

# Policies for Increasing Throughput and Decreasing Power Consumption in Bluetooth MAC

Indraneel Chakraborty, Abhishek Kashyap, Anupam Rastogi  
Huzur Saran, Rajeev Shorey and Apurva Kumar  
IBM India Research Laboratory,  
Block 1, Indian Institute of Technology,  
Hauz Khas, New Delhi 110016, India.  
Email: srajeev@in.ibm.com  
Phone: 91-11-6861100, Fax: 91-11-6861555.

**Abstract**— Bluetooth is a fast emerging standard for indoor pico-cellular wireless networks. Power is at a premium in typical Bluetooth devices like palmtops, PDAs, laptops and mobile phones. The Bluetooth standard defines various modes for reducing power consumption of the devices by reducing their transmission and reception activities. System throughput can be increased by keeping Bluetooth devices in low power mode in case of low data rates at those devices, by avoiding unnecessary polling packets. In this paper we propose policies for scheduling and switching of power modes of Bluetooth devices for increasing throughput and decreasing power consumption. All the proposed schemes, along with a policy where all the devices are always in *active* mode, and a policy with previous information of packet arrival times are implemented on a Bluetooth simulator for comparing their performance. Performance of the policies is compared for different traffic models and actual traffic traces. The policies are found to perform well in terms of power savings and throughput enhancement.

## I. INTRODUCTION

Mobile devices connected by wireless networks like Bluetooth [1], [2] typically have very limited battery power. It is thus required to design protocols and algorithms which facilitate conservation of power in mobile devices.

The chief sources of power consumption in mobile devices are the processing unit and the transceiver. The transceiver radio typically has three power modes : transmit, receive and standby. Radios consume more power during transmission than during reception, and consume the least power in standby mode. For example, the GEC Plessey DE6003 [8] 2.4 GHz radio requires 1.8 W, 0.6 W and 0.05 W of power in transmit, receive and standby modes respectively. Transceivers must therefore be in the standby mode for as much percentage of time as possible for lowest power consumption. For example, if a mobile device has no data to send and no device has data to send to that device, then the device should stay in standby mode as long as the period of *inactivity* lasts.

The time for which a transceiver (the transceiver of a mobile device) is in each mode is determined by the Medium Access Control discipline. In case of centrally driven systems like Bluetooth, this depends on the scheduling discipline at the *master*. In this paper, we propose scheduling disciplines to be used at the master of a Bluetooth network, which reduce power consumption in mobile devices while increasing the overall network throughput by polling the *slave* devices such that their transceivers can be in the standby mode for large times.

The paper is organized as follows. In Section II, the Bluetooth standard [1],[2] and the power modes of the connected devices defined by the standard are explained. Section III explains the criteria of switching of a slave between *active* and *sniff* modes. The proposed policies are explained in Section IV. In Section V, the simulation results are discussed. Section VI concludes the paper and Section VII discusses the future work.

## II. THE BLUETOOTH NETWORK

Bluetooth [1],[2] is a fast frequency hop (1600 hops/sec) Master driven Time Division Duplex (TDD) MAC wireless network. A Bluetooth *piconet* comprises of a master, and at most seven slaves connected to it, which receive and transmit data in the form of packets. The time is divided in to slots, with each slot of the duration of 625  $\mu$ seconds. There is a strict alternation of *slots* between the master and the slaves. The master can send packets to a slave only in even slots, while the slaves can send packets to the master only in the odd slots. Scheduling occurs in pairs of slots, and is carried out by the master. A slave can transmit a packet to the master only if it has received a packet from the master in the preceding time slot. All packets can be of sizes equal to one, three or five Bluetooth slot lengths.

The Bluetooth standard defines the *active* mode for a slave as the mode in which the slave is required to listen to the channel for master transmissions continuously. On

receiving a packet from the master, every *active* slave reads the destination slave address and packet length from the packet header. The addressed slave replies in the following slot. If the master has nothing to send during a slot when it polls a slave, it sends a *null* packet, and similarly the slave is always required to reply to the packet received from the master.

