An Adaptive Power-Conserving Service Discipline for Bluetooth

Hao Zhu, Guohong Cao, George Kesidis and Chita Das
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802

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 manner and it may waste a significant amount of power. 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 Bluetooth standard. Unlike other research work related to low-power, we also consider QoS of each flow. Theoretical analyses verify that our scheme can achieve throughput guarantees, delay guarantees, and fairness guarantees. Simulation results demonstrate that our scheme can save a significant amount of power compared to the Round Robin scheme and it shows that there exists a tradeoff between power and delay under varies traffic models.**

I. Introduction

Bluetooth is a promising technology which is aimed at supporting wireless connectivity among mobile devices. The technology enables the design of low-power, small-size, low-cost radios that can be embedded in existing portable devices. It is a frequency hopping system which can support multiple communication channels in a common area (each channel is defined by a unique frequency hopping sequence). In Bluetooth, a group of devices 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 (TDD) 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 [1]. In most of the current Bluetooth products, the master polls the slaves in a Round Robin (RR) manner. The polling based RR scheduling has a drawback when considering power consumption. If the slave being polled does not have any packet to send, two time slots will be wasted. As a result, if the slave's traffic density is low, there will be a large amount of power wasted due to excessive polling. Meanwhile, since a slave does not know when it will be polled, it has to keep listening and a large amount of power will be wasted.

There has been a lot of research on low-power control for wireless devices. On the hardware level, the communication device can adjust the power level used by the mobile transmitter during active communication [2]. On the software level, we can control the power consumption by communication devices [3]. The underlying principle is to estimate when the device will be used and suspend it for those idle intervals. Also, researchers proposed some schemes to minimize the power used by the device to transmit packets within a given amount of time [4].

Our work presented in this paper focuses on letting the master poll the slave when the slave has data to send and making the slave stay in the low power mode until the master wants to communicate with it. We use the *guaranteed service model* [5] as the underlying model of our approach and assume that every flow will ask for a required flow rate. Under 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's transmission 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 the active mode. 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 adaptively adjusts the predicted rate of the flow based on the power tuning knob and the flow's attribute parameters. While dealing with the power issue, we still take QoS of each flow into account. 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 remainder of this paper is organized as follows. Section II gives a brief introduction to the backgrounds and the motivations of this paper. In section III, 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 IV. Section V concludes the paper.

II. Background and Motivations

A. 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 slot. In the *Hold mode*, a slave goes into sleep 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 sleeps 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* [6].

0-7803-7400-2/02/$17.00 © 2002 IEEE

B. Guaranteed Service Model

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 [5]. 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 [5], even though the server is idle at that time.

C. Motivation

In current commercial Bluetooth products, the RR is the default scheduling scheme as well as specified in the Bluetooth specification [6]. Under this scheduling policy, the master works in the work-conserving manner and keeps polling the slaves of the piconet in turns.

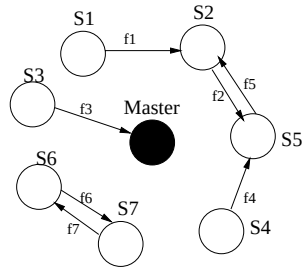


Fig. 1. A piconet example

For example, as shown in Figure 1, there are seven slaves and seven flows in the piconet. 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 doesn't know S_1's real-time traffic situation. If S_1 doesn't have any packet to send, still has to reply a response [6] that consumes one time slot. 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 toward S_1. Since other slaves don't know when they will be polled, they must 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[1] 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 the bandwidth to a flow based on the expected rate of the flow. Since the prediction may not be always accurate, it may incur extra 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.

[1]The reason of not using the sniff mode is that the sniff mode is suitable to inherently regular traffic and is less flexible to use when compared to the hold mode.

III. An Adaptive Power-Conserving Algorithm

In this section, we present the APCB[2] service model which is based on the following assumptions: error-free channel, single piconet and the system clock is synchronized among the master and the slaves.

