行政院國家科學委員會專題研究計畫　成果報告

藍芽個人無線區域網路上通訊議題之設計實作與分析(3/3)

計畫類別：個別型計畫
計畫編號：NSC94-2213-E-009-004-
執行期間：94 年 08 月 01 日至 95 年 07 月 31 日
執行單位：國立交通大學資訊工程學系(所)

計畫主持人：曾煜棋

報告類型：完整報告

處理方式：本計畫可公開查詢

中　華　民　國 95 年 10 月 24 日

摘要

藍芽（Bluetooth）技術是一個相當普及之個人區域網路標準，在無線通訊領域發展中扮演著重要的角色；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。在第一年當中，針對藍芽的主從式網路架構做改善。在藍芽微網的環境下，探討主裝置詢問（poll）從屬裝置之機制策略，儘量填滿所有的負載欄位（payload field），以及空的數據封包，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。

第二年的工作重點在於將數個微網連結成一個更大的網路結構，我們提出一個藍芽散網的環形結構，稱為藍環(BlueRing)。進而探討並實作藍環之形成(formation)，繞徑(routing)、維持管理(maintenance)以及提供一種回復機制（recovery mechanism），使藍環網路拓樸結構具有容錯能力，在出現錯誤時能迅速回復網路結構。此外，我們針對藍芽標準 V1.1 版及 V1.2 版，分析其頻率匹配(frequency-matching)的延遲時間，並針對分析的結果，提出了三種可以提升藍芽裝置搜尋時間的方法。

計畫的第三年中，我們研究在多個微網共存的環境（multi-piconet environment）中，從數學機率分析的角度，來評估因共同頻道干擾（co-channel interference）而導致封包碰撞（packet collision）所造成對整體網路效能的影響程度，以供藍芽網路佈署時之參考依據。

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(piconet)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；本計畫執行的第一年，我們在微網的環境下，探討主裝置（poll）從屬裝置之機制策略。我們提出一個***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）。現有藍芽規格中也未清楚定義如何將數個微網連結起來可以形成一個更大的網路結構，端視各家廠商如何設計。因此，如何規劃一個簡單又有通訊效率的藍芽散網（scatternet）拓樸結構，便成為藍芽是否能成為個人無線區域網路上主要標準技術元件的關鍵所在;在本計畫執行的第二年中，我們將提出一個藍芽散網之環形結構，稱為***藍環***（BlueRing），它能將數個微網以環狀的方式串接起來，並研究藍環（BlueRing）之形成（formation）、繞徑（routing）、及維運管理（maintenance）機制。在本計畫執行的第三年，我們研究在多個微網共存的環境（multi-piconet environment）中，不論這些微網有沒有彼此連結成一個散網，都無可避免地會遭遇到因***頻道干擾***（co-channel interference）而導致封包碰撞（packet collision）的問題，我們試著從數學機率分析的角度，來評估因為頻道干擾所造成對整體網路效能的影響程度，並配合先前在計畫第二年中實作出來具通訊能力之藍芽散網，提供理論與實測數據結果，以供將來藍芽網路佈署時有用之參考依據。最後，我們結合了藍芽及行動電話網路，實作出一套訪客系統，實際利用藍芽通訊的優點，將之實現在日常生活中。

II. 研究目的

本計畫第一年的工作重點在於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，期望藉由更有效率地使用每一個時槽（time slot）來提昇藍芽微網最大可容納之生產量（throughput)。

第二年的工作重點在於將數個微網連結成一個更大的網路結構，稱為散網（scatternet)。這個部分目前藍芽現有的標準規格裡並沒有清楚的定義，端視各家廠商如何設計。而在藍芽設備日益增加的情況下，欲延伸藍芽的使用，有效率地建立與維持藍芽散網將是關鍵技術。我們提出一個藍芽散網的環形拓樸結構，稱為藍環(BlueRing)。進而探討並實作藍環之形成(formation)，繞徑(routing)、維持管理(maintenance)以及提供一種回復機制（recovery mechanism)，使藍環網路拓樸結構具有容錯能力，在出現錯誤時能迅速回復網路結構。此外，針對藍芽搜尋裝置（device discovery）的時間過長的缺點，我們也提出分析和解決的方法。

在計畫的第三年度，我們研究在多個微網共存的環境（multi-piconet environment）中，從數學機率分析的角度，來評估因為共同頻道干擾（co-channel interference）而導致封包碰撞（packet collision）所造成對整體網路效能的影響程度，以供將來藍芽網路佈署時有用之參考依據。

III. 研究方法

計畫第一年：樣式配對和輪詢策略

我們提出一個有效率的樣式配對（Pattern Matching）的輪詢策略，適當地解決了其他相關研究效率的不足。輪詢樣式（Polling Patern）是一對主裝置和從屬裝置間產生不同種類的藍芽封包的組合(e.g., DH1/DH3/DH5/DM1/DM3/DM5)產生的序列，這個序列可以適當地反應出這對裝置流量的比例（比如說流量的不對稱性)。藉由明智地選擇適當的論詢樣式以及裝置連接的輪詢時刻，藍芽的頻寬可以被充分的被利用。另外，封包大小所能載送之資料量並不同，我們有效解決上／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之生產量（throughput)，使得更多的通訊活動能夠同時被支援。此外，我們還提出溢位機制（Overflow Mechanism）以解決無法預測的流量產生的變化。對於一般的流量或是流量擁擠的網路，我們進行了大規模的模擬來驗證樣式配對論詢策略的效能。

計畫第二年：藍環和裝置搜尋

藍芽的標準規格並未清楚定義如何將多個微網結合成一個大的散網。在較大規模的散網內，由於藍芽基頻（Baseband）層和介質存取控制（Medium Access Control）層的特性，直接採用現有的繞徑（routing）演算法會產生一些問題。我們提出了稱為藍環（BlueRing）的網路拓樸，它能將數個微網以環狀的方式串接起來，成為一個 ring of trees 之架構。連接微網與微網之藍芽主機稱為橋接器（bridge），橋接器負責跨微網（inter-piconet）之封包轉送；在這項研究裡我們也探討藍環的形成（formation）、繞徑以及維護管理（maintenance）等議題。藍環網路在中型的網路（約 50~70 個裝置）是有延展性的，而不會像樹狀或星狀網路在接近根點（root）時會形成瓶頸（bottleneck）的問題。此外，藍環網路具有容錯及回復機制（recovery mechanism），能夠在網路中的裝置失去功能時繼續維持網路架構。

另外，藍芽搜尋裝置的時間過長，這樣可能會影響一些藍芽上的應用以及影響藍芽通訊的效率。我們分析了藍芽 V1.1 及 V1.2 頻率匹配（frequency matching）的延遲，並且提出三個可以加速搜尋藍芽裝置的方法。分別為：

- Half Inquiry Interval (HII)：將詢問的週期縮減為一半，其效果可以減少前述的等待時間。
- Dual Inquiry Scan (DIS)：從屬裝置將其做詢問頻率增為兩倍，可以增加每次詢問時窗會被詢問到的機會，因而大量地減少頻率匹配的延遲。
- Combination of HII and DIS：結合上述兩種方法。

計畫第三年：多微網間封包碰撞頻率分析及藍芽訪客系統的實作

藍芽使用 2.4GHz 的 ISM 頻帶，在多個微網共存的環境中，各個微網必定會遭遇封包碰撞（collision）的情形，預測在多微網環境中封包碰撞的頻率是相當值得探討的問題。之前相關的研究僅針對單一時槽封包，然而藍芽規格中還提供了 3 個及 5 個時槽封包的傳輸，這樣的結果顯然是受限制的。我們將多時槽封包的情形納入考量，提出更加一般化的分析，有助於在實際佈建多微網的藍芽網路時提供較準確的參考。

最後，我們結合了目前相當普及的藍芽和環球行動通訊系統（GSM），提出一種以事件導向為主並整合行動電話與感測器（sensor）的網路，並以此架構實作出一套訪客系統。簡訊是行動電話上非常重要的運用，然而傳統的簡訊必須靠簡單的人為動作觸發，限制了簡訊的用途。在整合行動電話與感測器的網路的訪客系統中，我們利用感測器事件感知的能力來判斷訪客的進／出某個區域的事件，並且自動以簡訊告知主人，減少了約會時不必要的等待時間。這個系統的實作也展示出整合異質網路應用的潛力。

IV. 研究成果

計畫第一年：樣式配對和輪詢策略

在這項研究中，我們提出一個有效的樣式匹配輪詢（Pattern Matching Polling，PMP）策略。藉由估計每對藍芽主從裝置的封包到達頻率，我們可以在主裝置端找出一個輪詢式樣（polling pattern），有效地使用網路的頻寬。根據這個輪詢式樣，主裝置可以在適當的時間用適當的封包種類來輪詢從屬裝置；同樣地，從裝置也可以根據這個方法回覆適當種類的封包。PMP 策略適當地指出了上傳（uplink）和下傳（downlink）流量的不對稱。PMP 策略的另一個優點是它的計算複雜度低，在樣式長度（pattern length）為 3 或 4 時就有很好的效能。模擬的結果顯示，和其他的輪詢演算法相比，PMP 策略使藍芽網路能較有效率地使用頻寬，同時兼顧適當的封包延遲。

計畫第二年：藍環和裝置搜尋

我們提出了稱為藍環（BlueRing）的網路拓樸，用於連接藍芽微網，以及設計了藍環的形成（formation）、繞徑（routing）以及維護管理網路拓樸的方法。由於藍環的簡單性和規律性，藍環上的繞徑不需要各裝置維持繞徑表格（routing table），也不需要傳的繞徑發現的動作，非常適合手機等較簡單的行動裝置。此外，藍環的修復機制（recovery mechanism）也能夠在數個裝置損壞或是異常時重新連接網路。分析及模擬結果指出，藍環比起其他建構散網的方法提供較高的最大可容納之生產量（throughput）及較低的封包延遲（packet delay）。在縮短藍芽裝置的搜尋時間方面，結果證明結合 HII 和 DIS 的方法充分地減少了延遲。在使用結合 HII 和 DIS 的方法下，期望的頻率匹配延遲從平均的 23.55 秒降至 11.38 秒。

計畫第三年：多微網間封包碰撞頻率分析及藍芽訪客系統的實作

我們對於多微網間封包碰撞頻率分析進行了模擬，模擬的結果證明了這項分析的正確性。單時槽的封包很明顯地有較低的碰撞頻率，然而多時槽封包在傳輸上更有效率，因此我們將網路的最大容納流量做為評估的重點。最後模擬的結果證明，在流量負載皆為 70% 的狀況之下，使用較多的多時槽封包的表現得比使用較少多時槽封包好。

最後，我們成功地實作出結合藍芽與行動電話的訪客系統的原型，並且對這套系統從偵測到事件至告知主人的延遲進行分析和模擬。我們的系統也具有模組化的優點，可以讓程式設計者根據應用調整他們的系統，而不影響到底層和基本的架構，這樣的彈性在實作整合異質網路的系統是相當關鍵的。

V. 結論

本計畫進行順利，成果豐碩，相關論文發表如下：

- T.-Y. Lin, Y.-C. Tseng, and Y.-T. Lu, “An Efficient Link Polling Policy by Pattern Matching for Bluetooth Piconets”, ***The Computer Journal***, Vol. 47, No. 2, 2004, pp. 169-178. (SCIE, EI)
- T.-Y. Lin, Y.-C. Tseng, and K.-M. Chang, “A New BlueRing Scatternet Topology for Bluetooth with Its Formation, Routing, and Maintenance Protocols”, ***Wireless Communications and Mobile Computing***, Vol. 3, No. 4, June 2003, pp. 517-537. (SCIE)
- J.-R. Jiang, B.-R. Lin, and Y.-C. Tseng, "Analysis of Bluetooth Device Discovery and Some Speedup Mechanisms", ***Int'l J. of Electrical Engineering***, Vol. 11, No. 4, Nov. 2004, pp. 301-310. (EI)
- T.-Y. Lin and Y.-C. Tseng, “Collision Analysis for a Multi-Bluetooth Picocells Environment”, ***IEEE Communications Letters***, Vol. 7, No. 10, Oct. 2003, pp. 475-477. (SCI, EI)
- Y.-C. Tseng, T.-Y. Lin, Y.-K. Liu, and B.-R. Lin, "Event-Driven Messaging Services over Integrated Cellular and Wireless Sensor Networks: Prototyping Experiences of a Visitor System", ***IEEE J. on Selected Areas in Communications***, Vol. 23, No. 6, June 2005, pp. 1133-1145. (SCI, EI)

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	Bandwidt Efficienc (bytes/sl
DM1	1	0-17	2/3	yes	1
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

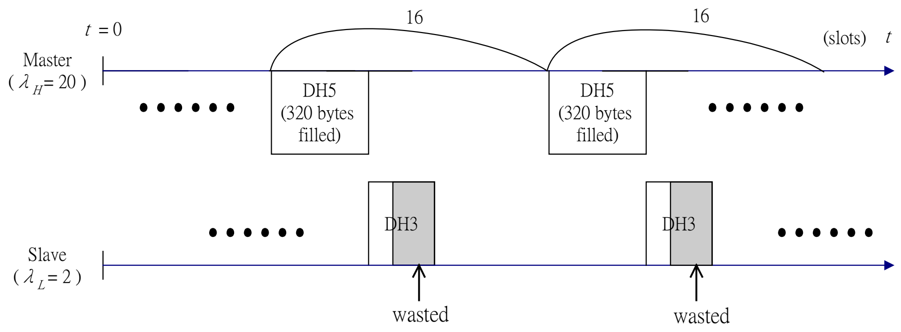


Figure 1: A naive greedy polling example.

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

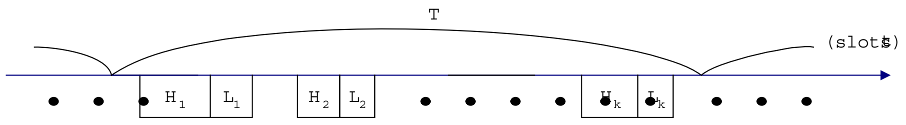


Figure 3: Illustration of our PMP policy with pattern length = k.

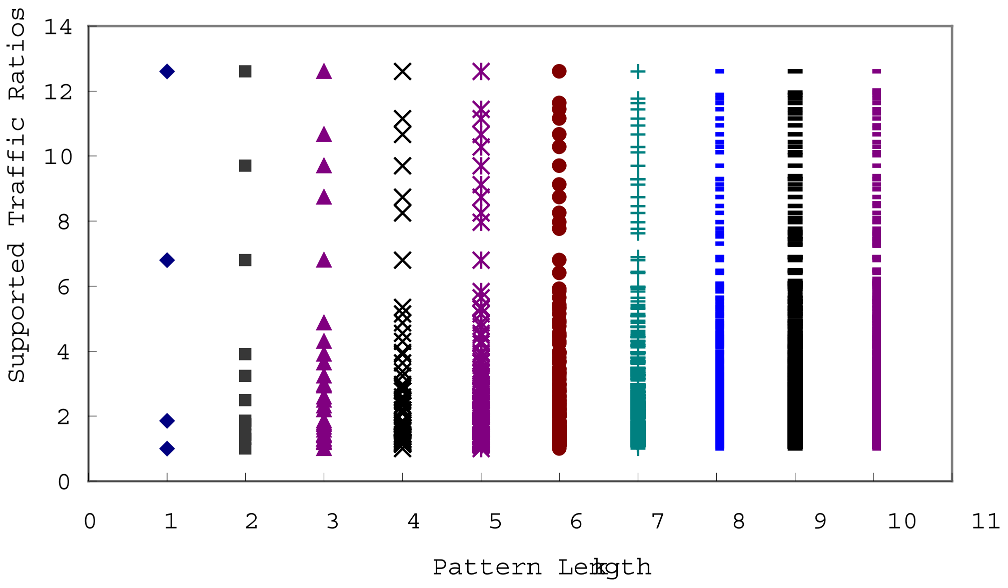


Figure 4: The distribution of supported traffic ratios for pattern lengths $k = 1 \dots 10$.

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.

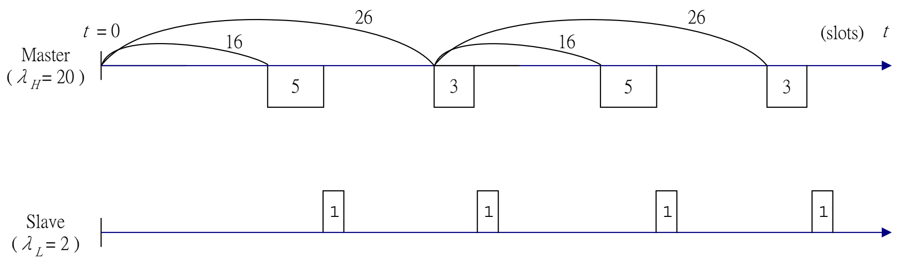


Figure 5: The PMP polling policy given $\lambda_H = 20$ for the master and $\lambda_L = 2$ for the slave ($K = 3$).

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.

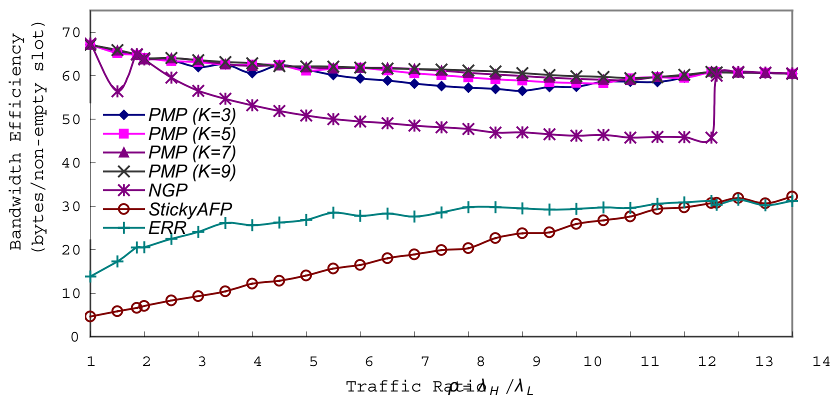
- For two overlapping polling activities, we compare their leading slots. The one with an earlier leading slot will be served first. The other one will be queued and served immediately after the earlier one is completed.

- 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.

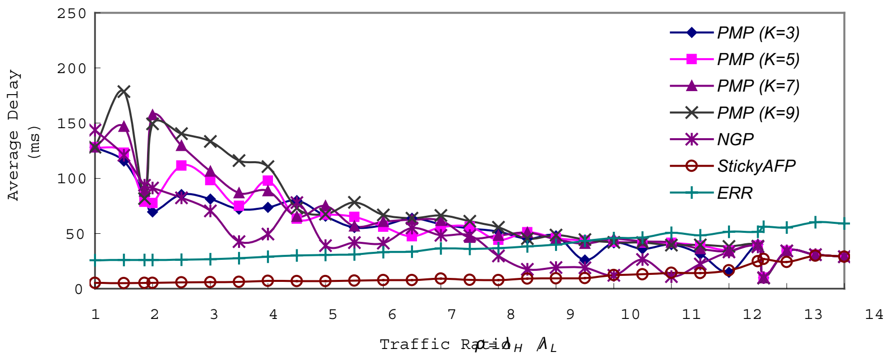
- In case of ties in both of the above rules, the AM_ADDRs of slaves are compared to break the ties such that the smaller one wins.

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.

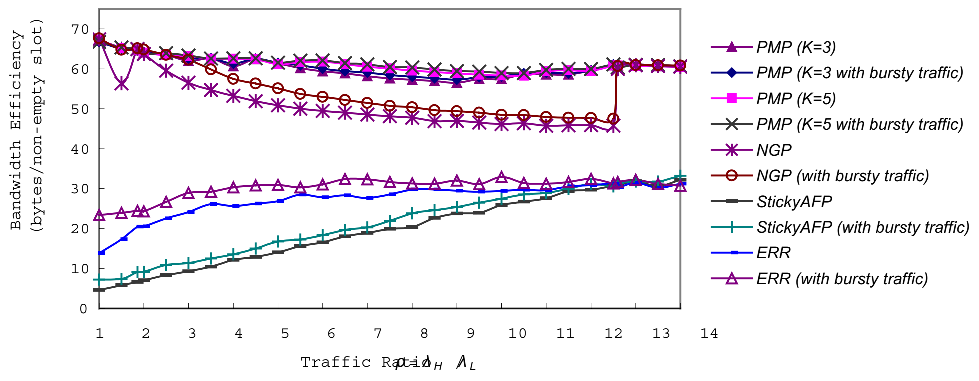
Note that for PMP, the case of $K = 7$ and $K = 9$ only slightly improves over $K = 5$ in terms of bandwidth efficiency. With $K = 3$, our PMP already outperforms other polling schemes significantly. Hence we conclude that it suffices to set K between 3 and 5 to balance between computational cost and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

4.2 Single Master-Slave Pair with Bursty Traffic

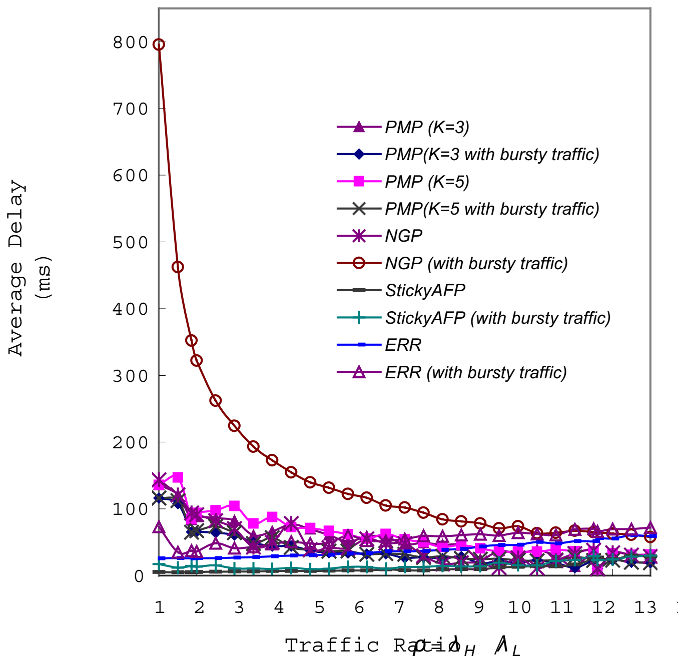
In this experiment, on top of the regular (Poisson) traffics at the master and slave sides, we also inject irregular bursty traffics. The bursty traffic occurs in average every 3000 slots with instant increase of $2048 \times 0.8 = 1638$ bytes to a buffer. As Fig. 7(a) shows, bursty traffic has very 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.

