Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Communications 27 (2004) 828–839
computer communications
www.elsevier.com/locate/comcom

An adaptive power-conserving service discipline for bluetooth (APCB) wireless networks☆

Hao Zhu*, Guohong Cao, George Kesidis, Chita Das

Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA

Received 11 September 2002; revised 20 January 2004; accepted 21 January 2004

Abstract

Bluetooth is a new short-range radio technology to form a small wireless system. In most of the current bluetooth products, the master polls the slaves in a round robin (RR) manner and it may waste a significant amount of power. Many solutions were proposed to reduce the power consumption of the slaves. However, these solutions cannot achieve a good balance between the power consumption and QoS provision. We propose an adaptive power-conserving scheme to address this problem. The proposed solution schedules each flow based on its predictive rate and achieves power optimization based on a low-power mode existing in the bluetooth standard. Unlike other research work related to low-power, we also consider QoS provision for each flow. Theoretical analyses verify that our scheme can achieve throughput guarantees, delay guarantees, and fairness guarantees. Simulation results demonstrate that our scheme outperforms the RR scheme and other existing works in terms of significant power saving and good QoS provision. It also shows that there exists a tradeoff between power and delay under varies traffic models.

Keywords: Bluetooth; Power-conserving service model; Guaranteed service model

1. Introduction

Bluetooth is a promising technology which is aimed at supporting wireless connectivity among mobile devices, such as cellular phones, headsets, PDAs, digital cameras, laptop computers and their peripheries. The technology enables the design of low-power, small-size, low-cost radios that can be embedded in existing portable devices [15]. Bluetooth is different from most of the existing wireless networks such as 802.11 wireless LANs, which use a CSMA [7] style MAC protocol. It is a frequency hopping system [6] which can support multiple communication channels in a common area (each channel is defined by a unique frequency hopping sequence). As a result, multiple communication channels can be geographically overlapped and the distance relationship cannot determine the topology of the network.

In bluetooth, a group of nodes sharing a common channel is called a *piconet*. Each piconet has a *master* and at most seven *slaves* as group participants. Within a piconet, the channel is shared using a slotted time division duplex protocol and is managed by the master. Bluetooth supports two types of channels: synchronous and asynchronous. For synchronous communications, the master and slaves communicate with each other at regular intervals of time which are reserved in advance. For asynchronous communications, the master uses a polling style protocol to allocate time slots to the slaves [17]. That means the master polls a slave and the slave responses in the next time slot and so on. The main advantage of the polling based scheduling scheme is simplicity, which makes the bluetooth device simpler, lower power and lower cost compared to other wireless communication devices (e.g. wireless LAN card). In most of the current bluetooth products, the master polls the slaves in a round robin (RR) manner. Many new scheduling schemes [2,9,10,12,13,18] have been proposed to improve the performance of the piconet. However, the polling based RR scheduling has another drawback when considering power consumption. The master polls each slave as quick as possible. If the slave being polled has no packet to send, two time slots will be wasted. As a result, if the slave's traffic density is low, a large amount of power will be wasted due

☆ This work was supported in part by the National Science Foundation (CAREER CCR-0092770 and ITR-0219711).
* Corresponding author.
E-mail addresses: hazhu@cse.psu.edu (H. Zhu), gcao@cse.psu.edu (G. Cao), kesidis@engr.psu.edu (G. Kesidis), das@cse.psu.edu (C. Das).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.01.027

to excessive polling. Meanwhile, since a slave does not know when it will be polled, it has to keep listening and wasting lots of power.

There has been a lot of research on low-power control for wireless devices. At the hardware level, the communication device can adjust the power level used by the mobile transmitter during active communication [16]. At the software level, we can reduce the power consumed by communication devices [11] and non-communication devices such as displays, disks, and CPUs [4]. The underlying principle is to estimate when the device will be used and suspend it for those idle intervals. Following this principle, some solutions [3] have been proposed to reduce the power consumption of slaves in a piconet by putting them to low-power mode periodically. Then, the slave stays in active only when it has packets to transmit or receive, which could significantly increase the power efficiency. These solutions work well when power consumption is the major concern. However, they may not be able to provide good QoS provision to each flow at the same time.

In this paper, we propose a new scheme which focuses on reducing the power consumption of slaves with a certain QoS provision. The basic idea is to let the master poll the slave when the slave has data to send and make the slave stay in the low-power mode until the master wants to communicate with it. In order to maintain QoS guarantees, we use the *guaranteed service model* [19] as the underlying model of our approach and assume that every flow is allocated a flow rate (in bps). Based on this model, we propose a *non-work-conserving* MAC layer scheduling scheme in which the master arranges a power efficient polling sequence based on the current prediction of the flow data rate. We use the *hold* operation mode of bluetooth to make the slave idle whenever there is no data addressed to it so that the slave can avoid unnecessarily staying in active. Intuitively, power conservation is achieved by reducing the number of unnecessary polling slots and unnecessary active periods. Since the prediction may not always be accurate, the slave may not have data to send while being polled, in which case, power will be wasted. In order to reduce this kind of mis-prediction, our scheme applies additive-increase multiplicative-decrease to adaptively adjust the predicted rate of the flow based on the power tuning knob and the flow attribute parameters. We show the throughput, delay, and fairness properties of our scheme via theoretical analyses and demonstrate the advantages of our scheme through extensive simulations.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to the background and the motivation of the paper. In Section 3, we describe our scheduling algorithm in details and give analytical results of throughput, delay and fairness. The performance of our approach is evaluated in Section 4. Section 5 concludes the paper.

2. Background and motivations

2.1. Bluetooth operation modes

Bluetooth defines four operational modes: *Active*, *Sniff*, *Hold* and *Park*. In the *Active mode*, a bluetooth device actively participates on the channel. In the *Sniff mode*, the native clock cycle of a slave's listen activity is reduced to specified periodic time slots, which are called sniff slots, and the master will poll the slave every sniff slots. In the *Hold mode*, a slave goes into sleep mode for a specified amount of time: *holdTO*. After *holdTO* time, the slave returns to active mode. This means that the slave temporarily leaves the channel for a time interval of *holdTO*. Before entering the hold mode, the master and the slave agree on the time duration that the slave should remain in the hold mode. After the slave wakes up, it will synchronize to the traffic on the channel and will wait for further information from the master. In the *Park mode*, the slave is in a sleep mode for an unspecified amount of time and gives up its active member address *AM_ADDR*. The master has to explicitly make the slave active at a future time by broadcasting through the *beacon channel* [1]. The parked slave wakes up at regular intervals to listen to the channel in order to re-synchronize and to check for beacon channel message.

2.2. The guaranteed service model

Each unit of data transmission at the network level in a packet-switch network is a packet. We refer to the sequence of packets transmitted by a source as a *flow* [5]. We consider the *guaranteed service model* as following: before the communication starts, the source needs to specify its flow traffic characteristics and the desired performance requirements. When the network admits the request, it guarantees that the specified performance requirements will be met provided that the source follows its traffic specification [19]. In addition, the network has an admission mechanism that may reject the source's connection request due to lack of resource or administrative constrains. Thus, this service contract is settled before the real data transfer during a connection establishment process and is kept valid throughout the life time of the flow. The network meets the requirements of all flows by appropriately scheduling its resource. A scheduling algorithm can be classified as either *work-conserving* or *non-work-conserving*. For working-conserving scheduling, a server is never idle when there is a packet to send. For non-work-conserving scheduling, each packet is not served until it is eligible [19], even though the server is idle at that time.

