藍芽 MAC 層排程機制效能之研究

廖啟宏
中華大學資訊工程學系
m9002020@chu.edu.tw

嚴力行
中華大學資訊工程學系
lhyen@chu.edu.tw

摘要

Bluetooth 是同時支援語音及資料傳輸的短距離無線網路。Bluetooth MAC 層頻寬分配是以 Polling 為基礎，由單一的 Master 裝置動態分配傳送資料封包的時槽給其他所有的 Slave 裝置。標準的做法中，Master 會以 Round-Robin 的方式對每一個 Slave 進行 Polling，而陸續也有許多學者提出各式的排程機制，分為預視與非預視兩大類。目前所見機制的效能評估，普遍建立在所有 Slave 送給 Master 的交通流量是一致的 (Uniform)，而且 Master 與 Slave 之間的交通流量是對稱的 (Symmetric) 兩項假設上。我們觀察發現，當這兩項假設不成立時，現有機制的效能表現會有很大差異。在本文中我們除了使用傳統的交通流量的量測方式，並加入了在藍芽排程機制於交通流量非一致、非對稱的環境下的測試。以及探討排程機制在不同交通流量模式下其頻道使用率、封包延遲以及佇列長度與排程效能的關連性，並針對所得結果加以分析。

關鍵詞：Bluetooth，Wireless Network，Round-Robin，MAC，Scheduling.

一、簡介

近年來，由於網路服務的多樣化和行動計算需求的普及，無線通訊漸漸在現代科技生活中扮演重要的角色。現有無線通訊網路中，除了商業性服務的長距離無線電信系統（如 GSM、GPRS）以外，還有運作於 2.4 Ghz ISM 頻段 (The Industrial, Scientific, and Medical band; The ISM band) 的短距離無線通訊網路。

其中短距離無線通訊網路中又可概分為無線區域網路 (Wireless Local Area Network; WLAN) 和無線個人網路 (Wireless Personal Area Networking; WPAN) 兩類 [2][6]。其中 WLAN 系統以 IEEE 802.11b/a 為代表，實體層傳輸技術主要是直接序列展頻 (Direct-Sequence Spread Spectrum)，多應用在網際網路的無線擷取服務。另外 WPAN 則以 Bluetooth 為代表，以跳頻展頻 (Frequency Hopping Spread Spectrum) 為實體層傳輸技術 [7]，以取代近距離同質／異質性週邊的纜線連接為主要目的，亦可藉由跨越 Bluetooth 配備週邊，擷取遠距離的資料及語音服務。一群 Bluetooth 裝置亦可形成無線隨意網路 (Ad hoc networks)，彼此互相進行通訊 。

Bluetooth 網路的最小運作單位稱為 Piconet。一個 Piconet 由 2 個到 8 個不等的 Bluetooth 裝置（配備有 Bluetooth 傳輸功能的周邊設備，如手機、PDA、印表機等）組成。Piconet 的形成是由一個扮演 Master 角色的 Bluetooth 裝置發起，由它負責召集在其通訊距離內的其它 Bluetooth 裝置（最多可到 7 個）形成，而被召集的 Bluetooth 裝置稱為 Slave。一個 Piconet 中只能存在一個 Master，其餘裝置則皆扮演 Slave 的角色。Master 以 TDD scheme 將傳輸時間切割成長度為 0.625ms (1/1600 秒)的時槽(Time slot)，在既定的 79 個頻道上，以跳頻的方式傳送資料。偶數編號之時槽固定作為下行 (Master-to-Slave) 資料的傳送，奇數時槽則為上行 (Slave-to-Master) 資料的傳送。

Bluetooth 的 Master 在 MAC 層負責兩項任務：1.轉送(switchig)：轉送自己與 Slave 產生的封包。2.排程(scheduling)：安排 Slave 的封包傳送權。Master 與 Slave 之間建立的連結(Link)與排程的方式亦有所不同。Master 與 Slave 之間的連結分為兩種，一種是 Synchronous Connection Oriented (SCO) 連結，另一種則是 Asynchronous Connection-Less (ACL) 連結。SCO 連結是全雙工 (Full Duplex) 的連結，主要用於需要較高服務品質的語音資料傳送。為了保證頻寬，Master 須預留 SCO 連結所需之雙工時槽 (Duplex Slots)。

ACL 連結主要用於傳送資料封包，是由 Master 動態分配時槽的連結。Bluetooth 定義了數種 ACL 資料封包格式，分別需要佔用一、三、或五個時槽的傳送時間。Master 可在偶數時槽傳送資料封包給某個 Slave。此 Slave 接收封包後，如有要傳送給 Master 的封包，可在緊接著的奇數時槽傳送(如圖 1 所示)。而如果 Master 未傳送任何封包給某個 Slave，該 Slave 即無機會傳送資料給 Master。由上述描述可知，Bluetooth ACL 的 MAC 機制乃是以 Polling 為基礎。