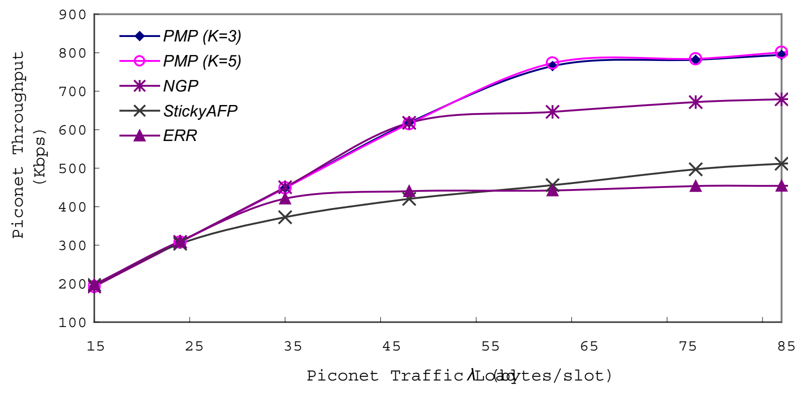


(a)

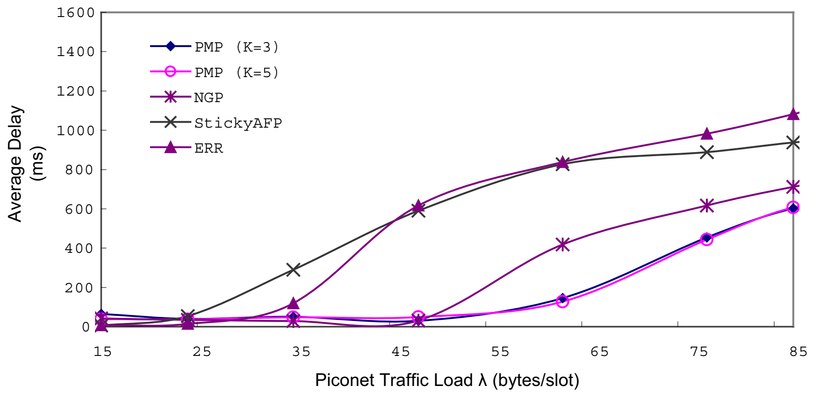


(b)

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



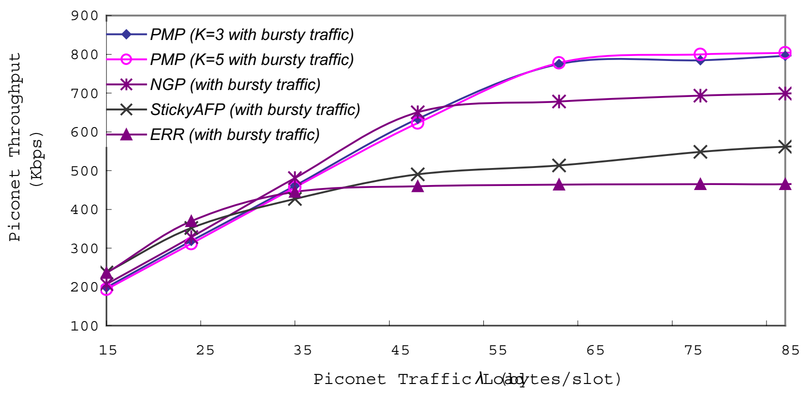
(a)



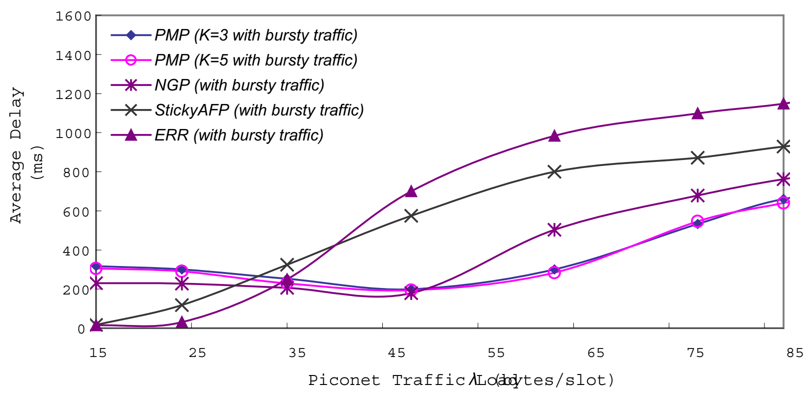
(b)

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.

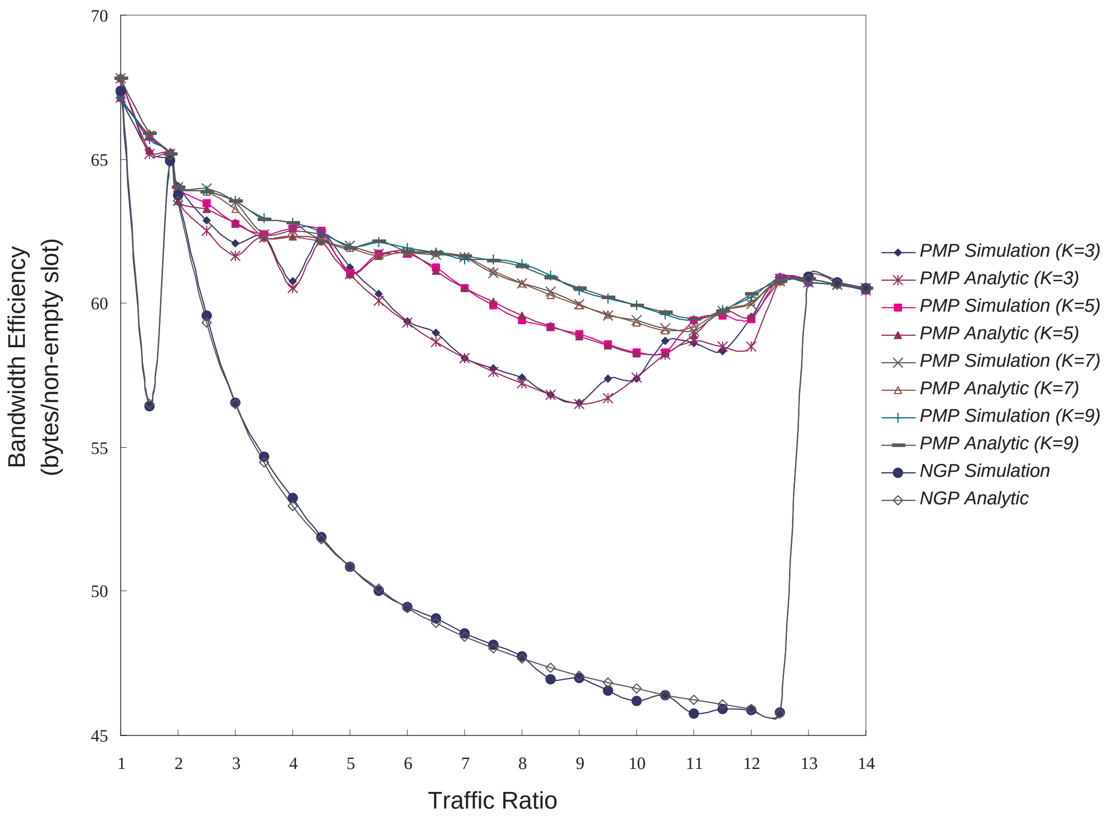


Figure 10: Comparison of simulation results and analytic values.

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.

Formation, Routing, and Maintenance Protocols for the BlueRing Scatternet of Bluetooths*

Ting-Yu Lin[1], Yu-Chee Tseng[1], Keng-Ming Chang[2] and Chun-Liang Tu[3]

[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
[3]CCL, Industrial Technology Research Institute, Taiwan

Abstract

The basic networking unit in Bluetooth is *piconet*, and a larger-area Bluetooth network can be formed by multiple piconets, called *scatternet.* However, the structure of scatternets is not defined in the Bluetooth specification and remains as an open issue at the designers' choice. It is desirable to have simple yet efficient scatternet topologies with well supports of routing protocols, considering that Bluetooths are to be used for *personal-area networks* with design goals of simplicity and compactness. In the literature, although many routing protocols have been proposed for *mobile ad hoc networks*, directly applying them poses a problem due to Bluetooth's special baseband and MAC-layer features. In this work, we propose an attractive scatternet topology called *BlueRing* which connects piconets as a ring interleaved by bridges between piconets, and address its formation, routing, and topology maintenance protocols. The BlueRing architecture enjoys the following nice features. First, routing on BlueRing is stateless in the sense that no routing information needs to be kept by any host once the ring is formed. This would be favorable for environments such as Smart Homes where computing capability is limited. Second, the architecture is scalable to median-size scatternets easily (e.g., around 50$\sim$70 Bluetooth units). In comparison, most star- or tree-like scatternet topologies can easily form a communication bottleneck at the root of the tree as the network enlarges. Third, maintaining a BlueRing is an easy job even as some Bluetooth units join or leave the network. To tolerate single-point failure, we propose a protocol-level remedy mechanism. To tolerate multi-point failure, we propose a recovery mechanism to reconnect the BlueRing. Graceful failure is tolerable as long as no two or more critical points fail at the same time. As far as we know, the fault-tolerant issue has not been properly addressed by existing scatternet protocols yet. In addition, we also evaluate the ideal network throughput at different BlueRing sizes and configurations by mathematical analysis. Simulations results are presented, which demonstrate that BlueRing outperforms other scatternet structures with higher network throughput and moderate packet delay.

*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. Part of this work was done while Y.-C. Tseng was supported by the Institute of Information Science, Academia Sinica, Taiwan, as a visiting scholar.

1 Introduction

Wireless communication is perhaps the fastest growing industry in the coming decade. It is an enabling technology to make computing and communication anytime, anywhere possible. Depending on whether base stations are established or not, a wireless network could be classified as *infrastructure* or *ad hoc.* According to the radio coverage and communication distance, it can be classified as *wide-area*, *local-area*, *personal-area*, or even *body-area.*

This paper focuses on Bluetooth [1], which is an emerging PAN (Personal Area Network) technology, and is characterized by indoor, low-power, low-complexity, short-range radio wireless communications with a frequency-hopping, time-division-duplex channel model. Main applications of Bluetooths are targeted at wireless audio link, cable replacement, and ad hoc networking. The basic networking unit in Bluetooth is called *piconet*, which consists of one master and up to seven active slaves. For a larger wide-spread deployments, multiple piconets can be used to form a *scatternet.* A host may participate in two piconets to relay data, to which we refer as a *bridge* in this paper.

In the Bluetooth specification, the structure of scatternets is not defined, and it remains as an open issue at the designers' choice. In the literature, although many routing protocols have been proposed for *mobile ad hoc networks* based on wireless LAN cards [9], directly applying them poses a problem due to Bluetooth's special baseband and MAC-layer features [2]. It is desirable to have simple yet efficient scatternet topologies with well supports of routing protocols, considering that Bluetooths are to be used for PAN with design goals of simplicity and compactness. According to [3, 4], Bluetooth-based mobile ad hoc networks need routing protocols closely integrated with underlying scatternet topologies. The reason stems from the physical and link-level constraints of Bluetooth technology, making legacy routing protocols for mobile ad hoc networks (e.g., [9]) unsuitable for scatternets.

Several previous papers [5, 8, 11] have addressed the performance issues, which motivate studies of the scatternet formation problem. References [6, 10, 12, 13] propose various scatternet formation mechanisms (refer to the review in Section 2.1). However, all these works fail to provide clear and efficient routing protocols to run over the proposed scatternet topologies. Until recently, reference [7] proposes a routing protocol based on a 2-level hierarchical scatternet. Two types of local networks are defined: *PAN (Personal Area Network)* and *RAN (Routing Area Network)*. RAN is responsible of interconnecting PANs. All traffic from a PAN needs to go through the RAN to reach another PAN. However, this approach suffers from two drawbacks. First, the number of participating Bluetooth units is limited. Second, the RAN may become the bottleneck of the whole network, in terms of both communication delays and fault tolerance capability.

In this work, we propose an attractive topology called *BlueRing* for scatternet structure, and address its formation, routing, and maintenance protocols. While similar to the IEEE 802.5 token-ring in topology, our BlueRing differs from token ring in several aspects due to Bluetooth's special baseband features. First, the ring consists of multiple piconets with alternating masters and slaves and thus can de facto be regarded as a *ring of trees* since each master can connect to multiple active slaves. Second, no token is actually running on the ring. Third, since each piconet has its unique frequency hopping sequence, multiple packets may be relayed on the ring simultaneously. Routing protocols to support unicast and broadcast on BlueRings are proposed. For bridges (slaves connecting two piconets), a bridging policy is clearly defined so as to relay packets efficiently.

The BlueRing architecture enjoys the following nice features. First, routing on BlueRing is stateless in the sense that no routing information needs to be kept or constructed by any host once the ring is formed. This would be favorable for environments such as Smart Homes where computing capability is limited. Second, the architecture is scalable to median-size scatternets (e.g., around 50∼70 Bluetooth units). In comparison, most star- or tree-like scatternet topologies can easily form a communication bottleneck at the root of the tree as the network enlarges. Third, maintaining a BlueRing is an easy job even if some bluetooth units join or leave the network. To tolerate single-point failure, we propose a protocol-level remedy mechanism. To tolerate multi-point failure, we propose a recovery mechanism to reconnect the BlueRing. Graceful failure is tolerable as long as no two or more critical points fail at the same time. As far as we know, the fault-tolerant issue has not been properly addressed by existing scatternet protocols yet.

The rest of this paper is organized as follows. Preliminaries are in Section 2. The formation, routing, and maintenance protocols for BlueRing are proposed in Section 3, Section 4, and Section 5, respectively. In Section 6, we present some analysis and simulation results. Finally, Section 7 summarizes the paper and points out our future work.

2 Preliminaries

Bluetooth is a master-driven, short-range radio wireless system. The smallest network unit is a *piconet*, which consists of one master and up to 7 active slaves. Each piconet owns one frequency-hopping channel, which is controlled by its 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. Four important operational modes are supported by Bluetooth: *active*, *sniff*, *hold*, and *park*. The active mode is most energy-consuming, where a bluetooth unit is turned on for most of the time. The sniff mode allows a slave to go to sleep and only wake up periodically to check possible traffic. In the hold mode, a slave can temporarily suspend supporting data packets on the current channel; the capacity can be made free for other things, such as scanning, paging, inquiring, and even attending other piconets. Prior to entering the hold mode, an agreement should be reached between the master and slave on the hold duration. When a slave does not want to actively participate in the piconet, but still wants to remain synchronized, it can enter the park mode. The parked slave has to wake up regularly to listen to the beacon channel, for staying synchronized or checking broadcast packets.

This paper proposes a new topology called BlueRing for scatternet structure. Since a scatternet must involve multiple piconets, some devices must participate in more than one piconet. Such devices are called *bridges* in this paper, and a bridging policy is needed for them to efficiently relay packets from piconets to piconets. A bridge host has to frequently pause activities in one piconet and switch to another piconet. In this paper, we propose to adopt the park mode for this purpose.

Below we give the reason why we choose park mode. The Bluetooth specification provides three options for a device to temporarily pause its current activity: sniff, hold, and park modes. The sniff mode has a periodical, prearranged wakeup pattern, and thus is more suitable for a device to switch from piconets to piconets with a regular pattern. It is not selected here because with a regular pattern time slots may easily get wasted. Moreover, with our BlueRing, which chains a sequence of piconets, determining a good sniffing pattern is very difficult. The hold mode would be favorable if the amount of time that a bridge should stay in each piconet can be predetermined. Unfortunately, this assumption is unrealistic, especially in a dynamic environment. The park mode is more favorable in our case since it allows a device to temporarily give up its current activity in one piconet for an arbitrary period of time until an unpark request is issued. The unpark request can be master-activated or slave-activated, but should be approved by the master. Hence, we adopt the park mode in our bridging policy, considering its simplicity and flexibility.

2.1 Review of Scatternet Formation Algorithms

Below, we review some existing scatternet formation schemes. Scatternet formation is explored in [6, 10, 12, 13]. In [10], a 2-stage distributed randomized algorithm is proposed to form a network of star-shaped clusters, where each cluster is a piconet with at most 7 active slaves. The goal is to maximize the number of nodes into each piconet so that the number of clusters is minimized. However, how these piconets are interconnected is not addressed. A similar work is in [13], where a fast scatternet formation algorithm is proposed to connect piconets as a tree. How to form a tree-like scatternet with bounded time and message complexities is presented in [6]; there is no limitation on the number of participant nodes. Clearly, the center/root host in a star/tree scatternet can become a communication bottleneck of the network. Furthermore, designing fault-tolerant routing protocols on a star/tree-like network is a difficult job since any single fault will partition the network. In [12], assuming that all devices are within each other's radio coverage, a fully-connected scatternet is constructed such that connectivity exists between each pair of piconets. At most 36 Bluetooth devices can participate in the scatternet.

We note that all the above works [6, 10, 12, 13] do not clearly address the corresponding routing and bridging protocols to be run over the proposed scatternets. While there is no standard criteria for good scatternet topologies, we conclude some guidelines for scatternet construction:

- The number of piconets should be kept as small as possible, so as to reduce inter-piconet interference and communication complexity.
- A node should participate in at most two piconets, so as to reduce switching overheads.
- To reduce redundant inter-piconet links, two piconets should not be connected by more than one bridge.
- Simple and efficient routing.
- Good mobility- or fault-tolerant capability.

3 The BlueRing Formation Protocol

3.1 Network Architecture

In this subsection, we propose the BlueRing structure. A BlueRing is a scatternet consisting of a cycle of piconets which form a ring. Although physically the ring is undirected, logically we impose a direction on it (say, clockwise). So each piconet has a *downstream piconet* in the forward direction, and an *upstream piconet* in the backward direction. Packets will flow following the direction of the ring, until destinations are reached. In each piconet, two of the slaves are designated as *bridges*, one for connecting to the upstream piconet, called the *upstream bridge*, and one for connecting to the downstream piconet, called the *downstream bridge*. For instance, as shown in Fig. 1, nodes 4 and 6 are upstream and downstream bridges of master M2, respectively. Similarly, each bridge also has a upstream and a downstream masters. Therefore, each bridge host serves as an upstream bridge in one piconet and a downstream bridge in another piconet, and each piconet should have at least two slaves. Fig. 1 illustrates the BlueRing architecture.

3.2 Initial Formation

In order to construct a BlueRing, we adopt a centralized formation mechanism similar to [12]. Assume that all Bluetooth devices are within the radio coverage of each other. We define a parameter RING_MEM for each Bluetooth to indicate whether it has become a member of the BlueRing

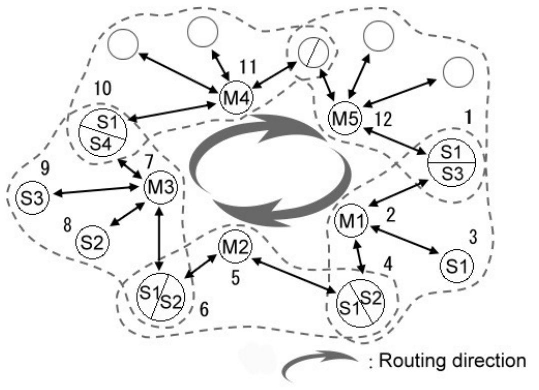


Figure 1: The BlueRing architecture with nodes 2, 5, 7, 11, and 12 serving as masters.

or not (1 for "yes", 0 for "not yet"). Initially, RING_MEM equals 0. The construction has two stages.

