行政院國家科學委員會專題研究計畫　期中進度報告

藍芽個人無線區域網路上通訊議題之設計實作與分析(1/3)

計畫類別：個別型計畫
計畫編號：NSC92-2213-E-009-076-
執行期間：92年08月01日至93年07月31日
執行單位：國立交通大學資訊工程學系

計畫主持人：曾煜棋

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 93年6月10日

摘要

藍芽（Bluetooth）技術係一新興崛起之個人區域網路標準，並預期在未來無線通訊領域發展中將扮演著重要的角色；藍芽當初原本是設計用來取代纜線設備（例如：滑鼠，鍵盤與個人電腦之連接線)，但在後來的演進中發現，藍芽的應用面並不僅止於此；隨著市面上低價藍芽模組的陸續出現，大規模的藍芽設備佈署將很有可能在近期內實現。針對上述趨勢，我們將分別就藍芽相關通訊議題，提出一系列的解決方案或分析結果，並實作出真正具有通訊功能的藍芽散網(scatternet)與微網(piconet)，評估不同網路結構對整體通訊效能的實際影響，以達到驗證理論分析之目的。

本計畫的主要目標，是建立有效率的藍芽通訊網路並以理論加以評估。本報告將提出第一年的研究成果。

關鍵詞： 藍芽，散網，微網

目錄

報告內容

I. 前言

藍芽係採主從裝置之網路結構（master-slave configuration），其基本的網路單元稱為微網（piconet），主裝置（master）係微網之中心控制設備，並以輪詢（polling）的方式來解決從屬裝置間競爭無線頻道（wireless channel）的問題，然而現有藍芽規格裡並無明確交代主裝置該以何種策略來輪詢旗下從屬裝置，使得頻寬能很有效率地被運用；因此，我們將在微網（piconet）的環境下，探討主裝置輪詢（poll）從屬裝置之機制策略；由於每個從屬裝置有著不同的數據流量，使得主裝置送往從屬裝置（master-to-slave，即 downlink 方向）與從屬裝置送往主裝置（slave-to-master，即 uplink 方向）邏輯連結之頻寬需求通常亦不對稱，亦即交通量有 asymmetric 現象，因此主裝置如何根據各從屬裝置之不同需求，聰明地執行輪詢機制，有效率地分配頻寬給旗下所有從屬裝置，便是一個重要的問題；在這項研究裡，我們將提示一個***樣式配對***（Pattern Matching）的輪詢策略，根據主裝置與從屬裝置之相對流量，我們將定義一組最有效率的輪詢樣式（polling pattern），並計算排程相對應的輪詢時間點（polling timings）。其中輪詢樣式是一個有序數列，定義了主裝置詢問（polling）與從屬裝置回應（replying）所必須使用的數據資料封包型態（data packet type），即藍芽規格裡主要提供的三種數據封包大小，分別是佔 1 個，3 個，及 5 個時槽（1-/3-/5-slot）。藉由封包大小所能載送之資料量並不同，我們將有效解決上／下行鏈結（up-／down-link）之交通量不對稱的問題，並減少負載欄位（payload field）沒填滿，甚至空的（null）數據封包，更有效率地使用每一個藍芽時槽（time slot），提昇藍芽微網最大可容納流量（throughput），使得更多的通訊活動能夠同時被支援。

II. 研究目的

本計畫的目的為：

針對藍芽的主從式網路架構做改善。在藍芽微網（piconet）的環境下，探討主裝置詢問（poll）從屬裝置之機制策略，減少負載欄位（payload field）沒填滿，或甚至空的數據封包，期望更有效率地使用每一個藍芽時槽（time slot），藉此提昇藍芽微網最大可容納之生產量（throughput），使得更多的通訊活動能夠同時被支援。

III. 研究方法

藍芽係採主從裝置之網路結構（master-slave configuration）主裝置（master）係微網之中心控制設備，並以輪詢的方式（polling scheme）來解決從屬裝置間競爭無線頻道（wireless channel）的問題。然而現有藍芽規格裡並無明確交代主裝置該以何種策略來輪詢旗下從屬裝置，使得有限頻寬能很有效率地被運用。

很明顯的，輪詢的方式的優劣，將會決定一個微網的的效率和生產量。一旦輪詢到佇列(queue)沒有資料要送的從屬裝置，或分配給從屬裝置過大的封包容量，使封包在未填滿的狀況下送出，都會使藍芽時槽的使用效率不彰。

我們就下列兩種狀況分別提出輪詢策略(polling policy)來增進藍芽網路的效率:

1. 一個樣式配對(Pattern Matching)的輪詢策略(polling policy)
2. 樣式配對的詢問策略(polling policy)適用在多主從配對架構(Multiple Master-Slave pair)下

兩種策略分述如下:

1.建構一個樣式配對(Pattern Matching)的輪詢策略(polling policy)

藍芽規格裡主要提供了三種數據封包大小，分別是佔 1 個，3 個，及 5 個時槽（1-/3-/5-slot），每一個封包大小所能載送之資料量並不同，如下表。我們可以利用這樣的機制來解決主裝置與從屬裝置資訊流量不對稱的問題。

在一個微網中，可以從過往的紀錄中，或者由逼近的方式，得知的每一對主從裝置的 average traffic arrival rates。當然，這些頻率未必相等。

首先我們提出一個 naive greedy 的解決方式作為比較的對象。DH5 的封包格式是藍芽中最有效率的使用方式，故我們使主裝置永遠等待一段時間 T，使得佇列中的資料量盡可能的填滿 DH5 的封包。

假設主裝置的 data rate 為 20(bytes/sec)，從屬裝置的 data rate 為 2(bytes/sec)。naive greedy 的策略是使主裝置等待一段時間，收集（339/20）* 20 =320 位元組，盡可

能地填滿一個 DH5 封包後才詢問其相對應的從屬裝置。而此時從屬裝置欲傳的資料量大約是(339/20)*2 =32 位元組，因而應該回傳給主裝置一個 DH3 的封包。若 data rate 並不改變，每 16 個藍芽時槽會重複一次上述(DH5, DH3)的封包傳送。很明顯的，在這個方法中從屬裝置回傳的封包的效率並不佳，將導致整個系統的生產量的下降。

以下將描述我們的樣式配對（Pattern Matching）的詢問策略（polling policy）。

首先，以λ_M和λ_S表示主裝置和從屬裝置的交通流量，單位為 bytes/slot。令$\lambda_H=\max\{\lambda_M,\lambda_S\}$,$\lambda_L=\min\{\lambda_M, \lambda_S\}$，而$\rho=\lambda_H/\lambda_L$我們以 N_H和 N_L 來表示交通流量（data rate）分別為λ_M和λ_L 之藍芽裝置。而詢問樣式(polling pattern)是指一連串主從裝置間交換的封包格式。長度 *k* 的樣式有兩個 *k* 維度的向量 $(H_1,H_2,\cdots,H_K)$及$(L_1,L_2,\cdots,L_k)$，其中 H_i，$L_i=1$, 3 或 5。$(H_1,L_1,H_2,L_2,\cdots,H_k,L_k)$為主從間交換的一連串封包的格式，且這個序列會在每隔一段時間後重複。在ρ不改變的情況下，表示沒有 bursty traffic。

很顯然的，在 *k* 增大的情況下，可以提供的ρ的值也呈現指數的成長。在下表二中，可以看出 *k* 與ρ值分佈的關係。然而，實際上只需要讓ρ值精確到一定的程度即可，在超過一定的瓶頸後，*k* 對效能的影響有限，只會徒然增加計算的複雜度。因而我們可以將 K 視為一個固定的系統參數，限制住最長的樣式長度。

而在給定兩個樣式$(H_1,H_2,\cdots,H_K)$及$(L_1,L_2,\cdots,L_k)$後，我們可以推導出一個頻寬使用的效率β。首先，先定義出一段時間 *T*，表示經過 *T* 時槽後，樣式會重複。基於儘量有效率的利用封包，我們可以得到：

$$T=\min\{\frac{f(H_1)+f(H_2)+\ldots+f(H_K)}{\lambda_H},\frac{f(L_1)+f(L_2)+\ldots+f(L_K)}{\lambda_L}\}$$

其中：

$$f(i)=\begin{cases}27 & \text{for } i=1\\ 183 & \text{for } i=3\\ 339 & \text{for } i=5\end{cases}.$$