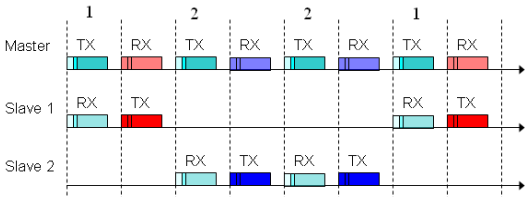

圖 1. 藍芽 Master Polling 的排程機制

在此機制中，每個 Slave 都有一 Slave-to-Master Queue (簡稱 Slave Queue)，存放 Slave 欲送至 Master 的資料封包。Master 也為每個 Slave 準備一個對應的 Master-to-Slave Queue (簡稱 Master Queue)，存放 Master 欲送至各個 Slave 的資料封包。Master 藉由拜訪每一個佇列的方式來服務排隊等候的顧客，而其拜訪佇列的順序規則，即為 Polling 機制的規則集合 (Rules Set) [9]。Master 必要時會傳送沒有任何實際有用資料的 Poll 封包給某個 Slave，以讓該 Slave 有機會傳送其資料封包。當 Slave 接收 Poll 封包後，若無資料封包傳送，亦需在下一個時槽回覆 Null 封包，以作為 Master 的回覆確認。顯而易見地，Bluetooth Polling 的規則集合將會影響其效能，如服務延遲 (Delay) [3,8,10,16]、頻寬利用率(Utilization) [3,8,13]、公平性(Fairness) [16,15]、佇列等待長度(Queue length)等。

依照 Bluetooth 規格書 [1] 中所訂定基本的作法，Master 會以 Round-Robin (RR / PRR) [3,10-11,12,14]的方式對每一個 Slave 進行 Polling，亦有許多學者陸續提出各式的排程(Scheduling)機制 [3,4]。依據是否假設 Master 知道 Slave 的佇列狀態，現有 Bluetooth MAC 排程機制大致可分為非預視排程[3,10]與預視排程[5][11]兩類。預視排程為理想化排程方式，要求 Master 事先知道 Slaves 的佇列狀態等資訊，並不切實際，在此文中不予考慮。在非預視排程中，Master 無法事先預知 Slave queue 的資訊，Master 只能依據 Master 單方面的資訊進行排程，或是與 Slave 交換封包後，藉由 Slave 回傳的訊息得知，較符合實際藍芽的運作狀況。

許多 MAC 排程機制宣稱有較高的產出量 [5,8,12]、較低的封包延遲、以及較佳的公平性 [4,12,14]。不過我們觀察到，其所謂效能上的增進，普遍建立在一個不客觀的運作環境上，即假設每個 Slave 之間送給 Master 的交通流量是一致的 (Uniform)。直覺上，如果每個 Slave 之間送給 Master 的交通流量並非一致時，這些排程機制的效能表現應該會有差異。此外，目前所有的排程機制的效能評估也假設每個 Slave 送至 Master 與 Master 送至該 Slave 的交通流量是對稱的 (Symmetric)。但由於實際藍芽的運作環境，是在多樣化不同性質的周邊，隨著周邊設備本身的特性和使用率，以及需作跨越網路轉送封包的設備而言，我們認為此假設並不符合真實藍芽的運作狀況。如果 Master 與 Slave 之間的交通流量是非對稱的 (Asymmetric)，只根據 Master 的佇列狀態進行排程，效能可能不佳。

在模擬量測上我們除了使用了傳統的一致且對稱的交通流量模式外，並對非對稱與非一致的交通流量模式進行測試分析，加強在不同流量下排程的客觀性。模擬中，我們採用現有藍芽排程中的 PRR [1,3,10-11,12,14]、ERR [3,10]、RR-FCFS [9]。因我們假設 Master Queue 長度沒有限制，所以在 [3] 提及的 EPM 不考慮為排程方法之一。在實驗中，假設在交通流量為一致、非一致、對稱、非對稱的各種情況下，三種排程在藍芽 MAC 的頻道使用率 (Channel utilization) 、封包延遲(Packet delay)、佇列長度(Queue length)與排程效能的關連性進行模擬測試，並針對所得結果與排程在各種交通流量模式下的結果加以分析，並推測解釋可能的原因，以提供以後更多研究的參考測試。

二、排程機制與實驗環境

我們進行模擬實驗的環境如圖 2 所示。所有的實驗均假設 Piconet 中 Slave 個數為 7 個。每個 Slave 個別維護自己的 Slave-to-Master Queue，存放欲傳送給 Master 的封包。Master 也各自為每個 Slave 維護 Master-to-Slave Queue，存放下行封包。

我們以 VC++來撰寫排程模擬程式，對 PRR、ERR 與 RR-FCFS 三種非預視排程進行效能研究。這三種排程方式簡述如下：

- PRR (Pure Round-Robin)

  Master 依照特定順序輪流送對各個 Slave 送出資料封包或 Poll 封包。收到資料封包或 Poll 封包的 Slave 可以回傳一個資料封包。即使 Slave 沒有資料封包要傳送，仍然要回傳一個 Null 封包。

- ERR (Exhaustive Round-Robin)

  Mater 仍依照特定順序輪流 Polling 各 Slave。與PRR不同的是Master可以持續送封包給同一個Slave，而被Polling到的Slave也可以在回傳資料封包時，夾帶訊息要求Master繼續Polling此Slave，直到Master與此Slave的傳送佇列清空為止。在這過程中即使Master已沒有資料封包要傳送給此Slave，Slave仍可以要求Master

繼續傳送Polling封包給它。

◆ RR-FCFS (Round-Robin with First-Come-First-Serve)

Master依據Master Queue的狀態採用兩種不同的排程。當Queue不為空時，Master依照封包產生的順序傳送資料封包，即FCFS(或稱FIFO)；當Queue為空時，Master則使用RR的方式Polling Slave。此排程機制中的RR為接續式，即當Master中途搶先執行FCFS後，再度執行RR時，Master會由上一次執行完RR服務順序的下一個Slave接著進行Polling。