- **Stage I:** Each Bluetooth chooses to inquiry (I) with probability p and to inquiry scan (IS) with probability $1 - p$. When some I matches with some IS, the two Bluetooths establish a temporary piconet. Three parameters are exchanged between them: RING_MEM, number of acquired Bluetooths, and BD_ADDR. First, their RING_MEMs are compared. The one with RING_MEM=1 wins if the other's RING_MEM=0. In case of a tie, the one that has gathered more Bluetooths' information wins. If the above cannot determine a winner, tie is broken by their unique BD_ADDRs. The loser should provide the winner with all Bluetooths' information it has gathered. After the information exchange, the temporary piconet is torn down. The potential winner can claim itself as a leader if no further I/IS message is received within an *inquiry timeout (IT)*. Then the (potentially only) leader enters the page state, trying to collect other non-leaders, which must enter the page scan state, waiting to be paged. The details are in Stage II.
- **Stage II:** Based on the desired ring topology, the leader designates several Bluetooths as masters by paging them and setting up a temporary piconet. For each designated master, the leader provides it with the information of its slaves, including assigned downstream and upstream bridges. Upon receiving such information, each master pages its slaves and establishes its piconet. A unit serving as a bridge should make sure that both its downstream and upstream masters have connected to it properly. Once becoming part of the ring, a Bluetooth sets its RING_MEM to 1.

The resultant BlueRing is quite fault-tolerable and scalable. We will discuss the maintenance protocol to handle Bluetooths leaving and joining in Section 5.

4 The BlueRing Routing Protocol

4.1 Packet Relaying Procedures

This subsection discusses the detailed packet relaying procedures and the packet formats. As mentioned earlier, data

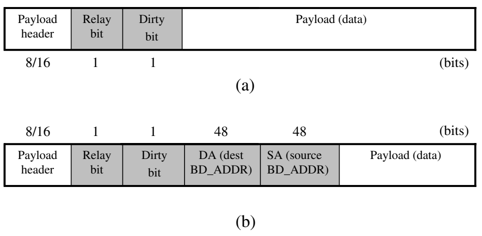


Figure 2: Payload formats in BlueRing: (a) single-hop communication and (b) multi-hop communication. The fields in gray are what added by BlueRing.

packets will be routed following the direction of the BlueRing. Since a packet flowing around the ring will eventually reach its destination piconet, no route discovery process is required. So routing on BlueRing is stateless since no routing table needs to be maintained (on the contrary, most routing protocols for ad hoc networks need to keep routing tables [9]).

To understand how packets are routed, we need to discuss the packet formats in more details. The general Bluetooth baseband data packet format is described as follows. Each packet has a 72-bit access code, which can uniquely identify a piconet, followed by a header and a payload. The header carries 18 bits of information, and is encoded by the 1/3 FEC (Forward Error Correction) code, resulting in a 54-bit header. The payload can range from 0 to 2745 bits.

Data packets supported by the ACL (Asynchronous ConnectionLess) link can occupy 1, 3, or 5 time slots. Type DM1/DH1 packets cover a single time slot, type DM3/DH3 packets 3 slots, and type DM5/DH5 packets 5 slots. On the payload field, there is also a payload header. Bluetooth adopts different payload headers for single-slot and multi-slot packets. For single-slot packets, the payload header is 1-byte long with no reserved bit. As for multi-slot data packets, the payload header is 2-byte long with 4 reserved bits.

To route packets on our BlueRing, several control/information bits should be appended after the payload header. There are two formats for the payload field in BlueRing, as shown in Fig. 2. These fields are explained below:

- Relay bit: equal to TRUE whenever the packet needs to be relayed by the receiving side, and equal to FALSE otherwise.
- Dirty bit: equal to TRUE once the master of the source host touches this packet.
- DA: 48-bit destination Bluetooth Device ADDRess (BD_ADDR).
- SA: 48-bit source Bluetooth Device ADDRess (BD_ADDR).

The relay bit is to help correctly route packets. Whenever a sending host determines that a packet needs to be relayed by the receiving side, the relay bit is set to 1. Only at the last hop or for single-hop packets, this bit is set to 0. The dirty bit is to detect the presence of orphan packets to prevent packets from endlessly circulating on the BlueRing

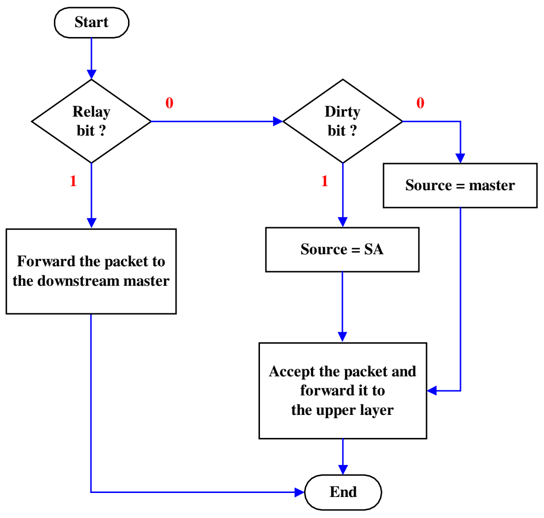

Start
Relay bit ?
0
1
Dirty bit ?
0
1
Source = master
Forward the packet to the downstream master
Source = SA
Accept the packet and forward it to the upper layer
End

Figure 3: The BlueRing routing protocol for slaves.

(which may happen when the destination host leaves the network). For single-hop communications, only the first two fields are required (note that this does not include the last-hop transmission for those multi-hop packets); otherwise, the source's and destination's BD_ADDRs are required.

We describe the BlueRing routing protocol for slaves and masters separately. Fig. 3 shows the operations to be taken by a slave upon receiving a packet. If the relay bit is 0, this is the last hop and the packet should be accepted and forwarded to the upper layer. The dirty bit can be used to determine the origin of the packet. If the dirty bit is 0, this packet is directly from the master; otherwise, it has been relayed and its origin can be found from the field SA. Lastly, if the relay bit is 1, the packet is forwarded to the downstream master.

Fig. 4 shows the operations to be taken by a master upon receiving a packet. If the relay bit is 0, the packet is accepted. Otherwise, we check the dirty bit. For a packet with dirty bit = 1, we need to check if the SA (source host address) is in this piconet. If so, this is an orphan packet and should be deleted from the network. Also, whenever the first master touches a packet with dirty bit = 0, the dirty bit is set to 1 (so as to detect future orphan packets). Then the master needs to decide whether the packet should be forwarded or not. If the DA field is equal to the master itself, the packet is accepted. Otherwise, the DA field is compared to the list of BD_ADDRs belonging to this piconet. If so, the packet is forwarded to host DA in the local piconet; otherwise, the packet is forwarded to the downstream bridge.

4.2 Bridging Policy

Below, we propose the bridging policy used in our BlueRing. A threshold-based strategy is adopted to initiate park/unpark requests. Three parameters are used in the bridging policy: (1) T_b: a threshold value to evaluate the queued packets in a bridge, (2) T_m: a threshold value to evaluate the queued packets in a master, and (3) T_{out}: a timeout value to evaluate when a bridge should switch piconets. Intuitively, under normal situations, a bridge will connect to its upstream piconet for most of the time. When

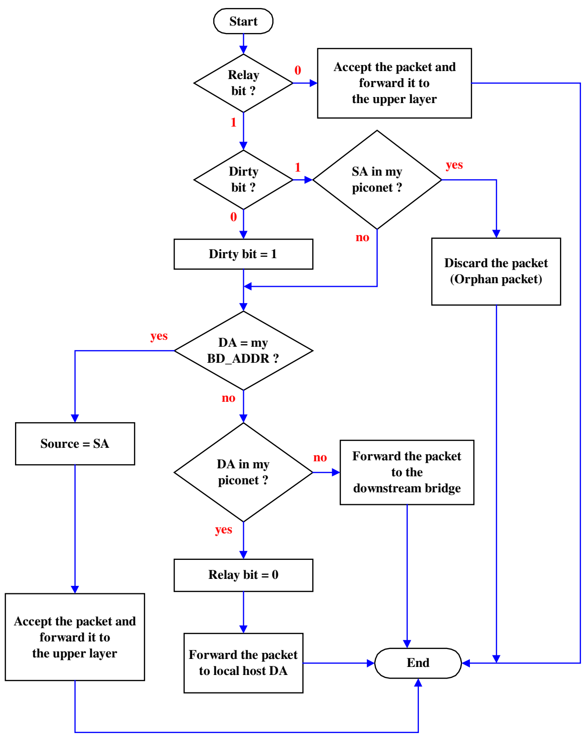

Start
Relay bit ?
0
Accept the packet and forward it to the upper layer
1
Dirty bit ?
1
SA in my piconet ?
yes
0
Dirty bit = 1
no
Discard the packet (Orphan packet)
yes
DA = my BD_ADDR ?
no
Source = SA
DA in my piconet ?
no
Forward the packet to the downstream bridge
yes
Relay bit = 0
Accept the packet and forward it to the upper layer
Forward the packet to local host DA
End

Figure 4: The BlueRing routing protocol for masters.

some threshold conditions become true, it will switch to its downstream piconet. Once connected to its downstream piconet, the bridge will be treated with higher priority by its downstream master so as to drain the packets in its buffer. The detailed switching strategy is described below.

- **From upstream to downstream:** A bridge connecting to its upstream piconet should switch to its downstream piconet when: (i) the number of queued packets to be relayed exceeds T_b, or (ii) the clock T_{out} expires. In this case, a park request should be sent to its upstream master and an unpark request should be sent in the next available access window in the downstream piconet. The downstream master should treat this bridge with higher priority to drain its buffered packets.

- **From downstream to upstream:** A bridge connecting to its downstream piconet should switch to its upstream piconet when: (i) all its buffered packets have been drained by its downstream master, or (ii) its upstream master has queued a number of packets exceeding the threshold T_m. In the former case, the unparking request is initiated by the bridge itself, while in the latter case, the unparking request is initiated by the upstream master to call the slave back. A bridge called by its upstream master should park its current piconet immediately, and switch to the calling piconet channel.

5 The BlueRing Maintenance Protocol

Fault tolerance is an essential issue in packet routing, especially under a mobile environment. In BlueRing, when any master or bridge leaves the network, the ring will become broken and reduce to a linear path. New Bluetooth units may join the network. In this section, we show how to maintain a BlueRing. To tolerate single-point failure, Section 5.1 proposes a protocol-level remedy mechanism. To tolerate multi-point failure, Section 5.2 proposes a recovery mechanism to reconnect the BlueRing. Note that the former one does not try to reestablish the BlueRing so the network may become a linear path. The later one will conduct local reconnection. Graceful failure is tolerable as long as no two or more critical points fail at the same time.

5.1 Single-Point Failure

Suppose that one host serving as a master or a bridge fails. Since there is a default routing direction on BlueRing, a host is unable to reach another host in the backward direction if packets are always sent in the forward direction. The basic idea here is to add a new control bit called *Direction* after the payload header. This bit assists hosts to determine which routing direction (forward/backward) to be followed. Below, we summarize the necessary enhancements on our BlueRing protocol for the fault-tolerant routing.

- The default value for *Direction* is 0, which indicates the forward direction. When a master/bridge on the BlueRing detects that the next hop on the ring is non-existing any more, it simply sets *Direction* = 1 and relays the packet backwards.

- Any master/bridge on receiving a packet with *Direction* = 1 should relay the packet in the backward direction.

- The condition for discarding orphan packets should be revised as follows. Observe that a packet with *Direction* = 1 may reach the source piconet more than once. It is erroneous to discard such a packet by the source piconet master on observing *Dirty* = 1. In this case, the packet should be allowed to continue traveling on the ring until the destination is reached or the other end of the BlueRing is reached. Therefore, the condition for determining a packet to be an orphan should be done by a master/bridge with no upstream node when observing a packet to be undeliverable with *Direction* = 1.

- (Optional) One optimization which can be done here is to have each master keep a list of destination addresses that are unreachable on the forward direction of the BlueRing. On seeing a packet destined to any host in the list, the packet can be sent directly on the backward direction to save communication bandwidth.

Note that if the failure point is a bridge, the whole network remains connected. If a master fails, the non-bridge slaves of the master will become orphans. The other masters should execute the inquiry process from time to time to collect such orphan slaves. The details are in the next subsection.

5.2 Multi-Point Failure

The fault-tolerant routing protocol proposed above ensures routing unaffected, but leaving the broken point unfixed. In

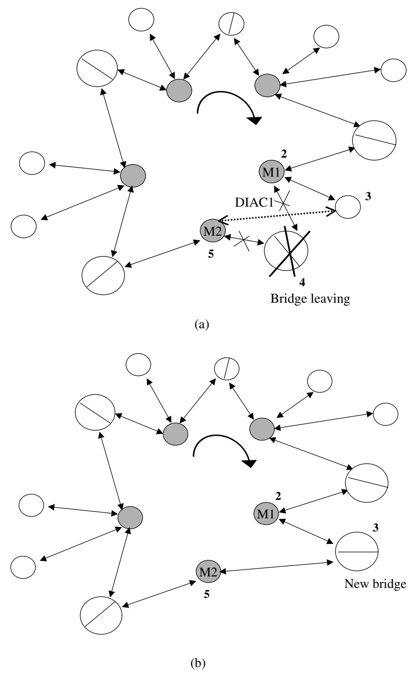


Figure 5: An example to illustrate bridge leaving recovery procedures: (a) DIAC1 discovering and (b) the reconnected BlueRing.

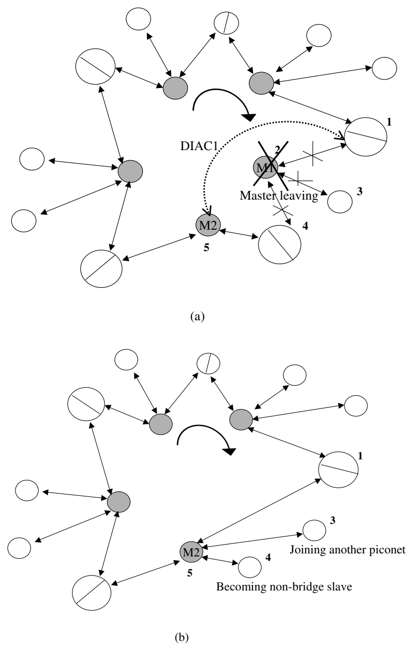


Figure 6: An example to illustrate master leaving recovery procedures: (a) DIAC1 discovering and (b) the reconnected BlueRing.

this subsection, we propose a recovery mechanism that can reconnect the network as a BlueRing. New hosts can join an existing BlueRing too. Only local reconnection is required. As long as no two critical points fail simultaneously, the protocol can work correctly.

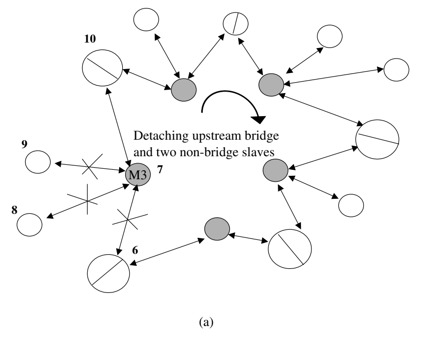
The Bluetooth specification provides 63 reserved DIACs for discovering certain dedicated units in range. Here, we propose to use 2 reserved DIACs, say DIAC1 and DIAC2, to facilitate BlueRing recovery. Also, the GIAC will be used to invite new hosts to join an existing BlueRing.

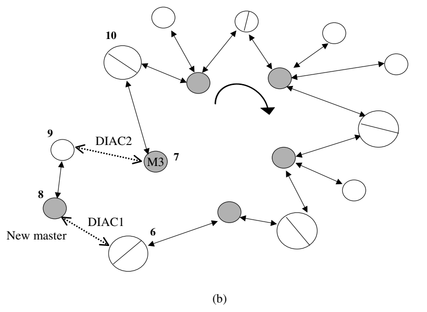
Below, we show how to manage cases of bridge leaving and master leaving. In some cases we will have to extend a BlueRing by creating more piconets.

- **Bridge Missing:** When a bridge leaves, its downstream master performs DIAC1 inquiry, hoping to connect with another upstream bridge. On the other hand, the leaving bridge's upstream master checks if it has any other non-bridge slaves that can serve as its new downstream bridge. If this is the case, the master notifies this bridge such and commands it to enter DIAC1 inquiry scan. Otherwise, the upstream master should tear down its current piconet and wait to be discovered by other masters. This case will induce a missing master, which can be cured by the "**Master Missing**" procedure in the subsequent paragraph. Fig. 5 illustrates the recovery procedures when the bridge node 4 leaves. Upstream master node 2 reassigns node 3 as its new downstream bridge and then node 3 enters DIAC1 inquiry scan. Meanwhile, the downstream master node 5 performs DIAC1 inquiry, and discovers node 3. A new connection is formed between nodes 5 and 3, healing the BlueRing.

- **Master Missing:** When a master leaves, all of its slaves, except the downstream and upstream bridges, become orphans. The downstream bridge of the leaving master should change its state to a non-bridge and inform its downstream master to perform DIAC1 inquiry, in hope of finding a new upstream bridge. On the other hand, the upstream bridge of the leaving master enters DIAC1 inquiry scan, hoping to be discovered. Fig. 6 shows the scenarios when master node 2 leaves, leaving node 3 isolated from the ring. The downstream bridge node 4 reduces a non-bridge slave and informs its downstream master node 5 to execute DIAC1 inquiry. Upstream bridge node 1 starts DIAC1 inquiry scan, and is discovered by node 5. A new connection is established between nodes 5 and 1. Moreover, by GIAC inquiry/inquiry scan, node 3 is discovered and invited to join node 5's piconet, thus becoming part of the ring again. So the BlueRing is reconnected.

- **Piconet Splitting:** Recall that in order to include more Bluetooths into the BlueRing, each master should execute GIAC inquiry from time to time. This happens when new Bluetooth units join or orphan Bluetooth units appear. When the number of slaves belonging to a master exceeds a certain limit, we will split it into two piconets. The procedure is as follows. Assume that the desirable maximum number of slaves per piconet is α ($\alpha \geq 4$). Whenever the number of slaves a master possesses reaches α, the master sends out a *split_request* token to obtain split permission from all other masters. The *split_request* packet traverses the ring to ensure that concurrent splitting operations on the BlueRing do not exist. Once the split request is approved by all piconets on the ring, the master detaches its upstream bridge and two non-bridge slaves (this must succeed since $\alpha \geq 4$). Then the master starts





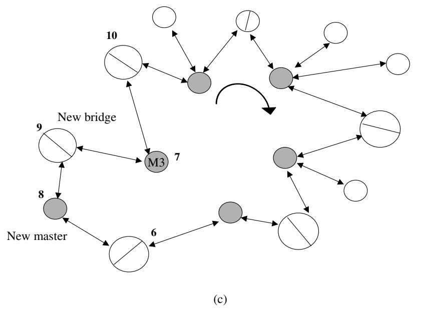


Figure 7: A BlueRing extension example with $\alpha = 4$.

DIAC2 inquiry. Upon discovering master missing, the upstream bridge enters DIAC1 inquiry scan in search of new downstream master. On the other hand, one of the two detached non-bridge slaves is designated as new master and provided with the information of the other detached slave, which is informed to enter page scan. The former will then page the later, thus setting up a new piconet consisting of two members (one master and one slave). The new master designates its only slave as its downstream bridge, and starts DIAC1 inquiry to discover an upstream bridge. Meanwhile, the new master orders its downstream bridge to enter DIAC2 inquiry scan. Fig. 7 shows a splitting example, assuming $\alpha = 4$. After obtaining the permission to split, master node 7 disconnects from nodes 6, 8, and 9, and designates node 8 as the new master. Immediately, node 8 sets up a new piconet with node 9, which serves as its downstream bridge. Then node 8 discovers node 6 by DIAC1 inquiry, while node 7 discovers node 9 via DIAC2 inquiry. The ring is now connected again with one more piconet. By creating more piconets, more new Bluetooths can join the ring, making BlueRing extensible.

We remark on two points. First, every split operation needs to detach 2 non-bridge slaves. Plus downstream and upstream bridges, only piconets with no less than 4 slaves can request for splitting. That explains why we enforce $\alpha \geq 4$. Second, the *split_request* token should compete with other potential *split_request* tokens while traveling on the ring. The can be easily done by allowing a requesting master with a higher BD_ADDR to inhibit other requesting masters' tokens with lower BD_ADDRs.

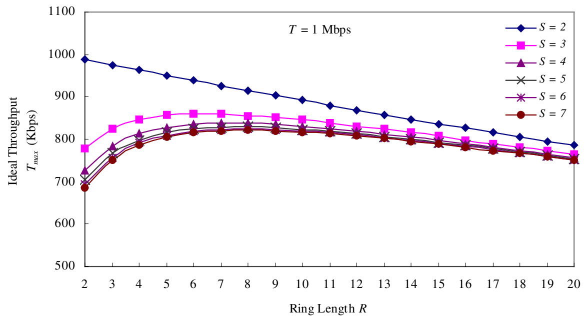


Figure 8: Ideal BlueRing throughput for different R's and S's.