故可以推得頻寬使用效率β：

$$\beta=\frac{\lambda_H\times T+\lambda_L\times T}{(H_1+H_2+\ldots+H_k)+(L_1+L_2+\ldots+L_k)}$$

下面將描述在已知參數λ_H及λ_L的狀況下，決定詢問樣式的機制。並評估其效率。

1)將 *t* 設定初始值為 0，*i* 設定為 1

2)令 *j* =((*i*-1)mod *k*)+1 則下一個詢問的時間 Γ_j應該為：

$$\Gamma_j=\begin{cases} max\{\frac{H_1+H_2+\cdots+H_j}{\lambda_H},\frac{L_1+L_2+\cdots+L_j}{\lambda_L}\} & \text{for } j=1,\cdots,k-1\\ min\{\frac{H_1+H_2+\cdots+H_k}{\lambda_H},\frac{L_1+L_2+\cdots+L_k}{\lambda_L}\} & \text{for } j=k\end{cases}$$

則在 $t + \Gamma_j$ 時，主裝置將選擇封包格式為 H_j 或 L_j 來詢問並傳送資料從屬裝置，相對的，從屬裝置也以 H_j 或 L_j 來回傳資料給主裝置。至選擇 H_j 或 L_j，在此主從配對的負載來決定，負載較大的一方選擇 H_j，反之則選擇 L_j。

3)如果 $j = k$，則將 t 重新設定，使 $t = t + T$（T 即為前面敘述的樣式週期），且使 $i = i+1$，並重複 2)的動作。

無限重複上面的動作，直到負載改變為止，即可決定輪詢的樣式。

為了預防 bursty traffic 造成 buffer 的溢位，可在藍芽預留的兩個位元中，增加一個溢位(overflow)位元。當系統的佇列中的資料到達預先規定的量時，即將溢位位元設為 TRUE。在發現溢位位元設為 TRUE 時，即忽略詢問策略，立刻以儘可能有效率利用的封包來處理這個主從配對。直到主從兩方的佇列皆清空，才重新回到 PMP 的策略。

2.將樣式配對的詢問策略使用在多主從配對架構(Multiple Master-Slave pair)下

在藍芽的系統中，可能會出現多主從配對架構。即一個網路中有不只一對的主裝置從裝置。在這個部分，我們希望把上述的方法延伸到多主從配對的架構中

首先依舊對每一對主從架構選擇最有效率的樣式(pattern)的使用方式。而輪詢的時機如同項目一的定義。在需要考慮有多個從屬裝置的狀況下，主裝置需要將所有的詢問依序排出，對所有的從屬裝置一一詢問。然而，針對每一對的詢問所決定的時間互相之間可能有衝突，我們以下列的規則解決之。

1)若有兩個衝突的詢問動作，比較兩個動作的第一個時槽的服務時間。第一個時槽較早的動作優先傳送。另一個動作先暫時放在佇列中。

2)如果有兩個動作擁有相同的第一時槽服務時間，則考慮兩個從屬裝置上次被詢問的時間。若從屬裝置兩次(本次和上次)被詢問間隔較短，則擁有較高的優先權來執行本次的動作。這個部分考量的是若詢問的間隔較短，可以視為較為緊急，且較容易溢位(overflow)故因較早被服務。

3)若上述兩項規則都仍無法決定，怎比較從屬裝置的 AM_ADDR，較小數字者擁有較高的優先權。而 AM_ADDR 在微網中為唯一，故最後必可做出決定。

IV. 研究成果

本計畫依時程順利進行，並獲致豐碩成果，相關論文發表如下，論文全文詳如附錄：

1. T.-Y. Lin, Y.-C. Tseng, and Y.-T. Lu, ”An Efficient Link Polling Policy by Pattern Matching for Bluetooth Piconet ” , ***The Computer Journal***, Vol. 47, No. 2, 2004, pp. 169-178. (SCIE, EI)

V. 可供推廣之研發成果資料表

□ 可申請專利　☑ 可技術移轉　日期：93年05月25日

國科會補助計畫	計畫名稱：藍芽個人無線區域網路上通訊議題之設計實作與分析 計畫主持人：曾煜棋 計畫編號：NSC 92-2213-E-009-076　學門領域：
技術/創作名稱	藍芽個人無線區域網路上通訊議題之設計
發明人/創作人	曾煜棋
技術說明	中文：針對藍芽的主從式網路架構做改善。在藍芽微網（piconet）的環境下，探討主裝置詢問（poll）從屬裝置之機制策略，減少負載欄位（payload field）沒填滿，或甚至空的數據封包，期望更有效率地使用每一個藍芽時槽（time slot），藉此提昇藍芽微網最大可容納之生產量（throughput），使得更多的通訊活動能夠同時被支援。 我們提出輪詢策略(polling policy)來增進藍芽網路的效率。
	英文：Bluetooth has a master-slave configuration called a piconet. Unspecified in the Bluetooth standard the link polling policy adopted by a master may significantly influence the bandwidth utilization of a piconet. We propose an efficient Pattern Polling policy for data link scheduling that improve the throughput of Bluetooth piconets.
可利用之產業及可開發之產品	Bluetooth 相關無線通訊設備開發業者及相關服務提供廠商。
技術特點	減少負載欄位沒填滿，或甚至空的數據封包，期望更有效率地使用每一個藍芽時槽，藉此提昇藍芽微網最大可容納之生產量，使得更多的通訊活動能夠同時被支援。
推廣及運用的價值	由於藍芽的生產量(throughput)為用以作為網路通訊的關鍵之一，本計畫所提之成果，可應用在未來行動設備的設計之中，將可使所開發之產品的應用範圍更為廣泛。

An Efficient Link Polling Policy by Pattern Matching for Bluetooth Piconets*

Ting-Yu Lin[1], Yu-Chee Tseng[1], and Yuan-Ting Lu[2]

[1]Department of Computer Science and Information Engineering
National Chiao-Tung University, Hsin-Chu, 300, Taiwan
[2]Department of Computer Science and Information Engineering
National Central University, Chung-Li, 320, Taiwan

Abstract

Bluetooth has a master-slave configuration, called a *piconet*. Unspecified in the Bluetooth standard, the link polling policy adopted by a master may significantly influence the bandwidth utilization of a piconet. Several works have been dedicated to this issue [2, 3, 4, 7, 8]. However, none of them addresses the asymmetry of traffics between masters and slaves, and the different data packet types provided by Bluetooth are not fully exploited. In this paper, we propose an efficient *Pattern Matching Polling (PMP)* policy for data link scheduling that properly resolves these deficiencies. A *polling pattern* is a sequence of Bluetooth packets of different type combinations (e.g., DH1/DH3/DH5/DM1/DM3/DM5) to be exchanged by a master-slave pair that can properly reflect the traffic ratio (i.e., asymmetry) of the pair. By judiciously selecting a proper polling pattern together with polling times for the link, the precious wireless bandwidth can be better utilized. The ultimate goal is to reduce the unfilled, or even null, payloads in each busy slot. In addition, an overflow mechanism is included to handle unpredictable traffic dynamics. Extensive simulations are presented to justify the capability of PMP in handling regular as well as bursty traffics.

Keywords: Bluetooth, home networking, Personal-Area Network (PAN), piconet, polling, wireless communication.

1 Introduction

With master-driven, short-range radio characteristics, Bluetooth [1] is a promising wireless technology for *Personal-Area Networks (PANs)*, and has attracted much attention recently [5, 6]. The

---

*This work is co-sponsored by the Lee and MTI Center for Networking Research at the National Chiao-Tung University and the MOE Program for Promoting Academic Excellence of Universities under grant numbers A-91-H-FA07-1-4 and 89-E-FA04-1-4.

smallest network unit in Bluetooth is a *piconet*, which consists of one master and one or more slaves. A piconet owns one frequency-hopping channel, which is controlled by the master in a time-division duplex manner. A time slot in Bluetooth is $625\mu s$. The master always starts its transmission in an even-numbered slot, while a slave, on being polled, must reply in an odd-numbered slot. By interconnecting multiple piconets, a larger-area network can be formed, called *scatternet*. In the literature, the scatternet performance issues are addressed in [9, 12, 14]. How to form scatternets is discussed in [10, 13, 15, 17]. In this paper, we will focus on the data link polling issue within a piconet involving one master and multiple slaves.

