A FAIR AND TRAFFIC DEPENDENT POLLING SCHEME FOR BLUETOOTH

ROHIT KAPOOR, ANDREA ZANELLA, MARIO GERLA

University of California, Los Angeles (UCLA)
3803, Boelter Hall, UCLA, CA 90095
Tel: 310-206-4772 Fax: 310-825-7578
E-Mail of Corresponding Author: rohitk@cs.ucla.edu

Bluetooth is a universal radio interface in the 2.45Ghz frequency band, which will enable users to connect a wide range of small electronic devices such as notebook computers, cellular phones etc. Two or more Bluetooth-enabled devices that come within range of each other can set up an ad hoc network, called a piconet. Within a piconet, one unit becomes the master and controls access to the channel. The rest of the units (up to seven in number) act as slaves. The master sends a data or POLL packet to poll a slave and the slave responds with a packet in the next time slot. The bandwidth of the piconet is divided among the slaves according to a polling algorithm used by the master. This polling algorithm has a significant impact on the system performance. In this paper, we propose a polling algorithm for Bluetooth that adapts to changes in traffic load and divides the piconet bandwidth among slaves in a max-min fair manner. We prove the fairness of the algorithm analytically and also verify it by simulations. We also show that the algorithm achieves a high efficiency in use of the piconet bandwidth.

1. Introduction

Bluetooth [1] is a universal radio interface in the 2.4 GHz ISM frequency band, which will enable users to connect a wide range of small electronic devices such as notebook computers, cellular phones and other portable handheld devices easily and quickly, without the need for cables. The key distinguishing features of Bluetooth are its minimal hardware dimensions, low complexity, low price and low power consumption [2].

Bluetooth is based on a centralized connection-oriented approach. Bluetooth devices sharing a wireless channel form a *piconet*. One device in a piconet has the role of the master and controls access to the channel, while the others are slaves. There may be up to 7 slaves in a piconet. Bluetooth uses a Time-Division Duplex (TDD) scheme to divide the channel into 625us time slots. Master and slave units transmit alternately. Each piconet is characterized by a particular fast frequency-hopping pattern; the frequency is uniquely determined by the master's address and is followed by all the devices participating in the piconet.

There are two types of connections that can be established between a master and a slave: the Synchronous Connection-Oriented (SCO) link, and the Asynchronous Connection-Less (ACL) link. SCO connections provide a circuit-oriented service with

Proc. of IEEE International Conference on Wireless LANs and Home Networks (ICWLHN 2002) and IEEE International Conference on Networking (ICN 2002), Atlanta, USA, 26-29 August 2002

constant bandwidth based on a fixed and periodic allocation of slots. ACL connections provide a packet-oriented service and span over 1, 3 or 5 slots [2]. For ACL links, Bluetooth uses a fast acknowledgment and retransmission scheme to ensure reliable transfer of data. The master controls the traffic on ACL links by employing a polling scheme to divide the piconet bandwidth among the slaves. A slave is only allowed to transmit after the master has polled it. The manner in which the master polls the slaves has a significant impact on the system performance.

In this paper, we propose a polling scheme for Bluetooth that adapts to changes in traffic load and divides the bandwidth among slaves in a max-min fair manner. In Section 2, we discuss previously proposed polling schemes for Bluetooth. In Section 3, we present a definition of max-min fairness and then describe our polling scheme and prove its fairness. We validate the traffic-adaptation and max-min fair behavior of the scheme through simulation experiments in Section 4. In Section 5, we present conclusions and discuss some future work.

2. Related Work

In this section, we review related polling schemes for Bluetooth. These schemes broadly fall under two categories: ideal and practical. The ideal schemes assume that the master has complete and updated knowledge of the queue status of the slaves. The practical schemes do not make any such assumption and are practically realizable; the ideal schemes serve as good performance benchmarks.

The ideal schemes presented in [3] and [4] assume that a master has updated knowledge of the status of slave's queues. A slave is not polled only if both the master and slave queues have no data packets. Another ideal scheme is presented in [5]. In this scheme, the master keeps polling the same slave until both the master and slave queues are empty. The next slave to be chosen is the one for which the sum of the master and slave queue lengths is the largest.