A. The APCB Service Model

We adopt the idea of Virtual Clock service model [7] and assume every sender of flows will provide its attribute parameters, such as flow rate and burst degree [3], to the master before the communication starts. Since the master cannot get the real-time knowledge of the arrival time of a flow's packets, it only uses the *expected arrival time* (EAT) of the packets of the flow to predict when the sender will have data to send. We first introduce some notations before presenting our algorithm:

- L_i^k denotes the k^{th} packet length (in bit) of flow i and L_i^{max} denotes the maximum packet length of flow i;
- $EAT(p_i^k)$ is the EAT of the k^{th} packet of flow i.
- r_i denotes the max bandwidth admitted to flow i;
- $r(p_i^k)$ denotes the *Expect Transmission Rate* of the k^{th} packet of flow i, and it is initialized to r_i;
- α_i denotes the power tuning knob for flow i and $0.0 < \alpha_i \leq 1.0$
- σ_i denotes the burst degree of flow i and $\sigma_i \geq 1$;
- $\langle start, sleep \rangle$ means the node will hold from time $start$ and last $sleep$ second(s).
- $clock$ is the system time clock;
- δ denotes the time for the end nodes of flow i to resynchronize the channel from hold to active state.[4].

The master works in a 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. For every flow i, the whole algorithm is related to the master and the end nodes, who are $Sender_i$ and $Reciever_i$. Applying the same algorithm, the master, $Sender_i$, and $Reciever_i$ can mutually agree on the next polling time based on the current packet length L_i^k and the current expected transmission rate $r(p_i^k)$. Note that the end nodes also adjust $r(p_i^k)$ as the master does. If there is a mis-prediction, 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 in order to let the scheme work more power efficiently. 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 reduces 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* needs to be further explained. Suppose a slave is the end node of flow set

[2]APCB stands for Adaptive Power Conserving service discipline for Bluetooth

[3]These parameters are assumed to come from the application layer

[4]The length of δ is set to be 1 time slot ($625\mu s$) in this paper

$\mathcal{M}$, 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. This can eliminate the situation where the master sends a packet to a holding slave. In $Reciever_i$, if the receiver gets no packet addressed to it until the channel is idle, which means the master didn't forward data to the receiver, p will be set to NULL. The variable $WakeupTime$ is used when p is set to $NULL$. Since the receiver may need some time to wait until the channel is idle (other flow's packet(s) 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 up on time and get further forwarded packets of flow i from the master. The algorithm is shown in Figure 2.

B. Analysis of the APCB Service Model

In this section, we demonstrate the QoS properties of the APCB service model. Let $EFT(p_i^k)$ denote the *Expected Finish Time* of the k^{th} packet of flow i, and it is defined as: $EFT(p_i^k) = EAT(p_i^k) + L_i^k/r(p_i^k)$. 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 $[t_1, t_2]$. Due to space limit, we only give the results and part of the proofs. More details can be found in [8].

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

$$W_f(v_1, v_2) \geq r_f(v_2 - v_1 - \Delta t_f) - L_f^{max}(\alpha_i^{-1} - \phi_f) \quad (2)$$

where $v_1 = v_s(t_1), v_2 = v_f(t_2)$, $\phi_f = \lfloor \frac{(1-\alpha_i)r_f}{\sigma_i L_f^{max}} \rfloor$, 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. Let $\hat{t_A}$ denote the time needed to serve packet set A and $\hat{t_A} = \sum_{i\in A} \frac{L_f^i}{r(p_f^i)}$.
- The set, denoted by B, consists of packets that have the expected rate equal to r_f. 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 ($v_1 \leq t_0 \leq v_1 + \frac{L_f k}{r(p_f^k)}$). Since flow f is backlogged in the interval $[v_1, v_2]$, we can get:

$$W_f(v_1, v_2) = W_f(t_0, \hat{t_A}) + r_f \hat{t_B} \quad (3)$$

Since $\hat{t_A} + \hat{t_B} \leq v_2 - t_0$, 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 $\lfloor \frac{(1-\alpha_i)r_f}{\sigma_i L_f^{max}} \rfloor$. As a result, in the worst case,

$$W_f(t_0, \hat{t_A}) = \phi_f * L_f^{max} \quad (4)$$

$Master$: Select flow i that its k^{th} packet has the smallest EAT.
if $clock < EAT(p_i^k)$ **then**
    idle until $EAT(p_i^k)$;
$EAT(p_i^k) = clock$
poll $sender_i$ and get packet p;
if $p = NULL$ **then**
/*sender has no packet to send*/
    $r(p_i^k) = \alpha_i * r_i$
    $EAT(p_i^k) = EAT(p_i^k) + L_i^{max}/r(p_i^k)$
else
    $r(p_i^k) = min(r(p_i^k) + L_i^{max} * \sigma_i,\ r_i)$
    $EAT(p_i^{k+1}) = EAT(p_i^k) + L_i^k/r(p_i^k)$
    FORWARD(p)