2.3. Motivations

In current commercial bluetooth products, the RR scheduling scheme is applied as well as specified in the bluetooth specification. Under this scheduling policy,

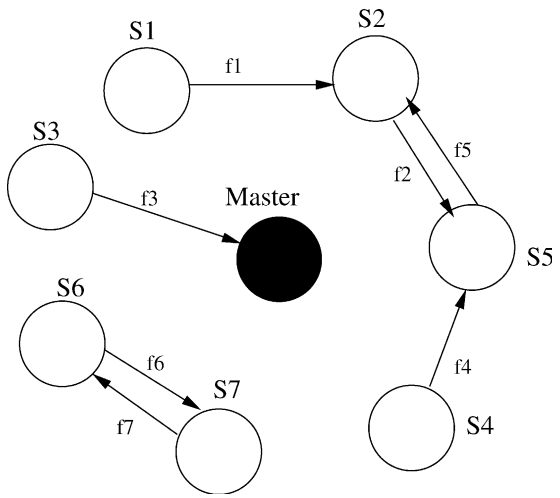


Fig. 1. A piconet example.

the master works in the work-conserving manner and keeps polling the slaves of the piconet in turns. For example, as shown in Fig. 1, there are seven slaves and seven flows in the piconet. Under the RR policy, the master's schedule sequence is: $S_1-S_2-S_3-S_4-S_5-S_6-S_7$. Suppose it is slave S_1's turn, the master has to poll S_1 no matter S_1 has a packet to send or not since it does not know S_1's real-time traffic situation. If S_1 does not have any packet to send, it will reply a NULL packet. Thus, a couple of time slots and some amount of power are wasted due to this polling. As a result, if flow f_1's rate is low, lots of power can be wasted due to the excessive polling towards S_1. Because packet length might vary, other slaves cannot know the exact time when it will be polled. Therefore, they need keep listening to the channel and waste a large amount of power. In order to fix these problems, we propose a MAC layer scheduling scheme and use the hold mode to optimize the power consumption while providing guaranteed service for the flows in the piconet. The basic idea is to let the master poll the slave when the slave has a packet to send. Also, we want to let the slave be active only when the master accesses it. The master allocates time slots to a flow based on the predicted rate of the flow. Since the prediction may not always be accurate, it may incur more power consumption under the scheme due to the mis-prediction. We use an adaptive method to adjust the predicted rate of the flow in order to reduce the cost of mis-predictions. This policy is power-conserving because it can reduce the number of unnecessary polling and let the slave stay in the idle mode as long as possible.

3. An adaptive power-conserving algorithm

In this section, we present the *adaptive power-conserving service discipline for bluetooth* (APCB) which is based on the following assumptions: (i) error-free channel, (ii) single piconet and (iii) the system clock is synchronized among the master and the slaves.

3.1. The APCB service model

We adopt the idea of the virtual clock service model [20] and require each flow to provide its attribute parameters, such as flow rate and burstiness degree, before the communication starts. The sender/receiver of the flow and the master (if different) will keep these parameters. The flow has a queue at its sender side, and the master (if different) forwards the flow's packets to the receiver (if different) and does not buffer the packets. If the master is different from the sender of the flow, which is true in most cases, it cannot get the real-time knowledge of the arrival time of a flow's packets; i.e. when the sender has the next packet to send. As a result, the master only uses the *expected arrival time* (EAT) of the packets of the flow to predict when the sender will have packet to send. Without loss of correctness, when the master is the sender of a flow, the master can still use EAT to indicate when the next packet will be sent. For clarity, we introduce some notations in Table 1. The pseudo-code of APCB is shown in Fig. 3.

In the APCB service model, the master works in non-work-conserving manner in our scheduling policy. It will try to serve the flow which has the packet with the smallest EAT, and use the node's address to break the tie. The whole algorithm is related to the master and the end nodes, which are $Sender_i$ and $Receiver_i$ of flow i. Applying the same algorithm, the master, $Sender_i$, and $Receiver_i$ can mutually agree on the next polling time based on the current packet length L_i^k and the current expected data rate $r(p_i^k)$. If there is a mis-prediction, which means $Sender_i$ does not have data to send while being polled, they will adaptively adjust $r(p_i^k)$ and prolong the interval of the next polling time to $L_i^{\max}/r(p_i^k)$. We use $L_i^{\max}$ as the current packet length when a mis-prediction happens in order to let the scheme work more power efficiently. Because the end nodes also adjust $r(p_i^k)$ as the master does, mutual agreement can be established. Since the nodes apply the same algorithm, if the master is different from the end nodes, unlike the hold

Table 1
Notations used in Section 3

Notation	Description
L_i^k	The length of the kth packet of flow i
$L_i^{\max}$	The maximum packet length of flow i
$\mathrm{EAT}(p_i^k)$	The *expected arrival time* of the kth packet of flow i
r_i	The maximum rate admitted to flow i (in bps)
$r(p_i^k)$	The *expected data rate* of flow i (in bps), where the kth packet is the head-of-line packet. It is initialized to r_i
α_i	The power tuning knob for flow i and 0. $0 < \alpha_i \leq 1.0$
σ_i	The burstiness degree of flow i and $\sigma_i \geq 1$
$\langle start, sleep \rangle$	The node will hold from time *start* and last *sleep* second(s)
clock	The system time clock
δ	The time for the end nodes of flow i to re-synchronize the channel from hold to active, which is set to be 1 time slot (625 μs) in this paper

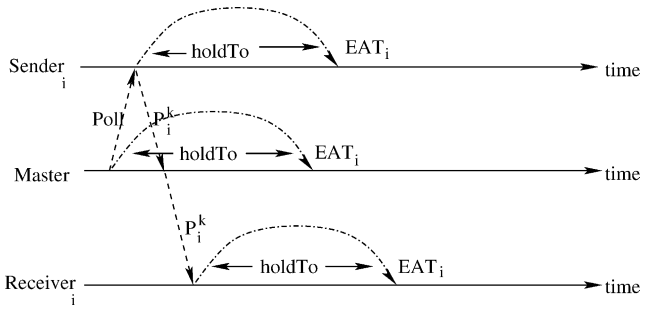


Fig. 2. An example of mutual agreement among the nodes.

operation defined in the specification, no extra messages are required to explicitly coordinate the hold period between the master and the end nodes under the APCB scheme, which is shown in Fig. 2. As shown in the figure, $Sender_i$ and $Receiver_i$ calculate the EAT of flow i independently, and know that the master will not serve flow i until the EAT_i. Therefore, the master, $Sender_i$ and $Receiver_i$ can agree on the hold period, which is equal to *holdTO*. If no other flow is connected to $Sender_i$ and $Receiver_i$, they can go into hold mode and wake upon EAT_i. Since the master manages the piconet, it does not turn off to sleep at any time.

The parameter α_i is used to control how much $r(p_i^k)$ decreases. When $\alpha_i = 1.0$, $r(p_i^k)$ does not change. With a smaller α_i, $r(p_i^k)$ drops much faster, and then the device could reduce the power consumption. Similarly, σ_i is used to control how much $r(p_i^k)$ increases. If flow i is bursty, the delay can be reduced by selecting a larger σ_i. The function *GetActualIdlePeriod* in Fig. 3 needs to be further explained. Suppose a slave is the end node of flow set $\mathcal{M}$, its actual hold period $\langle start, sleep\rangle$ is calculated according to:

$$\langle start, sleep\rangle = \cap_{i\in\mathcal{M}}\langle start_i, sleep_i\rangle \quad (1)$$

If $sleep \leq \varepsilon$, where ε is the threshold for holding, the slave will not hold. Using this function can eliminate the situation where the master sends a packet to a holding slave. In $Receiver_i$, if the receiver does not receive a packet addressed to it until the channel is idle, which means the sender did not have data to send, p will be set to NULL. The variable *WakeupTime* is used when p is set to NULL. Since the receiver may wait some time until the channel is idle (a packet of another flow may be on the fly during this period), we use *WakeupTime* and *clock* to adjust the *sleep* interval length of the receiver so that it can wake upon time and get further forwarded packets of flow i from the master.

3.2. Analysis of the APCB service model

In this section, we demonstrate the QoS properties of the APCB service model. Let $\mathrm{EFT}(p_i^k)$ denote the *Expected Finish Time* of the kth packet of flow i, and it is defined as:

$$\mathrm{EFT}(p_i^k) = \mathrm{EAT}(p_i^k) + \frac{L_i^k}{r(p_i^k)} \quad (2)$$

Two functions are used to convert the real time to virtual time: $v_s(t)$ and $v_f(t)$, where $v_s(t)$ is the EAT of the packet in service at time t, and $v_f(t)$ is the EFT of the packet in service at time t. $W_f(t_1, t_2)$ is the aggregated length of the packets served in the interval $[t_1, t_2]$.

