頻寬的使用效率。傳送速率的調整方法考慮到下層的傳輸延遲和接收端的處理速度，忠實反映傳輸的瓶頸。
- 低成本傳輸：考慮到 WPAN 中無線裝置的特性，盡量降低流量控制方法中的運算複雜度，並在連線短暫中斷時進行網路偵測，停止不必要的資料傳輸。藉此減少裝置在運算和傳輸上的負擔。
- 在 SunSPOT 上實作 WPAN-ATP，遵循 J2ME 的通用連接框架(GCF)，讓 SunSPOT 網路程式開發者可以透過通用的方式使用 WPAN-ATP。提供除了 SunSPOT 現有的傳輸協定外，另一個選擇。

5.2 未來方向

本研究未來的可行方向，能使本研究更好、更完整，敘述如下：

- 為了增進效率，流量控制方法可以將速率(rate)改成延遲時間(delay time)，以減少時間轉換成速率的除法運算。
- 本研究沒有深入分析效能。未來可以針對不同的連線數目、不同的封包遺失率做效能分析，如此，能夠從分析結果得知協定的瓶頸所在，可以進一步做最佳化。
- 若要達成穩定傳輸多頻道寬頻生理訊號(例如：EEG)的目標，下一步要做的是在 WPAN-ATP 之上增加中介層(middle layer)，此構想在 1.2 開頭有提到。WPAN-ATP 是比較通用的傳輸協定，中介層可以針對此類訊號的特性來設計，目的在加強 WPAN-ATP 的不足。

參考文獻

[1] Ye Tian, Kai Xu, and Nirwan Ansari, “TCP in Wireless Environments: Problems and Solutions”, IEEE Radio Communications, March 2005.

[2] Karthikeyan Sundaresan, Vaidyanathan Anantharaman, Hung-Yun Hsieh, and Raghupathy Sivakumar, “ATP: A Reliable Transport Protocol for Ad-hoc Networks”, *IEEE Transactions on Mobile Computing*, vol. 4, no. 6, November/December 2005.

[3] SunSpotWorld - Home of Project SunSPOT, http://www.sunspotworld.com

[4] Project SunSPOT Documentation, http://www.sunspotworld.com/docs

[5] The Generic Connection Framework, http://developers.sun.com/mobility/midp/articles/genericframework

[6] “Transmission Control Protocol”, Internet Engineering Task Force (IETF), ser. RFC793, November 1981.

[7] Jim Kurose, Keith Ross, and Addison-Wesley, *Computer Networking: A Top Down Approach Featuring the Internet, 3nd edition*.

[8] “TCP Selective Acknowledgment Options”, Internet Engineering Task Force (IETF), ser. RFC2018, October 1996.