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

$Reciever_i$: When wake up
$WakeupTime = clock$
wait for the forwarded packet p
if p=NULL **then**
    $r(p_i^k) = \alpha_i * r_i$
    $\langle start_i, sleep_i\rangle = \langle clock, L_i^{max}/r(p_i^k) - \delta - (clock - WakeupTime)\rangle$
else
    $r(p_i^k) = min(r(p_i^k) + L_i^{max} * \sigma_i,\ r_i)$
    $\langle start_i, sleep_i\rangle = \langle clock, L_i^k/r(p_i^k) - \delta\rangle$
$\langle start, sleep\rangle$=GetActualIdlePeriod()
Hold($\langle start, sleep\rangle$)

Fig. 2. The Algorithm of APCB

$$\hat{t_A} = \sum_{i=1}^{\phi_f} \frac{L_f^{max}}{\alpha_i r_f + \sigma_i L_f^{max} i} \quad (5)$$

From (3), (4), (5), and $t_0 \leq v_1 + \frac{L_f max}{\alpha_f r_f}$, we can get:

$$\begin{aligned} W(t_1, t_2) &\geq r_f(v_2 - v_1 - \frac{L_f^{max}}{\alpha_f r_f} - \hat{t_A}) + \phi_f L_f^{max} \\ &= r_f(v_2 - v_1 - \hat{t_A}) - L_f^{max}(\alpha_f^{-1} - \phi_f) \quad (6) \end{aligned}$$

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:

$$W_f(v_1, v_2) \leq r_f(v_2 - v_1) + L_f^{max} \quad (7)$$

$where\ v_1 = v_s(t_1), v2 = v_f(t_2).$

B.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:

$$|\frac{W_f(v_1,v_2)}{r_f} - \frac{W_m(v_1,v_2)}{r_m}| \leq$$
$$max_{i,j\in\{f,m\}}\{\frac{L_i^{max}}{r_i} + \frac{L_j^{max}(\alpha_j^{-1}-\phi_j)}{r_j} + \Delta t_j\} \quad (8)$$

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

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

$$\lim_{v\to\infty} \frac{W_f(0,v)}{v} = r_f \quad (9)$$

B.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:

$$W_f(t_1,t_2) \geq r_f(t_2-t_1-\Delta t_f) - \frac{r_f}{C}\sum_{i\in Q} L_i^{max} - L_f^{max}(\alpha_i^{-1}-\phi_f) \quad (10)$$

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

B.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 N^{th} 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 $DP(p_f^j)$, is given by:

$$DP(p_f^j) \leq EAT(p_f^{HOL}) + \sum_{i=1}^{min(\phi_f, N-1)} \frac{L_f^{max}}{\alpha_f r_f + \sigma_f L_f^{max} i}$$
$$+(max(\phi_f, N-1) - \phi_f)\frac{L_f^{max}}{r_f} + \frac{\sum_{i\in Q} L_i^{max}}{C} \quad (11)$$

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

IV. 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 layer. We assume that the channel is error-free and the baseband packet limit is 5-slot packet (339 Bytes), which is specified in the specifications [6]. The simulation topology is shown in Figure 1, and the parameters of each flow are listed in Table I. The holding threshold ϵ is set to one time slot. 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 dependent. In order to show this relationship, we evaluate our policy under two traffic models, the CBR model and the ON/OFF model. For the CBR model, the bursty degree $\sigma_i = 1.0$. For the ON/OFF model, $\sigma_i = 2.0$.

TABLE I
Connections Parameters

Flow	Flow Rate (bps)
$S_1 \rightarrow S_2$	27120
$S_2 \rightarrow S_5$	40680
$S_3 \rightarrow Master$	81360
$S_4 \rightarrow S_5$	54240
$S_5 \rightarrow S_2$	62376
$S_6 \rightarrow S_7$	27120
$S_7 \rightarrow S_6$	108480

