Round-Robin with FCFS Preemption: A Simple MAC Scheduling Scheme for Bluetooth Piconet

Li-Hsing Yen
Dept. Computer Science & Information Engineering
Chung Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: lhyen@chu.edu.tw

Chi-Hung Liao
Dept. Computer Science & Information Engineering
Chung Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: Teerrence@pchome.com.tw

***Abstract*— Bluetooth is a short-range TDD (Time Division Duplex) wireless network that supports both circuit- and packet-oriented applications. A piconet is composed of a device configured as master and at most seven other devices acting as slaves. At Medium Access Control (MAC) layer, the master can select a slave to send a data packet and until then, the slave is not allowed to transmit. Round-Robin (RR) and Exhaustive Round-Robin (ERR) are two elementary MAC scheduling schemes that are both simple and efficient. This paper proposes RR-FCFS, a simple MAC scheduling scheme that has the same advantages as RR and ERR. RR-FCFS acts as RR if the master's queue is empty and starts transmitting packets in first-come-first-serve order otherwise. The simulation results show that RR-FCFS's performance in terms of packet delay and queue length is comparable with those of RR and ERR.**

I. Introduction

Bluetooth [1] is a short-range wireless technology that was intended to replace numerous proprietary cables. It provides both voice and data transmission services, which allow users to interconnect their laptops, handsets, and other devices forming a small indoor picocellular wireless network (piconet). A piconet is an arbitrary collection of Bluetooth-enabled devices which are close enough to be able to communicate and are exchanging information in a regular way [2]. A device configured as master forms and manages a piconet, while other devices acting as slaves listen to the master and adopt the master's timing. All devices share the radio channel in a time-division manner and transmissions are TDD (Time Division Duplex). The master sends a packet to a slave on even-numbered time slot, while the slave sends packet to the master on an odd slot. A slave is allowed to take a slot for packet transmission only right after it is polled by the master, i.e., it receives a packet from the master. The master may send POLL packet rather than data packet to a slave. A POLL packet does not convey any user information and serves only for the purpose of giving the polled slave the privilege of sending packet in the succeeding slot. When the polled slave has no data packet to send, it responds to the master with a NULL packet which does not convey any user information.

Two types of physical links are supported by Bluetooth: SCO (Synchronous Connection-Oriented) and ACL (Asynchronous ConnectionLess). SCO links support circuit-oriented applications with constant bit rate and must use reserved time slots in a periodic fashion. An ACL link, on the other hand, provides a packet-switched connection between the master and a slave and can only be allocated unreserved slots. For any unreserved even slot, the master may either select one device to transmit ACL traffic or simply do nothing. The selection rule corresponds to MAC (Medium Access Control) scheduling. The performance issues we consider here include packet delay and queue length. A straightforward scheduling policy would be Round Robin (RR), which polls each slave in a cyclic order and transmits POLL/NULL packet when necessary. Exhaustive Round Robin (ERR) [3] is another simple scheduling policy which polls each slave in a cyclic order as well but does not switch to the next slave until the current master-slave pair has no more data to send in either direction. Several researchers have studied Bluetooth MAC scheduling problem [4], [5], [3], [6], [7], [8], [9]. Some works assume that the master has up-to-date knowledge of the slave queues' status [4], [3], [10], which makes these works impractical. Some approaches utilize Bluetooth's power-saving mode [5], [6] or control master's polling frequency [8] to reduce power consumption. Lin *et al.* [9] proposed a scheme that aims to increase link utilization by exploiting different combinations of Bluetooth packet types. This approach is too complex to be practical and it will increase packet delay. In [7], Das *et al.* utilized flow information at upper-layer to give more slots to slaves that have more queued data packets. However, this approach requires cross-layer information and it lacks a systematic way to determine the optimal number of slots that should be given.

In this paper, we propose a simple MAC scheduling scheme that requires neither slave queues' status nor cross-layer information. It operates in two folds: when the master's queue is empty, RR is used to poll each slave; when the master's queue is non-empty, queued packets are served in a FCFS (First-Come-First Serve) fashion. Our simulation results show that this scheme's performance is comparable with those of RR and ERR.

The rest of the paper is organized as follows. The MAC scheduling problem and related work are described in Section 2. Section 3 details our MAC scheduling scheme and analyzes stability conditions of network. Simulation model and numerical results are presented in Section 4. Section 5 concludes our

work.