$\underline{Master}$**:** Select flow i with the smallest $EAT(p_i^k)$.
if $clock < EAT(p_i^k)$ **then**
idle until $EAT(p_i^k)$;
$EAT(p_i^k) = clock$
poll $sender_i$ and receive packet p;
if $p = NULL$ **then**
*/*sender has no packet to send, do multiplicative decrease*/*
$r(p_i^k) = \alpha_i * r_i$
$EAT(p_i^k) = EAT(p_i^k) + L_i^{max}/r(p_i^k)$
else
*/*sender has a packet to send, do additive increase*/*
$r(p_i^k) = min(r(p_i^k) + L_i^{max} * \sigma_i,\ r_i)$
$EAT(p_i^{k+1}) = EAT(p_i^k) + L_i^k/r(p_i^k)$
*/*forward the packet to it receiver directly*/*
FORWARD(p)

$\underline{Sender_i}$**:** When wake up
wait until being polled
if DeQueue()=NULL **then**
reply a NULL packet
$r(p_i^k) = \alpha_i * r_i$
$\langle start_i, sleep_i\rangle = \langle clock, L_i^{max}/r(p_i^k) - \delta_i\rangle$
else
send the packet
$r(p_i^k) = min(r(p_i^k) + L_i^{max} * \sigma_i,\ r_i)$
$\langle start_i, sleep_i\rangle = \langle clock, L_i^k/r(p_i^k) - \delta_i\rangle$
$\langle start, sleep\rangle$=GetActualIdlePeriod()
Hold($\langle start, sleep\rangle$)

$\underline{Reciever_i}$**:** When wake up
$WakeupTime = clock$
wait for the forwarded packet p
if p=NULL **then**
*/*there is no packet addressed to it*/*
$r(p_i^k) = \alpha_i * r_i$
*/*deduct time spent in waiting from the sleep interval*/*
$\delta_i = \delta_i + (clock - WakeupTime)$
$\langle start_i, sleep_i\rangle = \langle clock, L_i^{max}/r(p_i^k) - \delta_i\rangle$
else
$r(p_i^k) = min(r(p_i^k) + L_i^{max} * \sigma_i,\ r_i)$
$\langle start_i, sleep_i\rangle = \langle clock, L_i^k/r(p_i^k) - \delta_i\rangle$
$\langle start, sleep\rangle$=GetActualIdlePeriod()
Hold($\langle start, sleep\rangle$)

Fig. 3. The algorithm of APCB.

Lemma 1. *If flow f is backlogged through the interval $[t_1, t_2]$, then in the APCB service model*

$$W_f(v_1, v_2) \geq r_f(v_2 - v_1 - \Delta t_f) - L_f^{\max}(\alpha_i^{-1} - \phi_f) \quad (3)$$

where

$$v_1 = v_s(t_1),\ \ v_2 = v_f(t_2),\ \ \phi_f = \left\lfloor \frac{(1-\alpha_i)r_f}{\sigma_i L_f^{\max}} \right\rfloor,$$

$$and\ \ \Delta t_f = \sum_{i=1}^{\phi_f} \frac{L_f^{\max}}{\alpha_i r_f + \sigma_i L_f^{\max} i}$$

Proof. From the APCB algorithm, the packets served in the interval $[v_1, v_2]$ can be partitioned into two sets:

- The set, donated by A, consists of packets that have the expected rate lower than r_f. Formally,

  $$A = \{k | r(p_f^k) < r_f\} \tag{4}$$

  Let $\hat{t}_A$ denote the time needed to serve packet set A and $\hat{t}_A = \sum_{i \in A} L_f^i / r(p_f^i)$.
- The set, denoted by B, consists of packets that have the expected rate equal to r_f. Formally,

  $$B = \{k | r(p_f^k) = r_f\} \tag{5}$$

  Let $\hat{t}_B$ denote the time needed to serve packet set B.

Suppose the server serves the first packet of flow f, which is p_f^k, at time t_0 ($v_1 \le t_0 \le v_1 + L_f k / r(p_f^k)$). Since flow f is backlogged in the interval $[v_1, v_2]$, we can get:

$$W_f(v_1, v_2) = W_f(t_0, \hat{t}_A) + r_f \hat{t}_B \tag{6}$$

Since $\hat{t}_A + \hat{t}_B \le v_2 - t_0$, and $r(p_f^i) < r_f (i \in A)$, we can easily find that $W_f(v_1, v_2)$ increases as $\hat{t}_A$ drops. Thus, $W_f(v_1, v_2)$ has the smallest value when $r(p_f^k) = \alpha_i r_f$ and $L_f^i = L_f^{\max} (i \in A)$. According to the APCB algorithm, the number of packets in set A, which is denoted by ϕ_f, is equal to

$$\left\lfloor \frac{(1 - \alpha_i) r_f}{\sigma_i L_f^{\max}} \right\rfloor.$$

As a result, in the worst case

$$W_f(t_0, \hat{t}_A) = \phi_f L_f^{\max} \tag{7}$$

$$\hat{t}_A = \sum_{i=1}^{\phi_f} \frac{L_f^{\max}}{\alpha_i r_f + \sigma_i L_f^{\max} i} \tag{8}$$

From Eqs. (6)–(8), and $t_0 \le v_1 + L_f^{\max} / (\alpha_f r_f)$, we can get:

$$\begin{aligned} W(t_1, t_2) &\ge r_f \left(v_2 - v_1 - \frac{L_f^{\max}}{\alpha_f r_f} - \hat{t}_A \right) + \phi_f L_f^{\max} \\ &= r_f (v_2 - v_1 - \hat{t}_A) - L_f^{\max} (\alpha_f^{-1} - \phi_f) \end{aligned} \tag{9}$$

By substituting $\hat{t}_A$ with Δt_f, the Lemma follows. □

Lemma 2. *If flow f is backlogged through the interval $[t_1, t_2]$, then in the APCB service model*

$$W_f(v_1, v_2) \le r_f (v_2 - v_1) + L_f^{\max} \tag{10}$$

where $v_1 = v_s(t_1)$, $v_2 = v_f(t_2)$.

Proof. From the definition of EAT, the set of flow f packets served in the interval $[v_1, v_2]$ have EAT value at least v_1 and at most v_2. Hence, the set can be partitioned into two sets:

- The set, denoted by D, consists of packets that have EAT at least v_1 and AFT at most v_2. Formally,

  $$D = \{k | v_1 \le \mathrm{EAT}(p_f^k) \le v_2 \wedge \mathrm{AFT}(p_f^k) \le v_2\} \tag{11}$$

  Since $r(p_f^k) \le r_f$, we can conclude

  $$\sum_{k \in D} L_f^k \le r_f (v_2 - v_1) \tag{12}$$

- This set, denoted by E, consists of packets that have EAT at most v_2 and AFT greater than v_2. Formally,

  $$E = \{k | v_1 \le \mathrm{EAT}(p_f^k) \le v_2 \wedge \mathrm{AFT}(p_f^k) \ge v_2\} \tag{13}$$

Clearly, at most one packet can belong to set E and

$$\sum_{k \in E} L_f^k \le L_f^{\max} \tag{14}$$

Hence, combined with Eqs. (12) and (14), Lemma 2 follows.

3.2.1. Fairness guarantees

Theorem 1. *(Short Term Fairness) For any interval $[t_1, t_2]$ in which flows f and m are backlogged during the entire interval. The difference in the service received by two flows in the APCB service model is given as*

$$\begin{aligned} &\left| \frac{W_f(v_1, v_2)}{r_f} - \frac{W_m(v_1, v_2)}{r_m} \right| \\ &\quad \le \max_{i,j \in \{f,m\}} \left\{ \frac{L_i^{\max}}{r_i} + \frac{L_j^{\max} (\alpha_j^{-1} - \phi_j)}{r_j} + \Delta t_j \right\} \end{aligned} \tag{15}$$

where $v_1 = v_s(t_1)$, $v_2 = v_f(t_2)$, and Δt_i, ϕ_i are defined in Lemma 1.