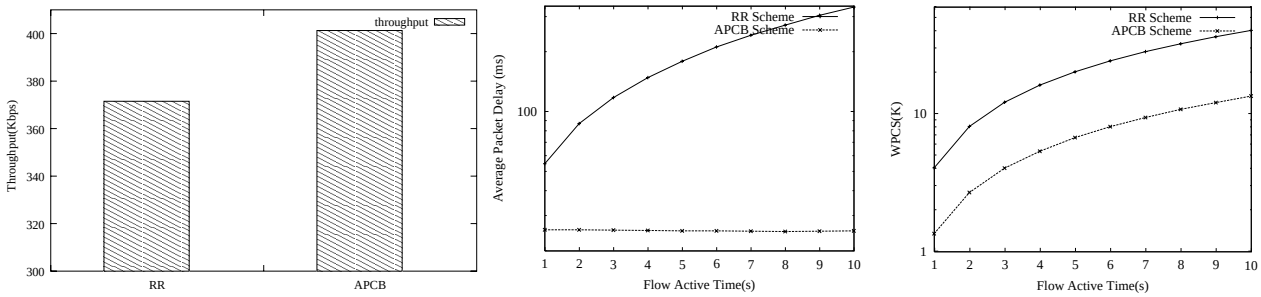
The following metrics are used to evaluate the algorithm: the total throughput for all the flows, the average packet delay and the total Weighted Power Consumption Slots (WPCS) of all the slaves in the piconet. Here, we define the WPCS as the weighted number of slots with power consumption by a device in the time period of T. For a Bluetooth device, it has four communication modes: *Tx, Rx, Active, Sleep*. We assume the weight of the four modes are 1.0, 0.5, 0.2, and 0.0, respectively.

A. The CBR Model

In this model, the master does not mis-predict the EAT of flows. Thus, α has no effect in this scenario and we set it to 1.0. The performance results are shown in Figure 3. As can be seen, our scheme has a much larger throughput. Our scheme not only reduces the power consumption by half, but also significantly reduces the packet delay compared to the RR scheme. This can be explained by the fact that APCB doesn't 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, the packet delay of the RR scheme increases continuously as the flow active time increases.

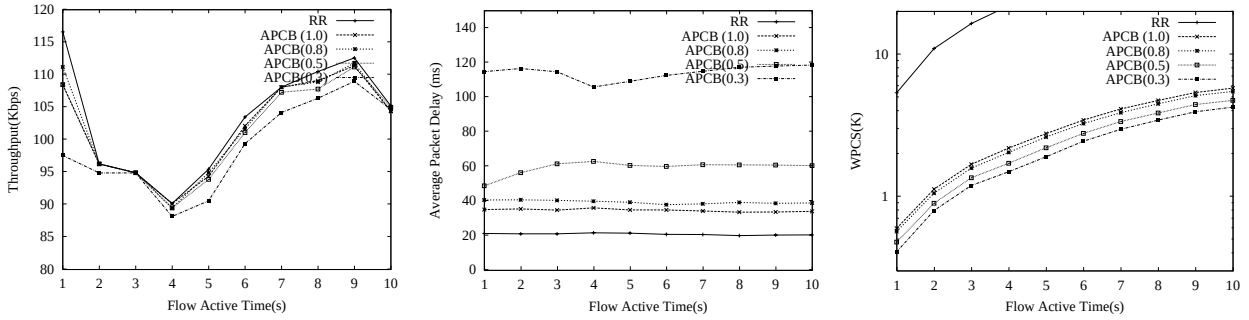
B. The ON/OFF Model

ON/OFF is an interrupted process. We choose the mean idle period and the mean active period to be 200 ms and 100 ms, respectively. They are assumed to follow a Gaussian Marginal distribution, and the random numbers are generated similarly to [9]. 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 Figure 4. 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



(a) Throughput (b) Delay (c) WPCS

Fig. 3. Performance comparisons between RR and APCB under the CBR traffic model



(a) Throughput (b) Delay (c) WPCS

Fig. 4. Performance comparisons between RR and APCB under the ON/OFF traffic model

approach, the power consumption of the APCB approach are reduced as much as 85% compared to the RR approach. 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. From Figure 4, 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 delay increase. 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 $\alpha = 0.5$.

V. Conclusions and Future Works

In this paper, we proposed an adaptive power-conserving scheduling approach 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 Round Robin 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 the 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 dynamically get a good balance between power and delay based on a certain criteria. 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] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks," *IEEE INFOCOM'01*, April 2001.

[2] J. Monks, V. Bharghavan, and W. -M Hwu, "A Power Controlled Multiple Access Protocol for Wireless Packet Networks," *IEEE INFOCOM'01*, March 2001.

[3] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication," *MobiCOM'98*, October 1998.

[4] B. Prabhakar, E. Uysal-Biyikoglu, and A. El Gamal, "Energy-Efficient Transmission over a Wireless Link via Lazy Packet Scheduling," *IEEE INFOCOM'01*, March 2001.

[5] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, vol. 83, no. 10, October 1995.

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

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

[8] H. Zhu, G. Cao, G. Kesidis and C. Das, "An Adaptive Power-Conserving Service Discipline for Bluetooth (APCB) Wireless Networks," *Technical Report (CSE-01-019), Pennsylvania State University*, 2001.

[9] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang, "A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks," *ACM SIGCOMM'95*, September 1995.