II. Problem Definition and Related Work

MAC scheduling problem in Bluetooth piconet can be modeled as follows. The master has logically 7 individual master-to-slave queues storing packets destined for slaves, one for each slave. Each slave has its own slave-to-master queue storing packets to be sent to the master. A slave can send a packet to the master on slot i if and only if it has received data from the master on slot $i-1$. The problem is to schedule packet transmission in master-to-slave queues so higher performance (lower packet delay, shorter queue length, etc.) can be achieved. When some master-to-slave queue is selected to be served but no data packet is pending there, the master can either send a POLL packet or omit that slave.

RR is simple and straightforward, but it wastes bandwidth on transmitting POLL/NULL packets under light traffic load. It also forces an equal partition of bandwidth capacity to each slave. Therefore, RR may suffer from performance degradation if traffic load from each slave are non-uniform. ERR performs well under light traffic load but suffers from long packet delay in case of heavy load. It may also need to transmit POLL/NULL packets when the load on master-to-slave and on slave-to-master queues are not symmetric. There are "preview" schemes assuming that the master has up-to-date knowledge of the slave-to-master queues' status [4], [3], [10]. This is only possible in an ideal scenario even if an explicit signaling between the master and slaves were provided [3]. This kind of preview schemes will not be discussed further. Some related researches aim to reduce power consumption by utilize Bluetooth Sniff operation mode [5], [6]. In Sniff mode, a slave listens for a master transmission every predefined period (a sniff interval) for a specified number of slots (an active window). Garg *et al.* [5] proposed to dynamically adjust sniff interval and/or active window on the basis of slot utilization. Other schemes proposed by Chakraborty *et al.* [6] try to predict slaves' traffics to decide the criterion of switching to or back from Sniff mode, under the assumption that the inter-arrival time till the next packet is drawn from the same distribution as those that have been observed. ASP [8] assumes that sources send data in constant bit rate so the master can implicitly learn appropriate share ratio of bandwidth allocated to each slave. The bandwidth share is controlled by limiting the polling success rate to a desired range. Though effective in reducing power consumption, these power efficient schemes all have the disadvantage of increasing queueing delay of packets.

Lin *et al.* [9] proposed a scheme that uses different combinations of Bluetooth packet types to match the traffic characteristics of master and slaves. This work assumes that the traffic arrival rates of each master-slave pair are known and the goal is to increase bandwidth utilization. This approach is too complex to be practical and it will increase packet delay. In [7], Das *et al.* exploited *flow bit* that is used to convey upper-layer (i.e., L2CAP) flow information as intended in the Bluetooth specification [1]. Their approach sets the flow bit when the number of packets buffered at the upper-layer reaches a predefined threshold, and sets variable *flow* when the flow bit for packets traveling in either direction is turned on. Based on the flow information, the authors proposed three approaches: Adaptive Flow-based Polling (AFP), Stick, and StickAFP. AFP dynamically changes polling intervals in favor of slaves that have more queued data packets. Sticky operates in RR fashion and StickyAFP is similar to AFP. They allow a slave with *flow* set to transmit at most *num_sticky* packets when the slave is polled. It was reported that AFP and Sticky have higher throughput than RR on a particular simulated network. The disadvantage of this work is that it requires cross-layer information and it has no systematic way to determine the appropriate value for *num_sticky*.

III. Proposed Scheme and Analysis

Our MAC scheduling scheme operates in accordance with the state of the master-to-slave queues. All master-to-slave queues are viewed as a global one where packets are stored in their arrival order. When the queue is empty, this scheme polls slaves in round-robin fashion. Whenever the queue becomes non-empty, the master suspends RR polling and starts transmitting packets in First-Come-First-Serve (FCFS) order. After emptying the queue, the master resumes regular RR polling. We denote this scheme by "RR-FCFS" hereafter.

RR-FCFS integrates RR with FCFS. The basic idea behind this integration is that FCFS has the shortest delay for master-to-slave packets and RR would acquire low delay average for slave-to-master packets. In fact, our preliminary study found the following result:

- In the downlink part (concerning master-to-slave traffic), FCFS has the shortest delay. RR-FCFS performs similar to but slightly worse than FCFS does. Both FCFS and RR-FCFS are significantly better than RR.
- In the uplink part (concerning slave-to-master traffic), RR has the shortest delay. RR-FCFS performs slightly worse than RR does but significantly better than FCFS does.