6 Analysis and Simulation Results

Table 1: Routing distances for packets originated at (a) a non-bridge slave, (b) a master, and (c) a bridge.

	No. of destinations	Distance
Intra-piconet-slave	$c_{s1} = S\text{-}3$	$d_{s1} = 2$
Ring-body	$c_{s2} = 2R$	$d_{s2} = R+1$
Inter-piconet-slave	$c_{s3} = (S\text{-}2)(R\text{-}1)$	$d_{s3} = R+2$

(a) Source is a non-bridge slave

	No. of destinations	Distance
Intra-piconet-slave	$c_{m1} = S\text{-}2$	$d_{m1} = 1$
Ring-body	$c_{m2} = 2R\text{-}1$	$d_{m2} = R$
Inter-piconet-slave	$c_{m3} = (S\text{-}2)(R\text{-}1)$	$d_{m3} = R+1$

(b) Source is a master

	No. of destinations	Distance
Intra-piconet-slave	$c_{b1} = S\text{-}2$	$d_{b1} = 2$
Ring-body	$c_{b2} = 2R\text{-}1$	$d_{b2} = R$
Inter-piconet-slave	$c_{b3} = (S\text{-}2)(R\text{-}1)$	$d_{b3} = R+1$

(c) Source is a bridge

In this section, we first evaluate the maximum achievable throughput of the BlueRing. Let the ring contain R piconets ($R \geq 2$) and each piconet contain S slaves (including both non-bridge slaves and bridges). So the total number of Bluetooths on the ring is $S \cdot R$. We shall derive the average number of hops that a data packet needs to travel before reaching its destination. This will depend on the roll of the source, which can be master, bridge slave, and non-bridge slave. Table 1 summarizes these three cases. In the tables, destinations are classified into ring-body, intra-piconet-slave, and inter-piconet-slave. Note that the ring-body contains all masters and bridges, and the rest of the slaves are classified into intra- and inter-piconet cases.

Based on Table 1, we derive the average distances d_s, d_m, and d_b for packets originated at a non-bridge slave, master, and bridge, respectively, as follows:

$$d_s = \frac{c_{s1} \cdot d_{s1} + c_{s2} \cdot d_{s2} + c_{s3} \cdot d_{s3}}{c_{s1} + c_{s2} + c_{s3}} = \frac{SR^2 + SR - 2}{SR - 1} \quad (1)$$

$$d_m = \frac{c_{m1} \cdot d_{m1} + c_{m2} \cdot d_{m2} + c_{m3} \cdot d_{m3}}{c_{m1} + c_{m2} + c_{m3}} = \frac{SR^2 - R}{SR - 1} = R \quad (2)$$

$$d_b = \frac{c_{b1} \cdot d_{b1} + c_{b2} \cdot d_{b2} + c_{b3} \cdot d_{b3}}{c_{b1} + c_{b2} + c_{b3}} = \frac{SR^2 + (S-3)R}{SR - 1} \quad (3)$$

Since there are $(S-2)R$ non-bridge slaves, R masters, and R bridges, the average traveling distance can be derived as:

$$\begin{aligned} D_{avg} &= \frac{(S-2) \cdot R \cdot d_s + R \cdot d_m + R \cdot d_b}{SR} \\ &= \frac{S^2R^3 + (S^2 - S - 4)R^2 + (4 - 2S)R}{S^2R^2 - SR}. \quad (4) \end{aligned}$$

Taking $S = 7$ and $R = 3$ for example, the average traveling distance D_{avg} will be 3.89.

The following analysis further considers interference between piconets. Assume that the maximum throughput of a single piconet under an interference-free environment is T. By extending to a scatternet, different piconets which choose the same FH channel in the same time slot result in a collision. Given that 79 frequencies are available in Bluetooth, the probability that a time slot of a piconet suffers no interference, denoted by P_S, can be approximated by $(78/79)^{R-1}$, where R is the number of piconets in the transmission range of each other.

The available network bandwidth is $T \cdot R \cdot P_S$. Dividing this by the average traveling distance D_{avg}, we obtain the maximum achievable throughput T_{max} of BlueRing:

$$T_{max} = \frac{T \cdot R \cdot P_S}{D_{ave}}. \quad (5)$$

Using the Bluetooth nominal bandwidth $T = 1$ Mbps, $S = 7$, and $R = 3$, we can compute $T_{max} = 751$ Kbps. Fig. 8 shows the ideal throughput T_{max} by varying R and S. From the curves, it can be seen that a BlueRing length of $R = 5 \sim 8$ would be quite cost-effective.

In the following, we present our simulation results to verify the above theoretical analysis and to compare our BlueRing with other scatternet topologies. We simulate only DH1 data packets. For a single-slot DH1 packet, 336 bits (including the access code, header, payload header, payload, and CRC) are transmitted over a time slot with 625μs duration, which contributes to a reduced $T = 538$ Kbps throughput/piconet. For simplicity, we assume that collisions due to frequency overlapping do not happen, and thus $P_S = 1$. Taking a 21-node BlueRing ($S = 7$, $R = 3$) for instance, we obtain $T_{max} = 415$ Kbps, which predicts the saturation point in throughput. This can be used to verify the correctness of our analysis.

In our simulation, the number of Bluetooth devices that may participate in the BlueRing could be $N = 10$, 20, 30, or 40. For each simulation instance, we initiate a number of data-link connections, each with a randomly chosen source-destination pair. Each connection is an ACL link and can be an intra- or inter-piconet communication. We also vary the ratio of the numbers of intra- to inter-piconet connections. Each connection has a data rate of 256 Kbps. A master keeps a separate buffer queue for each of its slaves. No mobility is modeled. Besides, physical properties such as fading and interference are not considered. Each simulation run lasts 80,000 time slots. Only DH1 data packets are simulated.

The bridging policy proposed in Section 4.2 is followed. Switching between piconets is realized using park/unpark procedures by following the proposed control message exchanges. We set the buffering threshold to 60% for bridges to switch between piconets. The maximum number of slaves per piconet is set to S. This factor will affect the ring length as well as packet delay. Two performance metrics are observed: *throughput* and *average packet delays*.

We compare BlueRing with two other scatternet structures. The first one is a simple *single-piconet* structure. According to the Bluetooth specification, at most 7 active slaves can be supported in a single piconet. To support more than 7 slaves, the extra slaves must enter the *park* mode. In our implementation, we let the channel be shared by slaves in a round-robin manner. In other words, communicating entities are parked/unparked periodically, taking turns to access the channel. So many extra control packets exchanges will take place. The second structure is the *star-shaped* scatternet proposed in [7]. One piconet is placed in the center. Each slave of the central piconet may be connected to another master, and if so, will act as a bridge of the two piconets. Non-central piconets do not extend to more piconet. So this can be regarded as a two-level hierarchy. All inter-piconet traffic must go through the central piconet, and thus this may present a traffic bottleneck, but the benefit is a less average number of hops that packets have to go through for inter-piconet communications. Although the bridging policy is not specified in [7], here we adopt our threshold-based policy in Section 4.2 by regarding the central piconet as upstream, and non-central piconets as downstreams. When $N = 20$, Fig. 9 compares our BlueRing with the star-shaped scatternet.

Fig. 10 demonstrates the network throughput and packet delay under different traffic loads. Here, load is reflected by the number of connections initiated. Each connection has a data rate of 256 Kbps, and could be an intra- or inter-piconet commuincation. Fig. 10(a) shows that BlueRing saturates at the highest point compared to the other two structures. The saturated throughput is about 415 Kbps, which is consistent with the prediction of our analysis. For the star-shaped structure, its throughput outperforms that of the single-piconet model when the number of simultaneous connections exceeds 6. This is because multi-piconet has the advantage of using multiple frequency-hopping channels at the same time. Fig. 10(b) demonstrates that, when the number of simultaneous connections is below 10, both BlueRing and star scatternet suffer higher delays compared to the single-piconet case due to bridging costs and larger network diameters. However, when traffic load becomes higher, the packet delay of single-piconet structure raises dramatically. The reason is that the throughput of single-piconet struc-

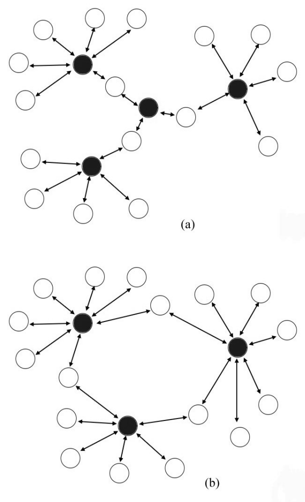


Figure 9: Two scatternet topologies with $N = 20$ hosts: (a) star-shaped structure and (b) BlueRing (black nodes are masters).

ture has reached a saturated point, which leads to significant increase of packet delays.

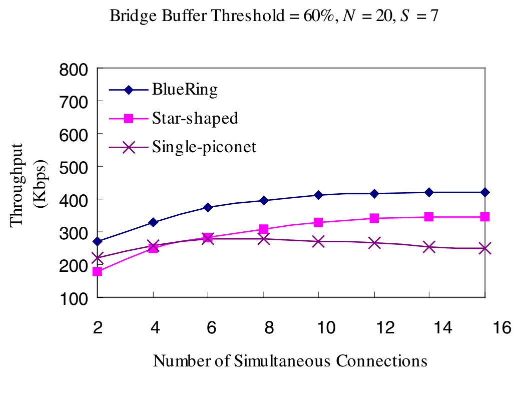
We also investigate the impact of different intra- to inter-piconet connection ratios on the network performance. Note that with more inter-piconet connections, the traffic load is higher. Fig. 11 compares the throughput and average packet delay of the three scatternet structures under different traffic loads. Here the ratio of intra-picont to inter-piconet traffic is 3:1. The figure shows that BlueRing yields the highest throughput with moderate packet delay. For the single-piconet structure, the throughput saturates when the number of simultaneous connections reaches 6. This is because many time slots are wasted on exchanging control packets for park/unpark procedures. Furthermore, a single piconet can only utilize one frequency-hopping sequence, and thus the maximum frequency utilization is only 1/79. On the other hand, BlueRing may utilize multiple FH sequences over the same space. This is what limits the capability of the single-piconet structure. With more intra-piconet connections, the BlueRing structure allows more simultaneous transmissions and shorter routing distances. For the star-shaped structure, the throughput is between those of BlueRing and single-piconet structures. With more intra-piconet connections, the bottleneck effect, incurred by the central piconet, of the star-shaped scatternet becomes less significant.

Fig. 12 illustrates the case when there are more inter-piconet connections (with intra- to inter-piconet connection ratio equal to 1:3). The performance of the star-shaped structure drops significantly due to more serious bottleneck effect. For BlueRing, the throughput also declines, but is still superior to the star-shaped structure. The single-piconet structure remains unaffected since all traffic is intra-piconet.

7 Conclusions

In this paper, we have designed the corresponding formation, routing, and topology maintenance protocols for BlueRing. Due to BlueRing's simplicity and regularity, routing on it is *stateless*, in the sense that no routing table needs to be kept by any host (and thus no route discovery procedure needs to be conducted prior to sending any packet). A protocol-level remedy is developed to keep the network alive when there is a single-point failure on the ring. To tolerate multi-point failure, a recovery protocol is devised to reconnect the BlueRing. We believe that the important fault-tolerant issue has not been properly addressed by existing proposed scatternet protocols. To demonstrate the scalability of BlueRing with respect to network size, analyses and simulation experiments have been conducted. The results do indicate that BlueRing outperforms other network structures, such as single-piconet and star-shaped scatternet, with higher throughput and moderate packet delay. To conclude, we believe that the BlueRing is an efficient topology in terms of both network performance and fault-tolerant capability.

Our future works include analyzing the fault tolerance capability of BlueRing and devising mechanisms to deal with more than one simultaneous failure on the ring. Moreover, a real implementation of BlueRing is also being planned in the National Chiao-Tung University and will be reported in our future work.



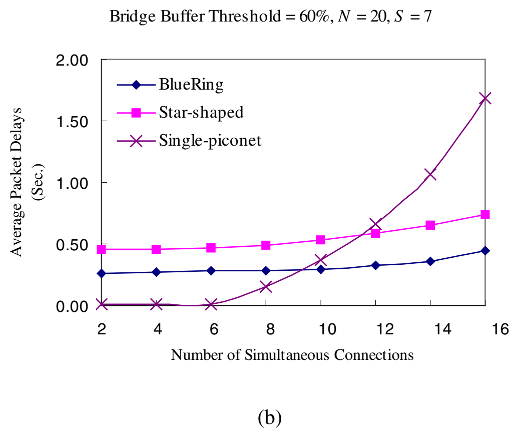
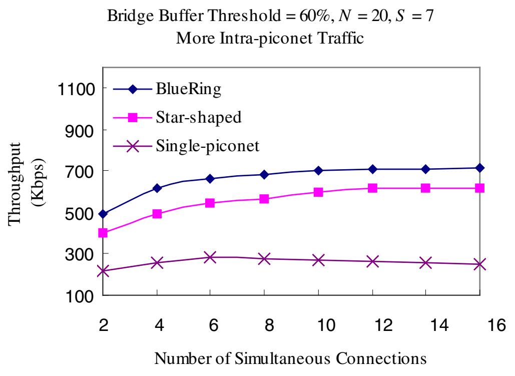


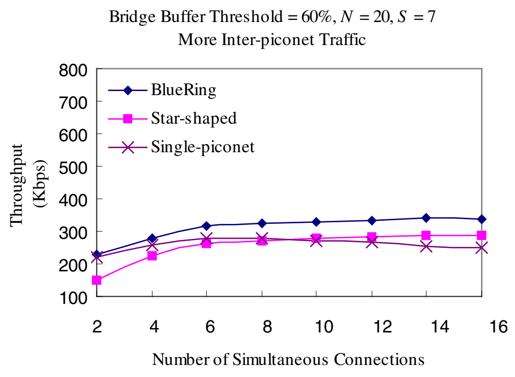
Figure 10: Performance comparison by varying the number of connections: (a) throughput and (b) packet delay.

References

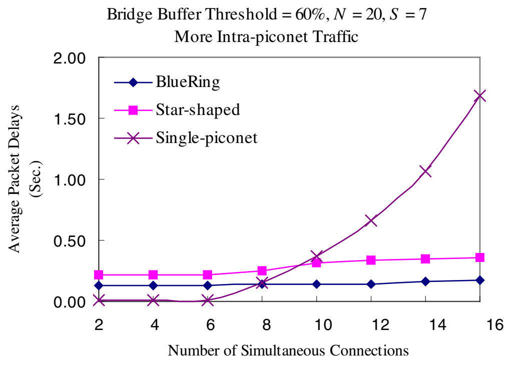
[1] Bluetooth SIG Bluetooth Specification v1.1,



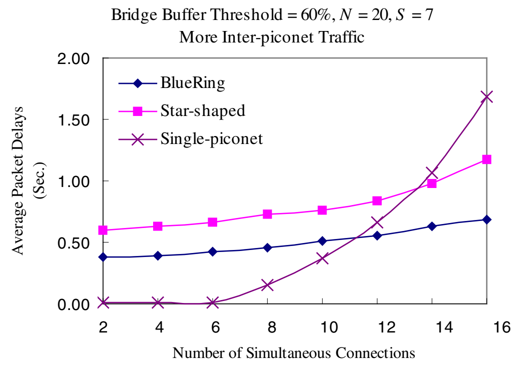
(a)



(a)



(b)



(b)

Figure 11: Performance comparison by varying the number of connections (with more intra-piconet traffic): (a) throughput and (b) packet delay.

Figure 12: Performance comparison by varying the number of connections (with more inter-piconet traffic): (a) throughput and (b) packet delay.

http://www.bluetooth.com. February, 2001.

[2] P. Bhagwat and A. Segall. A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets. *IEEE Int'l Workshop on Mobile Multimedia Communications (MoMuC)*, 1999.

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

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

[5] 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.

[6] 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.

[7] W. Lilakiatsakun and A. Seneviratne. Wireless Home Networks based on a Hierarchical Bluetooth Scatternet Architecture. *IEEE Int'l Conference on Networks (ICON)*, 2001.

[8] 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.

[9] C. E. Perkins. Ad Hoc Networking. *Addison-Wesley*, 2001.

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

[11] 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.

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

[13] 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.

Collision Analysis for a Multi-Bluetooth Picocells Environment

Ting-Yu Lin and Yu-Chee Tseng

***Abstract*—Operating in the unlicensed 2.4-GHz ISM band, a Bluetooth piconet will inevitably encounter the interference problem from other piconets. With a special channel model and packet formats, one research issue is how to predict the packet collision effect in a multi-piconet environment. In an earlier work by El-Hoiydi in 2001, this problem is studied, but the result is still very limited, due to the assumptions that packets must be single-slot ones and that time slots of each piconet must be fully occupied by packets. A more general analysis is presented in this work by eliminating these constraints.**

***Index Terms*—Bluetooth, collision analysis, frequency hopping (FH), piconet, Wireless Personal-Area Network (WPAN).**

I. Introduction

As a PROMISING Wireless Personal-Area Network (WPAN) technology, Bluetooths are expected to be used in many applications, such as wireless earphones, keyboards, wireless access points, etc., [3]. Operating in the unlicensed 2.4-GHz ISM band, multiple Bluetooth piconets are likely to coexist in a physical environment. With a frequency-hopping (FH) radio and without coordination among piconets, transmissions from different piconets will inevitably encounter the collision problem. In a previous work [1], the author investigates the co-channel interference between Bluetooth piconets and derives an upper bound on packet error rate. The analysis in [1] has two limitations. First, all packets are assumed to be single-slot ones. Second, it is assumed that each piconet is fully loaded, in the sense that packets are sent in a back-to-back manner. These constraints greatly limit the applicability of the result in [1].

Also focusing on the same problem, this paper derives a more general analysis model where all packet types (1-slot, 3-slot, and 5-slot) can coexist in the network, and the system is not necessarily fully-loaded. The latter is achieved by modeling idle slots as individual single slots with *no* traffic load. So the result greatly relax the constraints in [1].

Manuscript received February 27, 2003. The associate editor coordinating the review of this letter and approving it for publication was Prof. S. Pierre. This work was supported by the MOE, Taiwan, R.O.C., under Grant A-91-H-FA07-1-4 and Grant 89-E-FA04-1-4.

The authors are with the Department of Computer Science and Information Engineering National Chiao-Tung University, Hsin-Chu 300, Taiwan, R.O.C. (e-mail: yctseng@csie.nctu.edu.tw).

Digital Object Identifier 10.1109/LCOMM.2003.817313

II. Problem Statement

Bluetooth is a master-driven, time-division duplex (TDD), FH wireless radio system [2]. The smallest networking unit is a *piconet*, which consists of one master and no more than seven active slaves. Each picocell channel is represented by a pseudo-random hopping sequence comprised of 79 or 23 frequencies. In Bluetooth, the hopping sequence is determined by the master's ID and clock value. The channel is divided into time slots, each corresponding to one random frequency. In the following discussion, we assume 79 frequencies.

In each piconet, the master and slaves take turns to exchange packets. While the master only transmits in even-numbered slots, slaves must reply in odd-numbered slots. Three packet sizes are available: 1-slot, 3-slot, and 5-slot. For a multislot packet, its frequency is fully determined by the first slot and remains unchanged throughout.

We consider N piconets coexisting in a physically closed environment. Since no coordination is possible between piconets, each piconet has $N-1$ potential competitors. In any time instance, if two piconets transmit with the same frequency, the corresponding two packets are considered damaged. Our goal is to derive an analytic model to evaluate the impact of collisions in such a multi-piconet environment.

We assume a uniform traffic in each piconet, and let λ_1, λ_3, and λ_5 be the arrival rates of 1-, 3-, and 5-slot packets per slot, respectively, to a piconet. Note that for a multi-slot packet, only the header slot counts as arrival. It is easy to see that $\lambda_1+3\lambda_3+5\lambda_5 \le 1$. Further, we can regard the remaining vacant slots as "dummy" single-slot packets. Thus, the arrival rate of such dummy (1-slot) packets is $\lambda_0 = 1-(\lambda_1+3\lambda_3+5\lambda_5)$.

III. Collision Analysis in a Multi-Piconet Environment

Let us consider a piconet X and another competitor piconet Y, which is regarded as the unique source of interference to X. With the interference from Y, we first derive the success probability $P_S(i)$ of i-slot packets in X, where $i = 1, 3, 5$. We start by introducing the concept of "slot delimiter." Consider any slot in X. One or two slot delimiters in Y may cross X's slot. However, since we are considering continuous probability, the possibility of two crossing slot delimiters can be ignored, and thus we will deal with one crossing delimiter in the rest of the discussion. For example, for a 1-slot packet in X, it succeeds only if there is no interference from the two slots before and after the delimiter, so the success probability of X's packet could be 1, 78/79, or $(78/79)^2$, depending on whether Y transmits or not. Below, we denote the constant factor 78/79 by P_0.