According to the Bluetooth protocol stack, the bottom layer is the Bluetooth Baseband, which controls the use of the radio. On top of the Baseband is the Link Manager (LM), which is responsible for link configuration and control, security functions, and power management. The corresponding protocol is called the Link Manager Protocol (LMP). The Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless datagram services to upper-layer protocols. Two major functionalities of L2CAP are protocol multiplexing and segmentation and reassembly (SAR). The SAR function segments a L2CAP packet into several Baseband packets for transmission over the air, and reassembles those at the receiving side before forwarding them to the upper layer.

Two physical links are supported in Bluetooth: ACL (Asynchronous ConnectionLess) for data traffic and SCO (Synchronous Connection-Oriented) for time-bounded voice communication. SCO voice links always have higher priority than ACL data connection does. Three SCO packets are defined: HV1, HV2, and HV3. HV stands for High-quality Voice. An HV1 packet carries 10, HV2 carries 20, and HV3 carries 30 information bytes. To achieve the specified 64 Kbps speech rate, the HV1 packet has to be delivered every two time slots, while HV2 and HV3 need to be delivered every four and six time slots, respectively. These packets are all single-slot and are transmitted over reserved intervals without going through L2CAP. The remaining slots can be used by the ACL link. Section 2.1 will detail the ACL packets. The coexistence of SCO and ACL links is modeled and evaluated in [11, 16]; the result demonstrates that the existence of SCO links does significantly reduce the data rate of ACL connections.

This paper focuses on the management of the Bluetooth ACL link involving one master and multiple slaves. Unspecified in the Bluetooth standard, the link polling policy adopted by the master may significantly influence the bandwidth utilization of a piconet. A number of works have addressed the polling issue in a piconet [2, 3, 4, 7, 8]. References [7, 8] consider the coexistence of ACL link with a SCO link (HV3). Since the HV3 link will partition time slots into a number of free segments each of 4 slots, each master-slave pair can only exchange data by 1-to-1, 3-to-1, or 1-to-3 slot patterns. According to the available patterns and the leading packet sizes at the heads of the buffers, each master-slave pair is prioritized properly, based on which the polling policy is decided. A *K-fairness* scheme is further proposed to guarantee channel access for master-slave pairs with low priorities (starvation avoidance). A learning function is proposed in [3] to predict the polling interval for each master-slave pair. So the bandwidth waste is reduced. Since the next polling time is known, the slave may go to the low-power sniff mode to save energy. Also, bounded packet delay is guaranteed. However, the learning function is pretty complex and the cost of control messages could be significant.

More practical polling policies are proposed in [2, 4]. In [2], three polling schemes are proposed: *Pure Round Robin (PRR)*, *Exhaustive Round Robin (ERR)*, and *Exhaustive Pseudo-cyclic Master queue length (EPM)*. Assuming a fixed serving order, PRR naively polls each slave sequentially. Also with a fixed order, ERR will exhaust each master-slave pair's payloads in both sides in each polling before moving onto the next slave. As to EPM, it is similar to ERR except that the polling order is dynamically adjusted in each round based on the master's queues for slaves. The SAR and polling issues are addressed in [4]. Three polling strategies are proposed: *Adaptive Flow-based Polling (AFP)*, *Sticky*, and *Sticky Adaptive Flow-based Polling (StickyAFP)*. A new *flow* bit is defined for each master-slave pair. The bit is set to true if the buffered data at any entity is above a threshold. AFP then dynamically adjusts each slave's polling interval based on the corresponding flow bit. Whenever the flow bit is 1, the polling interval is reduced to the minimum, and whenever a poll is replied by a NULL packet, the polling interval is doubled if a certain upper bound is not exceeded. The Sticky strategy defines a new parameter, *num_sticky*, to indicate the maximum number of consecutive polls that a master-slave pair can be served, under the condition that the corresponding flow bit is 1. Finally, the StickyAFP policy is a combination of the above two.

From the above reviews, we observe two deficiencies associated with existing works. First, they all fail to address the asymmetry of traffics between masters and slaves. That is, each master-slave pair may exhibit distinct traffic load in each direction. Second, the different packet types provided by Bluetooth are not fully exploited to match the traffic need.

In this paper, supposing that the traffic ratio between each master-slave pair can be approximated, we propose a *Pattern Matching Polling (PMP)* policy for ACL link scheduling. A *polling pattern* is a sequence of Bluetooth packets of different type combinations (e.g., DH1/DH3/DH5/DM1/DM3/DM5) to be exchanged by a master-slave pair. Since each Bluetooth packet has its payload efficiency, different patterns can reflect different traffic ratios of the two sides. We show how to judiciously select the polling pattern, as well as the polling time, that best matches each master-slave pair's traffic characteristics. The ultimate goal is to reduce the unfilled, or even null, payloads in each packet. As a result, the traffic asymmetry problem can be properly handled, and the precious wireless bandwidth can be better utilized. We demonstrate how to apply this policy to single- and multi-slave environment. In addition, an overflow mechanism is included to handle unpredictable traffic dynamics. This further enhances the robustness of our PMP policy to deal with bursty traffics. Extensive simulation results are presented to justify the capability of the proposed PMP policy in processing regular as well as bursty traffics.

The rest of this paper is organized as follows. Preliminaries are provided in Section 2. Section 3 proposes the PMP policy. Performance evaluation is presented in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Bluetooth Data Packets

Since our main focus is on ACL connections, we need to introduce the available packet types in Bluetooth. Table 1 summarizes all supported packet types. DM stands for Data-Medium rate, and DH for Data-High rate. DM packets are all 2/3-FEC encoded to tolerate possible transmission errors. Not encoded by FEC, DH packets are more error-vulnerable, but can carry more information.

Table 1: Summary of Bluetooth ACL data packets.

Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Bandwidth Efficiency (bytes/slo
DM1	1	0-17	2/3	yes	17
DH1	1	0-27	no	yes	27
DM3	2	0-121	2/3	yes	40
DH3	2	0-183	no	yes	61
DM5	2	0-224	2/3	yes	44
DH5	2	0-339	no	yes	67
AUX1	1	0-29	no	no	29

DM1/DH1 packets occupy one time slot, while DM3/DH3 and DM5/DH5 packets occupy three and five time slots, respectively. The AUX1 packet is similar to DH1, but has no CRC code. We define *bandwidth efficiency* as the number of payload bytes per slot. From Table 1, we see that DH5 has the highest efficiency, which is followed subsequently by DH3, DM5, DM3, AUX1, DH1, and DM1.

By monitoring the channel conditions, a Bluetooth unit can pick the proper packet types (DM or DH) to use. However, in this paper, we assume an error-free environment and only consider DH1/DH3/DH5 packets. For an error-prone environment, our PMP policy can be tailored to include DM1/DM3/DM5 packets easily.

2.2 The ACL Link Polling Problem

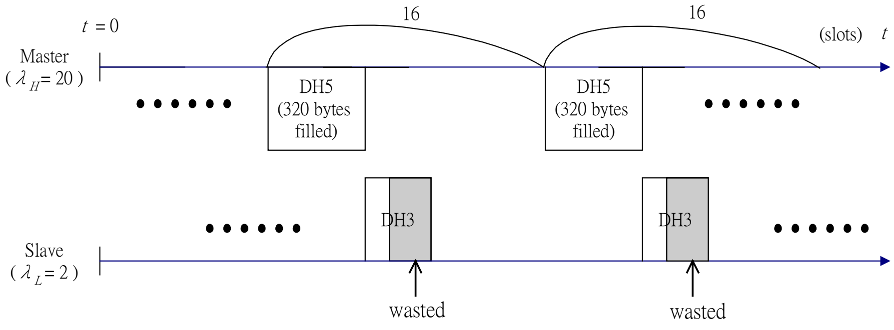


Figure 1: A naive greedy polling example.

In this work, we consider a polling problem as follows. Suppose a long-term scenario (e.g., remote data exchange through TCP) where communication traffics in both directions (up-/down-link) have been stable and approached certain average arrival rates. In a piconet, we assume that from history, or by approximation, the average traffic arrival rates of each pair of master and slave are known factors. Note that these rates are not necessarily the same for all master-slave pairs. In addition, unpredictable, but rare, bursty traffics may appear in any side. The objective is to determine a good polling policy that should be adopted by a master as well as a replying policy of a slave, when being polled. The ultimate goal is to increase bandwidth efficiency while keeping delays low.