A practical polling scheme, called Limited and Weighted Round Robin (LWRR) is also presented in [5]. This scheme achieves a high efficiency by reducing the rate of visits to queues that have been found empty in previous visits. LWRR gives a weight equal to Max_Priority (MP) to each slave at the beginning. Each time a slave is polled and no data is exchanged between the master and the slave, the weight of the slave is reduced by 1. The lowest value of the weight of a slave is 1, in which case the slave has to wait a maximum of MP-1 cycles to get a chance to be polled. Even though LWRR is expected to show reasonably fair behavior, no results regarding the fairness of the scheme are presented. The polling scheme presented in this paper is proved to be max-min fair and this fairness is also verified by simulations.

3. Description of Algorithm and its Fairness

3.1 Max-Min Fairness [6]

Definition (max–min fairness): An allocation of rates $\eta_1, \eta_2, \ldots, \eta_s$ among S units is max–min fair if it is feasible[1], and for each unit i, η_i cannot be increased (while maintaining feasibility) without decreasing η_j for some other unit j for which $\eta_j <= \eta_i$.

3.2 Polling Algorithm

The polling algorithm is based on the master estimating the traffic rate between each slave and itself. This traffic rate is the sum of the traffic rates from the master to a slave and in the reverse direction. We assume, in order to simplify the explanation of the algorithm, that traffic flows only from slaves to master; masters generate no traffic to slaves. The same algorithm also applies with little change when traffic flows in both directions (explained later).

The master uses a Round Robin polling scheme, with the modification that a slave is skipped if it does not belong to the "active list" of the master. The slaves are moved in and out of the active list on the basis of two variables that the master maintains for each slave. These two variables are:

r – estimate of the rate of traffic generated by the slave
N – estimate of the queue length of the slave

When a slave is polled, the master–slave pair gets a chance to exchange a maximum amount of data denoted by M (in each direction). At the end of this poll, the master updates the values of N and r in the following manner:

For the slave just polled:

(1) $$N = N + r\tau - x$$

[1] A rates distribution is feasible if rates are non-negative, the aggregate rate is not greater than one, and no unit receives a higher rate than required.

$$r = \begin{cases} \alpha r + (1-\alpha)\frac{x}{T}; & x < M \\ \alpha r + (1-\alpha)\frac{x}{T} + \delta; & x = M \end{cases} \quad (2)$$

For other slaves:

$$N = N + r\tau \quad (3)$$

where τ is the time elapsed since the last update, x is the amount of data exchanged during the poll phase, T is the total time elapsed since the last poll of the same slave, α is a parameter used to smooth the rate estimation and δ is a parameter used to probe for more bandwidth. Note that x is the actual amount of data exchanged, which may be less than or equal to M, depending upon the number of packets in the slave's queue. Since N is an estimate of the slave's queue length and r is an estimate of the rate at which traffic is generated, N is increased at the rate of r (as in Eqs. 1 and 3). Also, when a slave is polled, N is decreased by the amount of data exchanged (Eq. 1).

After updating these values, the master determines the changes to be made to the active list. A slave is added or deleted from the active list depending upon whether its value of N is greater or smaller than a threshold. The value of this threshold is the minimum amount of data that the master would like the slave to have in order to poll it. We choose a value equal to the payload of a DH5 packet for the threshold since a 5–slot Bluetooth packet incurs least overhead. Thus, a slave is present in the active list if the master's estimate of the value of N for the slave is greater than the threshold. This makes the simple Round Robin polling strategy adaptive to traffic and enables it to utilize bandwidth more efficiently, particularly when slaves have different rates of traffic. The maximum amount of data that can be exchanged at each poll, M, is also set equal to the threshold.

The master now goes to the next slave according to the Round Robin ordering of slaves. If the slave is present in the active list, it is polled. Else, the procedure is repeated for the next slave in the Round Robin ordering.

Also, note that if the amount of data sent by the slave x is equal to M, r is increased by a small amount, δ. This is basically an attempt by the slave to probe for more bandwidth if it is able to send data at the present rate. The usefulness of this increase is evident in the proof of fairness in the next section. The value of δ chosen is

0.15 and that of α is 0.65. We also discuss the rationale behind choosing these values in the proof of fairness.

If traffic flows in both directions, i.e., from slaves to master and in the reverse direction, x is the average of the data exchanged in the two directions, r refers to the average of the rate-estimations of the two directions and N refers to the average of the queue length estimates from the slave to the master and vice versa.

Another advantage of such a scheme is that it may allow the master to go into a power–saving mode if it realizes that no slave has sufficient packets to send, i.e., if N is smaller than the threshold for all slaves. Though we do not explore this option in this paper, it may be useful since Bluetooth devices are expected to work in power–constrained environments.