Proof. Since unfairness between two flows in any interval is the maximum possible service difference between them. From Lemmas 1 and 2, we can get the throughput lower bound and upper bound of a flow. Let the flow receives the maximum service and the other minimum service. This Theorem follows directly. □

Theorem 2. *(Long Term Fairness) For a continually backlogged flow f, it achieves the following long-term throughput in the APCB service model*:

$$\lim_{v \to \infty} \frac{W_f(0, v)}{v} = r_f \tag{16}$$

Proof. From Lemmas 1 and 2, we have the following:

$$r_f (v - \Delta t_f) - L_f^{\max} (\alpha_f^{-1} - \phi_f) \le W_f(0, v) \le r_f v + L_f^{\max} \tag{17}$$

When divided by v, we have:

$$r_f - \frac{\Delta t_f}{v} - \frac{L_f^{\max}(\alpha_f^{-1} - \phi_f)}{v} \le \frac{W_f(0,v)}{v} \le r_f + \frac{L_f^{\max}}{v} \tag{18}$$

The result follows directly by letting $v \to \infty$. □

3.2.2. Throughput guarantees

Theorem 3. *If Q is the set of flows served in the APCB service model, and if a flow f is continually backlogged over a real-time interval $[t_1, t_2]$, flow f's aggregated service $W_f(t_1, t_2)$ is bounded by*

$$W_f(t_1,t_2) \ge r_f(t_2 - t_1 - \Delta t_f) - \frac{r_f}{C}\sum_{i\in Q} L_i^{\max} - L_f^{\max}(\alpha_i^{-1} - \phi_f) \tag{19}$$

where $C = \sum_{i\in Q} r_i$, C is less than the system capacity, and Δt_f and ϕ_f are defined in Lemma 1.

Proof. Let $v_1 = v_s(t_1)$ and let $\hat{W}(v_1, v_2)$ denote the aggregate length of packets served by the server in the virtual time interval $[v_1, v_2]$. Then from Lemma 2 we get:

$$\hat{W}(v_1,v_2) \le \sum_{i\in Q} r_i(v_2 - v_1) + \sum_{i\in Q} L_i^{\max} \tag{20}$$

Since the server is non-work-conserving, the system bandwidth $C = \sum_{i\in Q} r_i$

$$\hat{W}(v_1,v_2) \le C(v_2 - v_1) + \sum_{i\in Q} L_i^{\max} \tag{21}$$

Define v_2 as

$$v_2 = v_1 + t_2 - t_1 - \frac{\sum_{i\in Q} L_i^{\max}}{C} \tag{22}$$

Then from Eq. (21), we conclude:

$$\hat{W}(v_1,v_2) \le C(t_2 - t_1) \tag{23}$$

Let $\bar{t}_2$ be such that $v_f(\bar{t}_2) = t_2$. Therefore, the real time taken by the server to serve packets with aggregated length $\bar{W}(v_1, v_2)$ is:

$$\bar{t}_2 - t_1 = \frac{\hat{W}(v_1,v_2)}{C} \le \frac{C(t_2 - t_1)}{C} = t_2 - t_1 \tag{24}$$

So, $\bar{t}_2 \le t_2$ Therefore

$$W_f(t_1,t_2) \ge \hat{W}_f(t_1,\bar{t}_2) = W_f(v_1,v_2) \tag{25}$$

According to Lemma 1

$$\begin{aligned} W_f(v_1,v_2) &\ge r_f(v_2 - v_1 - \Delta t_f) - L_f^{\max}(\alpha_i^{-1} - \phi_f) \\ &= r_f\left(t_2 - t_1 - \frac{\sum_{i\in Q} L_i^{\max}}{C} - \Delta t_f\right) - L_f^{\max}(\alpha_i^{-1} - \phi_f) \\ &= r_f(t_2 - t_1 - \Delta t_f) - \frac{r_f}{C}\sum_{i\in Q} L_i^{\max} - L_f^{\max}(\alpha_i^{-1} - \phi_f) \end{aligned} \tag{26}$$

Combine Eqs. (25) and (26), the theorem follows. □

3.2.3. Delay guarantees

Theorem 4. If Q is the set of flows served in the APCB service model, and if packet p_f^j is the Nth packet in flow i's outgoing buffer and p_f^{HOL} is the head-of-line packet in the buffer, then the departure time of packet p_f^j, which is denoted by $\mathrm{DP}(p_f^j)$, is given by:

$$\mathrm{DP}(p_f^j) \le \mathrm{EAT}(p_f^{\mathrm{HOL}}) + \sum_{i=1}^{\min(\phi_f, N-1)} \frac{L_f^{\max}}{\alpha_f r_f + \sigma_f L_f^{\max} i} + (\max(\phi_f, N-1) - \phi_f)\frac{L_f^{\max}}{r_f} + \frac{\sum_{i\in Q} L_i^{\max}}{C} \tag{27}$$

where $C = \sum_{i\in Q} r_i$, C is less than the system capacity, and ϕ_f is defined in Lemma 1

Proof. Let packet p_f^k and p_f^j be the first and the last packet served in the same busy period of server. Also, let $v_1 = \mathrm{EAT}(p_f^k)$ and $v_2 = \mathrm{EFT}(p_f^j)$. Thus, the set of flow f packets served in the interval $[v_1, v_2]$ have EAT at least v_1 and at most v_2. There are two sets:

$$D_f = \{m | v_1 \le \mathrm{EAT}(p_f^m) \le v_2 \wedge \mathrm{EFT}(p_f^m) \le v_2\} \tag{28}$$

Since the estimated rate of flow f may be variable in a time interval, let $r_f(v)$ be the function for flow f at time v as the rate assigned to the packet that is being transmitted at time v. Formally,

$$r_f(v) = \begin{cases} r(p_f^i), & \text{if } \exists i\ (\mathrm{EAT}(p_f^i) \le v < \mathrm{EFT}(p_f^i)) \\ 0, & \text{else} \end{cases} \tag{29}$$

So the aggregated length of flow f packets served by the server in the interval $[v_1, v_2]$, denoted by $\mathrm{AP}_f(v_1, v_2)$ is given as:

$$\mathrm{AP}_f(v_1,v_2) \le \int_{v_1}^{v_2} r_f(v)\mathrm{d}v \tag{30}$$

Hence aggregated length of packets in set Q is:

$$\sum_{n\in Q} \mathrm{AP}_n(v_1,v_2) \le \sum_{n\in Q} \int_{v_1}^{v_2} r_n(v)\mathrm{d}v \tag{31}$$

$$\sum_{n\in Q} \mathrm{AP}_n(v_1, v_2) \le C(v_2 - v_1) \tag{32}$$

Since it is the busy period in the interval $[v_1, v_2]$, and by the definition of EAT and EFT, we get

$$v_2 - v_1 = \sum_{i=k}^{j} \frac{L_f^i}{r(p_f^i)} \tag{33}$$

and

$$\sum_{n\in Q} \mathrm{AP}_n(v_1, v_2) \le C \sum_{i=k}^{j} \frac{L_f^i}{r(p_f^i)} \tag{34}$$

• This set, denoted by E, consists of packets that have EAT at most v_2 and EFT greater than v_2. Formally,

$$E = \{m | v_1 \le \mathrm{EAT}(p_f^m) \le v_2 \wedge \mathrm{EFT}(p_f^m) > v_2\} \tag{35}$$