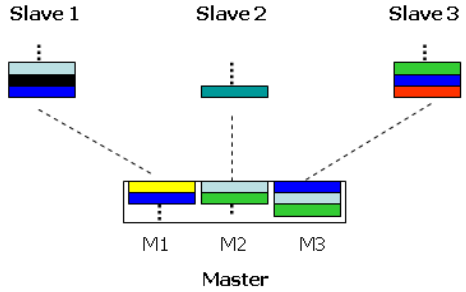

圖 2. Slave-to-Master & Master-to-Slave queues

我們假設每個 Master-to-Slave Queue 和 Slave-to-Master Queue 的資料封包均依波氏程序 (Poisson process) 獨立產生。我們令 λ 為波氏程序的平均產生率，單位為 packet/slot-pair。且令 $\lambda_M(i)$ 為 Master 送給 Slave i 的封包產生率，$\lambda_S(i)$ 為 Slave i 送給 Master 的封包產生率。實驗所用到的封包類型為 DH1 資料封包、Poll 封包、Null 封包，均佔用一個時槽的傳送時間。此外 μ 為封包最大服務率，其值固定為 1 packet/slot-pair。

我們實驗所使用的交通模式 (Traffic Pattern)，將包含以下各種情形：

- **一致與非一致 (Uniform and Non-uniform) 流量**：各個 Slave 之間送給 Master 的交通流量或 Master 送給各個 Slave 之間的交通流量可相同或不相同。
- **對稱與非對稱 (Symmetric and Asymmetric) 流量**：Slave 送至 Master 與 Master 送至該 Slave 的交通流量可相同或不相同。

我們將對現有的三種非預視排程 PRR、ERR 及 RR-FCFS，在假設 Master 與 Slave 的封包產生率分別為，一致、非一致、對稱、非對稱的各種交通流量情況下，測試並比較分析下列幾項的效能表現：

1. **頻道利用率 (Channel Utilization)**：亦即時槽利用率，為實際傳輸資料時槽數目與最大可用時槽數目的比值。在我們的模擬實驗中，是一個排程運作穩定性的重要參考指標。
2. **封包延遲 (Packet Delay)**：為封包到達佇列至離開佇列之間所花的時間。平均封包延遲表示排程機制在此項目的平均值。這項效能應該是評估排程機制優劣的最重要考量因素[8]。
3. **佇列長度 (Queue Length)**：佇列長度定義為佇列中封包的數量，我們測量佇列的平均長度(Average Queue Length)。

三、實驗結果與分析

對稱且一致型 (1m1s)

此實驗中，我們令 $\lambda_M(i) = \lambda_S(i) = \lambda/7$ 對所有 i 成立。即 Master 與 Slave 的封包到達率呈對稱而且各個 Slave 之間的封包到達率也呈一致。

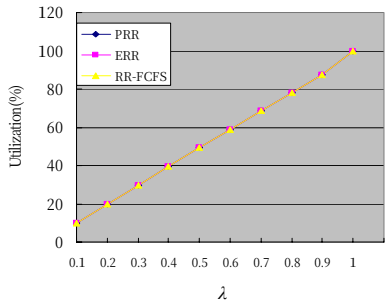

圖 3. Channel utilization in 1m1s

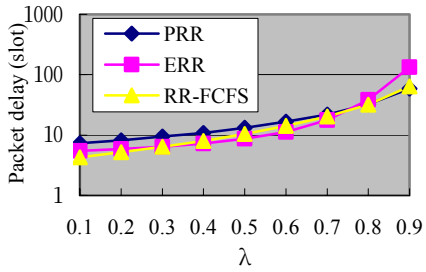

圖 4. Average packet delay in 1m1s.

頻寬利用率

圖 3 顯示 PRR、ERR 及 RR-FCFS 三種排程於不同 λ 值下的頻寬利用率。可以看到頻寬利用率與 λ 成正比，且三種排程效能幾乎相同，這是由於 $\sum\lambda_M(i) = \sum\lambda_S(i) < \mu$。事實上，由於三種排程除了傳送資料封包外，剩餘時槽皆用來傳送 Poll 和 Null 封包，頻寬利用率與發送 Null 與 Poll 封包的比例恰好相反。

Packet Delay

圖 4 顯示三種方法的封包延遲。RR-FCFS

在 λ 值小於 0.3 與大於 0.75 時，Packet delay 較小。這是由於封包到達率較低時，Master-to-Slave Queue 產生的資料封包即可立即以 FCFS 方式處理，Slave-to-Master Queue 也有較高機會在執行 RR 時被服務到，比起 PRR 或 ERR 來有較小延遲。在 λ > 0.75 時，Master-to-Slave Queue 被執行 FCFS 的頻率較高，相對 Slave-to-Master queue 被 polling 的頻率也增高，所以整體封包延遲還算不錯。

ERR 整體表現還算不錯，只有由圖 4 中在 λ > 0.8 明顯較差，顯示 ERR 在 λ 值高時，容易停留在處理單一 Slave 過長的時間，而造成其他 Slave 的 Packet delay 明顯增加。

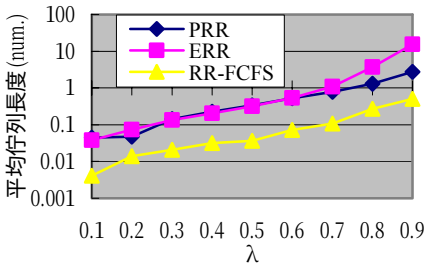

圖 5. 1m1s 中 Master 的平均佇列長度

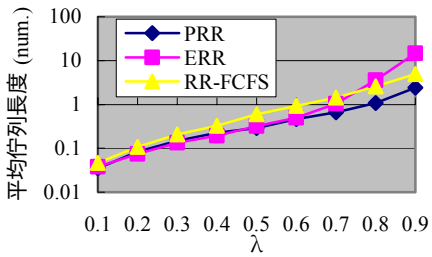

圖 6. 1m1s 中 Slave 的平均佇列長度.