To improve the working of the algorithm, we add a heuristic to it. The maximum number of polling cycles that a slave is not polled is bounded. If a slave generates a large burst of data occasionally and then does not generate any data for a long time, the value of r for the slave may be very low. This may cause the value of N for the slave to be lower than the threshold for a long time. By limiting the maximum number of cycles missed by the slave, we make sure that such a behavior of the slave does not lead to its starvation.

3.3 Proof of Fairness

We now prove that the algorithm is max-min fair. Let us introduce the following notation:

S : number of slave units in the piconet;

g_i : rate–demand of the i–th unit;

$\overline{\eta_i}$: rate achieved by the i–th unit;

$\overline{r_i}$: rate–estimation of the i–th unit (as defined in Eq. 2);

where $\overline{\eta_i}$ and $\overline{r_i}$ are average values.

Slave unit i is referred to as “satisfied”, if it achieves it rate demand, i.e., $\overline{\eta_i} = g_i$; else, the slave unit is referred to as “unsatisfied”. Also, in the proof that follows, “slot” refers to “Bluetooth slot”; “unit” and “slave unit” may be used interchangeably.

If there is one slave unit in a piconet, then it will always get polled and hence, the algorithm is fair. We prove the fairness when there are two or more slave units.

We first make the following observations:

a) If a unit has a rate-estimation, r >= 0.25, it will never achieve a lesser rate than another unit with the same rate demand.

r is an estimation of the average number of slots of traffic that a master-slave pair will generate per slot in each direction. Thus, a rate of 0.25 means that a master-slave pair generates, on the average, 5 slots of traffic in each direction in every 20 slots. Suppose a piconet has two slaves, and the first has a rate-estimation, r >= 0.25, then the first slave will be polled at least once in every 20 slots, i.e., will get at least ten polling slots out of every twenty, regardless of the r of the other slave. (Since N increases at the rate of r, N will increase by at least 0.25*20 = 5, which is equal to the threshold; thus, the slave will enter into the "active list" in 20 slots). Thus, it will never be treated unfairly with respect to the other slave. It is easy to see that this property would be true if there were more than two slaves (two slaves is the worst case).

b) For δ >= 0.1 and α >= 0.6, an unsaturated slave will tend to a rate-estimation of at least 0.25.

For an unsaturated slave, the second part of Eq. 2 (when x = M) is always used for updating the rate. Thus, if r_i is the i^{th} rate-estimation:

$$r_{n+1} = \alpha r_n + (1-\alpha)\frac{M}{T_n} + \delta$$

This leads to (as n becomes very large):

$$r = (1-\alpha)M\sum_{k=0}^{\infty}\frac{\alpha^{n-k}}{T_k} + \frac{\delta}{1-\alpha} >= \frac{\delta}{1-\alpha}$$

Thus, for δ >= 0.1 and α >= 0.6, for any value of T, the value of r tends to at least 0.25.

c) As long as there is an unsaturated unit, the utilization of the system capacity is 1 (for δ >= 0.15 and α >= 0.65).

Consider a piconet consisting of seven slave units, in which the first unit, $unit_1$ is unsaturated. From (a) and (b), $unit_1$ will be treated fairly with respect to any other unit; this means that it will be polled at least once for each time the other slaves are polled. The value of T (as in Eq. 2) for $unit_1$ is thus, at most, 70. For this value of T and

for $\delta = 0.15$ and $\alpha = 0.65$, r for $unit_1$ tends to at least 0.5. A value of r = 0.5 for a slave unit means that it can be polled all the time. Thus, the system capacity is totally utilized. If there were less than 7 slave units, the value of T would be smaller (than 70), and r would tend to a higher value (than 0.5).

We choose values of and α to satisfy the above properties, i.e., $\delta = 0.15$ and $\alpha = 0.65$.

To Prove:

(i) Units with the same rate–demand achieve the same average rate: $\overline{g_i} = \overline{g_j} \Rightarrow \overline{\eta_i} = \overline{\eta_j}$;

We prove this by contradiction. Suppose there are two units, $unit_1$ and $unit_2$ with rate demands g_1 and g_2 respectively, such that $\overline{g_1} = \overline{g_2}$. Also, suppose one unit achieves a higher average rate than the other, $\overline{\eta_1} > \overline{\eta_2}$.