Clearly, at most one packet of flow f can belong to this set. Hence, the maximum aggregate length of packets in flow set Q is:

$$\sum_{i\in Q \wedge i\neq f} L_i^{\max} + L_f^j \tag{36}$$

Hence, the aggregate length of packets served by the server in the interval $[v_1, v_2]$, denoted by $\hat{W}(v_1, v_2)$, is:

$$\hat{W}(v_1, v_2) \le C \sum_{i=k}^{j} \frac{L_f^i}{r(p_f^i)} + \sum_{i\in Q\wedge i\neq f} L_i^{\max} + L_f^j \tag{37}$$

Let $\bar{t}$ be the time taken by server to serve packets with aggregate length $\hat{W}(v_1, v_2)$ in a busy period. We get:

$$\bar{t} = \frac{\hat{W}(v_1, v_2)}{C} \le \sum_{i=k}^{j} \frac{L_f^i}{r(p_f^i)} + \frac{\sum_{i\in Q\wedge i\neq f} L_i^{\max}}{C} + \frac{L_f^j}{C} \tag{38}$$

$$\bar{t} \le \sum_{i=k}^{j} \frac{L_f^i}{r(p_f^i)} + \frac{\sum_{i\in Q} L_i^{\max}}{C} \tag{39}$$

Since packet p_f^j departs no later than v_2 and all the packets served in the virtual time interval $[v_1, v_2]$ are served in the same busy period of the server, we get:

$$\mathrm{EAT}(p_f^k) + \bar{t} \ge \mathrm{DP}(p_f^j) \tag{40}$$

From Eq. (40), we get:

$$\mathrm{DP}(p_f^j) \le \mathrm{EAT}(p_f^k) + \sum_{i=k}^{j} \frac{L_f^i}{r(p_f^i)} + \frac{\sum_{i\in Q} L_i^{\max}}{C} \tag{41}$$

Let p_f^j be the Nth packet in the outgoing buffer of flow f, and p_f^k is the head-of-line packet of the buffer. For the set of packet p_f^m, where $m \in \{a | k \le a \le j\}$, we can use the arguments in Lemma 1 to get that the worst case for $\mathrm{DP}(p_f^j)$ happens when $L_f^m = L_f^{\max}$ and $r(p_f^k) = \alpha_f r_f$. Hence, using the same notations in Lemma 1, we can get:

$$\mathrm{DP}(p_f^j)_{\max} = \mathrm{EAT}(p_f^k) + \sum_{i=1}^{\min(\phi_f, N-1)} \frac{L_f^{\max}}{\alpha_f r_f + \sigma_f L_f^{\max} i} + (\max(\phi_f, N-1) - \phi_f)\frac{L_f^{\max}}{r_f} + \frac{\sum_{i\in Q} L_i^{\max}}{C} \tag{42}$$

By substituting p_f^k with p_f^{HOL}, the Theorem follows. □

4. Performance evaluations

In this section, we evaluate the scheme by simulations. We simulate both core components of bluetooth standard: the Baseband layer and the L2CAP layers. With the assumption of error-free channel, we do not consider error bit rate. The baseband packet limit is 5-slot packet (339 bytes), which is specified in the specifications [1]. For simplicity, the sender of each flow generates packets with fixed length of 339 bytes, and we do not consider flow control so that the buffer size of each sender is large enough. We compare the APCB scheme with the RR scheme and the *queue status based polling interval* (QSPI) scheme [2], which is a power-conserving scheduling algorithm for bluetooth piconets. With APCB, the service unit is flow and the master forwards packets directly without buffering. With RR and QSPI, the service unit is node and the master maintains a buffer for each slave and itself. The simulation topology is shown in Fig. 1, and the parameters of each flow are listed in Table 2.

In QSPI, the *sniff* mode[1] is utilized to aid the master and the slave to reach an agreement on when the slave will be served by the master. Whenever there is no data in a particular queue, the master switches the slave to sniff mode with a fixed value of the polling interval. Each slave has two sniff states: state I and state II. The polling interval of a slave in state II doubles that in state I. When a slave is switched from active to sniff, it is in state I. It goes to state II when it has no data to send when being polled, and is moved back to active from state I if its queue contains more than one packet. When a slave staying in state II is polled, it goes back to state I if there are two packets in the queue. In the case that there are more than two packets in the queue, the slave is resumed to be active. We set the fixed polling interval to be 62.5 ms.

The holding threshold ϵ for APCB is set to one time slot (625 μs). For simplicity, we only consider the homogeneous traffic sources in our simulation, so that each flow sender uses the same traffic parameters and sets up the same value of α. Intuitively, the selection of α is traffic model

[1] Essentially, the sniff mode used in QSPI is the same as the hold mode used in APCB.

Table 2
Connections parameters

Flow	Flow rate (bps)
$S_1 \rightarrow S_2$	27,120
$S_2 \rightarrow S_5$	40,680
$S_3 \rightarrow Master$	81,360
$S_4 \rightarrow S_5$	54,240
$S_5 \rightarrow S_2$	62,376
$S_6 \rightarrow S_7$	27,120
$S_7 \rightarrow S_6$	108,480

dependent. In order to show this relationship, we evaluate our policy under three traffic models, the Constant Bit Rate (CBR) model, the Poisson Process (PP) model and the ON/OFF model. For the CBR model and PP model, the burstiness degree σ_i is equal to 1.0. For the ON/OFF model, $\sigma_i = 2.0$.

The following metrics are used to evaluate the algorithm: the throughput, the average packet delay and the total Weighted Power Consumption Slots (WPCSs) of all the slaves in the piconet. The throughput is equal to the total throughput of all flows. We measure these data along with different length of flow active time, during which the flow is actively transmitting packets. Here, we define the WPCS as the weighted number of the slots with power consumption by a device in the time period of T. For a bluetooth device, it has four operational modes: *TxMode*, *RxMode*, *ActiveMode*, *SleepMode*. It has been shown that transmitting packets (for the same duration) consumes almost the same amount of power as receiving packets, and power consumption in TxMode and RxMode is higher than that in ActiveMode and SleepMode. According to the data from Ref. [14], we assume the weight of the four modes are 1.0, 0.6, 0.2, and 0.0, respectively.

4.1. The CBR model

Under this model, we assume that the sender of each flow generates packets at a constant interval, which is determined by the flow rate. For example, flow $S_1 \rightarrow S_2$ has 10 packets per second (Note that the packet size is fixed to 339 bytes) so that the packet interval is 0.1 s. With APCB, the master does not mis-predict the EAT of flows in this model. Thus, α has no effect in this scenario and we set it to 1.0. The performance results are shown in Fig. 4. As can be seen, our scheme has the largest throughput. Compared to RR, APCB