PRR 整體則是偏差，但是也顯示公平分配時槽的特性，在 λ > 0.8，每一個 Slave 與 Master 都有較頻繁的封包傳送，相對也減少浪費在傳送空封包的機會，所以在 λ 值高時，反而有較好表現。RR-FCFS 在 λ 值高時，仍然能保持與 PRR 相當接近。

Average queue length

圖 5-6 分別為 Master queue 與各個 Slave queue 的平均佇列長度。圖 5 中，RR-FCFS 的 Master queue 平均佇列長度明顯遠較其他兩個方法度小，這是由於 FCFS 能即時清空佇列的結果。ERR 在 λ > 0.6 後，Master queue 的長度就較 PRR 明顯增長。這是由於 ERR 在 λ 值大時，需要停留在某對 Maste/Slave 做清空 queue 的時間加長，導致要傳送給其他 Slave 的 Master queue 長度平均較長。

在 Slave queue 方面，圖 6 中 RR-FCFS 在 λ > 0.35 ，則明顯較 PRR 略大。這是由於 λ 值較大時 Mater 執行 FCFS 頻率較高，Slave 在 FCFS 過程較少機會被 Polling 到，需在 Master queue 清空時執行 RR 才有較多機會被 Polling 到。但整體仍與 ERR 相近。而 PRR 在整體上則保持較為穩定且不錯的 queue 長度。

圖 5 與圖 6 比較，由於 PRR 與 ERR 都是屬於 Master 與 Slave 較為平均兼顧的排程，所以在 Master 與 Slave 的平均佇列長度的堆積曲線是相似的，而 RR-FCFS 則是 Master 明顯較好，而 Slave 平均整體表現則是與 ERR 相近。因為 Master-to-Slave Queue 平均而言是 Slave-to-Master Queue 長度的七倍，所以在 Master-to-Slave Queue 表現較佳的方法，如 RR-FCFS，實際上應會有較佳的效能表現。

非對稱、一致型 (2m1s)

此實驗中，我們令 $\lambda_M(i) = \lambda/7$ 且 $\lambda_S(i) = \lambda/14$ 對所有 i 成立。即 Master queue 的封包到達率(下行)為 Slave queue(上行)的 2 倍，而各個 Slave 之間 Master-Slave 各對(Pair)的封包到達率為一致的情形。

頻道利用率

三種排程的頻道利用率大致情形與 1m1s 相同，因為此時 $\sum\lambda_M(i) = \sum\lambda_S(i) < \mu$ 也成立。不過此處 Slave 的封包到達率較 1m1s 時為低，所以其頻道利用率也較 1m1s 時為低，在 λ = 0.9 時，頻道利用率約只有 66.2%。另外，當 $\sum\lambda_M(i) > 0.9$ 時，雖然 $\sum\lambda_S(i)$ 才大於等於 0.45，但由於 $\sum\lambda_M(i)$ 已接近穩定排程的負荷值，所以不被考慮。

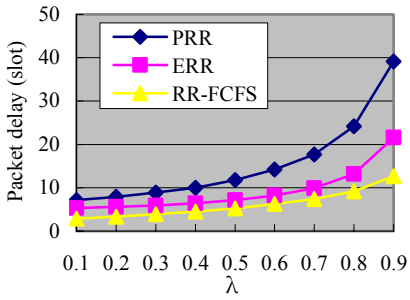

圖 7. 2m1s 中的平均封包延遲.

Packet Delay

圖 7 為三種排程的平均封包延遲。RR-FCFS 效能明顯較好。這是由於在 Master 的 λ 值比例較高時，優先執行 FCFS 的次數較頻繁，能迅速的處理 Master queue 的封包，連帶消耗部分 Slave queue 的封包的頻率也較高，加上 Slave 的 λ 值較小，使得整體的封包延遲降到最低。而 ERR 也因為在一次清空要 Master 與連帶清空 λ 值較低的 Slave 才換下一個 Slave 的特性下，明顯比 PRR 好。

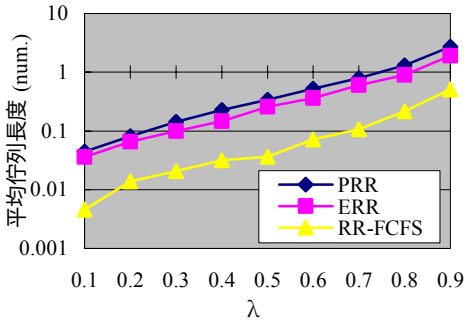

圖 8. 2m1s 中 Master 平均佇列長度.