The result is obtained from simulations with the following settings:

- uniform traffic load among slaves,
- symmetric traffic load between the master to a salve and the slave to the master, and
- various traffic load ranging from 0.1 to 0.9 packet/slot-pair.

In terms of average packet delay, RR-FCFS outperforms either RR or FCFS. More comprehensive results will be presented in the next section.

In the following, we analyze the stability condition for a piconet with RR, ERR, and RR-FCFS. Let λ_m^i be the mean data traffic load from the master to slave i and λ_i^m be that from slave i to the master. Let us assume seven slaves and define system configuration $\sigma = \{\lambda_m^i, \lambda_i^m\}_{i=1}^7$ to be a particular set of all traffic load. We say that σ is stable if no queue in either direction of the link grows unlimitedly, which can be guaranteed if traffic load on any channel does not exceed the bandwidth allocated to that channel.

TABLE I

A SCENARIO FOR WHICH THE SYSTEM IS UNSTABLE WITH ERR.

i	1	2	3	4	5	6	7
λ_m^i	$C/7$	0	$2C/7$	0	$C/7$	$C/7$	0
λ_i^m	0	$2C/7$	0	$2C/7$	0	0	$C/7$

Denote C the total bandwidth capacity of all uplink channels. Due to TDD transmission, the total bandwidth capacity of all downlink channels is also C. Since RR forces an equal bandwidth share on either direction, the condition for σ staying in stable with RR is

$$\forall i : \lambda_m^i < \frac{C}{7} \wedge \lambda_i^m < \frac{C}{7}. \tag{1}$$

With RR, it is possible to have an overloaded channel (*i.e.*, traffic load on a channel is higher than the bandwidth allocated to that channel), while others remain stable (traffic load is lower than allocated bandwidth).

ERR allocates bandwidth $\max(\lambda_m^i, \lambda_i^m)$ to both uplink and downlink channels of master-slave pair i, so the condition for system's stability with ERR is

$$\sum_{i=1}^{7} \max(\lambda_m^i, \lambda_i^m) < C. \tag{2}$$

With ERR, it is possible that neither the total traffic load on all uplink channels nor that on all downlink channels exceeds C but the system is still unstable. Table I illustrates a scenario where total traffic load on either direction is only $5C/7$ but the system is unstable with ERR.

The 'FCFS' part of RR-FCFS allocates bandwidth to each downlink channel according to the channel's traffic load. Therefore, the total traffic load on all downlink channels must not exceed C for system stability. The remaining available downlink bandwidth is equally distributed to all downlink channels by the 'RR' part of RR-FCFS. Since each uplink channel receives the same amount of bandwidth as the corresponding downlink channel does due to TDD, the condition for system's stability with RR-FCFS is

$$\sum_{j=1}^{7} \lambda_m^j < C \wedge \forall i : \lambda_i^m < \lambda_m^i + \frac{C - \sum_{j=1}^{7} \lambda_m^j}{7}. \tag{3}$$

The following two theorems show that ERR has a wider range of operation retaining stability then either RR or RR-FCFS has.

Theorem 1: For any given system configuration σ, if σ is stable using RR, σ is also stable using ERR.

Proof: Since σ is stable using RR, Eq. (1) is ensured. It follows that $\sum_{i=1}^{7} \max(\lambda_m^i, \lambda_i^m) < C$. So σ is also stable using ERR. ■

Theorem 2: For any given system configuration σ, if σ is stable using RR-FCFS, σ is also stable using ERR.