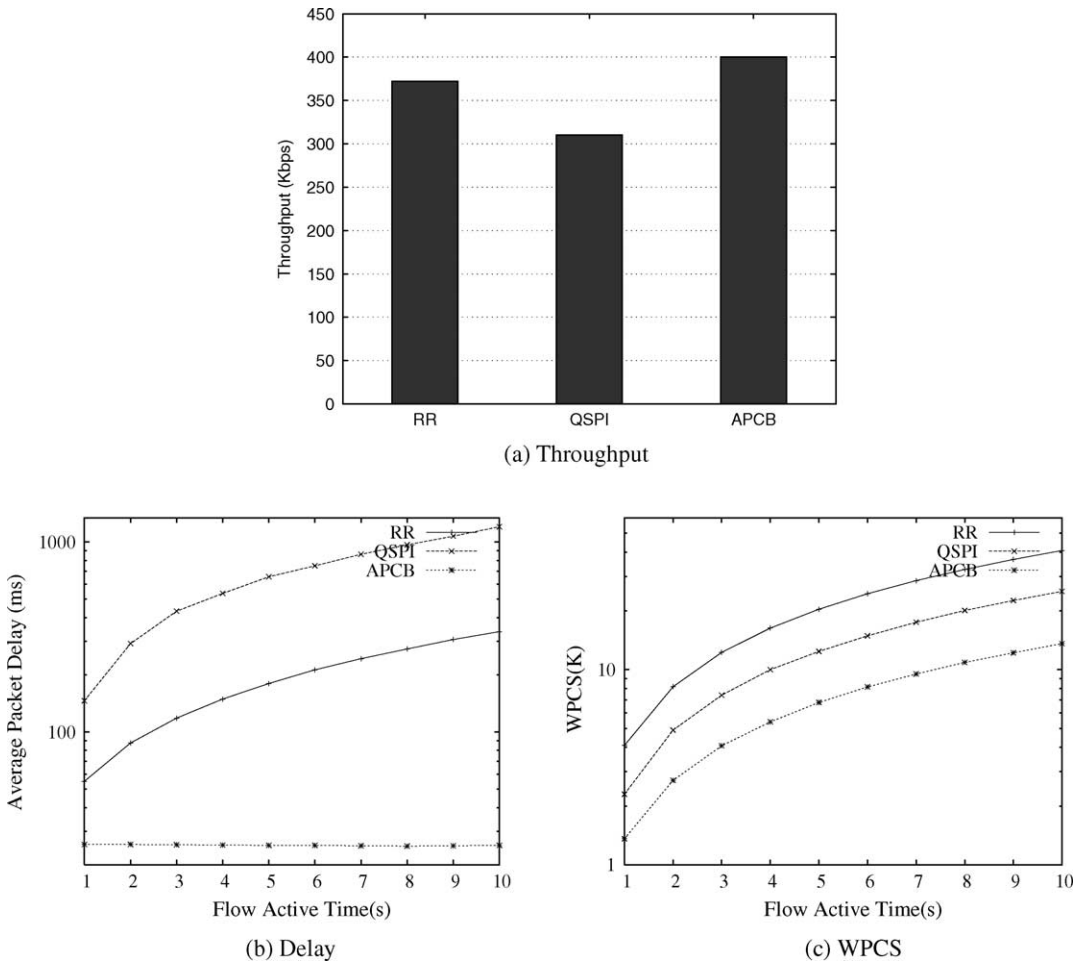


Fig. 4. Performance comparisons between RR and APCB under CBR traffic.

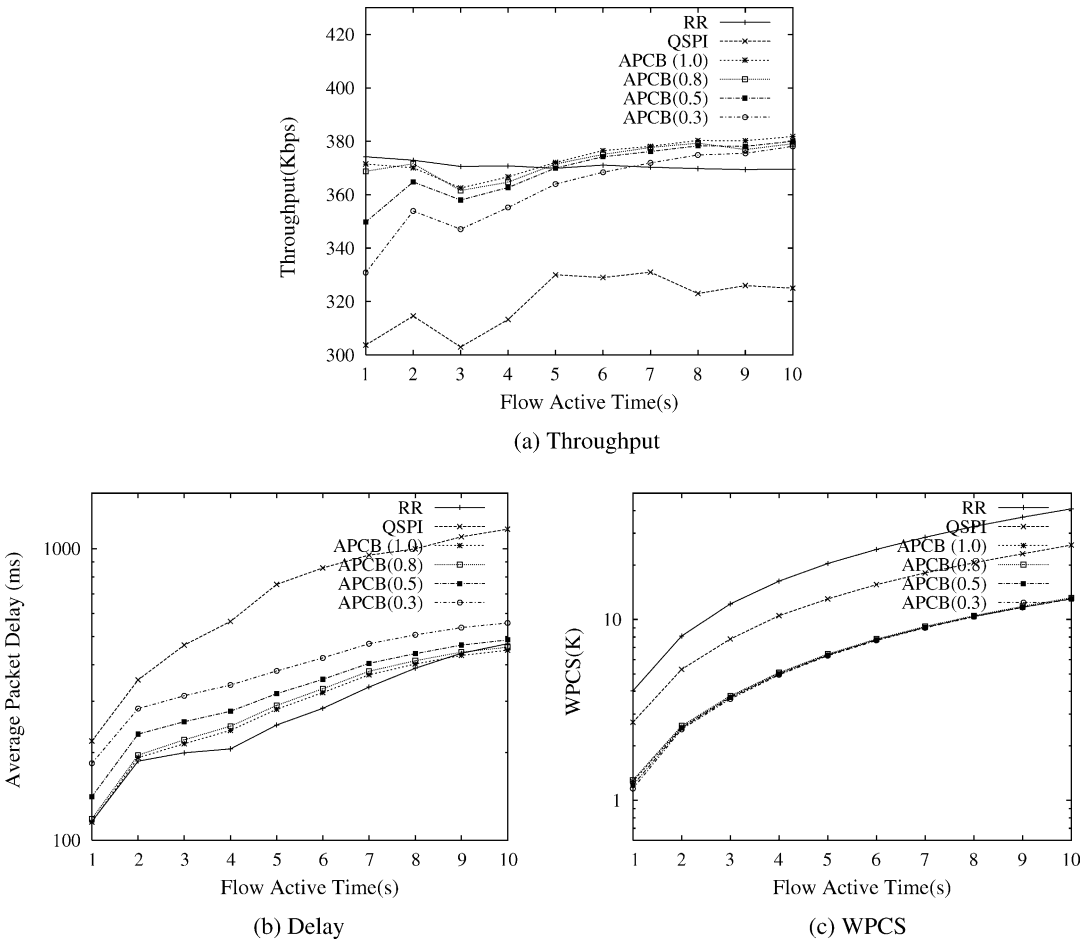
not only reduces the power consumption by 65%, but also significantly reduces the packet delay. This can be explained by the fact that APCB does not waste any time slot, whereas RR treats each slave equally and wastes many slots due to excessive polling. Since we do not apply flow control in the simulation, due to queuing delay, the packet delay of the RR scheme increases continuously as the flow active time increases.

Compared to RR, as shown in Fig. 4, QSPI can significantly reduce the power consumption. However, its throughput and packet delay is the worst among the three schemes. With QSPI, the polling interval of QSPI is adapted according to the queue length of a slave, a flow connected to the slave may have a long packet delay (more than twice the fixed polling interval) if the slave has been put into *sniff*. On the other hand, when several slaves have more than two packets in their queues, they are in active simultaneously, and the master needs to serve these slaves with RR. If the number of such slaves is large, the power efficiency of QSPI could be significantly degraded since the number of time slots used for unnecessary polling may be high. In contrast, APCB adjusts the polling interval of each slave based on the corresponding flow rate, so that the master can have a precise estimate of when the sender will have data, especially in the case of CBR. As a result, the end nodes of the flow can stay in *hold* as long as possible without violating the packet delay too much.

4.2. The PP model

We use the Poisson Process (PP) model to emulate some variable bit rate (VBR) flows. If the traffic model of a flow is a simple PP with mean λ, then the time interval between two adjacent packets has an exponential distribution with parameter λ. In this section, the $1/\lambda$ of each flow is equal to the packet interval used in the CBR model. PP is a pure memoryless process which means there exists no relationship between any adjacent events. This means that the master will definitely encounter mis-predict the EAT of a PP flow.

The simulation results are shown in Fig. 5, and the legend APCB(1.0) means the $\alpha = 1.0$. Under PP model, we can see that as the flow active time lasting longer, the throughput of APCB scheme with different α settings, except $\alpha = 0.3$, are similar and are greater than that under RR. The delay under APCB(1.0) is just little higher than that under RR. From Fig. 5, we can see that APCB outperforms QSPI in terms of throughput, packet delay and power efficiency. The reason



(a) Throughput

(b) Delay

(c) WPCS

Fig. 5. Performance comparisons between RR and APCB under PP traffic.

has been explained before and is still valid here. For APCB, it is clear that $\alpha = 0.3$ brings a longer delay. This is caused by the longer queuing delay due to the longer catching up time after one mis-prediction. It is interesting to see that the WPCSs are similar when α is different. The reason is that the traffic is quite intensive and the occurrences of mis-predictions are small since the queue of a flow can have some packets backlogged for many time slots after an mis-prediction (Please note that the standard deviation of the exponential distribution is $1/\lambda$). So setting $\alpha = 1.0$ is the best choice under the PP model.