To motivate this problem, we demonstrate a naive greedy solution below (later on we will show a better solution). Suppose that a master-slave pair has traffic loads of 20 and 2 bytes/slot in each direction. Since DH5 is most bandwidth-efficient, a greedy approach may work as follows. The master may always delay its polling time until a DH5 packet is full or close to full. A possible scenario is shown in Fig. 1, where the master always polls the slave whenever it has collected $\lfloor \frac{339}{20} \rfloor \times 20 = 320$ bytes, which fit into a DH5 packet. On the other side, the slave may have collected $\lfloor \frac{339}{20} \rfloor \times 2 = 32$ bytes, and will reply with a DH3 packet. Then the same polling pattern will be repeated every 16 time slots. As can be observed, although all forward packets are almost fully loaded, the backward packets are hardly filled, resulting in a lot of bandwidth waste. Since $16(20+2)$ bytes are delivered in every $5+3$ slots, the bandwidth efficiency is 44.

In general, suppose that the master and slave have loads of λ_H and λ_L (bytes/slot), respectively, and $\lambda_H \geq \lambda_L$. In every $\frac{339}{\lambda_H}$ slots, the master will poll the slave with a DH5 packet. In response, the slave may return $\alpha = \frac{339 \cdot \lambda_L}{\lambda_H}$ bytes with a smallest possible packet of $f(\alpha)$ slots, where

$$f(\alpha) = \begin{cases} 1 & \text{if } \alpha \leq 27 \\ 3 & \text{if } 27 < \alpha \leq 183 \\ 5 & \text{otherwise} \end{cases} .$$

Then the bandwidth efficiency is

$$\beta = \frac{339 + \alpha}{5 + f(\alpha)}. \tag{1}$$

The value of β heavily depends on λ_H and λ_L. Taking the above example, we have $\beta = 46.6$. This is still far beyond the best possible efficiency 67.8 offered by DH5.

3 The Pattern Matching Polling (PMP) Policy

The basic idea of PMP is to use different combinations of Bluetooth packet types to match the traffic characteristics of masters and slaves. For ease of presentation, only DH1/DH3/DH5 will be used (however, our result can be extended to other packet types easily).

3.1 Polling Patterns

In this subsection, we consider only one master-slave pair. Under long-term steady communication patterns, let λ_M and λ_S be their traffic loads, respectively (unit = bytes/slot). Let $\lambda_H = \max\{\lambda_M, \lambda_S\}$ and $\lambda_L = \min\{\lambda_M, \lambda_S\}$. Also, let ratio $\rho = \lambda_H/\lambda_L$. We denote by N_H and N_L the units with loads λ_H and λ_L, respectively. Note that in reality, traffic arrival is by packets, not by bytes. Our assumption is that even if traffic arrives in packets, in the long run, it will still exhibit some steady arrival pattern that can be modeled by a byte arrival process. It is based on this model that we derive our results. For simplicity, we may use numbers 1/3/5 to represent DH1/DH3/DH5 packets.

A *polling pattern* is a sequence of packet types that will be exchanged by a master-slave pair. Let k be a positive integer. A length-k pattern consists of two k-tuples: $(H_1, H_2, \ldots, H_k)$ and $(L_1, L_2, \ldots, L_k)$, where H_i, L_i = 1, 3, or 5, each representing a packet type. The former are packet types used by unit N_H, and the latter by N_L. Intuitively, the sequence of packets $(H_1, L_1, H_2, L_2, \ldots, H_k, L_k)$ will be exchanged by N_H and N_L, and the sequence will be repeated periodically, as long as the ratio ρ is unchanged and there is no bursty traffic. For instance, when length $k = 1$, there are four available patterns, as shown in Fig. 2(a), which offer four different traffic ratios. Note that other patterns not listed in the table also exist, such as $H_1 = 3$ and $L_1 = 3$. However, since the offered ratio will be equal to that of $H_1 = 5$ and $L_1 = 5$ and the bandwidth efficiency will be lower, we omit such possibility in the table. By increasing the pattern length to $k = 2$, Fig. 2(b) summaries all possible patterns. Fig. 3 illustrates the concept.

As k grows, the number of offered traffic ratios ρ will increase exponentially. On the other hand, the computational complexity to obtain all available traffic ratios also increases exponentially for larger k. In reality, we would not use a k value that is too large. This issue will be further

	Pattern 1	Pattern 2	Pattern 3	Pattern 4
N_H	(5)	(5)	(3)	(5)
N_L	(5)	(3)	(1)	(1)
Traffic Ratio (ρ_i)	$\rho_1 = 1.0$	$\rho_2 = 1.86$	$\rho_3 = 6.8$	$\rho_4 = 12.6$

(a)

	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Pattern 6
N_H	(5, 5)	(5, 5)	(5, 3)	(5, 1)	(5, 5)	(5, 3)
N_L	(5, 5)	(5, 3)	(5, 1)	(3, 1)	(5, 1)	(3, 1)
Traffic Ratio (ρ_i)	$\rho_1 = 1.0$	$\rho_2 = 1.3$	$\rho_3 = 1.43$	$\rho_4 = 1.75$	$\rho_5 = 1.86$	$\rho_6 = 2.49$

	Pattern 7	Pattern 8	Pattern 9	Pattern 10	Pattern 11
N_H	(5, 5)	(3, 1)	(3, 3)	(5, 3)	(5, 5)
N_L	(3, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)
Traffic Ratio (ρ_i)	$\rho_7 = 3.23$	$\rho_8 = 3.9$	$\rho_9 = 6.8$	$\rho_{10} = 9.7$	$\rho_{11} = 12.6$

(b)

Figure 2: Traffic ratios supported by pattern lengths (a) $k = 1$ and (b) $k = 2$.

investigated through simulations. Fig. 4 illustrates the distribution of all supported traffic ratios for $k = 1 \dots 10$. As can be expected, with a larger k, our PMP policy could be more flexible. However, note that the set of traffic ratios supported by a larger k is not necessarily a superset of that of a smaller k. Hence a longer pattern does not necessarily better match the traffic need than a shorter one.

Let K be a system parameter, which represents the largest allowable pattern length that can be used. Below, we derive the bandwidth efficiency β given a pattern $(H_1, H_2, \dots, H_k)$ and $(L_1, L_2, \dots, L_k)$, where $k \leq K$. First, we need to define a period T during which we can execute one iteration of the pattern. The basic idea is to fill the payloads of all available packets as much

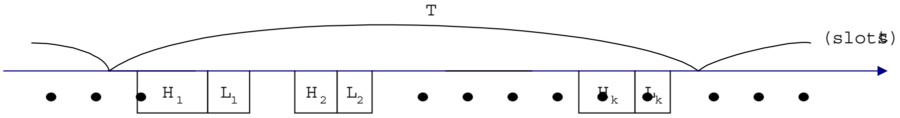


Figure 3: Illustration of our PMP policy with pattern length $= k$.