TABLE II

A SCENARIO ILLUSTRATING DAMAGE CONTROL.

i	1	2	3	4	5	6	7
λ_m^i	$C/14$	$C/14$	$2C/7$	$C/14$	$C/14$	$C/14$	$C/7$
λ_i^m	$C/14$	$2C/7$	$C/14$	$C/7$	$C/14$	$C/14$	$C/14$

Proof: Let M denote the set of slaves i such that $\lambda_m^i < \lambda_i^m$ and let N be $\{1, 2, \cdots, 7\} - M$. We have

$$\sum_{i=1}^{7} \max(\lambda_m^i, \lambda_i^m) = \sum_{i \in N} \lambda_m^i + \sum_{i \in M} \lambda_i^m. \tag{4}$$

Since σ is stable using RR-FCFS, by (3) we have

$$\begin{aligned} \sum_{i \in M} \lambda_i^m &< \sum_{i \in M} \left(\lambda_m^i + \frac{C - \sum_{j=1}^{7} \lambda_m^j}{7} \right) \\ &= \sum_{i \in M} \lambda_m^i + |M| \left(\frac{C - \sum_{j=1}^{7} \lambda_m^j}{7} \right). \end{aligned}$$

Note that $|M| \leq 7$, so

$$\sum_{i \in M} \lambda_i^m < \sum_{i \in M} \lambda_m^i + C - \sum_{j=1}^{7} \lambda_m^j. \tag{5}$$

By (4) and (5), it is clear that

$$\sum_{i=1}^{7} \max(\lambda_m^i, \lambda_i^m) < \sum_{i=1}^{7} \lambda_m^i + C - \sum_{j=1}^{7} \lambda_m^j = C.$$

So σ is also stable using ERR. ■

Both RR and RR-FCFS are superior to ERR in 'damage control', which refers to the ability of confining the impacts of an overloaded channel to that channel. With ERR, whenever the system becomes unstable, all queues grow without bound. In contrast, only queues that correspond to overloaded channels grow unlimitedly with RR or RR-FCFS. There is always some queue remaining bounded as long as the overall downlink/uplink traffic load does not exceed the associated bandwidth capacity. As an example, all queues grow unlimitedly with ERR in the scenario of Table II, while only two do so with either RR or RR-FCFS.

IV. SIMULATION AND RESULTS

We conducted simulations to evaluate the performance of RR-FCFS. We assume a piconet consisting of one master and seven slaves numbered from 1 to 7. The master generates packets to slave i by a Poisson process with rate λ_m^i and slave i generates packets to the master by a Poisson process with rate λ_i^m. We say that the traffic is *symmetric* if $\lambda_m^i = \lambda_i^m$ for all i and *asymmetric* otherwise. The traffic is *uniform* if $\lambda_m^i = \lambda_m^j$ and $\lambda_i^m = \lambda_j^m$ for all i, j, and the traffic is *non-uniform* otherwise. Table III lists the experiment settings which cover possible combinations of different traffic patterns. A variable, λ, is used to control traffic load on both downlink and uplink channels. Its value is ranged from 0.1 to 0.9 packet/slot-pair. Total 60,000 packets are generated for each simulation.

TABLE III

EXPERIMENT SCENARIOS (λ IS A TUNABLE PARAMETER).

ID	Traffic type	Setting
S1	Symm. & Uniform	$\lambda_m^i = \lambda_i^m = \lambda/7$ for all i
S2	Asym. & Uniform	$\lambda_m^i = \lambda/7$ and $\lambda_i^m = \lambda/14$ for all i
S3	Asym. & Uniform	$\lambda_m^i = \lambda/14$ and $\lambda_i^m = \lambda/7$ for all i
S4	Symm. & Non-uni	$\lambda_m^1 = \lambda_m^3 = \lambda_m^5 = \lambda_m^7 = \lambda/12$; $\lambda_m^2 = \lambda_m^6 = \lambda/2$; $\lambda_m^4 = \lambda/3$; and $\lambda_i^m = \lambda_m^i$ for all i
S5	Asym. & Non-uni	$\lambda_m^1 = \lambda_m^3 = \lambda_m^4 = \lambda_m^6 = \lambda/12$; $\lambda_m^2 = \lambda_m^5 = \lambda/6$; $\lambda_m^7 = \lambda/3$; $\lambda_1^m = \lambda/3$; $\lambda_3^m = \lambda_6^m = \lambda/6$; and $\lambda_2^m = \lambda_4^m = \lambda_5^m = \lambda_7^m = \lambda/12$

All packets are assumed of one-slot length (i.e., DH1 packets [2]).

We investigated packet delay and queue length of RR-FCFS. The results are compared with those of RR and ERR. RR and ERR are chosen simply because they both are simple and efficient methods.