4.3. The ON/OFF model 1

The ON/OFF model is an interrupted process (i.e. the inter-arrival time changes based on the state of the system). The idle period and the active period are assumed to be exponentially distributed. In the active period, the packet interval is assumed to be constant. We choose the mean period of the idle period and the active period to be 200 and 100 ms, respectively. The rate of flow i is treated as its average rate so that the inter-arrival time in flow i's active period is $3r_i$. The simulation results are shown in Fig. 6. From the results, we can find out that the throughput with APCB, except $\alpha = 0.3$, is generally similar to that of the RR policy. This is because the average flow rates are quite high and the buffer of a flow is rarely empty in its idle periods. Following the same reason, mis-prediction is rare so that there is no difference in terms of power saving with different value of α. There is a slight difference between the delay under APCB(1.0) and that under RR. The reason for longer delay under APCB with other α settings is the same as that in the PP model. From this example, we find that if the traffic is quite intensive under the ON/OFF model, $\alpha = 1.0$ is also the best choice.

Similar to the cases under the previous two traffic models, QSPI has better power efficiency than RR, whereas it cannot maintain a comparable throughput or packet delay. Compared to APCB, QSPI is worse in all aspects of the performance metrics. This is still due to the relatively static adaptation policy of QSPI.

4.4. The ON/OFF model 2

In this scenario, we still use the ON/OFF model. The differences are: first, the idle and active periods are assumed to follow a Gaussian Marginal distribution, and the random numbers are generated similarly to Ref. [8].

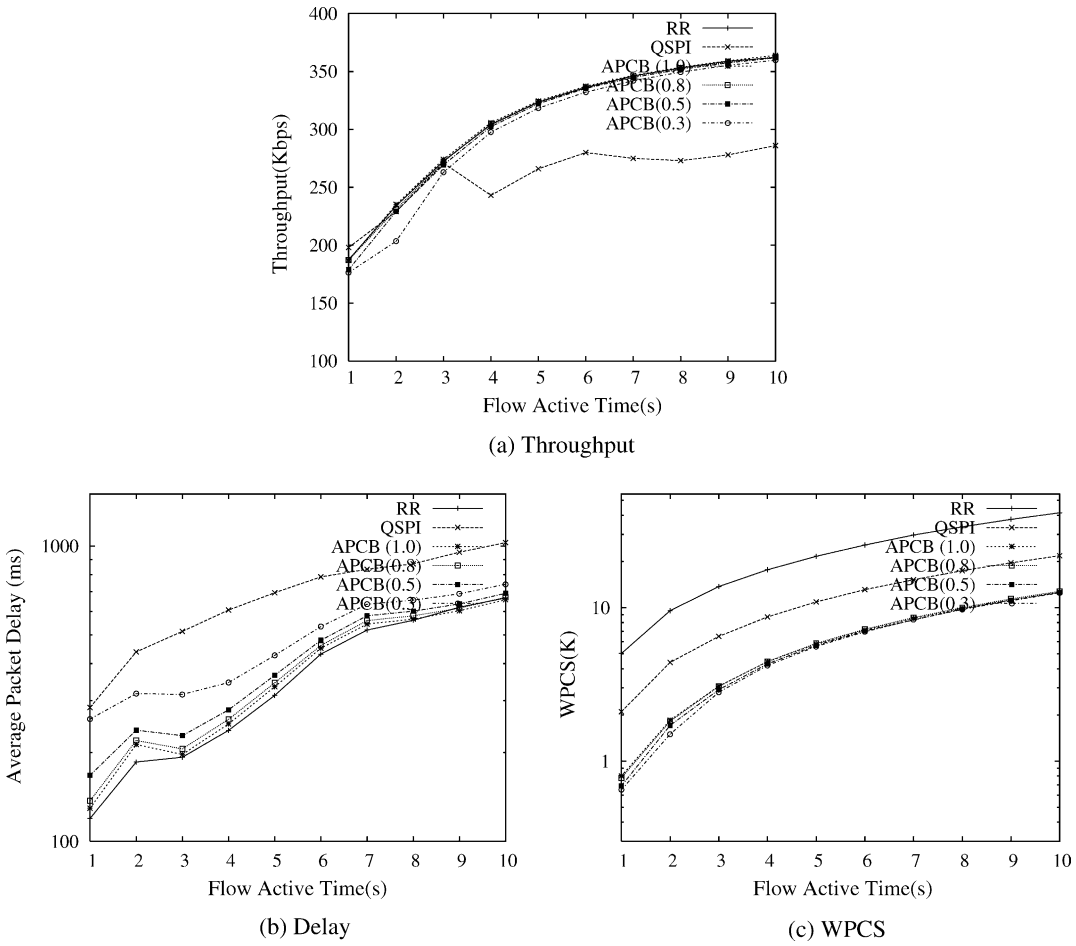


Fig. 6. Performance comparisons between RR and APCB under ON/OFF traffic.

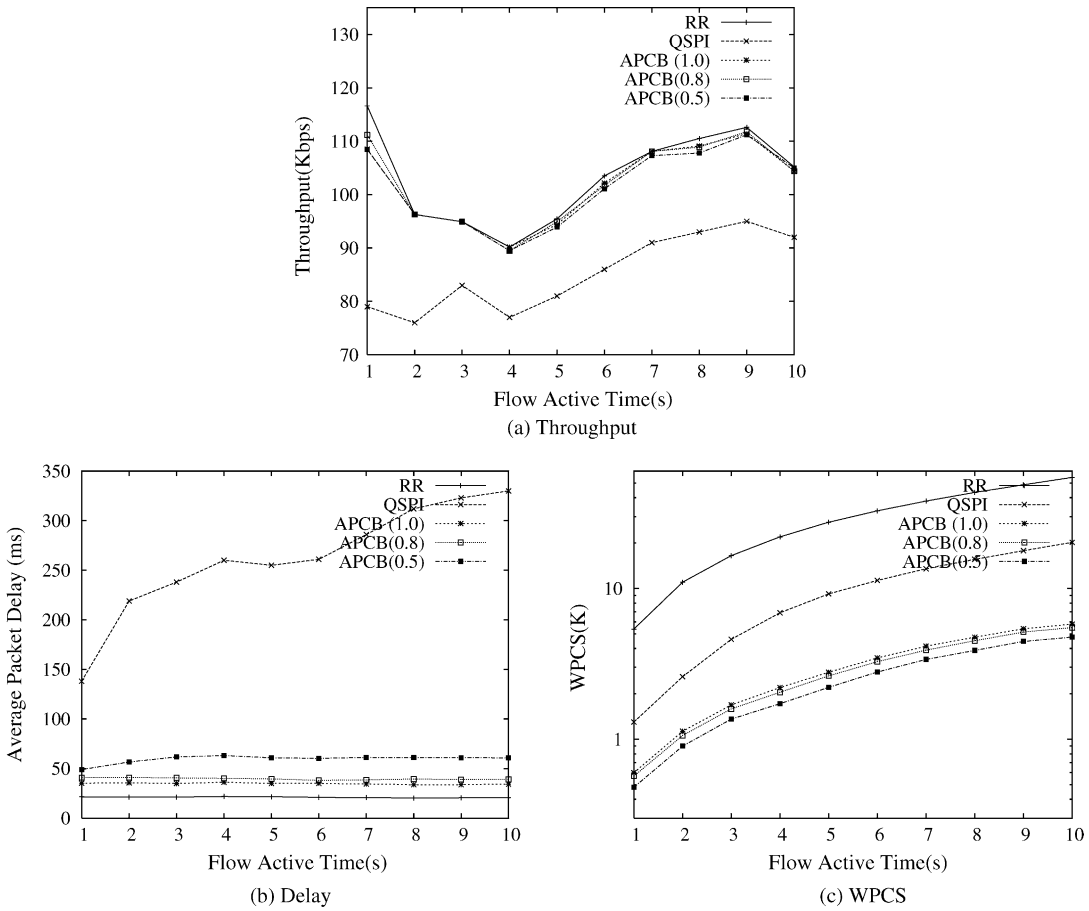


Fig. 7. Performance comparisons between RR and APCB under train traffic.