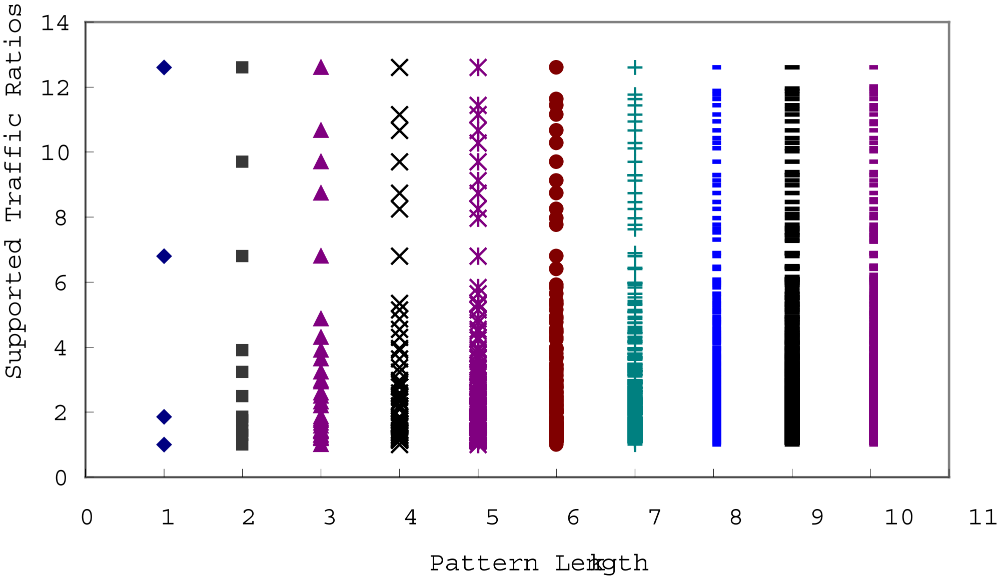


Figure 4: The distribution of supported traffic ratios for pattern lengths $k = 1 \ldots 10$.

as possible. As a result, we define T to be (unit = slots)

$$T = min\left\{\frac{f(H_1)+f(H_2)+\cdots+f(H_k)}{\lambda_H}, \frac{f(L_1)+f(L_2)+\cdots+f(L_k)}{\lambda_L}\right\}, \tag{2}$$

where

$$f(i) = \begin{cases} 27 & \text{for } i = 1 \\ 183 & \text{for } i = 3 \\ 339 & \text{for } i = 5 \end{cases}.$$

Here we take a min function because otherwise buffer overflow may occur after long time. In a period of T slots, the expected number of bytes that will be transmitted is $\lambda_H \cdot T + \lambda_L \cdot T$. Divided by the total number of slots used, the bandwidth efficiency is

$$\beta = \frac{\lambda_H \cdot T + \lambda_L \cdot T}{(H_1+H_2+\cdots+H_k)+(L_1+L_2+\cdots+L_k)}. \tag{3}$$

3.2 Polling Policy for One Master-Slave Pair

We have derived the bandwidth efficiency of a polling pattern. Given traffic loads λ_H and λ_L of a master-slave pair, we propose to choose the polling pattern that gives the highest bandwidth efficiency for use. Let $(H_1, H_2, \ldots, H_k)$ and $(L_1, L_2, \ldots, L_k)$ be the best pattern. Below, we present the corresponding polling policy. Note that here a time unit is one time slot, and we assume for simplicity that our protocol starts from slot 0.

Step 1. Initially, let $t = 0$ and $i = 1$.

Step 2. Define $j = ((i-1) \bmod k) + 1$. The next polling is expected to appear Γ_j time slots after t, where

$$\Gamma_j = \begin{cases} max\{\frac{H_1+H_2+\cdots+H_j}{\lambda_H}, \frac{L_1+L_2+\cdots+L_j}{\lambda_L}\} & \text{for } j = 1, \cdots, k-1 \\ min\{\frac{H_1+H_2+\cdots+H_k}{\lambda_H}, \frac{L_1+L_2+\cdots+L_k}{\lambda_L}\} & \text{for } j = k \end{cases}.$$

Then at time slot $t + \Gamma_j$, the master polls the slave with a proper packet type H_j or L_j (depending on whether it has the higher or lower load). In return, the slave replies with a proper packet type H_j or L_j.

Step 3. If $j = k$, then move t ahead by setting $t = t + T$, where T is as defined in Eq. (2). Finally, let $i = i + 1$ and go to Step 2.

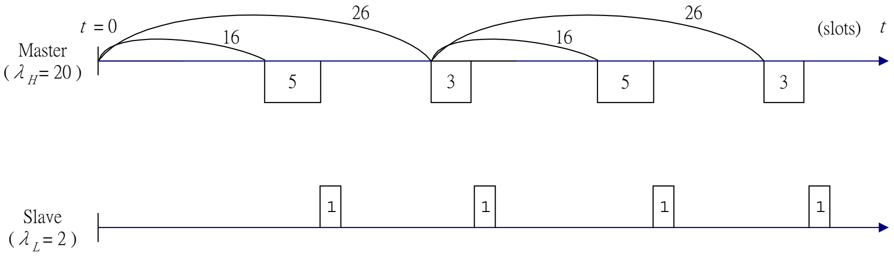


Figure 5: The PMP polling policy given $\lambda_H = 20$ for the master and $\lambda_L = 2$ for the slave ($K = 3$).

The above steps may be repeated infinitely until the master determines that the traffic loads have changed. Note that in our protocol, a master and a slave will determine their own traffic loads λ_M and λ_S. These load information can be exchanged by a user-defined control packet. Since both the master and the slave will run the same algorithm to determine the polling pattern, a consistent polling pattern will be used. So only the load information needs to be exchanged, and there is no special packet to notify the chosen polling pattern. When the traffic rate at either side changes, the master and slave should exchange with each other by piggybacking the new traffic load information. This implies that a user-defined control packet format is needed for this purpose. Then the best polling pattern for this pair should be re-determined. In this work, we do not handle misbehaving slaves. Instead, we assume cooperative slaves, which always follow the polling algorithm based on computed polling patterns.

In the polling algorithm, index i is the current number of polls being counted starting from the very beginning, while index j represents the number of polls in every polling pattern cycle. For $j = 1 \ldots k-1$, Γ_j is the time slot when both entities already have sufficient data to fill the next predicted packet type (reflected by the max function). For $j = k$, $\Gamma_j = T$ and then completes one pattern cycle. Fig. 5 illustrates how our PMP policy solves the earlier example of $\lambda_H = 20$ and $\lambda_L = 2$. Assuming $K = 3$, Eq. (3) can be used to determine the best pattern to be (5, 3) for the master, and (1, 1) for the slave. Here, $\Gamma_1 = 16$ and $\Gamma_2 = 26$. This gives a bandwidth efficiency of $\beta = 57.2$, which is about 23% better than the earlier naive greedy policy.

The above policy is derived based on an ideal assumption that the traffic pattern behaves perfectly as we predicted. However, in practice, traffics may not be as regular as we expected, and in some cases bursty traffic may appear. For this reason, we further enhance our policy by defining an *overflow* bit to prevent buffer overloading. The *overflow* bit is set to TRUE whenever an entity (master or slave) finds its buffer reaching a pre-defined threshold value. On discovering such situation happening, the entity will ignore the polling pattern and immediately sends out a DH5 packet to relieve its backlog. Here we assume that the buffer status is checked whenever an entity is scheduled to transmit data as requested by our PMP policy. In such case, the *overflow* bit will be piggybacked in the DH5 packets to inform the other entity. This *overflow* bit may be placed in one of the four reserved bits in the 2-byte payload header of DH5. The entity that does not have the overflow situation also stops its pre-defined pattern, when seeing *overflow*=1, and selects a packet type that can cover as many queued data as possible. The polling activity will be repeated in a back-to-back manner, until both sides' buffers are emptied, after which we will reset the polling pattern by letting $i = 1$ and goto Step 2. Also, we will move t to the current time slot.

3.3 Polling Policy for Multiple Master-Slave Pairs

For an environment with only one master-slave pair, bandwidth efficiency may not be an important factor, since we may have plenty of free slots and it may not be desirable to adopt the PMP policy to save bandwidth at the cost of longer packet delays due to waiting. However, for an environment with multiple master-slave pairs, bandwidth efficiency becomes more critical. How efficiently slots are utilized will significantly affect the maximum throughput that can be supported in a piconet. In Section 3.2, we first propose the polling policy for a single master-slave pair. In this section, we describe the polling policy for multiple master-slave pairs based on the approach for a single pair.