Now, $unit_2$ does not achieve its rate-demand (since $\overline{\eta_1} > \overline{\eta_2}$). $unit_1$ may or may not achieve its rate demand. From property (b), $unit_2$ will always tend to a value at least equal to 0.25, since it is an unsaturated slave. Using property (a), this implies that $\overline{\eta_2}$ cannot be less than $\overline{\eta_1}$. This is a contradiction.

(ii) Units with a higher rate–demand achieve an average rate at least equal to that achieved by units with a lower rate–demand: $\overline{g_i} > \overline{g_j} \Rightarrow \overline{\eta_i} \geq \overline{\eta_j}$.
This can be proved by contradiction in the same manner as in part (i).

Now, without loss of generality, let us partition the slave units into two sets, S1 and S2, in such a way that units in S1 are satisfied, while units in S2 are not.

- If the set S2 is empty, than all the units achieve their rate–demand and the system is fair.
- If the set S2 is not empty, then using statements (i) and (ii), all units share the bandwidth in a fair manner. Moreover, since S2 contains at least one unit, the total system capacity is utilized. Hence, it is not possible to increase the rate of a unit in S2 without decreasing the rate of some other unit.

4. Experiments and Results

In this section, we perform simulations in a piconet, varying the number of slaves and the traffic generated by each slave. We show that, under various conditions, our polling scheme achieves max-min fairness among the slaves and is able to adapt to changing traffic. We also show the algorithm to be highly efficient in terms of bandwidth usage in the piconet.

In the experiments, we specify the "rate of a slave", which refers to the sum of the rates at which a slave generates data for a master and the master generates data for a slave. Moreover, unless otherwise mentioned, we assume that the traffic rate from the slave to the master is equal to that from the master to the slave. Thus, a slave having a rate of 0.4 means that the slave generates data at the rate of 0.2 Bluetooth slots per slot and the master also has a rate of 0.2 towards the slave. As the experiments of this section show, the algorithm works well even if these two rates are not the same.

A Bluetooth simulator written in C++ is used in the experiments. The simulator models the Bluetooth baseband and L2CAP layers and enables the creation of piconets. All traffic generated is uniform. Each experiment is run for a system time of 32 sec. In the figures, "BW" stands for bandiwdth.

In the first experiment, we consider a piconet with seven slaves, in which one slave, slave1 varies its traffic rate and the other slaves have a traffic rate of 0.1. Fig 1 shows the fraction of the bandwidth obtained by each slave as the traffic rate of slave1 is varied. The figure also shows the fair share of slave1. It can be seen that the fraction of bandwidth obtained by slave1 shows a very close matching with its fair share. Also, the bandwidth obtained by each other slaves is 0.1, which is the same as the fair share.

Fig. 1: Sharing of bandwidth between slaves as traffic rate of one slave is varied

In the second experiment, we vary the number of slaves in a piconet. Each slave has a traffic rate of 0.5. Fig 2 shows the fair share and fraction of bandwidth of each slave as the number of slaves is varied. Again, one can see the fair sharing of bandwidth. Fig 2 also shows the efficiency in the piconet, defined as the total fraction of the piconet bandwidth used for transmitting data (as opposed to NULL and POLL packets). The efficiency can be seen to be nearly equal to 1.

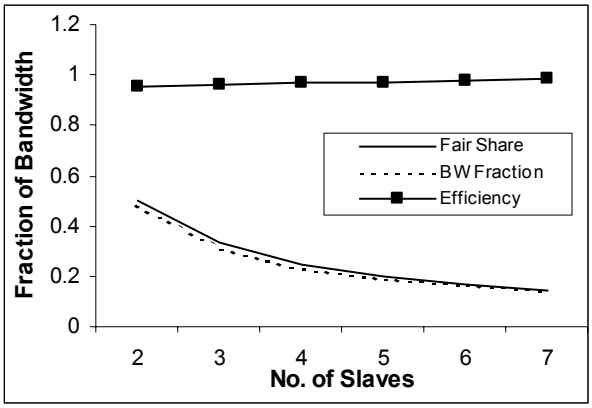

Fig. 2: Sharing of bandwidth between slaves as number of slaves is varied

In the next experiment, we show the effect of asymmetric traffic in a piconet in which the number of slaves is varied. Each slave generates data at the rate of 0.5, with the master-slave traffic rate being 0.4 and the slave-master traffic rate being 0.1. This scenario is very similar to that in the second experiment, except that the rates are asymmetric. Fig 3 shows bandwidth fraction of each slave in this experiment and also shows the bandwidth fraction obtained by each slave in the second experiment to compare with the case in which rates are symmetric. The bandwidth fraction obtained in the asymmetric case is slightly smaller than in the symmetric case, but this is expected since asymmetric traffic will lead to some wastage of slots (e.g., a NULL packet may have to be sent in one direction when data is being sent in the other). Also, the efficiencies in the symmetric (second experiment) and asymmetric cases are shown. Again, the efficiency is only slightly lower in the asymmetric case.