A. Stability Condition

Before presenting the numerical results, it is worth noting that all queues in Scenarios S1-S3 (Table III) are all stable with either RR, ERR, or RR-FCFS. In S4, all queues becomes unstable when $\lambda \geq 0.6$ with either ERR or RR-FCFS. With RR, some queue becomes unstable when $\lambda > 2/7$. In S5, the threshold value of λ (the maximum λ value for a given queue staying in stable) varies. With RR, some queue becomes unstable when $\lambda > 3/7$ while some others are stable even when $\lambda > 1$. The threshold value for all queues with ERR are 0.71. With RR-FCFS, $\sum_i \lambda_m^i = \lambda$, so slave-1-to-master queue (the most extreme one) grows unbound when

$$\frac{\lambda}{3} \geq \frac{\lambda}{12} + \frac{1-\lambda}{7}.$$

Thus the threshold value for slave-1-to-master queue is ~ 0.36. However, many other queues stay in stable through all experiments. This can be easily verified using the analytic results presented in the preceding section.

B. Packet Delay

Packet delay is the time (measured in slots) between a packet's generation and transmission. Figures 1-5 show the averaged results for S1 to S5, respectively, under various values of λ. Generally speaking, RR-FCFS performs better than RR in most cases, and ERR outperforms RR in all cases with the exception of S1 under heavy load. The results of RR-FCFS are similar to those of ERR in symmetric traffic cases (Figures 1 and 4). In asymmetric-and-uniform traffic cases, RR-FCFS outperforms ERR if the master has higher total packet rate than slaves have, i.e., $\sum \lambda_m^i > \sum \lambda_i^m$, (Figure 2) and performs worse than ERR otherwise (Figure 3). In S5 (Figure 5), ERR is superior to RR-FCFS because ERR has a higher threshold value of λ.

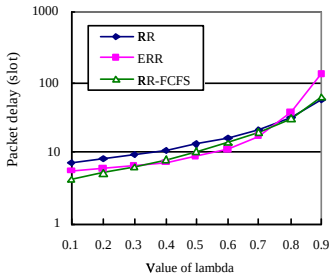

Fig. 1. Packet delay with S1

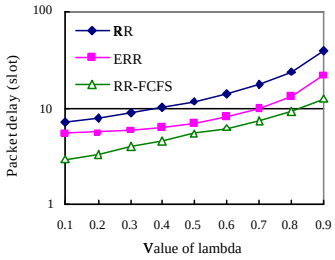

Fig. 2. Packet delay with S2

C. Queue Length

Queue length is defined to be the number of packets pending in queue. Figures 6-8 show the average queue lengths with S1, S2, and S3, respectively. Clearly, RR-FCFS always has the shortest master-to-slave queues. For S4, we look at slave 4 as it is the most extreme case. RR's master-to-slave-4 and slave-4-to-master queues grow unlimitedly when $\lambda > 0.43$ (Figure 9). The reason is that when $\lambda > 0.43$, $\lambda_m^4 = \lambda_4^m = \lambda/3$ is larger than $1/7$, the service rate allocated to slave 4. The average length of master-to-slave queue with RR-FCFS is shorter than that with ERR. ERR has shorter slave-to-master queue than RR-FCFS has when λ is below the threshold value.

For S5, Figures 10 and 11 show the average queue length concerning slaves 1 and 7, respectively. Again, RR-FCFS has the shortest master-to-slave queues in both cases. Concerning slave-1-to-master queues, ERR is superior to RR and RR-FCFS for its higher threshold value of λ. In Figure 11(a), the master-to-slave-7 queue with RR gets overwhelmed when $\lambda > 0.4$ and that with ERR does so when $\lambda > 0.8$. In contrast, RR-FCFS behaves stably. In Figure 11(b), ERR has the longest slave-7-to-master queue under heavy load due to the lack of