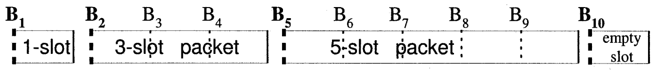


Fig. 1. Illustration of slot delimiters.

Next, we further elaborate on slot delimiters. Depending on what packet(s) is divided by it, a delimiter is classified into ten types (refer to Fig. 1):

- B_1, B_2, B_5: the beginning of a 1-, 3-, and 5-slot packet, respectively.
- B_3, B_4: the beginnings of the second and third slots of a 3-slot packet, respectively.
- B_6, B_7, B_8, B_9: the beginnings of the second, third, fourth, and fifth slots of a 5-slot packet, respectively.
- B_{10}: the beginning of a dummy slot.

It is easy to see that the rate of B_1 is λ_1 per slot; the rate of each of B_2, B_3, and B_4 is λ_3; the rate of each of B_5, B_6, B_7, B_8, and B_9 is λ_5; and the rate of B_{10} is λ_0. For ease of presentation, we denote the arrival rate of B_j by $\lambda(B_j)$, $j = 1 \ldots 10$. Given any B_j, we also define $g(j)$ to be the number of slots that follows delimiter B_j and belongs to the same packet. For example, $g(1) = 1$, $g(3) = 2$, $g(7) = 3$, and $g(10) = 1$.

Intuitively, when a packet in X is crossed by a delimiter of type $B_1/B_2/B_5$ in Y, there may exist two packets (of different frequencies) in both sides of the delimiter in Y which are potential sources of interference to X's packet. On the other hand, when the delimiter is of the other types, the interference source reduces to one.

Next, we formulate the success probability $P_S(i)$ of an i-slot packet, $i = 1, 3, 5$, in X, given the interference source Y. Toward this goal, we first introduce another probability function.

Definition 1: Given any i-slot packet in piconet X and any interference source piconet Y, define $L(k), k < i$, to be the probability that the packet of X experiences no interference from Y starting from the delimiter of Y crossing the $(i - k + 1)$-th slot of the packet to the end of the packet, under the condition that the aforementioned delimiter is of type $B_1/B_2/B_5/B_{10}$. For $k \leq 0$ (in which case the above definition is not applicable), $L(k) = 1$.

Intuitively, $L(k)$ is the success probability of the last k slots of X's packet excluding the part before the first delimiter of Y crossing these k slots, given the delimiter type constraint. With this definition, we can find $P_S(i)$ by repeatedly cutting off some slots from the head of X's packet, until there is no remaining slot. Specifically, we establish $P_S(i)$ by $L()$ as follows:

$$P_S(i) = \sum_{j=1}^{10} \lambda(B_j) \cdot f(j) \cdot L(i - g(j)) \tag{1}$$

where

$$f(j) = \begin{cases} (1 - \lambda_0) \cdot P_0^2 + \lambda_0 \cdot P_0, & \text{if } j = 1, 2, 5 \\ (1 - \lambda_0) \cdot P_0 + \lambda_0, & \text{if } j = 10 \\ P_0, & \text{otherwise.} \end{cases}$$

In the equation, we consider each type $B_j, j = 1 \ldots 10$, of the first delimiter in Y crossing X's packet. The corresponding probability is $\lambda(B_j)$. Function $f(j)$ gives the probability that the packet(s) of Y on both sides of the first delimiter B_j does (do) not interfere with X's packet. It remains to consider the success probability of the last $i - g(j)$ slots of X's packet, excluding the part before the first delimiter of Y crossing these $i - g(j)$ slots (which must be of delimiter type $B_1/B_2/B_5/B_{10}$). This is reflected by the last factor $L(i - g(j))$.

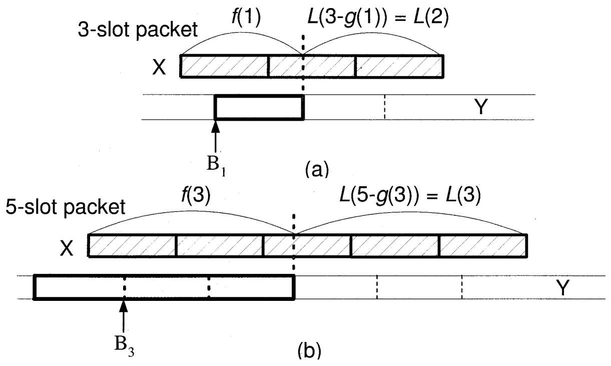


Fig. 2. Analysis of success probabilities for (a) 3-slot and (b) 5-slot packets.

For example, Fig. 2(a) illustrates a 3-slot packet in X. The first delimiter in Y crossing the 3-slot packet is of type B_1. The success probability of the first part in X is $f(1) = (1 - \lambda_0) \cdot P_0^2 + \lambda_0 \cdot P_0$. Intuitively, if the packet of Y before the delimiter B_1 is a dummy packet (of probability λ_0), the success probability is simply P_0; otherwise, there are two packets which are potential interference sources, and the success probability is P_0^2. Then we can move on to consider the success probability of the remaining part of X after the second delimiter in Y, which is given by $L(2)$. Another example of a 5-slot packet is shown in Fig. 2(b). The first delimiter in Y crossing the 5-slot packet is of type B_3. So the success probability from the beginning of the packet up to the third delimiter in Y crossing the packet is $f(3)$. For the remaining part, the success probability is $L(3)$. So the success probability of the 5-slot packet is $f(3) \cdot L(3)$

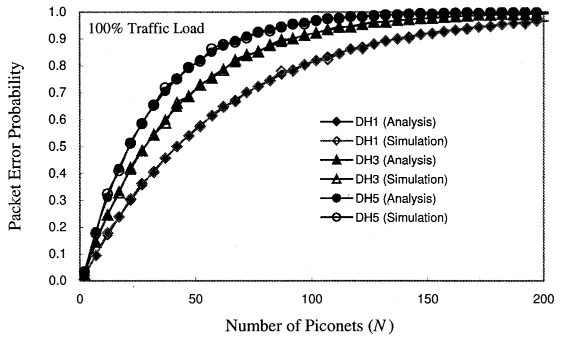
The remaining part of X's packet covered by $L(k)$ must start with a delimiter in Y of a restricted type of $B_1/B_2/B_5/B_{10}$. Since the packet in Y after the delimiter must be a complete packet, it can be solved recursively as follows $(k > 0)$:

$$\begin{aligned} L(k) = &\frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_3 + \lambda_5} \cdot L(k - g(10)) \\ &+ \frac{\lambda_1}{\lambda_0 + \lambda_1 + \lambda_3 + \lambda_5} \cdot P_0 \cdot L(k - g(1)) \\ &+ \frac{\lambda_3}{\lambda_0 + \lambda_1 + \lambda_3 + \lambda_5} \cdot P_0 \cdot L(k - g(2)) \\ &+ \frac{\lambda_5}{\lambda_0 + \lambda_1 + \lambda_3 + \lambda_5} \cdot P_0 \cdot L(k - g(5)). \end{aligned} \tag{2}$$

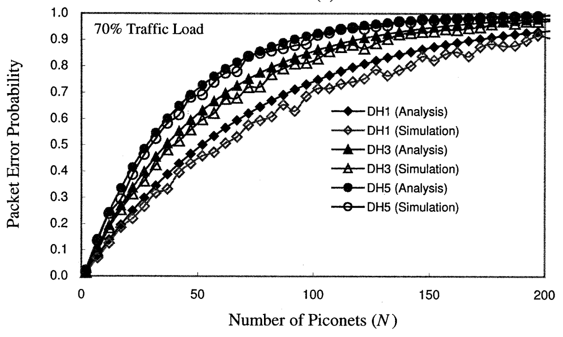
In each term, the first part is the probability of the corresponding packet type in Y. As to the boundary conditions, $L(k) = 1$, for $k \leq 0$.

Next, we consider an N-piconet environment. For each piconet X, there are $N - 1$ piconets each serving as an interference source. Since these interferences are uncoordinated and independent, the success probability of an i-slot packet in X can be written as $P_S(i)^{N-1}$. So the network throughput of X is:

$$\begin{aligned} T = \lambda_1 \cdot P_S(1)^{N-1} \cdot R_1 + 3 \cdot \lambda_3 \cdot P_S(3)^{N-1} \cdot R_3 \\ +5 \cdot \lambda_5 \cdot P_S(5)^{N-1} \cdot R_5, \end{aligned} \tag{3}$$

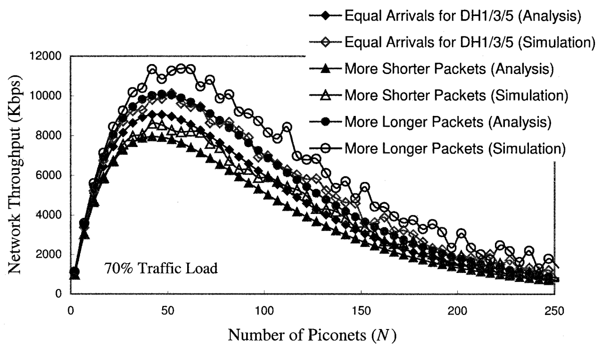


(a)

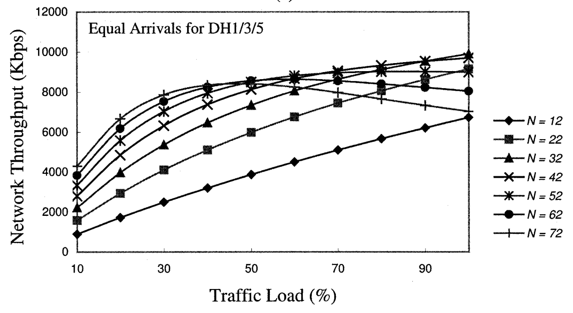


(b)

Fig. 3. Packet error probabilities under traffic loads of: (a) 100% and (b) 70%.



(a)



(b)

Fig. 4. Network throughput: (a) under 70% traffic load, and (b) against traffic loads for various network sizes.

where R_1, R_3, and R_5 are the data rates (bits/slot) of 1-, 3-, and 5-slot packets, respectively (for example, if DH1/DH3/DH5 are used, $R_1 = 216$, $R_3 = 488$, and $R_5 = 542.4$). The aggregate network throughput of N piconets is $N \times T$.

IV. Simulation and Analysis Results

From the above discussion, it can be seen that the result in [1] is in fact a special case of our analysis when $\lambda_1 = 1$ and $\lambda_3 = \lambda_5 = 0$.

To verify our analysis, simulations are conducted. We investigate DH1/3/5 packets. Assuming $\lambda_1 = \lambda_3 = \lambda_5$, we inject traffic loads of 100% and 70% (the percentage of busy slots) to reflect heavy and medium loads, respectively. That is, $\lambda_1 + 3\lambda_3 + 5\lambda_5 = 1$ and 0.7. Fig. 3 plots the error probabilities of DH1/3/5 packets under different numbers of picocells. The packet error probability increases as the traffic load or the number of piconets grows. Small packets (DH1) suffer less collisions than large ones (DH5) due to shorter transmission durations. However, larger packets are much more bandwidth-efficient than smaller ones (e.g., a DH5 carries 542.4/216 times more bits per slot than a DH1 does). This observation leads us to conduct the next experiment by using network throughput as the metric.

Here we evaluate aggregate network throughput (i.e., $N \times T$). We show the case of 70% traffic load. In addition to equating $\lambda_1 = \lambda_3 = \lambda_5$, we also set λ_1:λ_3:λ_5 as 3:2:1 to reflect the case of more shorter packets, and as 1:2:3 to reflect the case of more longer packets. The results are shown in Fig. 4(a). The aggregate throughput saturates at a certain point as the number of piconets increases, and then drops sharply. Different from the earlier observation, the figure reveals that longer packets are more preferable in terms of throughput because the collision problem can be compensated by the benefit of bandwidth efficiency.

Finally, Fig. 4(b) plots the throughput against traffic loads by fixing the value of N. It indicates that throughput goes up steadily as traffic load increases when $N \leq 32$. However, for larger N's, throughputs saturate at certain points, due to more serious collisions. The results suggest that at most 42 piconets can be placed in a physical area.

References

[1] A. El-Hoiydi, "Interference between bluetooth networks—Upper bound on the packet error rate," *IEEE Communications Letters*, June 2001.
[2] (2001) Bluetooth SIG Specification v1.1. [Online]. Available: http://www.bluetooth.com
[3] K. Sairam, N. Gunasekaran, and S. Reddy, "Bluetooth in wireless communication," *IEEE Commun. Mag.*, vol. 40, pp. 90–96, June 2002.

Event-Driven Messaging Services Over Integrated Cellular and Wireless Sensor Networks: Prototyping Experiences of a Visitor System

Yu-Chee Tseng, *Senior Member*, Ting-Yu Lin, Yen-Ku Liu, and Bing-Rong Lin

***Abstract*—As one of the killer applications, instant messaging has become a simple yet efficient tool for peer-to-peer communications in data networks. In telephone networks, short message services are gaining more popularity as well. However, this kind of services typically operates in their own respective networks and is triggered by fairly simple events (such as pushing a "send" button or prescheduling the transmission at later time). A promising direction is to trigger instant messages by environmental information from the physical world. In light of this, this paper proposes to establish an event-driven messaging service over a cellular-and-sensor-integrated network. We have prototyped a system which adopts Global System for Mobile Communication (GSM) as the cellular network, and Bluetooth technology as the sensor network. The latter is to realize a Bluetooth surveillance network with location-sensing capabilities to be deployed within an office building area. While using other technologies is possible, GSM and Bluetooth are two dominating technologies in telephone and data networks. So the proposed technology is immediately feasible, given the fact that many handsets are already Bluetooth-enabled.**

Through this combination, we demonstrate a *visitor system (VS)* that offers several attractive features/services for visitors arriving at an office. First, messaging services in VS are driven by preconfigured events which can be collected from the Bluetooth surveillance network. Simple events might be a person entering/leaving a space, while complicated events might be a compound logic statement involving multiple users and multiple locations in VS. Second, we believe that the proposed system justifies the potential of cross-network applications and services. Third, the proposed architecture takes a modular approach by dividing the system into several subsystems according to their functionality. Logically dispatching jobs is the key to future extensions and further value-added services. The system architecture and implementation details are reported. Performance analyses are presented to model the detection latency of a Bluetooth sensor network.

***Index Terms*—Bluetooth, context awareness, Global System for Mobile Communication (GSM), instant message, mobile computing, sensor network, short message service (SMS), wireless communication.**

Manuscript received February 1, 2004; revised December 1, 2004. The work of Y.-C. Tseng was supported in part by the NSC Program for Promoting Academic Excellence of Universities under Grant 93-2752-E-007-001-PAE, in part by Computer and Communications Research Laboratories, ITRI, Taiwan, in part by Intel, Inc., and in part by the Institute for Information Industry and MOEA, R.O.C., under the Handheld Device Embedded System Software Technology Development Project and the Communications Software Technology Project.

Y.-C. Tseng, Y.-K. Liu, and B.-R. Lin are with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu 30010, Taiwan (e-mail: yctseng@csie.nctu.edu.tw; yklau@csie.nctu.edu.tw; linnbiro@csie.nctu.edu.tw).

T.-Y. Lin is with the Computer and Communications Research Laboratory, Industrial Technology Research Institute, Hsinchu 310, Taiwan, R.O.C. (e-mail: tylin@csie.nctu.edu.tw).

Digital Object Identifier 10.1109/JSAC.2005.845623

I. Introduction

PEER-TO-PEER messaging has become a common way of communications in people's daily lives. In mobile phone systems, such as Global System for Mobile Communication (GSM) and NTT DoCoMo i-mode, short message services are gaining more revenues year-by-year according to the Internet service provider (ISP) statistics [13], [14]. For IP-based data networks, instant messaging software, such as ICQ and MSN [8], [10], is also popular for nowadays Internet users. In addition, an interesting project called Hubbub [15], which aims at providing mobile users with impromptu information exchange services, is recently reported. However, this kind of services typically operates in their own respective networks and is triggered by fairly simple events (such as pushing a "send" button or prescheduling the transmission at later time).

On the other hand, with the rapid development of wireless data networks, the potential service range of instant messages can be further extended, if cross-network solutions can be provided. In this work, we adopt Bluetooth [12], which is one of the dominating short-range wireless radio systems, as our platform. Bluetooth is characterized by small size and low cost, and has drawn much attention from both academia and business worlds [4], [7], [9], [18], [20]. It is also available immediately, given the fact that many laptops, PDAs, and handsets are already Bluetooth-enabled. Main applications of Bluetooths nowadays include cable replacement and simple voice/data exchange.

Motivated by the above observations, this paper proposes an *event-driven messaging service* over an integrated GSM-IP-Bluetooth network. We use Bluetooth as a sensor network for surveillance and location-tracking purposes in an indoor environment. GSM is used to support instant messaging services. These networks are integrated under Internet protocol (IP) networks. Targeting at providing visitor-friendly services in an office environment, our system is referred to as the visitor system (VS). VS offers several attractive features/services. First, messaging services in VS are driven by preconfigured events which can be collected from the Bluetooth sensor network. Simple events might be a person entering/leaving a space, while complicated events might be a compound logic statement involving multiple users and multiple locations in VS. Through logic operators such as "AND," "OR," and "INVERSE," we can provide new types of event-driven instant messaging services with novel applications. Second, we believe that the proposed system justifies the potential of cross-network applications and services. In addition, integrated services across different networks release people from a used-to-be-confined service area. Migration between networks is transparent to users in VS. Third, the proposed architecture takes a modular approach by dividing the system into several subsystems according to their functionality. Logically dispatching jobs is the key to future extensions and further value-added services. This feature provides application programmers with the flexibility to customize their own visitor systems. The deployment cost of VS should be low because it takes advantage of existing cellular networks and embedded Bluetooth devices that are already built in many portable devices. Only some servers are needed in the backbone networks.

The rest of this paper is organized as follows. In Section II, we review several existing messaging services and sensing systems. Section III discusses our VS in more details. Section IV addresses the performance aspects and possible extensions of VS. Finally, Section V concludes the paper.

II. Background and Related Works

Sections II-A and B review existing messaging systems and location-aware sensor networks, respectively. A new opportunity is to seamlessly bridge a messaging system and a location-sensing platform to provide a cross-network service. In Section II-C, we will review some existing cross-network architectures, and point out the main features that distinguish our VS from those works.

A. *Instant Messaging Services*

We present messaging services in telecommunication systems first, followed by those in IP-based networks.

- Short message service (SMS) in GSM [13]: As part of the GSM Phase 1 standard, SMS is a store-and-forward service providing text-based messages originated from and sent to mobile phones. Currently, the SMS center is in its second-generation, which is capable of handling more messages in a more efficient and reliable way. Furthermore, national SMS internetworking and international SMS roaming allow mobile users to communicate short messages freely between different operators. As GSM is evolving to encompass high-speed packet data services (e.g., GPRS), SMS is expected to support longer messages. In addition, potential multimedia applications are under development. For example, the emerging multimedia messaging service (MMS) supports combinations of text, audio, picture, and video.
- NTT DoCoMo i-mode/FOMA [14]: The i-mode platform developed by Japan NTT DoCoMo provides a convenient interface for content providers over mobile phone telecommunication systems. This mobile service allows users to access more than 75,000 Internet sites, as well as specialized services such as e-mail, online shopping and banking, ticket reservations, and restaurant advice. Charges in i-mode are based on the volume of data transmitted, rather than the amount of time remaining connected. With the success of i-mode, in October 2001, NTT DoCoMo further launched the FOMA services based on wideband-code-division multiple-access (CDMA) third-generation (3G) networks. FOMA supports high-speed data transmissions from 64 to 384 kb/s, thus enabling more attractive high-speed services.
- ICQ [10]: Originated from the pronunciation of “I seek you”, ICQ enables Internet users with capabilities of sending instant messages, online chatting, as well as files exchanges. Besides peer-to-peer instant messaging, ICQ also allows users to conduct group conferences or participate in online games. Furthermore, another ICQ-based software, ICQphone, allows users to enjoy voice communications over PCs/IP phones via IP telephony technologies.
- MSN messenger [8]: Similar to ICQ, the MSN messenger (or. NET messenger) delivered by Microsoft also creates an environment for users to pleasantly enjoy instant message, chat, file transfer, and video conference. An MSN messenger client program will connect to an MSN messenger server over the Internet. The client then exchanges data with other clients through the server, which keeps track of users’ current points of Internet attachment. The instant messages are forwarded by the server, and processed by the destined client itself. MSN messenger was not bundled with Windows operating systems until Windows XP, in which the messaging service is incorporated in a program known as Windows Messenger.

B. *Wireless Sensor Networks (WSNs)*

The WSN has many applications in disaster rescue and environment monitoring applications. Reference [2] discusses important design issues of WSN. Advances in microsensor and communication technologies have made it possible to manufacture small and cost-effective WSNs [17], [22]. Several attractive applications of WSNs have been reported [3], [19]. Below, we review several important systems.