We now test the adaptivity of the algorithm, i.e., how quickly the algorithm adapts to changing rates. We consider a piconet consisting of 3 slaves, in which 2 slaves have a traffic rate of 0.2 each and one slave varies its traffic rate; we vary the traffic rate of this slave as time progresses: for the first 2.5 seconds, the slave's rate is 0.1, for the next 2.5 seconds, it is 0.5 and for the remaining time, it is 0.3. Fig. 4 shows the actual

rate estimation of the slave and its ideal value against time. The algorithm is able to adapt the rate to the fair share very quickly.

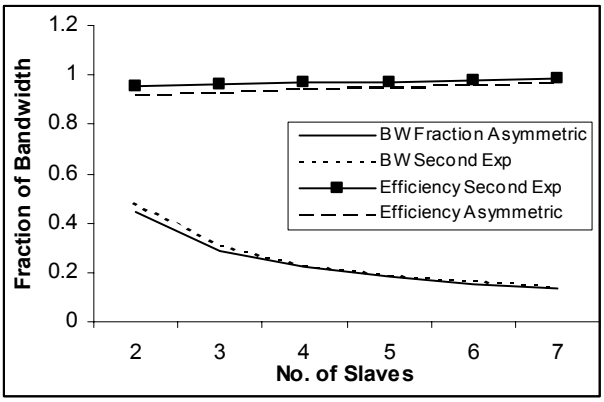

Fig. 3: Sharing of bandwidth when traffic is asymmetric

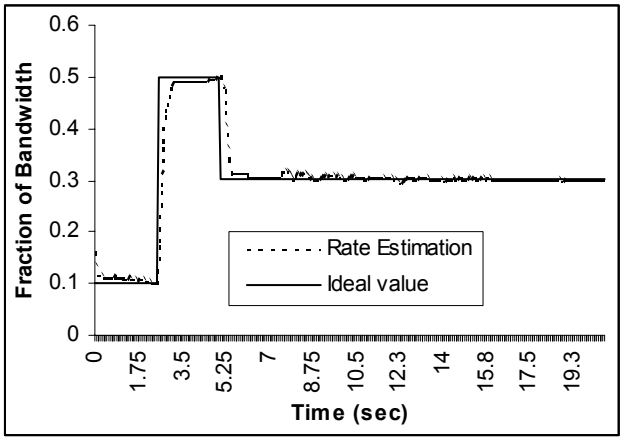

Fig. 4: Change in rate estimation as traffic rate changes

5. Conclusions

In this paper, we presented a traffic-adaptive and max-min fair polling algorithm for Bluetooth. We proved the fairness of the algorithm analytically and also verified it

experimentally. The experiments also show that the algorithm is able to adapt to changing traffic very quickly and achieves a high efficiency in terms of bandwidth usage of a piconet. In future, we would like to extend this algorithm to the case of a scatternet. In other words, the polling algorithm should be able to achieve fairness and traffic-adaptation between slaves and gateways. The algorithm would also need to schedule the presence of gateways in piconets appropriately.

References

1. Specifications of the Bluetooth System – Core vol.1 v1.1, www.bluetooth.com
2. J. Haartsen, Bluetooth – the universal radio interface for ad hoc wireless connectivity, Ericsson Review, n.3, 1998, pp. 110–117.
3. S. Garg, M. Kalia, R. Shorey, MAC Scheduling Policies and SAR policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System, MoMuc 99, pp. 384-386.
4. M. Kalia, D. Bansal, R. Shorey, Data Scheduling and SAR for Bluetooth MAC, IEEE VTC 2000-Spring Tokyo, pp. 717-720.
5. A. Capone, R. Kapoor, M. Gerla, Efficient Polling Schemes for Bluetooth Picocells, IEEE ICC 2001, Finland, June 2001.
6. A. Mayer, Y. Ofek, M. Yung, Approximating Max–Min fair Rates via Distributed Local Scheduling with Partial Information, In the proceedings of IEEE Infocom 1996.