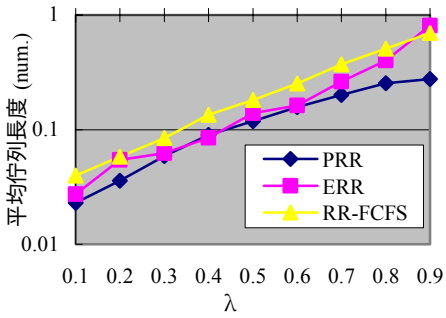

圖 9. 2m1s 中 Slave 平均佇列長度.

Average queue length

圖 8 是 Master queue 的平均佇列長度。在 2m1s 下 Master queue 雖然 λ 值的比例較高但 RR-FCFS，依然能迅速的處理 Master queue 的封包，讓 queue 的平均長度保持比其他兩個方法短。

在圖 9 中，Master 雖然較高 λ 比例連帶能讓 Slave 被 Polling 的次數較為頻繁，但是 Master 執行 FCFS 的次數較多，相對 RR 次數較少。而且 FCFS 是依照 Master 做排程，所以服務並無法顧及實際需要傳送封包的 Slave。因此 Slave queue 的平均堆積長度略長於 PRR。但是在 λ 值大時仍較 ERR 為佳。

非對稱、一致型 (1m2s)

我們令 $\lambda_M(i) = \lambda/14$ 且 $\lambda_S(i) = \lambda/7$ 對所有 i 成立。即上行封包到達率為下行的 2 倍，而各個 Slave 之間的封包到達率為一致的情形。

頻道利用率

三種排程的頻道利用率與 2m1s 類似，因為此時 $\sum\lambda_M(i) = \sum\lambda_S(i) < \mu$ 也成立。1m2s 的封包到達率以 Master 較低，在 λ = 0.9 時頻道利用率大約只有 66.6%。

Packet Delay

圖 10 為平均封包延遲。RR-FCFS 在 λ < 0.5 的情況下，還算比 PRR 為佳，但是大於 0.5 以後就明顯較差。這是由於 RR-FCFS 在 λ > 0.5 時 Slave 的累積封包已越來越多，但 Master 封包到達率仍然較低，所以多半只能執行 RR，而 RR 消耗 Slave Queue 的速度較為緩慢，再加上執行 FCFS 無法針對累積封包較多的 Slave 提供較多的服務，使得整體系統有較高的封包延遲。ERR 則由於在 Slave 的 λ 比例高的情況下，能一次清空所有 Slave Queue，較 PRR 浪費時槽傳送空封包，在 Packet Delay 上明顯較 PRR 和 RR-FCFS 低。

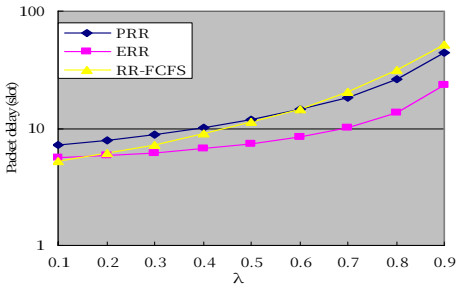

圖 10. 1m2s 中的平均封包延遲.

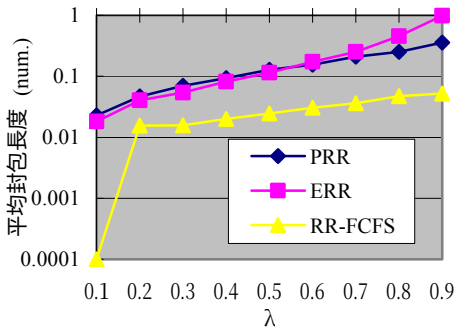

圖 11. 1m2s 中 Master 的平均佇列長度

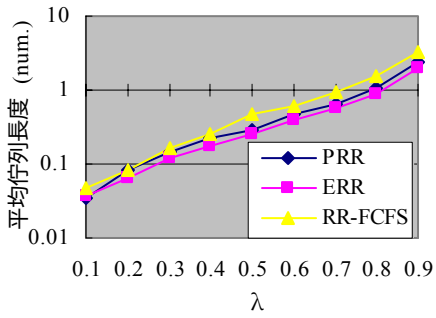

圖 12. 1m2s 中 Slave 的平均佇列長度.

Average queue length

圖 11 是 1m2s 下 Master Queue 的平均長度。由圖中可以明顯看出 RR-FCFS 的表現比其他兩種排程好，甚至比起在其他情況的交通模式下都還好，尤其在 λ 值小時更為明顯。這是由於在 1m2s 的交通流量下，Mater 相對 λ 比例低，產生的封包較少，再加上 Master Queue 一有封包便很快搶先用 FCFS 排程優先處理，所以 RR-FCFS 的表現到了最佳的狀態。

相反的圖 12 中 Slave queue 的平均累積長度中，RR-FCFS 顯示較差是可以預期的，由於 Slave 的 λ 比例較高，又要等到 Master Queue 清空時才能執行 RR。但並沒有預期的差，在 λ 值小時，由於執行 RR 的次數較多，所以還是能保持與 ERR 相當接近。

由圖 11 圖和圖 12 圖的比較可以得知在圖 10 中 RR-FCFS 的 Packet delay 平均值所產生較差的表現，應該是由 Slave Queue 所造成的 Queuing Delay。