- Active badge [27]: The active badge system provides a means for locating individuals within a building by determining the locations of active badges worn by them. An active badge is a small device that can transmit a unique infrared signal periodically. Important places within a building such as offices and corridors can be equipped with networked sensors (i.e., badge readers) to detect these badges. The locations of badges (and, hence, their wearers) are then sent from the sensors to a server on the backbone.
- RFID [11]: Radio frequency identification (RFID) technology has been used for years in transportation applications (rail car tracking, toll collection, and vehicular access control) and inventory management and monitoring. A typical RFID system consists of some tags, readers, and processing servers on the backbone network. A reader can transmit radio waves to trigger antennas of tags. A tag then can convert the signal into DC voltage to charge itself and in turn emit radio signals with identification information for readers to interpret. As a result, the sizes of tags can remain very small due to this batteryless design. Readers then can pass the collected data on to the backbone servers to exploit further applications.

- Berkeley Smart Dust [28]: The Smart Dust mote developed by U.C. Berkeley contains a microcontroller. Periodically the microcontroller gets readings from sensors, which can measure different physical or chemical stimuli such as temperature, ambient light, vibration, acceleration, and air pressure, and processes the data. It also occasionally turns on its optical receiver to see if anyone is trying to communicate with it. In response to a message or upon its own initiative, the microcontroller can use its corner cube retroreflector or laser to communication with other motes or base stations. A smart dust is designed to be only a few cubic millimeters in volume, with sensing, communicating, and self-powering capabilities. Given the limited available storage, one major challenge is to incorporate power-saving designs in all functional parts.
- MIT cricket [23]: The cricket indoor location system is developed by MIT. The system is based on TinyOS and supports continuous object tracking. Following the time difference of arrival (TDoA) model, the cricket compass uses a combination of radio frequency (RF) and ultrasound technologies to determine the position and orientation of a device. A program called listener is used to analyze beacons from objects. A maximum-likelihood estimator is used to correlate a sequence of location information.

C. Existing Cross-Network Services

In Sections II-A and II-B, we have reviewed several instant messaging and sensor networking technologies. We observe that combining these two technologies would be very powerful to provide many novel applications. The sensor networks can detect and report environment-related information and events. Then through instant messaging systems, these events can be sent to the outside world for processing immediately. These events may trigger human or application programs to take actions, which may be further fed back into the sensor networks. Cross-network services may be an important trend in the near future. Several works have addressed integration wireless local area network (WLAN) and 3GPP/CDMA2000/GPRS [1], [5], [24]. As far as we know, no much work has addressed the integration of instant messaging and sensor networking technologies as our work does.

III. Event-Driven Messaging Services: A Visitor System

We have prototyped a VS by combining a Bluetooth-based sensor network and the GSM SMS instant messaging service. Our VS targets at providing novel and friendly services for visitors coming to a building, such as a company headquarters. The devices at visitors' hands can be a Bluetooth-enabled laptop/palmtop or a Bluetooth-enabled GSM WAP-compatible handset. We deploy Bluetooths in important locations within a building, which are connected to a backbone wireline network, to form a sensor network. We use the Bluetooth-based sensor network to identify the locations of visitors/employees in the building, as well as to provide data services. Messaging services are delivered to visitors/employees via either Bluetooth or GSM SMS.

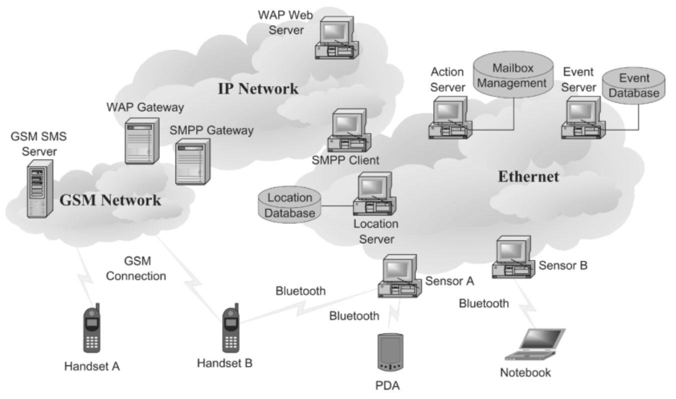


Fig. 1. Visitor system architecture.

GSM SMS has become a pervasive telecommunication data service, and has triggered many interesting applications. On the other hand, Bluetooth is characterized by low price, low-power consumption, and short-range communication, making it a potential choice for supporting positioning in an indoor, office environment. Nowadays, many mobile devices are already equipped with Bluetooths. As a result, we adopt Bluetooth and GSM as our prototyping platform. Through the implementation, we have developed several interesting location-based "event-driven" services.

A. System Architecture

The architecture of our VS is illustrated in Fig. 1, which includes user equipments, sensors, backbone networks, and several servers. Next, we detail each component separately.

In our VS, user equipment could be laptops, palmtops, or GSM handsets with WAP capability. Since we adopt Bluetooth as the sensing technology, all participant mobile terminals should be Bluetooth-enabled. One exception is a single GSM handset that is only WAP-enabled, in which case the user can only receive our instant messageing services, but can not be detected while traveling in our VS coverage area. Every mobile terminal registered in VS is assigned a unique logical ID. We classify user equipment into two categories: *program-loadable*, such as laptops and PDAs, and *program-unloadable*, such as mobile phones. For program-loadable terminals, a client program will be installed such that users can configure their own events and query databases. For program-unloadable terminals, a WAP interface is provided in VS so that users can still enjoy most functionalities provided to program-loadable devices.

A Bluetooth-based sensor network is deployed at important venues in an office building. These sensors are responsible for location sensing. At every predefined 30 s, each sensor will perform the "inquiry" procedure to discover newly arriving devices. The inquiry interval is in fact adjustable (we will further discuss this parameter in Section IV). Once a new device is discovered, the sensor will obtain its Bluetooth medium access control (MAC) address and logical ID, and notify the location server to establish a MAC-ID-location mapping in its database. Sensors also keep MAC-ID mapping at their local databases. Users' subsequent communications are done via logical IDs. Note that in addition to user location tracking, sensors are also responsible for relaying information between users and servers, which includes event configuration with the event server and instant message exchange with the action server.

The backbone network of VS is an integrated IP data network over the Bluetooth sensor network and GSM cellular network. The information exchange from network to network is transparent to end-users. In this way, communications are no longer confined to the limited office area. In particular, with the convenient and robust SMS service supported by GSM, we successfully extend the service area of VS. Two gateways, WAP and short message peer to peer (SMPP), are used to bridge these networks, which will be elaborated later on.

Our VS consists of a number of servers to support event-driven messaging services and a variety of functions. A modular system architecture is adopted in our design, so as to facilitate future extensions and improve flexibilities for third-party application developments. Note that those logically separated components can actually be executed on the same machine through multiple process threads. Next, we describe servers in VS in more details.

- **Location server**: The location server maintains the user-location mapping in a database. Whenever a sensor detects a newly joining device, it will register the device ID with the location server. There is a two-way handshaking between the location server and sensors. Only after the registration is completed can the corresponding mapping be committed to its local database. Once a device is detected absent by a sensor in a prespecified interval, the sensor will notify the location server to deregister that device. Other servers, such as event server or action server, will query the location server to obtain such user-location mapping information.
- **Event server**: In VS, actions are triggered by events. An event could be an instant message request, time event, location event, or their combinations. An instant message request is simply a message to be delivered to a terminal and it will be processed immediately. In VS, there are two basic events: *time event* and *location event* (detailed definitions to be presented in Section III-B). A user can configure a basic event or a compound event with the event server. The event server will parse users' requests into basic events and store them into its event database. Time events will be stored locally, while location events will be delivered to the Location server. Time events are triggered by a local clock at the event server. Location events are stored at the location server and triggered by users' movements. Once a location event becomes true, a signal will be sent from the location server to the event server. The event server will then check its event database and, on a user event becoming true, trigger the action server to take proper actions. Once the action request is sent to the action server, the corresponding event record will be removed from the event database.
- **Action server**: The responsibility of the action server is to really deliver the services. Basically, the Action server is triggered by commands from the event server. Upon receiving an action instruction, the action server will first issue a query to the location server to retrieve the receiver's location information. According to the receiver's profile, if it is recognized as a GSM handset, the service will be forwarded onto the GSM network via several assistant servers and delivered to the destined mobile phone. On the other hand, for services destined for a laptop/palmtop currently located in the VS coverage area, the action server acts as a gateway to relay the services to the receiver.

  To increase reliability, we associate a mailbox with each device ID. According to the classification of *mailbox-based schemes* in [6], our mailbox is stationary with "push" message delivery. However, we do not synchronize between mobile terminal migration and mailbox forwarding. Instead, we adopt a retransmission strategy. Before a message is successfully delivered, it is buffered locally at the action server. Each buffered message will be retransmitted at predefined time intervals. To avoid wasting too much resource on retransmissions, the retransmission interval is doubled each time a failure is experienced. We set up a lifetime of 2 hours for each message. Once the lifetime expires, the corresponding message will be deleted from the buffer.
- **SMPP client**: The short message peer to peer (SMPP) protocol is an open industry standard messaging protocol designed to simplify integration of data applications with wireless mobile networks such as GSM, time-division multiple-access (TDMA), and CDMA. The current system that we are using is SMPP v3.3 (the newest version is v5.0, dated February 2003). In order to provide messaging services to GSM handsets, we developed a SMPP v3.3 client program. To transmit, we need to specify the SMPP Gateway Host Name, Port Number, User Name, Password, and Telephone Number in the URL text. The SMPP client is responsible for creating a SMPP connection to SMPP gateway. The message content is attached at the end of the URL text, which will be formatted by the SMS protocol and forwarded to the GSM SMS server for delivery.
- **WAP web server**: WAP provides an interface for mobile handsets to access the Internet. GSM handset users can configure their requests/services through our WAP server. As a result, we set up a WAP web page to accept users' configuration request. This web page is developed using wireless markup language (WML) and WML script. Any WAP-compatible GSM handsets can connect to the WAP web server and enjoy available VS services. On receipt of a request, the WAP server will contact the event server, which is responsible for processing the request.
- **Other assistant servers**: In the prototyped system, we utilized several existing interfaces available in GSM telecommunication system. Those assistant servers include SMPP Gateway, WAP Gateway, and GSM SMS server.

B. Events and Actions

One of the main features that distinguishes our VS from other messaging systems is its event-driven and location-aware service capability. The concept of events is useful when determining whether to take an action or not. Furthermore, a combination of multiple basic events may support more complicated novel applications. For example, a general manager may want to be notified that a meeting is ready only when the scheduled time is up and all other meeting members have shown up in the meeting room. A staff may want to be notified if he/she is not in the meeting room within 3 min before the meeting time.

In light of this, services in VS are accomplished by event-driven actions, which can be formulated by a number of rules, each with the following format:

$$\langle EvntService\rangle = On\langle EvntVal\rangle Do\langle Action\rangle.$$

In a VS, there are two types of basic events.

- **L**ocation event: This simply includes someone entering or leaving an area.
- **T**ime event: Four types of temporal concept can be specified: absolute time, relative time, time interval, and periodical time. For example, we can specify "10 a.m. on 7/9/2003," "20 min. after ID leaves sensor X," "from 10:00 a.m. to 10:15 a.m.," and "every 5 min."

More complicated events can be combined from basic events through logical operators, such as "AND," "OR," and "NOT."

We summarize the definition of events in the following EBNF-like recursive grammar:

$$\begin{aligned}
\langle EvntVal\rangle &= \langle SubEvntVal\rangle AND\langle EvntVal\rangle \mid \\
&\quad\ \langle SubEvntVal\rangle OR\langle EvntVal\rangle \mid \\
&\quad\ \langle SubEvntVal\rangle \\
\langle SubEvntVal\rangle &= (\langle EvntVal\rangle) \mid NOT\langle SubEvntVal\rangle \\
&\quad\ \langle SnglEvntVal\rangle \\
\langle SnglEvntVal\rangle &= \langle LocEvnt\rangle \mid \langle TimeEvnt\rangle \\
\langle LocEvnt\rangle &= \langle ID\rangle\langle Rel\rangle\langle Sensor\_X\rangle \\
\langle Rel\rangle &= ENTER \mid LEAVE \\
\langle TimeEvnt\rangle &= \langle min/hr/dat/mon/yr\rangle \mid \\
&\quad\ \langle TimeOfEvnt(LocEvnt) + min/hr\rangle \mid \\
&\quad\ \langle min/hr/dat/mon/yr, \\
&\qquad min/hr/dat/mon/yr\rangle \mid \\
&\quad\ \langle min/hr/dat/mon/yr, period\rangle.
\end{aligned}$$

Note that $TimeOfEvnt(LocEvnt)$ returns the actual time when a particular location event happens.

Actions in a VS are all message transmission commands. The service types include unicast, multicast, geocast, and broadcast. The receiver ID (i.e., target of the transmission) can be a user, a group of users, or users in a particular space. The mapping between service type and receiver ID is illustrated in Table I.

C. Service Operations and Message Flows

Based on the above definition of event-driven actions, we next discuss the operations of a VS and some real message flows. We will explain through two examples in this section.

The first example demonstrates how one is reminded by an instant message when a meeting is ready. Suppose that manager Mike calls for a meeting involving staffs Alice and Bob at

TABLE I
SERVICE TYPES AND CORRESPONDING RECEIVER IDs IN VS

	Receiver ID	Service Type
1	Single user	Unicast
2	Group	Multicast
3	Sensor	Geocast
4	ALL	Broadcast

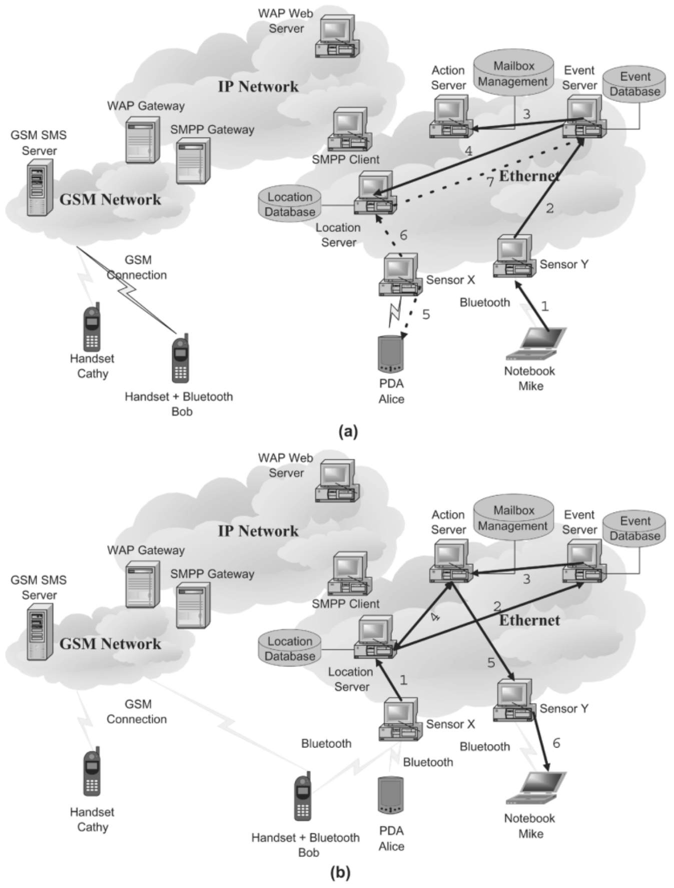


Fig. 2. Message flows for a meeting-reminding instant message service.

10:00 a.m. However, Mike does not want to be kept in waiting if Alice and Bob is late. So Mike uses a notebook installed with the VS client program and sets up an event service: "On (time $= \langle 10:00, 10:10\rangle$) AND (Alice ENTER Sensor X) AND (Bob ENTER Sensor X) Do Unicast(Mike, "staff meeting is ready")".

Suppose that Mike is now in sensor Y. We illustrate the message flow in Fig. 2(a). At first, Mike configures the above event-driven action through sensor Y. In this case, since Mike uses a notebook with a Bluetooth, the Bluetooth is used as a data network to connect to the event server. Through our VS client program, sensor Y will relay the request. On receiving the request, the event server parses the request, which includes a time event and two location events together with a unicast action. The time event will be stored locally at the event server, while the two location events will be forwarded to the location server. The action server will also be contacted to register the unicast action together with an index to the action. Note that the index will be stored at the event server, for which to trigger the corresponding action later on. Next, suppose that Alice, who has a PDA installed with VS, arrives at sensor X first. Then, such an event will be captured by sensor X, which will conduct a location update with the location server. (Note that this update action is independent of the content of the location server; it is conducted whenever a sensor finds a device entering or leaving its coverage.) On receiving the update of Alice's location, the location server checks its local database and finds that such an event should be signaled to the event server.

Later on, as shown in Fig. 2(b), as Bob, who has a Bluetooth-enabled handset, arrives at the meeting room. Such an entering event will also be captured by sensor X and triggered to the location server, and then the event server. Depending on whether the time event has become true or not, the Event server may trigger the unicast action immediately or at later time. Once all events are satisfied, the action server will be triggered by the proper action index. The action server then queries Mike's current location and sends the reminder to Mike. On receiving an acknowledgment message from Mike, all related information at these servers will be cleaned.

Next, suppose that another staff Cathy needs Mike to sign a document but finds that Mike is still in the meeting. Suppose that Cathy does not want Mike to be disturbed, but wants to send herself a message three minutes after Mike leaves the meeting room. Therefore, Cathy may submit a request: "On (time = TimeOfEvnt(Mike LEAVE Sensor X) + 3) Do Unicast(Cathy, "Mike is available now")."

In Fig. 3(a), we show the message flow for Cathy's request through a WAP-enabled handset. In this case, the handset does not need to be Bluetooth-enabled. The configuration procedure will be done through our WAP server. Note that in our implementation, we regard the cellular network as a single special sensor in our VS network. So the WAP server will go through our SMPP client to submit the event-driven action to the event server. In this way, the event server has a consistent view on users in VS, and all handset users are regarded as under a special "GSM sensor." Fig. 3(b) shows the rest of the message flow after Mike leaves the meeting room. Note that since Cathy is recognized as resident in the special "GSM sensor" network, the SMPP client will be contacted to provide the instant message service. The SMPP client will then contact the SMPP Gateway, which will connect to the GSM SMS server, who actually delivers the short message.

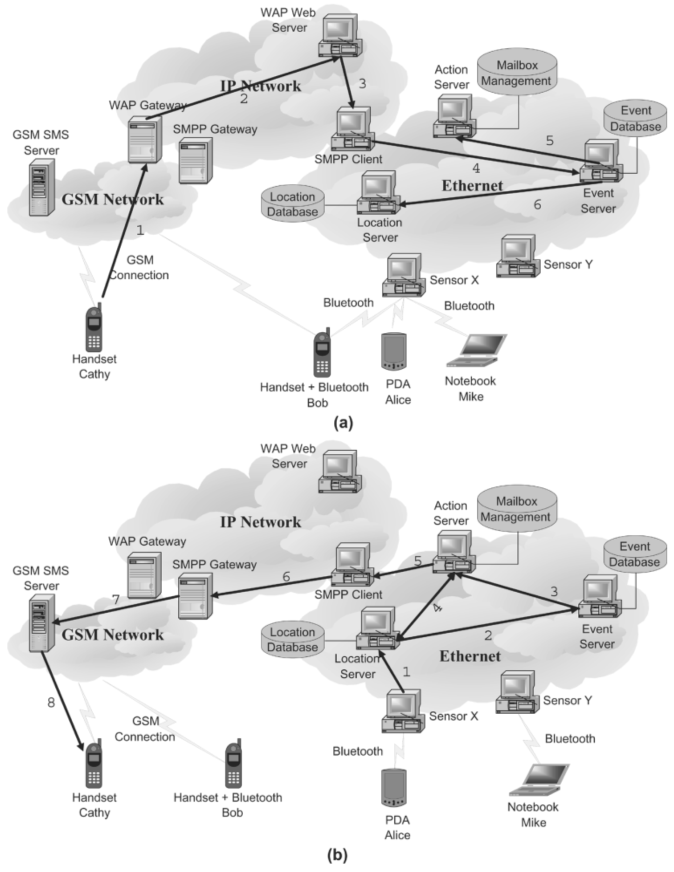


Fig. 3. Message flows for a relative-time instant message service.