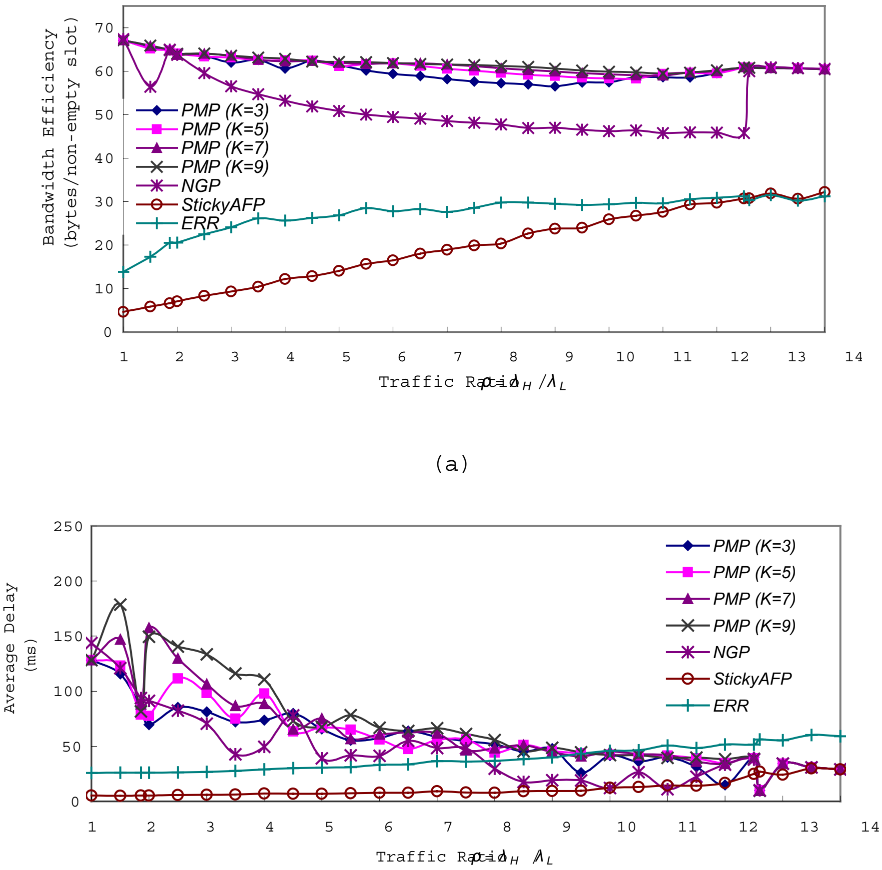
When there are multiple master-slave pairs in an ACL link, we will choose for each pair a most bandwidth-efficient pattern. From the pattern, the polling times are determined as mentioned earlier. As there are multiple master-slave pairs, the master should place all polling activities in a time line and conduct polling one by one. However, the polling activities of different master-slave pairs may overlap with each other in time. In this case, we adopt the following rules to determine the polling priorities.

- For two overlapping polling activities, we compare their leading slots. The one with an earlier leading slot will be served first. The other one will be queued and served immediately after the earlier one is completed.

- When the leading slots are the same, the last polling in a polling pattern has a higher priority. Intuitively, we consider such polling to be more urgent since it is supposed to consume all traffic loads of a master-slave pair in each pattern interval (i.e., T) to avoid buffer overflow.
- In case of ties in both of the above rules, the AM_ADDRs of slaves are compared to break the ties such that the smaller one wins.

4 Performance Evaluation

To demonstrate the effectiveness of the proposed PMP solution, we develop a C++ simulator to observe the performance. Two measurement metrics are evaluated: bandwidth efficiency and average delay time. We adopt the simulation assumptions suggested in [4] that the master keeps separate buffers for slaves, and that the buffer size for each entity is 2048 bytes. The buffer threshold to turn the overflow bit on is 80%. Each experiment lasts for 80,000 time slots. Three other policies are compared: NGP (the naive greedy protocol as described in Section 2.2), ERR [2], and StickyAFP [4]. In the ERR approach, the master can only observe its local queues without knowledge of slaves' buffer status. A control bit indicating buffer emptiness is piggybacked in slave-to-master packets, so that the master can decide to stop polling or not. In StickyAFP, the initial polling interval $P_0 = 14$ (slots), and the maximum allowable polling interval $P_{max} = 56$ (slots). The *flow* bit is set to TRUE whenever the buffer exceeds 80%. The parameter *num_sticky* is set to 16 packets as suggested in [4]. For both ERR and StickyAFP, whenever a master/slave decides to send, it will examine its queue and choose the most appropriate packet type that can consume as many bytes in its queue as possible. Traffic is modeled by a byte arrival process with a certain rate. From time to time, we also inject a large volume of data to model bursty traffic. [1]



(a)

(b)

Figure 6: Effect of traffic ratio ρ when there is one master-slave pair: (a) bandwidth efficiency and (b) average delay.

4.1 Single Master-Slave Pair without Bursty Traffic

We first simulate one master-slave pair with Poisson traffic arrival rates λ_H and λ_L (bytes/slot) at the master and slave sides, respectively. By fixing $\lambda_L = 1$, we adjust λ_H to observe how different traffic ratios affect the network performance.

Fig. 6 illustrates the bandwidth efficiency and average delay against various ratios $\rho = \lambda_H/\lambda_L$. Four values of K (3, 5, 7, and 9) for our PMP strategy are simulated. When $\rho \leq 12.6$, our PMP strategy successfully improves the bandwidth efficiency with moderate average delay. For NGP,

---

[1] We comment that the ERR and StickyAFP are designed based on a packet arrival process, but adopting a byte arrival process would not hurt their performance.

when $\rho \leq 12.6$, only three ρ's (1, 1.85, and 12.6) can be handled properly with high bandwidth efficiency. For $\rho > 12.6$, our PMP always selects the pattern $H_1 = 5$ and $L_1 = 1$, and thus acts the same as NGP. StickyAFP and ERR achieves low bandwidth efficiency due to too frequent polls and inadequate selections of packet types.

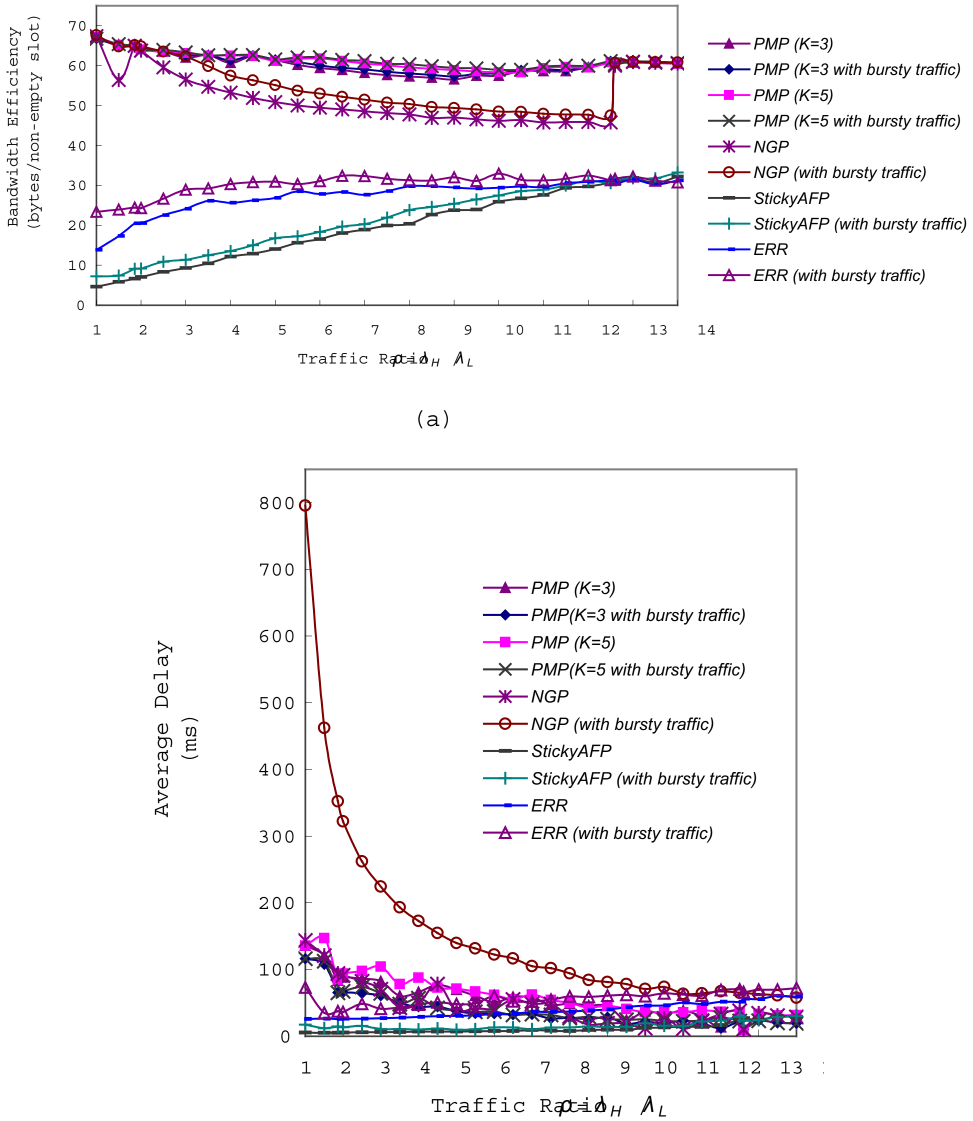
Note that for PMP, the case of $K = 7$ and $K = 9$ only slightly improves over $K = 5$ in terms of bandwidth efficiency. With $K = 3$, our PMP already outperforms other polling schemes significantly. Hence we conclude that it suffices to set K between 3 and 5 to balance between computational cost and performance.