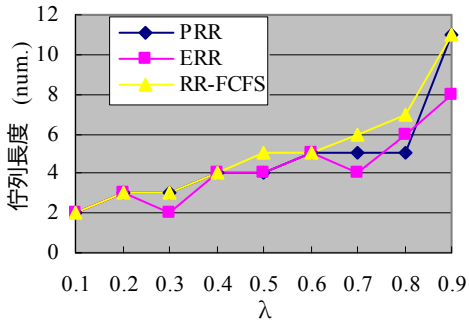

圖 13. 1m2s 中 Slave 的最大佇列長度

圖 13 為 Slave 的最大 Queue 長度。，三種方法在整體表現上相近，並沒有如圖 12 平均 Queue 長度上的表現 RR-FCFS 在 Slave Queue 上有特別明顯的較差。

對稱、非一致型 (Non-uniform)

我們令 $\lambda_M(1)= \lambda_M(3)= \lambda_M(5)= \lambda_M(7) = \lambda/12$; $\lambda_M(2) = \lambda_M(6) =\lambda/6$; $\lambda_M(4)= \lambda/3$; 且 $\lambda_S(i) =\lambda_M(i)$ 對所有 i 成立。即下行與上行封包到達率為對稱，而各個 Slave 之間的封包到達率為非一致的情形。

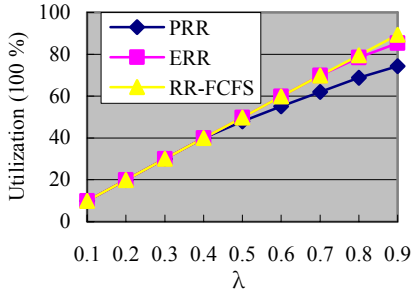

圖 14. Channel utilization in Non-uniform.

頻道利用率

圖 14 是對稱且非一致型的頻道利用率。在此實驗中雖然 $\sum\lambda_M(i) = \sum\lambda_S(i) < \mu$，但是由於比例分配的差異，每一個 Slave 的負荷也各不同。可以明顯看出 PRR 在 $\lambda > 0.4$ 以後就出現明顯不穩定的情況，整體的頻道利用率已無法達到預期的比例。而 ERR 也在 $\lambda > 0.7$ 以後，略有降低的情況產生，只有 RR-FCFS 仍然保持相當穩定的運作狀態。PRR 的運作方式，在對稱且非一致的情況下，對於某些 λ 比例分配較大的 Master-Slave pairs 已無法在固定運作時間內處理完所有封包，而仍然把時槽浪費在 λ 比例分配較小的 Master-Slave pairs 上傳送 Poll 和 Null 封包，導致整體的頻寬利用率下降。ERR 於每一個 Master queue 與 Slave queue 清空後才換下一對，對於 λ 比例高的 Master-Slave pairs 仍能保持不錯的效能，但由於 ERR 多半在系統 λ 值較大時表現較差，在對稱且非一致型的情況下，λ 值大時也出現不穩定的情況。

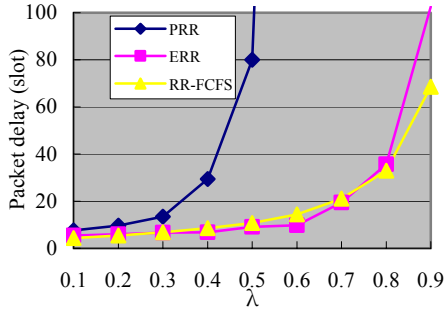

圖 15. Average packet delay in Non-uniform

Packet Delay

圖 15 為平均封包延遲的結果。我們可以看到 ERR 於 λ>0.8 時和 PRR 於 $\lambda > 0.4$ 時，封包延遲急速增加。這是由於第 4 對 Master 與 Slave 的封包產生速率 $\lambda_M(4) = \lambda_S(4) = \lambda/3$ 遠大於平均值 λ/7，使得 PRR 在 λ >0.4 時，便無法負荷。ERR 與 RR-FCFS 在 $\lambda < 0.8$ 時，大致與圖 4 的結果相近。

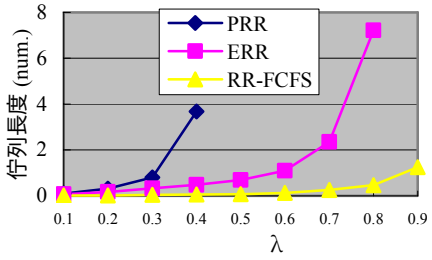

圖 16-1. Master-to-Slave 4 的平均佇列長度

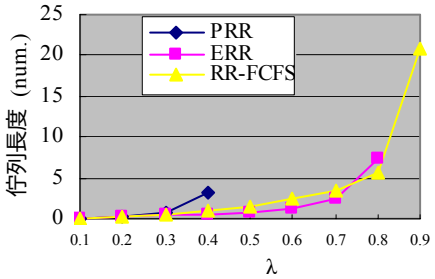

圖 16-2. Slave4-to-Master 的平均佇列長度

Average queue length

圖 16-1 與圖 16-2 為第四對 Master 與 Slave 的 Queue 平均長度。由於在這非一致型的比例下，第四對所分配 λ 的比例為最高的。圖 16-1 中 RR-FCFS 在 Master Queue 上仍有相當好的表現。但是在圖 16-2 中 Slave Queue 上當 λ > 0.4 時就差於 ERR。但就整體而言仍較 ERR 好，因為在系統 λ > 0.8 時 ERR 就出現 Queue 長度無限增長的趨勢。

非對稱、非一致型 (NuA)