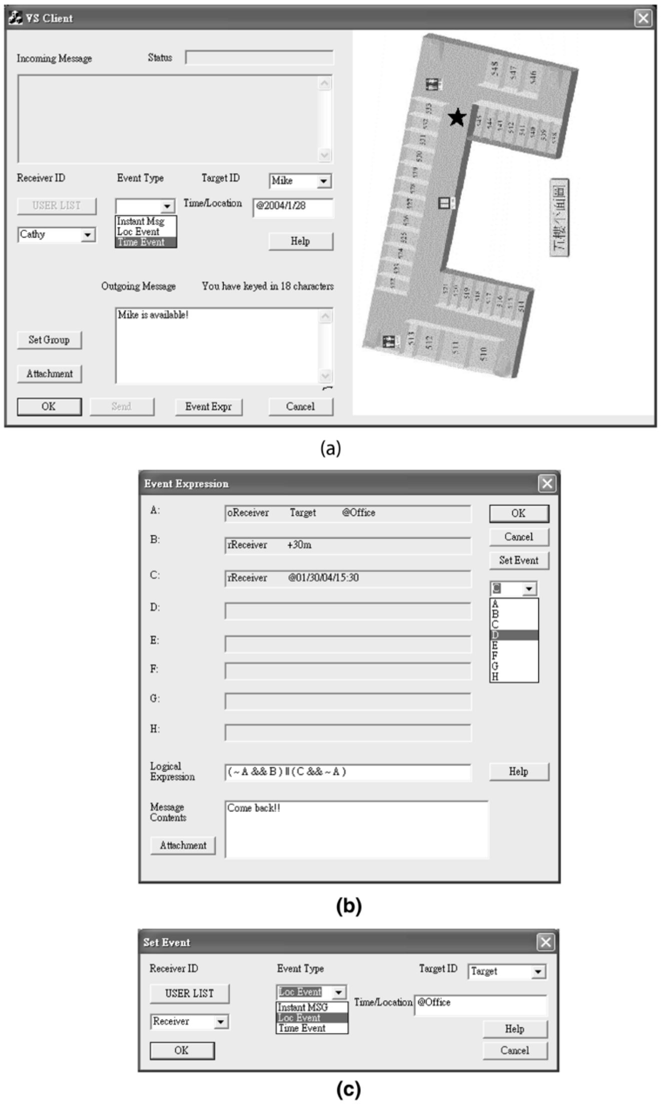
D. Implementation Details and User Interfaces

In this section, we demonstrate the configuration interfaces of VS. We first explain the interfaces for laptops/palmtops, followed by the interactive WAP web pages established for handsets.

Fig. 4(a) is the main user interface of the VS client software for program-loadable devices. On the right-hand side, the asterisk marks the current location of the user. The user can send an instant message by selecting "Instant Msg" as the event type. After filling in receiver ID and outgoing message, clicking the OK button will submit the request. The user can configure a basic event as follows. To set up a basic time event, the event type field should be declared first, followed by filling a parameter in the time/location field with one of the following formats: @mm/dd/yy/hh:xx (for absolute time), +xx{m,s,h} (for relative time, xx minutes/seconds/hours after the current time), and #xx{m,s,h} $\sim$ yy{m,s,h} (for time interval, every xx time units, lasting for yy time units). To set up a basic location event, we first declare the event type field followed by entering desired sensor with the format: @SensorID. Then, target ID field should be declared by filling in either a single device or a group of devices.

To configure a compound event, one should click on the "event expr" in the main window. Then, the event expression window, as shown in Fig. 4(b) will pop up. Up to eight basic events (numbered A–H) can be defined. To define a basic event, the "set event" button should be clicked, which will bring up the interface as shown in Fig. 4(c). The setup procedure is similar to what is discussed earlier. After configuring all basic events, the user can enter a logic statement in the field logical expression in Fig. 4(b).

We also provide online Help through the main window. In addition, buttons USER LIST and set group facilitate users to check users currently visible by VS and to set up a group ID for multicast services. The Attachment option, though not discussed yet, supports simple file transfers between users. At the current stage, VS only allows text-based messaging services due to the limited bandwidth and MAC restrictions imposed by


VS Client

Incoming Message Status

Receiver ID: Cathy
Event Type: Instant Msg / Loc Event / Time Event
Target ID: Mike
Time/Location: @2004/1/28

Outgoing Message — You have keyed in 18 characters
Mike is available!

(a)

Event Expression

A: oReceiver Target @Office
B: rReceiver +30m
C: rReceiver @01/30/04/15:30
D:
E:
F:
G:
H:
Logical Expression: (~ A && B) || (C && ~ A)
Message Contents: Come back!!

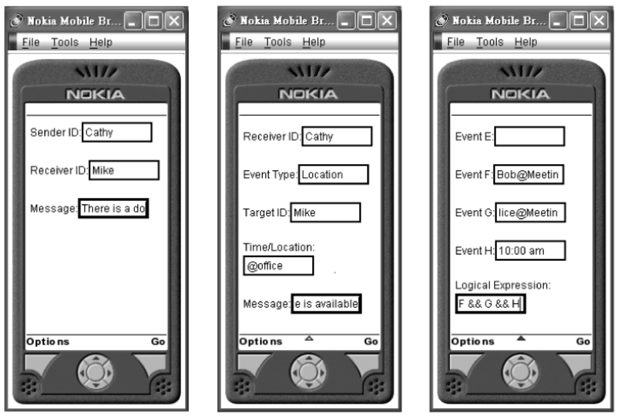
(b)

Set Event

Receiver ID: Receiver
Event Type: Loc Event (Instant MSG / Loc Event / Time Event)
Target ID: Target
Time/Location: @Office

(c)

Fig. 4. Interfaces for laptop/palmtop terminals in VS.


Sender ID: Cathy
Receiver ID: Mike
Message: There is a do

(a)

Receiver ID: Cathy
Event Type: Location
Target ID: Mike
Time/Location: @office
Message: e is available

(b)

Event E:
Event F: Bob@Meetin
Event G: lice@Meetin
Event H: 10:00 am
Logical Expression: F && G && H

(c)

Fig. 5. Interactive WAP web page interfaces for handsets in VS. (a) Instant message. (b) Single event configuration. (c) Hybrid event configuration.

Bluetooth. Multimedia applications are being considered and will be directed to our future work.

The WAP interface is designed following the similar philosophy as discussed above, except that the setup procedure is decomposed into several small WAP pages. We are unable to provide all details due to space limitation. Fig. 5 shows some WAP web pages for handset configurations.

Our current implementation is based on IBM X21/X24 notebooks with Bluetooth USB dongles (with Broadcom chipsets and firmware specification v1.1b), HP iPAQ H3630 with Bluetooth CF cards, and Sony Ericsson T68i handsets. Windows XP SP1 is used, and the server/client programming environment is Visual C++ 6.0. The number of active slaves per piconet is limited to 7. Although theoretically a lot of slaves can be supported in one piconet, since the current system does not support the *park* mode, only seven visitors can be tracked by one Bluetooth sensor. Users are required to register before they can use our system. We consider security and authentication as independent issues, so currently everyone can register into VS. Devices without going through the registration procedure will not be tracked by our system.

IV. Performance Evaluations and Discussions

Our VS relies on Bluetooth sensors to track the movement of objects. Below, we propose a model to evaluate the sensing capability of a given Bluetooth network and discuss some possible extensions of VS.

A. Sensing and Detecting Capability

In a VS, sensor nodes are required to perform Bluetooth inquiry periodically. We experiment on different inquiry intervals from 10 s to 1 min. Intuitively, the larger the inquiry interval, the longer the connection latency will be. However, a short inquiry interval means that sensors will spend significant time discovering devices and thus messages buffered at sensors may experience delays or even dropping. Due to its special inquiry frequency hopping rules, Bluetooth suggests that each inquiry should last for at least 10 s. Otherwise, the "inquiry" from a sensor and the "inquiry scan" from a device are likely to miss each other. Our experimental results suggest that an inquiry interval of 25–30 s performs better, resulting in an average connection time of about 15 s. Several works have indicated that Bluetooth suffers from long discovery time. (The Bluetooth v1.2 is claimed to have better connection efficiency. However, the firmware update is not available to us at the time of implementation. A lot of works have tried to improve the inquiry speed of a single Bluetooth piconet [16], [21], [25], [26], [29], but this is regarded as an independent issue in our work.)

Our VS is built upon a multipiconet environment. Below, we propose an analytical model to evaluate the sensing capability of such an environment. We are given a sensing field A with a number of arbitrarily deployed sensors. Each Bluetooth sensor has a circular sensing range, and sensors may have overlapping sensing ranges. Sensors are not synchronized in time (in terms of inquiry timing). Given any user appearing in any location in A, we would like to analyze the average latency L such that the user can be detected by any of the sensors after it appears.

1) Analytical Results: Let A_i be the area in A that is covered by exactly i sensors $1 \leq i \leq n$, where n is the total number of sensors. (We assume that every point in A is covered by at least

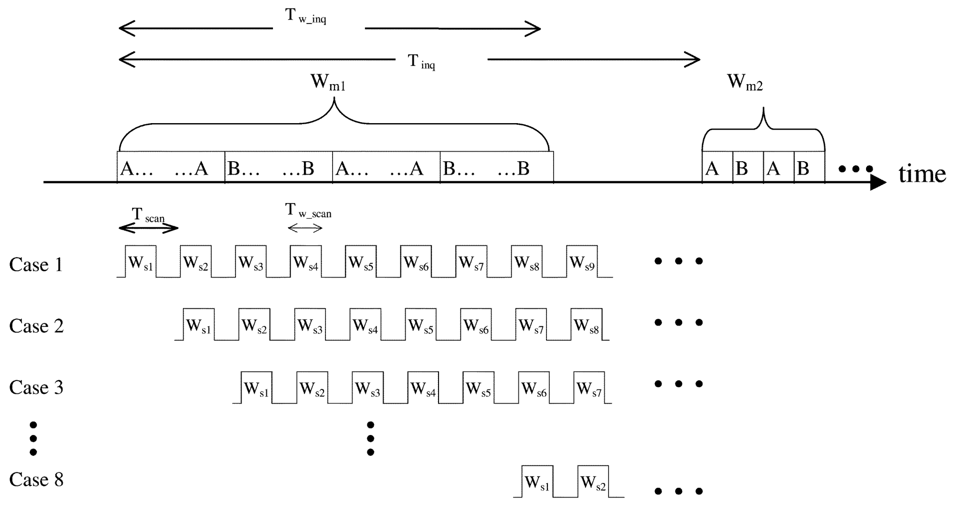

T w_inq
T inq
Wm1
Wm2
A... ...A B... ...B A... ...A B... ...B
A B A B
time
T scan
T w_scan
Case 1
Case 2
Case 3
Case 8
Ws1 Ws2 Ws3 Ws4 Ws5 Ws6 Ws7 Ws8 Ws9

Fig. 6. Eight possible cases for the slave to start its inquiry scan in the analysis of D.

one sensor; otherwise, analyzing the average detection latency is meaningless.) Also, let L_i be the latency such that a user is detected by any piconet/sensor after it appears in a point in A_i and starts it first inquiry scan window. It is easy to see that

$$L = \frac{A_1}{A} \times L_1 + \frac{A_2}{A} \times L_2 + \cdots + \frac{A_n}{A} \times L_n.$$

Below, we show how to derive L_i. The derivation is based [16], where a probabilistic model is given to analyze the inquiry latency between a master–slave pair. Here, we further extend the result to multiple-sensor-covered scenarios.

In the following analysis, let $T_{\text{inq}}, T_{w\_\text{inq}}, T_{\text{scan}}$, and T_{w_scan} be the lengths of one inquiry interval, one inquiry window, one inquiry scan interval, and one inquiry scan window, respectively. We follow the recommendation of Bluetooth that T_{w_inq} is one half of a sequence of 256 A/B trains [12]. Therefore, a slave's inquiry scan has two chances to match with the frequencies on which the master sends ID packets. First, we analyze the *expected inquiry latency*, denoted by D, for a slave to be found by a master if the former performs inquiry scan during the latter's inquiry window (note that this is a special case of inquiry). Let $W_{s1}, W_{s2}, \ldots,$ be the sequence of inquiry scan windows of the slave, and $W_{m1}, W_{m2}, \ldots,$ that of inquiry windows of the master. We consider all possible positions where the first window W_{s1} appears in the first window W_{m1}. This is illustrated in Fig. 6. Basically, there are eight cases to be considered. In the following, D_j, $1 \le j \le 8$, is the detection delay when W_{s1} appears in the jth $(1/8)$ window of W_{m1}

$$D_1 = \frac{1}{32} \times T_{\text{scan}} + \frac{1}{32} \times (4 \times T_{\text{scan}}) + \frac{14}{32} \times (2 \times T_{\text{scan}}) \quad (1)$$

$$D_2 = \frac{1}{32} \times (3 \times T_{\text{scan}}) + \frac{15}{32} \times T_{\text{scan}} \quad (2)$$

$$D_3 = \frac{1}{32} \times T_{\text{scan}} + \frac{1}{32} \times (4 \times T_{\text{scan}}) + \frac{14}{32} \times (2 \times T_{\text{scan}}) \quad (3)$$

$$D_4 = \frac{1}{32} \times (3 \times T_{\text{scan}}) + \frac{15}{32} \times T_{\text{scan}} \quad (4)$$

$$D_5 = \frac{1}{32} \times T_{\text{scan}} + \frac{1}{32} \times \left(T_{\text{inq}} - 4 \times T_{\text{scan}} + D_1^{(1)}\right) + \frac{14}{32} \times (2 \times T_{\text{scan}}) \quad (5)$$

$$D_6 = \frac{1}{32} \times (T_{\text{inq}} - 5 \times T_{\text{scan}} + D_1) + \frac{15}{32} \times T_{\text{scan}} \quad (6)$$

$$D_7 = \frac{1}{32} \times T_{\text{inq\_scan}} + \frac{15}{32} \times (T_{\text{inq}} - 6 \times_{\text{scan}} + D_1) \quad (7)$$

$$D_8 = \frac{1}{2} \times (T_{\text{inq}} - 7 \times T_{\text{scan}} + D_1). \quad (8)$$

To understand the meanings of D_j, we refer to the discussion of [16]. A slave will repeatedly scan all frequencies of train A in 16 consecutive inquiry scan windows, followed by all frequencies of train B in 16 consecutive inquiry scan windows. For example, for the calculation of D_1, there are 32 possibilities, depending on the frequency in W_{s1}, where the slave waits for the master's ID packets. These possibilities can be classified into four subcases, as illustrated in Fig. 7. It is even possible that the slave may have to wait until the next inquiry window due to frequency mismatch in the current inquiry window (e.g., D_5, D_6, D_7, and D_8 are such cases). So we get the expected delay of D as follows:

$$D = \frac{1}{8} \sum_{j=1}^{8} D_j. \quad (9)$$

Next, we extend the derivation for areas covered by α, $1 \le \alpha \le n$ sensors (i.e., areas of A_α). The situation is more complicated because the inquiry windows of the α masters may or may not overlap with each other in a T_{inq} interval. In the following analysis, for simplicity, we assume that the time axis is divided into slots such that each slot is of length T_{scan}, and T_{inq} is a multiple of T_{scan}. Also, we assume that each sensor's inquiry window starts at the beginning of a slot. Such simplification allows us to analyze the discovery latency on a discrete time axis.

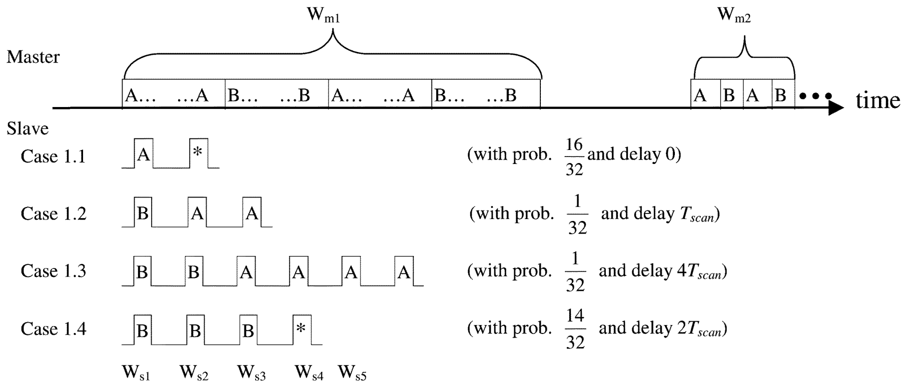

W_m1
W_m2
Master
A... ...A B... ...B A... ...A B... ...B
A B A B
time
Slave
Case 1.1
Case 1.2
Case 1.3
Case 1.4
(with prob. 16/32 and delay 0)
(with prob. 1/32 and delay T_scan)
(with prob. 1/32 and delay 4T_scan)
(with prob. 14/32 and delay 2T_scan)
W_s1 W_s2 W_s3 W_s4 W_s5

Fig. 7. Calculation of D_1. There are four subcases for frequency-matching between the master and the slave. A "$*$" means a "Don't Care" frequency, because a matching has already appeared in the previous inquiry scan window.

Without loss of generality, let the slave start its first inquiry scan window at time 0, and let T_i be the starting time of the ith sensor's first *complete* inquiry window[1] after time 0, $0 \le T_i < T_{\text{inq}}$, $i = 1, 2, \ldots, \alpha$. We first analyze the probability that the first frequency-matching appears between the slave and the ith sensor at the end of the slave's kth inquiry scan window, denoted by $P_1(k, T_i)$, $1 \le k$. Note that although the value of k appears to have no bound, from the above analysis of D, we see that $P_1(k, T_i) = 0$ for all $k > (T_{\text{inq}}/T_{\text{scan}}) + 5$. This is because in the design of Bluetooth's A/B trains, after a slave tries five trains to receive an ID packet from a master in the same inquiry window, a frequency-matching is guaranteed to appear (cases 1, 2, 3, and 4 in Fig. 6 are examples). So we only need to derive $P_1(k, T_i)$ for $k \le (T_{\text{inq}}/T_{\text{scan}}) + 5$. Next, we separate the discussion into two cases.

Case 1) $0 \le T_i \le T_{\text{inq}} - T_{w\_\text{inq}}$. In this case, the first frequency-matching must appear within the first five slots after time T_i. This is illustrated in Fig. 8(a). So we have

$$P_1(k, T_i) = \begin{cases} 0, & k = 1, 2, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6. \end{cases} \quad (10)$$

The above probabilities can be derived from the formulation of D_1.

Case 2) $T_{\text{inq}} - T_{w\_\text{inq}} < T_i < T_{\text{inq}}$. In this case, although the first complete inquiry window of the sensor appears after the first slot, the tail of the master's previous inquiry window should overlap with the first few inquiry scan windows of the slave. An example is illustrated in Fig. 8(b). The first frequency-matching may appear in the tail of the previous inquiry window or the current window. The probabilities can be derived from the formulation of $D_2, D_3, \ldots, D_8$

$$P_1(k, T_i = T_{\text{inq}} - 7 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ 0, & k = 2, 3, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{16}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{16}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{16}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{16}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{16}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \quad (11)$$

$$P_1(k, T_i = T_{\text{inq}} - 6 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{1}{32}, & k = 2 \\ 0, & k = 3, 4, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{17}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{17}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{17}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{17}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{17}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \quad (12)$$

$$P_1(k, T_i = T_{\text{inq}} - 5 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{15}{32}, & k = 2 \\ 0, & k = 3, 4, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{31}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{31}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{31}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \quad (13)$$

[1]Note that the first *complete* inquiry window does not mean the first frequency-matching has to appear after this point. Refer to the discussion of Case 2) for more details.

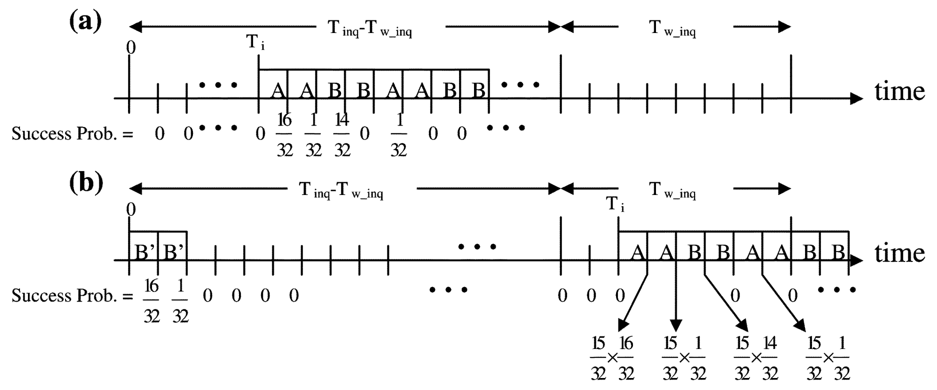


Fig. 8. Analysis of P_1. (a) Example of Case 1). (b) Example of (12) of Case 2).

$$P_1(k, T_i = T_{\text{inq}} - 4 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{1}{32}, & k = 2 \\ \frac{14}{32}, & k = 3 \\ 0, & k = 4, 5, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{31}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{31}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{31}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \geq \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \tag{14}$$

$$P_1(k, T_i = T_{\text{inq}} - 3 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{15}{32}, & k = 2 \\ 0, & k = 3 \\ \frac{1}{32}, & k = 4 \\ 0, & k \geq 5 \end{cases} \tag{15}$$

$$P_1(k, T_i = T_{\text{inq}} - 2 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{1}{32}, & k = 2 \\ \frac{14}{32}, & k = 3 \\ 0, & k = 4 \\ \frac{1}{32}, & k = 5 \\ 0, & k \geq 6 \end{cases} \tag{16}$$

$$P_1(k, T_i = T_{\text{inq}} - 1 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{15}{32}, & k = 2 \\ 0, & k = 3 \\ \frac{1}{32}, & k = 4 \\ 0, & k \geq 5 \end{cases}. \tag{17}$$

The example in Fig. 8(b) represents the calculation of (12). The sensor starts performing its current inquiry procedure at time $T_{\text{inq}} - 6 \times T_{\text{scan}}$ so the tail of its previous inquiry window will overlap with the slave's first and second inquiry scan windows. At the first slot, the frequency-matching will happen only if the slave listens on one of the 16 frequencies in train B, so the success probability for $k = 1$ is $(16/32)$. At the second slot, the first frequency-matching will happen only if the slave listens on the last frequency in train A' at the first slot and on the first frequency in train B' at the second slot. So the success probability for $k = 2$ is $(1/32)$. If the slave did not receive any ID packet during the first two slots, the next inquiry window will appear at time $T_i = T_{\text{inq}} - 6 \times T_{\text{scan}}$. Note that the frequencies in trains A and B have no relation with those in trains A' and B'. So the success probability for $k = (T_i/T_{\text{scan}}) + 1$ is conditional to the failure in the first two slots [with a probability = 1 − $(17/32)$]. Since there are 16 frequencies in train A, the success probability for $k = (T_i/T_{\text{scan}}) + 1$ is $(1 - (17/32)) \times (16/32)$. The rest of the derivation of (12) is similar.