4.2 Single Master-Slave Pair with Bursty Traffic

In this experiment, on top of the regular (Poisson) traffics at the master and slave sides, we also inject irregular bursty traffics. The bursty traffic occurs in average every 3000 slots with instant increase of $2048 \times 0.8 = 1638$ bytes to a buffer. As Fig. 7(a) shows, bursty traffic has very limited impact on our PMP. For NGP, StickyAFP, and ERR, the bandwidth efficiency gets improved, since bursty traffic helps fill those unfilled payloads. Note that, in Fig. 7(b), the delay of NGP increases sharply and remains the highest for all traffic ratios. The reason is that NGP does not implement overflow bit to handle sudden traffic burst. Due to the lack of overflow indication, NGP is unable to properly adapt to bulky data arrivals. This phenomenon is especially serious when traffic rates are low, which implies that NGP keeps the infrequent polling patterns without realizing that bursty traffic has occurred, thus resulting in long delays.

4.3 Multiple Master-Slave Pairs without Bursty Traffic

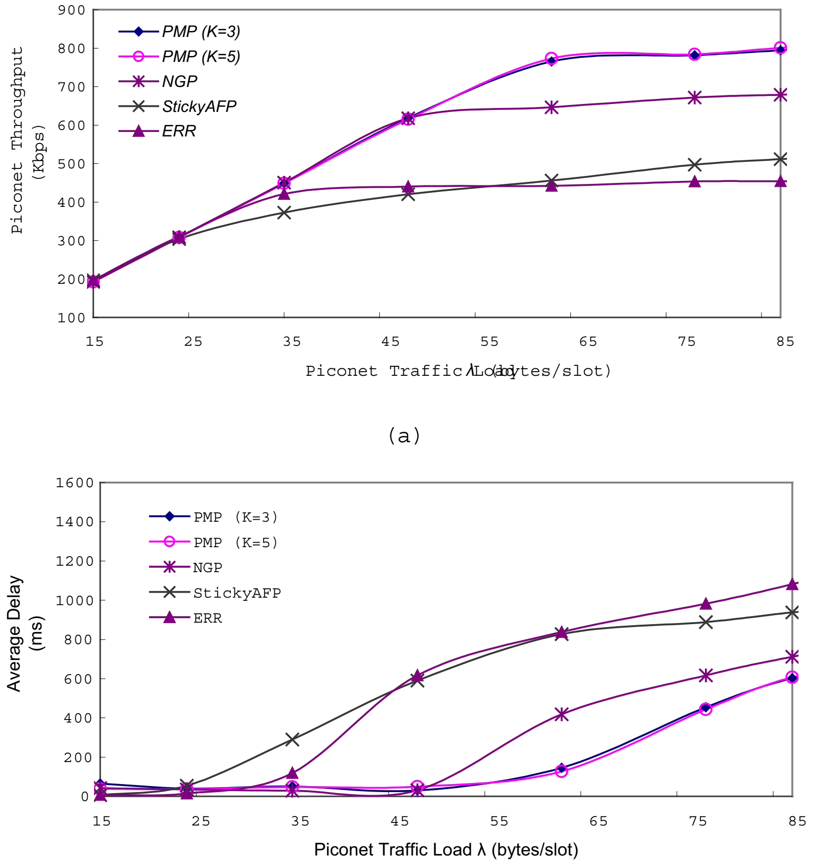
In the following experiments, we enlarge the piconet by including more slaves. Under such situation, the low efficiency of one master-slave pair may deprive the chances of other pairs from using the resource (i.e., slots), which is more likely to bring the network to the saturated level. Thus, slots should be used more cautiously. We simulate seven slaves in a piconet. The arrival rates of the seven master-slave pairs are denoted as $\lambda_{H1}/\lambda_{L1}$, $\lambda_{H2}/\lambda_{L2}$, $\ldots$, and $\lambda_{H7}/\lambda_{L7}$. To add heterogeneity, we let $\lambda_{H1}/\lambda_{L1} = 2$, $\lambda_{H2}/\lambda_{L2} = 4$, $\lambda_{H3}/\lambda_{L3} = 6$, $\lambda_{H4}/\lambda_{L4} = 8$, $\lambda_{H5}/\lambda_{L5} = 10$, $\lambda_{H6}/\lambda_{L6} = 12$, and $\lambda_{H7}/\lambda_{L7} = 14$. The total piconet traffic load λ is the sum of these rates.



(a)

(b)

Figure 7: Effect of bursty traffic when there is one master-slave pair: (a) bandwidth efficiency and (b) average delay.

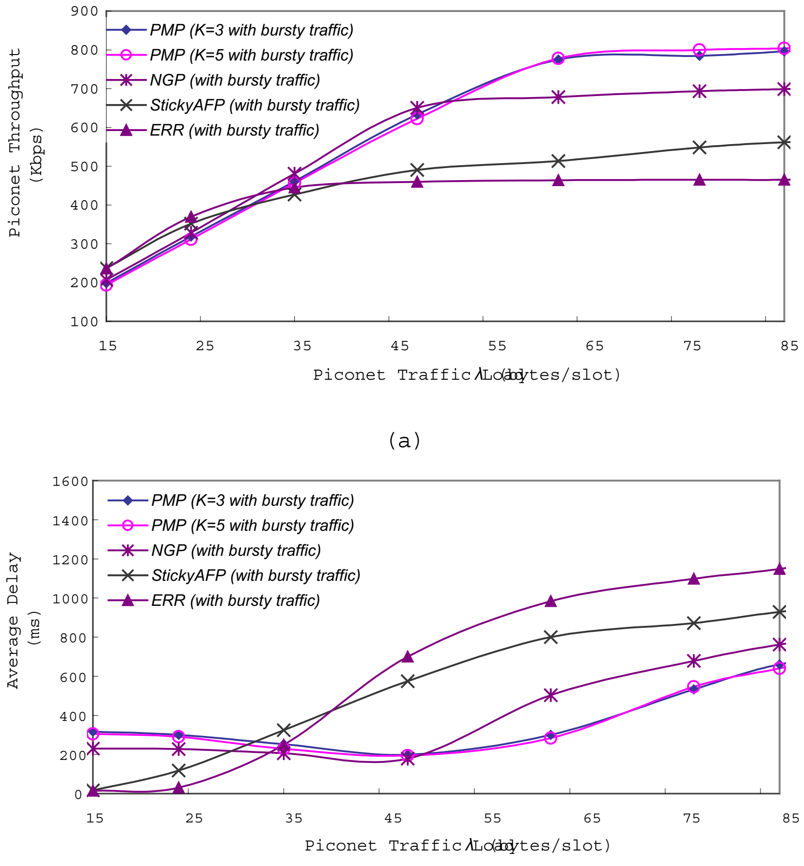


(a)

(b)

Figure 8: Piconet performance when there are multiple master-slave pairs: (a) throughput and (b) average delay.

Fig. 8 plots the piconet throughput and average delay against various total loads λ. We observe that the throughput of PMP saturates at the highest level compared to the other approaches. This is because PMP utilizes bandwidth more efficiently, thus saving more bandwidth space to accommodate more traffic. In other words, the proposed PMP effectively reduces unnecessary bandwidth waste, which improves piconet throughput. For the cases of $K = 3$ and $K = 5$, the differences are almost indistinguishable. This further confirms that a simple/short pattern length is sufficient to achieve very good performance. Note that after the saturation points, a lot of data bytes may be dropped. However, the delays of dropped bytes are not taken into account. This is why we do not see significant increase in delays in Fig. 8 after the network is saturated.