我們令 $\lambda_M(1)= \lambda_M(3)= \lambda_M(4)= \lambda_M(6) = \lambda/12$; $\lambda_M(2) = \lambda_M(5) = \lambda/6$; $\lambda_M(7)= \lambda/3$; 且 $\lambda_S(2) = \lambda_S(4) = \lambda_S(5)= \lambda_S(7) = \lambda/12$; $\lambda_S(3) = \lambda_S(6) = \lambda/6$; $\lambda_S(1) = \lambda/3$。

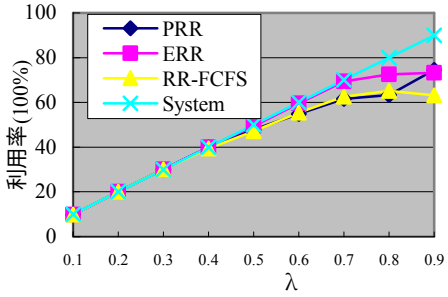

圖 17. Channel utilization in NuA (Non-uniform & Asymmetric).

頻道利用率

圖 17 顯示頻道利用率。圖中 System 是指在穩定運作下排程應該達到的理想利用率。三種方法都無法在非一致且非對稱的交通流量模式下穩定運作。其中以 ERR 的情況較佳，在 λ > 0.7 時才明顯出現不穩定的情況，RR-FCFS 則與 PRR 相近，在 λ > 0.4 就明顯出現 Utilization 下降的情形。這是由於 ERR 在這種流量模式下，較能針對 Master Queue 與 Slave Queue 不同比例的封包產生率，提供不同的服務率。RR-FCFS 則是受到 RR 的機置所影響而與 PRR 相接近。

Packet delay

圖 18 是三種排程在平均 Packet delay 上的情形。RR-FCFS 在 λ > 0.3 以後 Packet delay 明顯較大，而 PRR 和 ERR 則分別出現在 λ > 0.4 及 λ > 0.7 以後，與 Utilization 結果相似。主要的影響並非整體的 Packet delay 都變大所造成的，應該是由於某幾對 Master-Slave pair(如 Master-Slave1 與 Master-Slave7)在非對稱流量的落差較大所造成。其他的 Master-Slave pair 在 λ 值小時，仍然能保持穩定的運作。所以由圖 18 下圖，就整體來看雖然 RR-FCFS 因為 Master-Slave1 產生的延遲較大，導致整體 Packet delay 在 λ 較小時就明顯拉升。但在 λ 較大時卻能持平穩定，PRR 也是。而 ERR 雖然能保持在 λ 較大才拉升，但卻有 Packet delay 急速加大的情形。這是由於 ERR 在 λ 大時，會耗費更多的時間處理 Master-Slave1 與 Master-Slave7，而導致其他的 Master-Slave pair 平均的 Packet delay 明顯加大所導致。

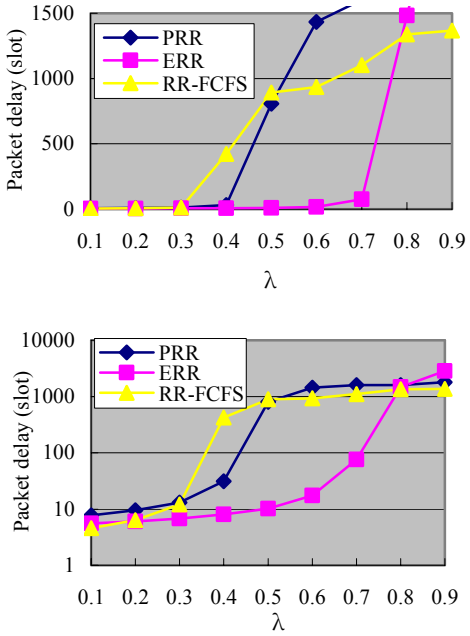

圖 18. Average packet delay in NuA. (上) Master. (下) Slave.

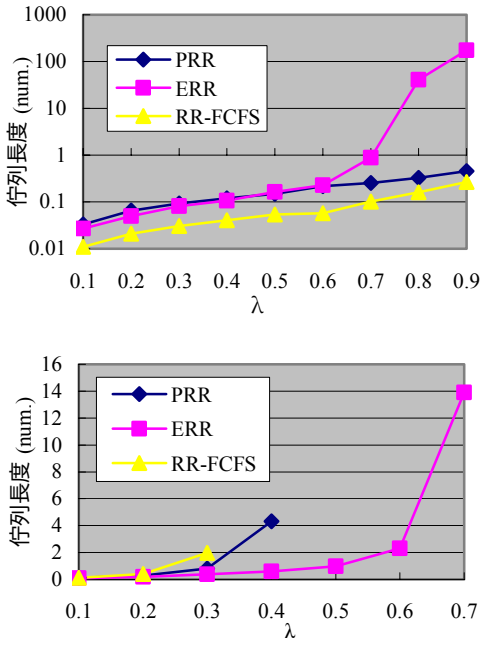

圖 19. Average queue length of Master (上) and Slave #1 (下) in NuA.