Next, we define $P_2(j, k, T_1, T_2, \ldots, T_\alpha)$ to be the probability that: 1) no frequency-matching appears in the first k slots between the slave and sensors numbered from 1 to $(j - 1)$; 2) no frequency-matching appears in the first $(k - 1)$ slots between the slave and sensors numbered from $(j + 1)$ to α; and 3) the first frequency-matching appears between the slave and the jth sensor in the kth slot. We derive

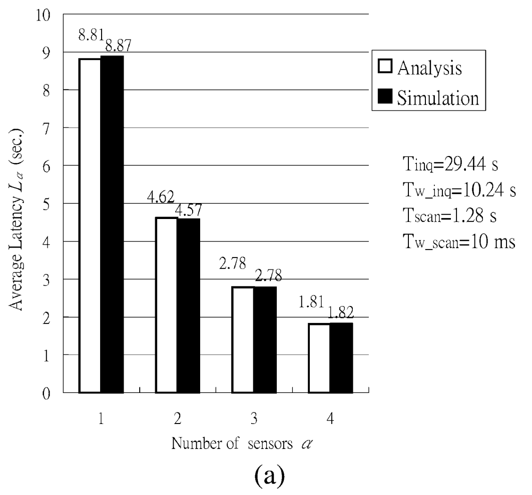
$$\begin{aligned} P_2(j, k, T_1, T_2, \ldots, T_\alpha) = &\prod_{i=1}^{j-1} \left(1 - \sum_{n=1}^{k} P_1(k, T_i)\right) \\ &\times \prod_{i=j+1}^{\alpha} \left(1 - \sum_{n=1}^{k-1} P_1(k, T_i)\right) \\ &\times P_1(k, T_j). \end{aligned} \tag{18}$$

Note that the first, second, and the last terms in (18) are the probabilities for the afore mentioned 1), 2), and 3), respectively. It follows that the probability that the first frequency-matching will appear in the kth slot is

$$P_3(k, T_1, T_2, \ldots, T_\alpha) = \sum_{i=1}^{\alpha} P_2(i, k, T_1, T_2, \ldots, T_\alpha). \tag{19}$$

Finally, we can derive the expected inquiry latency for the areas in A_α

$$\begin{aligned} L_\alpha = &\left(\frac{T_{\text{scan}}}{T_{\text{inq}}}\right)^\alpha \sum_{i_1=0}^{(T_{\text{inq}}/T_{\text{scan}})-1} \sum_{i_2=0}^{(T_{\text{inq}}/T_{\text{scan}})-1} \cdots \\ &\sum_{i_\alpha=0}^{(T_{\text{inq}}/T_{\text{scan}})-1} \sum_{k=1}^{(T_{\text{inq}}/T_{\text{scan}})+5} \\ &\times (P_3(k, i_1 \times T_{\text{scan}}, i_2 \times T_{\text{scan}}, \ldots, i_\alpha \times T_{\text{scan}}) \\ &\times (k - 1) \times T_{\text{scan}}). \end{aligned} \tag{20}$$



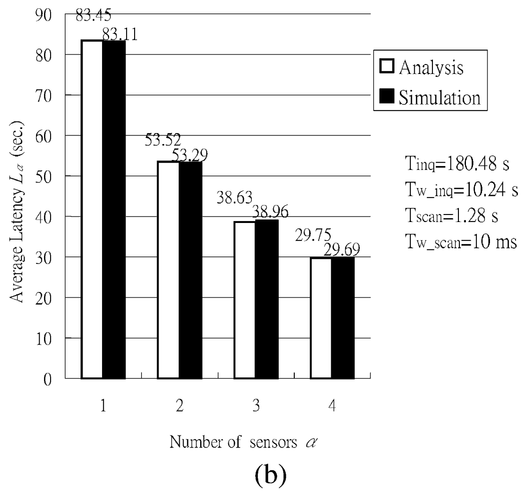
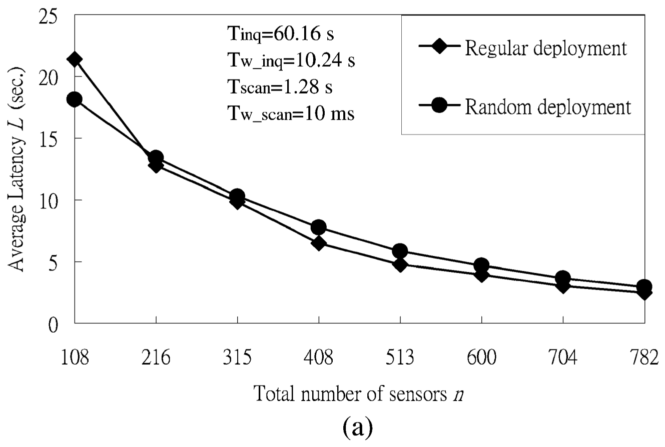


Fig. 9. Average latency $L\alpha$ versus number of sensors α with (a) $T_{\rm inq} = 29.44$ s and (b) $T_{\rm inq} = 180.48$ s.

Note that in (20), the first α $\sum$'s enumerate all combinations of $T_1, T_2, \ldots, T_\alpha$, and the last $\sum$ is to account for all nonzero probabilities for P_3(So k is bounded by $(T_{\rm inq}/T_{\rm scan}) + 5$).

2) Simulation Results: We have also conducted some simulations to verify our analytical results. We first evaluate the discovery latency L_α when an object appears in an area covered by α sensors. In our simulation, sensors do not synchronize their clocks. Fig. 9(a) and (b) shows the latency when $T_{\rm inq} = 29.44$ s and $T_{\rm inq} = 180.48$ s, respectively, with $T_{w\_{\rm inq}} = 10.24$ s, $T_{\rm scan} = 1.28$ s, and $T_{w\_{\rm scan}} = 10$ ms (note that according to our analysis, $T_{\rm inq}$ is a multiple of $T_{\rm scan}$). As can be seen, our analytical results are very close to simulation results.

We have also simulated the network-level latency. A sensing field of size 500 × 500 is simulated. The radius of each Bluetooth coverage area is 30 units. We vary the number of sensors n and observe the detection latency L of the sensing field. Sensors are deployed either in a regular manner or in a random manner. In the former case, the typical triangular deployment is adopted, where each three nearby sensors form a triangle of the same side length. We adjust the side length to adjust the value of n. Generally, as n increases, the latency L will decrease, which is reasonable. The regular deployment has a lower latency than the random deployment for both cases of $T_{\rm inq}$. One exception is for small values of n, where the random deployments have lower latency. This is because in random deployments, some areas could be uncovered (for example, when $n = 108$, 216, 315, and 408, the average ratios of uncovered areas are 31.51%, 10.42%, 3.81%, and 1.55%, respectively). Since such areas are not taken into account, what has shown in Fig. 10 is based on unfair comparison.



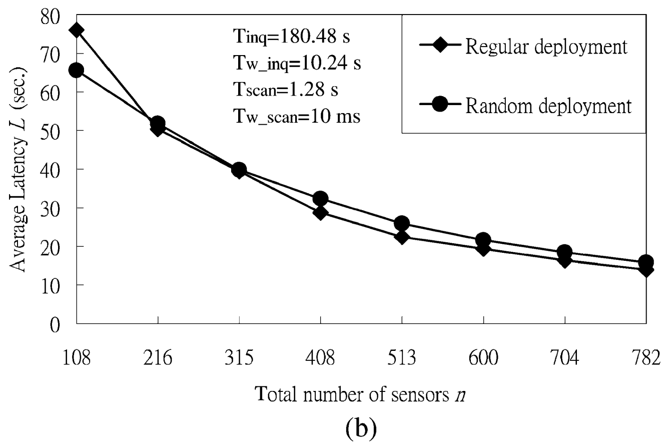
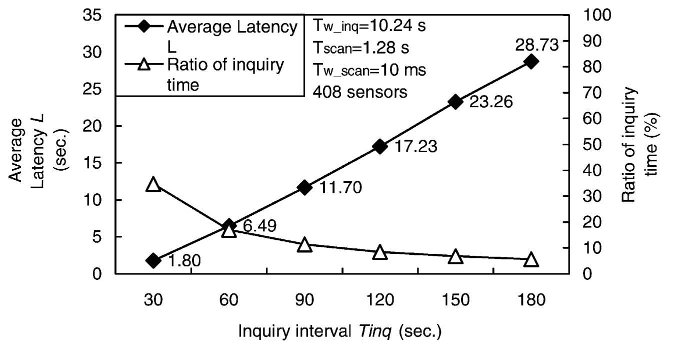


Fig. 10. Average latency L versus total number of sensors n for regularly and randomly deployed networks. (a) $T_{\rm inq} = 60.16$ s. (b) $T_{\rm inq} = 180.48$ s.

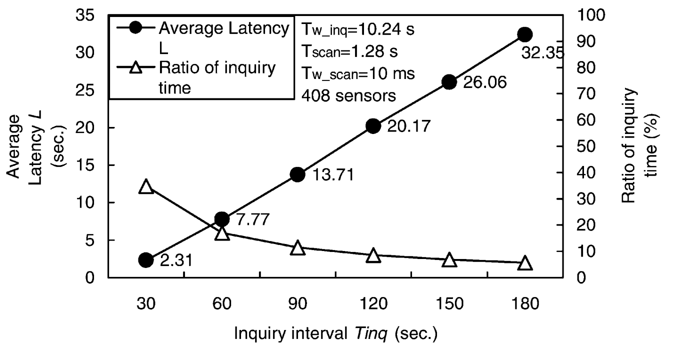
Finally, we observe the impact of $T_{\rm inq}$. In Fig. 11, we vary the value of $T_{\rm inq}$ (which is a multiple of $T_{\rm scan} = 1.28$ s). There are 408 sensors deployed in a regular or random manner. We observe the average latency L and the ratio of time that a sensor spends on inquiry (excluding the inquiry time, a sensor can do data communications.) As can be seen, a larger $T_{\rm inq}$ will increase the latency L linearly, but the inquiry overhead is reduced. So one should properly tune $T_{\rm inq}$ to account for both detection latency and the ratio of time that sensors can spend on data communications to support other activities of VS.

B. User Profile Management

Currently, VS recognizes senders/receivers by device IDs. However, one disadvantage is that device ID provides little information on the ownership of the user. It is also possible that a user owns multiple devices. The problem is that the system is unaware of which mobile device the user is carrying. One solution is to maintain a user profile, which registers all devices he/she



(a)



(b)

Fig. 11. Average latency L and ratio of inquiry time versus inquiry interval T_{inq} with 408 sensors. (a) Regular deployment. (b) Random deployment.

owns. Once we have such information, multicast could be a solution to this problem. With user profiles, several user-centric applications are possible.

C. More Value-Added Services

Limited by Bluetooth and GSM wireless bandwidth, the current VS supports mainly text-based services. We have experimented some graphics files, such as maps, and the transmission performance is acceptable. However, supporting multimedia streaming services requires much more works. With the popularity of GSM MMS services, one extension step is to implement MMS transmissions in VS. This will require modifications in the action server.

V. Conclusion

In this paper, we have reported how we prototyped an event-driven instant messaging application over integrated telecomm and datacomm networks. The proposed visitor system in its current stage exercises both GSM and Bluetooth technologies. With the recent release of Bluetooth v1.2, which emphasizes on faster connection and better interference mitigation, we expect to improve the performance of our system. Our VS aims to provide human-centric services. As we have predicted, unifying novel applications across heterogeneous networks is an attractive trend. From the prototyping experiences, we have shown how to separate logical servers (such as event, location, and action servers) for event-driven messaging services, and the importance of an open interface for third-party application development. Our module-based system design allows software programmers to configure their own systems by just adding or modifying components without changing the base architecture. This flexibility is critical when implementing a real system. Our VS relies on Bluetooth as sensors to track the movement of objects. As far as we know, there is no prior work which addresses the device discovery delay in an environment with multiple Bluetooth masters. Novel analyses have been presented to address this problem, and our simulation results do verify the accuracy of the analyses.

References

[1] K. Ahmavaara, H. Haverinen, and R. Pichna, "Interworking architecture between 3GPP and WLAN systems," *IEEE Commun. Mag.*, vol. 41, no. 11, pp. 74–80, Nov. 2003.
[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[3] A. Baptista, T. Leen, Y. Zhang, A. Chawla, D. Maier, W. C. Feng, W. C. Feng, J. Walpole, C. Silva, and J. Freire, "Environmental observation and forecasting systems: Vision, challenges and successes of a prototype," in *Proc. Int. Annu. Conf. Society Environ. Inf. Sci. (ISEIS)*, 2003.
[4] J. Beutel, O. Kasten, M. Ringwald, F. Siegemund, and L. Thiele, "Bluetooth smart nodes for mobile ad hoc networks," TIK Rep. 67, 2001.
[5] M. M. Buddhikot, G. Chandranmenon, S. Han, Y.-W. Lee, S. Miller, and L. Salgarelli, "Design and implementation of a WLAN/CDMA2000 interworking architecture," *IEEE Commun. Mag.*, vol. 41, no. 11, pp. 90–100, Nov. 2003.
[6] J. Cao, X. Feng, and S. K. Das, "Mailbox-based scheme for mobile agent communications," *IEEE Comput.*, vol. 35, no. 9, pp. 54–60, 2002.
[7] J. Haartsen, W. Allen, and J. Inouye, "Bluetooth: Vision, goals, and architecture," *Mobile Comput. Commun. Rev.*, vol. 1, no. 2, pp. 38–45, 1998.
[8] MSN Messenger. [Online]. Available: http://messenger.msn.com/
[9] (2001) MIT Blueware Project. [Online]. Available: http://nms.lcs.mit.edu/projects/blueware
[10] ICQ Homepage. [Online]. Available: http://web.icq.com/
[11] AIM RFID Technology. [Online]. Available: http://www.aimglobal.org/technologies/rfid/
[12] (2003) Bluetooth SIG Specification v1.2. [Online]. Available: http://www.bluetooth.com
[13] GSM Short Message Service. [Online]. Available: http://www.gsm-world.com/technology/sms/index.shtml
[14] NTT DoCoMo I-Mode. [Online]. Available: http://www.nttdocomo.com/corebiz/imode/global/
[15] E. Isaacs, A. Walendowski, and D. Ranganathan, "Mobile instant messaging through hubbub," *Commun. ACM*, vol. 45, no. 9, pp. 68–72, Sep. 2002.
[16] J.-R. Jiang, Y.-C. Tseng, and B.-R. Linn, "A mechanism for quick Bluetooth device discovery," presented at the Mobile Computing Workshop, Taiwan, 2004, . .
[17] J. Kahn, R. Katz, and K. Pister, "Mobile networking for smart dust," *Mobile Comput. Netw.*, pp. 483–492, 1999.
[18] O. Kasten and M. Langheinrich, "First experiences with Bluetooth in the smart-its distributed sensor network," in *Proc. Workshop Ubiquitous Comput. Commun.*, Oct. 2001.
[19] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Applicat.*, 2002, pp. 88–97.
[20] N. Milanovic, A. Radovanovic, B. Cukanovic, A. Beric, N. Tesovic, and G. Marinkovic, "Bluetooth ad hoc sensor network," Univ. Belgrade, Belgrade, Yugoslavia, Tech. Rep., 2002.
[21] P. Murphy, E. Welsh, and J. P. Frantz, "Using Bluetooth for short-term ad hoc connections between moving vehicles: A feasibility study," in *Proc. IEEE Int. Conf. Veh. Technol. Conf. (VTC)*, May 2002, pp. 414–418.
[22] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, pp. 51–58, May 2000.
[23] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. ACM/IEEE Int. Conf. Mobile Comput. Netw. (MobiCom)*, Aug. 2000, pp. 32–43.
[24] A. K. Salkintzis, C. Fors, and R. Pazhyannur, "WLAN-GPRS integration for next-generation mobile data networks," *IEEE Wireless Commun.*, pp. 112–124, Oct. 2002.
[25] T. Salonidis, P. Bhagwat, and L. Tassiulas, "Proximity awareness and fast connection establishment in bluetooth," in *Proc. Int. Conf. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2000, pp. 141–142.

[26] F. Siegemund and M. Rohs, "Rendezvous layer protocols for bluetooth-enabled smart devices," in *Proc. Int. Conf. Arch. Comput. Syst.*, 2002, pp. 256–273.
[27] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst. (TOIS)*, vol. 10, no. 1, pp. 91–102, Jan. 1992.
[28] B. Warneke, M. Last, B. Liebowitz, and K. S. Pister, "Smart dust: Communicating with a cubic-millimeter computer," *IEEE Comput.*, pp. 2–9, Jan. 2001.
[29] R. Woodings, D. Joos, T. Clifton, and C. D. Knutson, "Rapid heterogeneous connection establishment: Accelerating Bluetooth inquiry using IrDA," in *Proc. IEEE Int. Conf. Wireless Commun. Netw. Conf. (WCNC)*, 2002, pp. 342–349.

Yu-Chee Tseng (S'99–M'95–SM'03) received the B.S. degree in computer science from the National Taiwan University, Taipei, Taiwan, in 1985, the M.S. degree in computer science from the National Tsing-Hua University, Hsin-Chu, Taiwan, in 1987, and the Ph.D. degree in computer and information science from Ohio State University, Columbus, in January 1994.

He worked for D-LINK, Inc., as an Engineer in 1990. He was an Associate Professor at the Chung-Hua University (1994–1996) and at the National Central University, Taoyuan, Taiwan, (1996–1999), and a Full Professor at the National Central University (1999–2000). Since 2000, he has been a Full Professor at the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu, Taiwan. His research interests include mobile computing, wireless communication, network security, and parallel and distributed computing.

Dr. Tseng is a member of the Association for Computing Machinery (ACM). He is a two-time recipient of the Outstanding Research Award, National Science Council, R.O.C., from 2001 to 2002, and 2003 to 2005, and a recipient of the Best Paper Award at the International Conference on Parallel Processing in 2003. Several of his papers have been chosen as Selected/Distinguished Papers in international conferences. He has guided students to participate in several national programming contests and received several awards. He served as a Program Chair in the Wireless Networks and Mobile Computing Workshop, 2000 and 2001, as a Vice Program Chair in the International Conference on Distributed Computing Systems (ICDCS 2004), and as a Vice Program Chair in the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2004). He served as an Associate Editor for *The Computer Journal*, as a Guest Editor for *ACM Wireless Networks* (Special Issue on Advances in Mobile and Wireless Systems), as a Guest Editor for the IEEE TRANSACTIONS ON COMPUTERS (Special Issue on Wireless Internet), as a Guest Editor for the *Journal of Internet Technology* (Special Issue on Wireless Internet: Applications and Systems), as a Guest Editor for *Wireless Communications and Mobile Computing* (Special Issue on Research in Ad Hoc Networking, Smart Sensing, and Pervasive Computing), as an Editor for the *Journal of Information Science and Engineering*, as a Guest Editor for *Telecommunication Systems* (Special Issue on Wireless Sensor Networks), and as a Guest Editor for the *Journal of Information Science and Engineering* (Special Issue on Mobile Computing).

Ting-Yu Lin received the Ph.D. degree in computer science and information engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in December 2002.

She is currently a Software Engineer at the Computer and Communications Research Laboratory, Industrial Technology Research Institute. Her research interests include wireless communication, mobile computing, WLANs/WPANs, sensor networking, and energy conservation designs (software-level).

Dr. Lin was the recipient of the Phi Tau Phi Scholastic Honor Award.

Yen-Ku Liu received the B.S. and M.S. degrees in computer science from the National Chiao-Tung University, Hsinchu, Taiwan, in June 2002 and April 2004, respectively.

His research interests include wireless networks, sensor networks, and Bluetooth technology.

Bing-Rong Lin received the B.S. and M.S. degrees in computer science from the National Chiao-Tung University, Hsinchu, Taiwan, in June 2002 and June 2004, respectively.

His research interests include wireless networks, sensor networks, and Bluetooth technology.