(a)

(b)

Figure 9: Effect of bursty traffic when there are multiple master-slave pairs: (a) throughput and (b) average delay.

4.4 Multiple Master-Slave Pairs with Bursty Traffic

Again, we add bursty traffic to the regular Poisson traffic for each master-slave pair. As Fig. 9 illustrates, the PMP saturates at the highest throughputs with the lowest packet delays.

4.5 Comparison of Simulation and Analytic Results

Recall that analytic predictions have been derived in Eq. (1) and Eq. (3). In Fig. 10, we compare these analytic results against simulation results, under a single master-slave pair, for PMP ($K = 3, 5, 7, 9$) and NGP. Note that it is infeasible to do this for bursty traffics. The result verifies the consistency of our previous analyses with simulations.

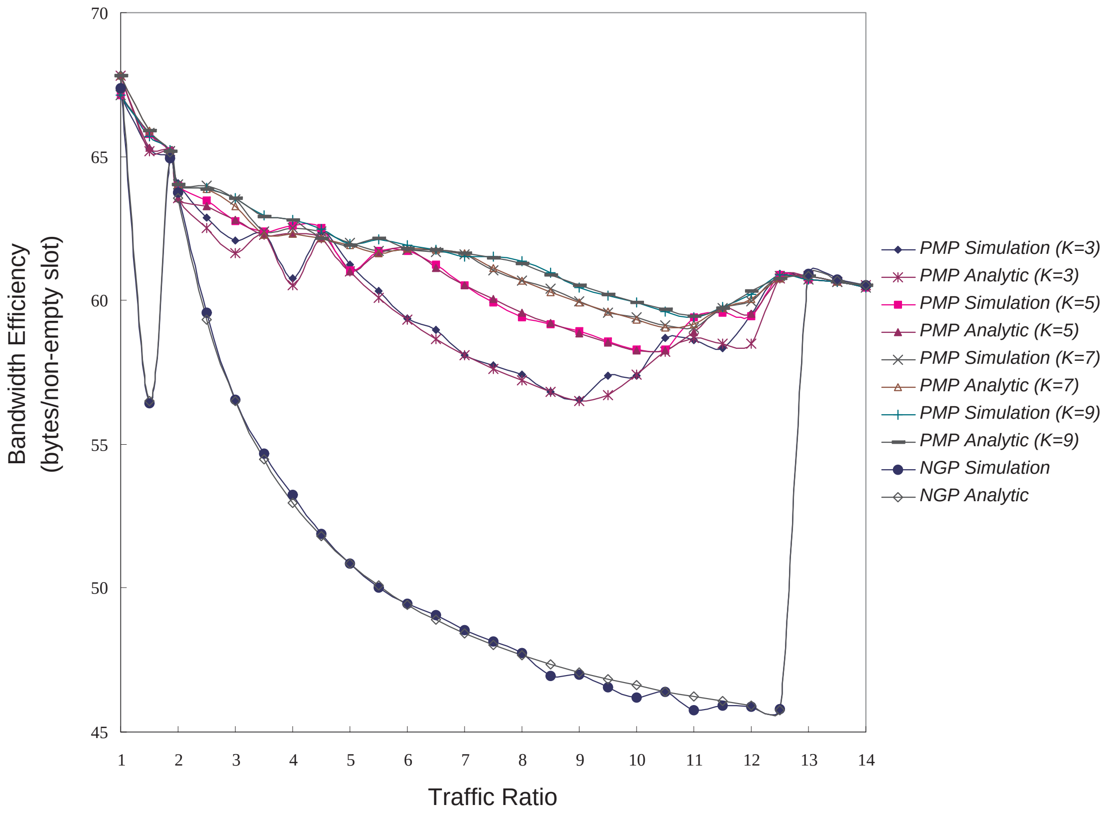


Figure 10: Comparison of simulation results and analytic values.

5 Conclusions

In this paper, we have proposed an efficient Pattern Matching Polling (PMP) policy for ACL connections in a Bluetooth piconet. For each master-slave pair, by estimating both sides' packet arrival rates, the master judiciously selects a polling pattern that can best utilize the network bandwidth. Based on the selected pattern, the master then polls the slave with proper packet types at proper time slots. In return, the slave also replies with proper packet types. The ultimate goal is to reduce the number of NULL packets and unfilled payloads so as to increase bandwidth efficiency. The PMP policy has properly addressed the asymmetry of up- and down-link traffics and the available packet types in Bluetooth. Another merit of PMP is its simplicity - a pattern length of $K = 3$ or 4 can already perform very well. So the computational complexity can be kept low.

Simulation experiments have demonstrated that the proposed PMP policy improves bandwidth efficiency and network throughput at the expense of moderate packet delays, compared to other polling approaches. In our discussion, only DH1/3/5 are considered. To include DM1/3/5, we propose to estimate the packet error probability. Whenever the probability is below a threshold, we will adopt DH1/3/5; otherwise, we will switch to DM1/3/5, and the derivation of polling patterns is similar.

In our current model, traffic is simulated by byte arrival, not packet arrival. So delay is computed based on bytes, not packets. Since we do not make explicit upper-layer traffic behavior, we were unable to translate from byte to packet delay. In order to provide further insight about the packet delay, higher-level traffic behavior must be modeled, and this may be directed to future work.

References

[1] Bluetooth Specification v1.1, http://www.bluetooth.com. February, 2001.

[2] A. Capone, M. Gerla, and R. Kapoor. Efficient Polling Schemes for Bluetooth Picocells. *IEEE Int'l Conference on Communications (ICC)*, 2001.

[3] I. Chakrabory, A. Kashyap, A. Kumar, A. Rastogi, H. Saran, and R. Shorey. MAC Scheduling Policies with Reduced Power Consumption and Bounded Packet Delays for Centrally Controlled TDD Wireless Networks. *IEEE Int'l Conference on Communications (ICC)*, 2001.

[4] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey. Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless Ad-hoc Network. *IEEE INFOCOM*, 2001.

[5] D. Groten and J. Schmidt. Bluetooth-based Mobile Ad Hoc Networks: Opportunities and Challenges for a Telecommunications Operator. *IEEE Vehicular Technology Conference (VTC)*, 2001.

[6] P. Johansson, M. Kazantzidis, R. Kapoor, and M. Gerla. Bluetooth: An Enabler for Personal Area Networking. *IEEE Network*, pages 28–37, September/October 2001.

[7] M. Kalia, D. Bansal, and R. Shorey. MAC Scheduling and SAR Policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System. *IEEE Int'l Workshop on Mobile Multimedia Communications (MoMuC)*, 1999.

[8] M. Kalia, D. Bansal, and R. Shorey. Data Scheduling and SAR for Bluetooth MAC. *IEEE Vehicular Technology Conference (VTC)*, 2000.

[9] M. Kalia, S. Garg, and R. Shorey. Scatternet Structure and Inter-Piconet Communication in the Bluetooth System. *IEEE National Conference on Communications*, New Delhi, India, 2000.

[10] C. Law, A. K. Mehta, and K.-Y. Siu. Performance of a New Bluetooth Scatternet Formation Protocol . *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001.

[11] T.-J. Lee, K. Jang, H. Kang, and J. Park. Model and Performance Evaluation of a Piconet for Point-to-Multipoint Communications in Bluetooth. *IEEE Vehicular Technology Conference (VTC)*, 2001.

[12] G. Miklos, A. Racz, Z. Turanyi, A. Valko, and P. Johansson. Performance Aspects of Bluetooth Scatternet Formation. *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2000.

[13] L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal. Clustering Algorithms for Wireless Ad Hoc Networks. *ACM DIAL M Workshop*, pages 54–63, 2000.

[14] T. Salonidis, P. Bhagwat, and L. Tassiulas. Proximity Awareness and Fast Connection Establishment in Bluetooth. *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2000.

[15] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. *IEEE INFOCOM*, 2001.

[16] V. Sangvornvetphan and T. Erke. Traffic Scheduling in Bluetooth Network. *IEEE Int'l Conference on Networks (ICON)*, 2001.

[17] G. V. Zaruba, S. Basagni, and I. Chlamtac. Bluetrees - Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks. *IEEE Int'l Conference on Communications (ICC)*, 2001.