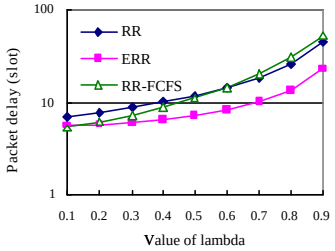

Fig. 3. Packet delay with S3

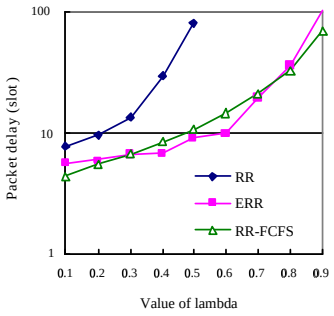

Fig. 4. Packet delay with S4

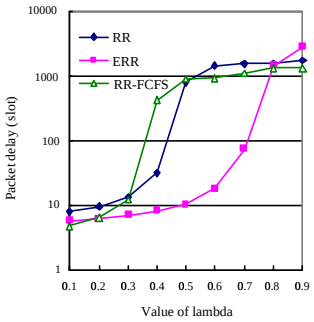

Fig. 5. Packet delay with S5

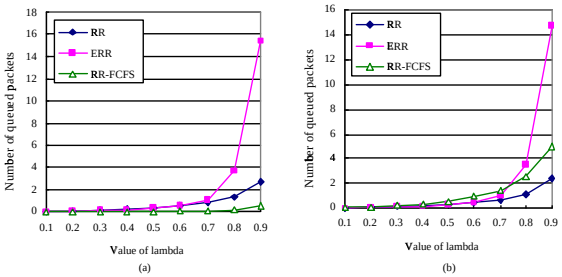

Fig. 6. Average queue length with S1. (a) master-to-slave (b) slave-to-master

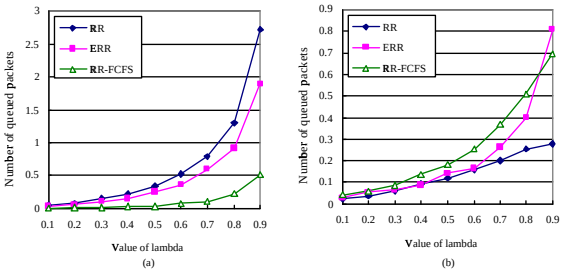

Fig. 7. Average queue length with S2. (a) master-to-slave (b) slave-to-master

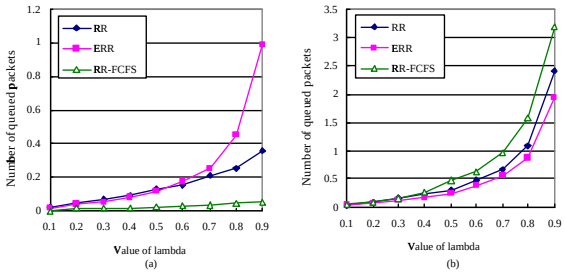

Fig. 8. Average queue length with S3. (a) master-to-slave (b) slave-to-master

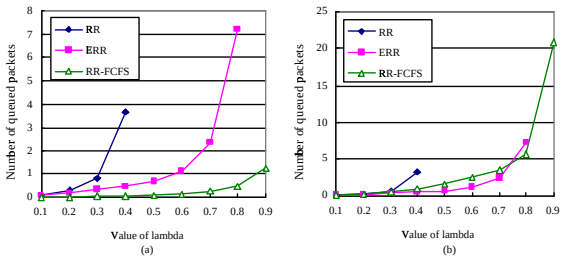

Fig. 9. Average queue length of master-slave-4 pair with S4. (a) master-to-slave-4 (b) slave-4-to-master

'damage control'. Both RR and RR-FCFS are better than ERR but RR-FCFS performs slightly better than RR does. The reason for RR-FCFS's superiority is that it provides higher service rate to slave 7 ($\lambda_m^7 = \lambda/3$) than RR does under moderate to high traffic load.

V. Conclusions

MAC scheduling scheme for Bluetooth piconet is critical to link-layer performance. RR and ERR are two simple and efficient MAC scheduling schemes that require neither slave queues' status nor cross-layer information. This paper proposes RR-FCFS, a simple MAC scheduling scheme that has the same advantages as RR and ERR. RR-FCFS acts as RR if the master's queue is empty and starts transmitting packets in FCFS order otherwise.