Average queue length

不同的 Master-Slave 對表現並不相同。以 Master-Slave1 而言(圖 19), $\lambda_M(1) = \lambda/12 < \lambda_S(1) = \lambda/3$。各個排程在 Master 佇列長度上，RR-FCFS 仍明顯比其他兩種方法好，其次是 PRR、然後 ERR。在 Slave 佇列長度上三種排程方式都出現不穩定的情況，其中 RR-FCFS 在 $\lambda > 0.3$，PRR 在 $\lambda > 0.4$，ERR 在 $\lambda > 0.7$ 時出現不穩定情形，與 Utilization 所測得的結果相近。但以 Master-Slave7 而言，恰與 Master-Slave 1 相反，是 $\lambda_M(1) = \lambda/3 > \lambda_S(1) = \lambda/12$。所以在 Master 佇列長度上 PRR ($\lambda > 0.4$ 時)與 ERR ($\lambda > 0.7$ 時), 種排程方式也出現不穩定的情況，而 RR-FCFS 仍然保持穩定的狀態。在 Slave 佇列長度上，RR-FCFS 與 PRR 相近，但在 λ 高時比 PRR 略好，這是由於 RR-FCFS 在 Mater 的 λ 值較高時, 連帶 Polling Slave 的頻率也增加的結果。

由 Master-Slave1 與 Master-Slave7 的比較可知，在 RR-FCFS 造成利用率 $< \lambda$ 主要是由 Master-Slave1 所造成，而 PRR 和 ERR 則除了 Master-Slave1 外還有 Master-Slave7 所影響。而 Master-Slave1 的 Master Queue Length 與 Master-Slave7 的 Slave Queue Length 是相近的。

四、結論

總結實驗結果，就 Packet delay 而言，ERR 除了在一致、對稱的 heavy load 時較 PRR 差外，其餘狀況均勝過 PRR。RR-FCFS 在對稱狀況下，效能與 ERR 在伯仲之間；在非對稱狀況下，如 Master 的封包到達率較高時，則效能較 ERR 好，反之則不然。就平均 queue 長度而言，無論在任何情況下 RR-FCFS 始終有最小的 Master queue 長度。

不同的藍芽排程方式，對於藍芽 Piconet 的運作效能有顯著的影響。而我們加入與先前提出所不同的模擬交通流量模式，非對稱型與非一致型，試圖使模擬更接近，各種不同網路交通情況的實際運作情形。並在這些交通模式下對不同排程的頻道使用率、封包延遲、佇列長度，的效能結果分析其與排程效能的關連性，並針對各排程在各種交通流量模式下的結果加以分析，並推測解釋可能的原因，以提供以後更多藍芽排程研究的參考和測試。

參考文獻

[1] Bluetooth SIG, *Specification of the Bluetooth System - Version 1.1 B*, Feb. 2001.

[2] A. Kamerman, "Coexistence Between Bluetooth and IEEE 802.11 CCK Solutions to Avoid Mutual Interference", IEEE 802.11-00/162, July 2000.

[3] A. Capone, M. Gerla and R. Kapoor, "Efficient polling schemes for Bluetooth picocells," in *Proc. of IEEE ICC*, vol. 7, pp. 1990-1994, Helsinki, Finland, June. 2001.

[4] A. Das et. al., "Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network," in *Proc. of IEEE INFOCOM'01*, pp.591-600, 2001.

[5] Daqing Yand, G. Nair, "Round robin with look ahead: a new scheduling algorithm for Bluetooth," in *Proc. IEEE International Conference on Parallel Processing Workshops*, pp. 45-50, 2002

[6] J. Lansford et. al., "Wi-Fi (802.11b) and Bluetooth: enabling coexistence," *IEEE Network*, 15(5): 20 -27 , Sep/Oct. 2001.

[7] J.C. Haartsen, "The Bluetooth radio system," *IEEE Personal Communications*, 7(1): 28-36, Feb. 2000

[8] Jong Soo Oh et. al., "An efficient and QoS-aware scheduling policy for Bluetooth," in *Proc. IEEE VTC 2002-Fall*, vol. 4, pp. 2445-2450, 2002

[9] Chi-Hung Liao, Li-Hsing Yen, "RR-FIFO : A new bluetooth MAC scheduling scheme," *2003 Symposium on Digital Life and Internet Technologies*, NCKU Taiwan, Sept 2003.

[10] Liron Har-Shai et. al., "Inter-piconet scheduling in Bluetooth scatternets," in *Proc. OPNETWORK 2002 Conference*, Aug. 2002.

[11] M. Kalia, D. Bansal, and R. Shorey, "Data scheduling and SAR for Bluetooth MAC," in *Proc. of IEEE VTC 2000-Spring*, vol. 2, pp. 716-720, Tokyo, Japan, May 2000

[12] M. Kalia, D. Bansal, and R. Shorey, "MAC scheduling and SAR policies for Bluetooth: A master driven TDD pico-cellular wireless system," in *Proc. Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)*, pp. 384-388, San Diego, USA, Nov. 1999.

[13] O. Fabin, H. Levy, "Polling system optimization through dynamic routing policies," in *Proc. of IEEE INFOCOM '93*, vol. 1, pp. 194-200, 1993.

[14] S. Garg, M. Kalia, and R. Shorey, "MAC scheduling policies for power optimization in Bluetooth: a master driven TDD wireless system," in *Proc. IEEE VTC 2000-Spring*, vol. 1, pp. 196–200, Tokyo, Japan, May 2000.

[15] S. Lu, V. Bharghavan and R. Srikant, "Fair scheduling in wireless packet networks," *ACM SIGCOMM'97*, August 1997.

[16] Z. Liu, P. Nain, D. Towsley, "On optimal polling policies," *Queuing Systems*, vol. 11, pp. 59-83, 1992.