Second, in the active mode, the inter-arrival time is randomly distributed between $0.5r_i$ and r_i. It is obvious that the system load is much less than that in the previous example. The simulation results are shown in Fig. 7. As can be seen, the throughput of APCB with varies values of α, except $\alpha = 0.3$, is almost the same as that of RR. Although the average packet delay of the APCB approach, except $\alpha = 0.3$, is moderately higher than that of the RR approach, the power consumption of the APCB approach has been reduced as much as 85% compared to the RR approach. Again, QSPI achieves better power efficiency than RR at the expense of much worse throughput and packet delay, when compared to RR and APCB. Thus, it is still not a good solution compared to APCB. Under the ON/OFF model, we should note that the system load is quite light, and the scheduler under the APCB approach works in the non-work-conserving manner. Thus, the capacity of the system is smaller than the RR approach. Also, since mis-prediction may happen, the corresponding rate adjustment could bring some extra packet delay due to the prolonged polling intervals (Note that the expected data rate of flow i is reduced by α_i upon a mis-prediction). From Fig. 7, we can see that there exists a tradeoff between power and delay. If power saving is the optimization goal, the $\alpha = 0.3$ approach is better than the $\alpha = 0.5$ approach. However, this power saving is at the cost of increasing the delay. To achieve a balance between power and delay, $\alpha = 0.5$ is better than $\alpha = 0.3$. For example, in terms of power consumption, the $\alpha = 0.3$ approach reduces the power consumption by about 13% compared to the $\alpha = 0.5$ approach; however, the $\alpha = 0.3$ approach doubles the delay. Certainly, if delay is an important issue, $\alpha = 0.8$ is a better option compared to $\alpha = 0.5$.

5. Conclusions and future works

In this paper, we proposed an adaptive power-conserving scheduling algorithm for bluetooth networks. Through theoretical analyses, we showed that the APCB scheme can guarantee the performance of a flow. From the simulation results, we found that the APCB scheme can save a significant amount of power under varies traffic models. Under the CBR traffic model, the APCB scheme always outperforms the RR policy. Under the PP traffic model, if the system load is heavy, our scheme may have a little bit longer delay, but the throughput is more than that under the RR scheme. Under the ON/OFF traffic model, if

the system load is heavy, the throughput and the delay are almost the same as those of the RR scheme. Under the ON/OFF traffic model, if the system load is light, our scheme may have a moderately longer packet delay and slightly less throughput due to the non-work-conserving nature and the prediction nature of APCB. However, our scheme can reduce the power consumption as much as 85%. We found that α has significant effects on power saving and delay. Properly choosing the values of α is helpful for achieving a good balance between system performance and power consumption. Our future work will focus on automatically adapting α for each flow so that the flow can automatically get a good balance between power and delay based on a certain criteria. For example, the master can monitor the performance of a flow. When the packet delay increases above a threshold, the master and the end nodes of the flow will increase α to serve the flow faster. Otherwise, they can decrease α to save more power.

References

[1] Bluetooth Special Interest Group, Specification of the Bluetooth System 1.0b, Volume 1: Core, 1999, http://www.bluetooth.com.

[2] A. Capone, M. Garla, R. Kapoor, Efficient polling schemes for bluetooth picocells, IEEE ICC'01 (2001).

[3] I. Chakraborty, A. Kashyap, A. Rastogi, H. Saran, R. Shorey, A. Kumar, Policies for increasing throughput and decreasing power consumption in bluetooth MAC, IEEE International Conference on Personal Wireless Communications (ICPWC) (2000).

[4] K. Govil, E. Chan, H. Wesserman, Comparing algorithms for dynamic speed-setting of a low power CPU, MobiCOM'95 November (1995).

[5] P. Goyal, H. Vin, H. Cheng, Start-time fair queuing: a scheduling algorithm for integrated services packet switching networks, ACM SIGCOMM'96 August (1996).

[6] J. Haartsen, The bluetooth radio system, IEEE Personal Communications 7 (1) (2000) 28–36.

[7] IEEE, Wireless LAN medium access control (MAC) and physical layer (PHY) Sp ec, IEEE 802.11 Standard (1999).

[8] S. Jamin, P.B. Danzig, S. Shenker, L. Zhang, A measurement-based admission control algorithm for integrated services packet networks, ACM SIGCOMM'95 September (1995).

[9] M. Kalia, D. Bansal, R. Shorey, MAC scheduling and SAR policies for bluetooth: a master driven TDD pico-cellular wireless system, Proceedings of Workshop on Mobile Multimedia (1999) 384–388.

[10] M. Kalia, S. Garg, R. Shorey, Efficient policies for increasing capacity in bluetooth: an indoor pico-cellular wireless system, Proceedings of IEEE Vehicular Technology Conference (VTC 2000) (2000) 907–911.

[11] R. Kravets, P. Krishnan, Power management techniques for mobile communication, MobiCOM'98 October (1998).

[12] J.-B. Lapeyrie, T. Turletti, FPQ: a fair and efficient polling algorithm with QoS support for bluetooth piconet, IEEE INFOCOM'03 (2003).

[13] T.-Y. Lin, Y.-C. Tseng, Y.-T. Lu, An efficient link polling by pattern matching for bluetooth piconets, 36th Hawaii International Conference on System Sciences, 2002 (2002).

[14] J. Linskey, Bluetooth and power consumption: issues and answers, 1st November, 2001, http://images.rfdesign.com/files/4/1101Linsky74.pdf.

[15] B. Miller, C. Bisdikian, Bluetooth Revealed, Prentice-Hall, Upper Saddle River, NJ, 2000.

[16] J. Monks, V. Bharghavan, W.-M. Hwu, A power controlled multiple access protocol for wireless packet networks, IEEE INFOCOM'01 March (2001).

[17] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, Distributed topology construction of bluetooth personal area networks, IEEE INFOCOM'01 April (2001).

[18] V. Sangvornvetphan, T. Erke, Traffic scheduling in bluetooth network, Ninth IEEE International Conference on Networks, 2001 (2001).

[19] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, Proceedings of the IEEE 83 (10) (1995).

[20] L. Zhang, Virtual clock: a new traffic control algorithm for packet switching networks, ACM SIGCOMM'90 September (1990).

Hao Zhu: Hao Zhu received his BS degree from Xian Jiaotong University, Xian, China and his MS degree in computer science and engineering from the Institute of Software, Chinese Academy of Sciences, Beijing, China. He is currently working towards the Ph.D degree at the Pennsylvania State University. His research interests include resource management, QoS and power efficient protocols in wireless networks.

Guohong Cao: Guohong Cao received his BS degree from Xian Jiaotong University, Xian, China. He received the MS degree and PhD degree in computer science from the Ohio State University in 1997 and 1999 respectively. Since Fall 1999, he has been an Assistant Professor of computer science and engineering at the Pennsylvania State University. His research interests are mobile computing, wireless networks, and distributed fault-tolerant computing. He was a recipient of the Presidential Fellowship at the Ohio State University in 1999, and a recipient of the NSF CAREER award in 2001.

George Kesidis: George Kesidis received his M.S. and Ph.D. in EECS from U.C. Berkeley in 1990 and 1992 respectively. He was a professor in the E&CE Dept of the University of Waterloo, Canada, from 1992 to 2000. Since April 2000, he has been an associate professor in both the EE and CS&E Departments of the Pennsylvania State University. His research interests include Internet (IPV4) security, high-speed packet scheduling, evolving (IPV6) Internet economics and TCP, energy-efficient communication, mobility management for sensor networks and intrusion detection.

Chita Das: Dr. Das joined the faculty at Penn State as an assistant professor of Computer Engineering in the Department of Electrical Engineering in August 1986. He received his doctoral degree in computer science from the Center for Advanced Computer Studies, University of Louisiana in 1986. His research interests include computer architecture, parallel and distributed computing, design and analysis of routing algorithms, performance evaluation, fault-tolerant computing, Internet QoS, multimedia servers, and mobile computing. Dr. Das is a Fellow of the IEEE.