Our analysis shows that ERR has a wider range of operation retaining stability then either RR or RR-FCFS has. But RR and RR-FCFS has better 'damage control' in the sense that the impacts of an overloaded channel can be confined to that channel. The simulation results show that ERR and RR-

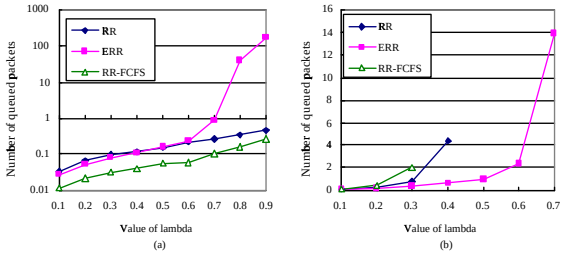

Fig. 10. Average queue length with S5. (a) master-to-slave-1 (b) slave-1-to-master

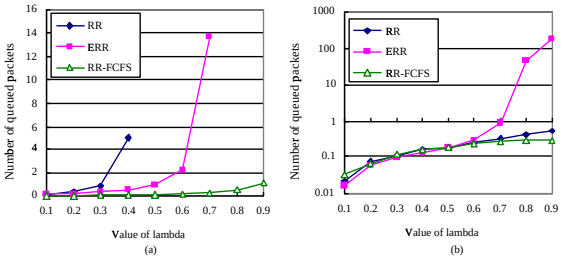

Fig. 11. Averaged queue length with S5. (a) master-to-slave-7 (b) slave-7-to-master

FCFS generally outperforms RR in terms of packet delay. The packet delay with RR-FCFS is similar to that with ERR under symmetric traffic, better than ERR under asymmetric-and-uniform traffic, and worse than ERR in other cases.

RR-FCFS always has the shortest queue for downlink channels. RR's master-to-slave or slave-to-master queues may easily grow unlimitedly under non-uniform traffic, even overall downlink/uplink traffic load does not exceed bandwidth capacity. ERR generally has shorter slave-to-master queue than RR-FCFS has, since some slave-to-master queue with RR-FCFS may become unstable when the traffic load on uplink channels is higher than that on downlink channels. On the other hand, RR-FCFS may perform the best at slave queues where it provides higher service rate than necessary (at the cost of overwhelming other slave queues).

In short, RR-FCFS's performance is comparable with those of RR and ERR, and none is definitely better than others in all circumstances.

ACKNOWLEDGEMENT

This work has been jointly supported by the National Science Council, Taiwan, under contract NSC-92-2213-E-216-004 and by Chung Hua University under contract CHU-93-TR-004.

REFERENCES

[1] *Specification of the Bluetooth System Version 1.1: Core*, Bluetooth Special Interest Group, February 2001, http://www.bluetooth.org.

[2] J. Bray and C. F. Sturman, *Bluetooth: Connect Without Cables*. Prentice Hall PTR, 2001.

[3] A. Capone, M. Gerla, and R. Kapoor, "Efficient polling schemes for Bluetooth picocells," in *IEEE International Conference on Communications*, June 2001, pp. 1990–1994.

[4] M. Kalia, D. Bansal, and R. Shorey, "Data scheduling and SAR for Bluetooth MAC," in *Proceedings of IEEE Vehicular Technology Conference*, Tokyo, Japan, May 2000, pp. 716–720.

[5] S. Garg, M. Kalia, and R. Shorey, "MAC scheduling policies for power optimization in Bluetooth: A master driven TDD wireless system," in *Proceedings of IEEE Vehicular Technology Conference*, Tokyo, Japan, May 2000, pp. 196–200.

[6] I. Chakraborty, 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," in *IEEE International Conference on Communications*, June 2001, pp. 1980–1984.

[7] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, "Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc networks," in *INFOCOM*, 2001, pp. 591–600.

[8] M. Perillo and W. B. Heinzelman, "ASP: An adaptive energy-efficient polling algorithm for Bluetooth piconets," in *Proc. of the 36th Hawaii International Conference on System Sciences*, 2003.

[9] T.-Y. Lin, Y.-C. Tseng, and Y.-T. Lu, "An efficient link polling policy by pattern matching for Bluetooth piconets," in *Proc. of the 36th Hawaii International Conference on System Sciences*, 2003.

[10] Y.-I. Joo, J.-S. Oh, O.-S. Kwon, Y. Kim, T.-J. Lee, and K. H. Tchah, "An efficient and QoS-aware scheduling policy for Bluetooth," in *Proc. of 56th Vechicular Technology Conference (VTC 2002-Fall)*, vol. 4, 2002, pp. 2445–2450.