In the *sniff* mode (figure 1), the duty cycle of the listen activity of the slave is reduced. In this mode, the master can only start transmission in specified time slots, thus a slave does not need to listen to the channel continuously. The slave remains in standby mode for the remaining time slots. These *sniff* slots are placed at an interval  $T_{sniff}$ . A slave in *sniff* mode listens for a transmission every sniff period,  $T_{sniff}$  for a specified number of slots, which is  $N_{SNIFF\_ATTEMPT}$ .

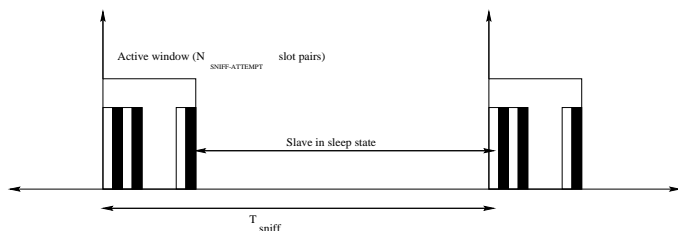


Fig. 1. The *sniff* mode in Bluetooth Standard

The other two modes defined in the Bluetooth standard are *hold* mode and the *park* mode. In the *hold* mode, a slave goes to the standby mode for some time and then switches back to *active* mode. In the *park* mode, a slave goes to standby mode for long period of time and switches back to *active* mode only after communicating with the master. No data can be transferred between master and slaves in *hold* and *park* mode. Therefore, using these two modes will increase the end-to-end packet delays to a very high value, thus making them unsuitable for use. Thus, only the *sniff* mode is suitable for power conservation along with satisfying the delay constraints.

### III. CRITERIA OF SWITCHING A SLAVE BETWEEN ACTIVE AND SNIFF MODES

According to the Bluetooth standard [1],[2], switching of the slave between *active* and *sniff* modes can be based on mutual agreement between master and slave, or the master can switch a slave to a power mode forcefully. Either node (the master or a slave) can request for switching of the slave's power mode.

In our policies, the request for switching of a slave from *active* to *sniff* mode can be rejected or accepted by the node which receives the request from the other node, whereas the switching of a slave from *sniff* to *active* mode cannot

be rejected by the node which receives the request, as their may be data enqueued at the request initiating node. A slave will be switched to *sniff* mode only if there is no data in the queues of both the nodes.

### IV. PROPOSED POLICIES

We propose scheduling policies which can be used by the master to schedule slaves so as to reduce power consumption and increase throughput in a Bluetooth piconet. The central idea of these policies is to switch a slave to *sniff* mode whenever the power overhead of switching the device state from *active* to *sniff* and back is overshadowed by the power saving achieved by keeping the device in *sniff* mode. For this, the master requires to predict the arrival time of the next burst of data packets for the slave. This is also required for setting the polling intervals ( $T_{sniff}$ ) for the slave in *sniff* mode.

To reduce the power consumption, when the master sends a packet, the slaves, which are not addressed, read the length of the packet from the header and move to standby mode for that period of time.

The following policies differ in the method of predicting the next arrival time. We compare the performance of these policies with the policy in which all the slaves are always in the *active* mode, and the master polls the slaves using Earliest Deadline First (EDF) scheduling (we call this policy as Always Active policy), and with the policy in which the scheduling is optimally done by using previous knowledge of packet arrival times (the Offline Optimum Policy). The latter policy is unimplementable practically, and is used for getting lower bounds on power consumption.

#### A. Mean Policy: MEAN

In this policy, the master and the slave store the mean of the last several inter-arrival times of data packet bursts. The assumption is that the next inter-arrival time is correlated to the inter-arrival times of previous packet bursts. The next expected inter-arrival time, as calculated from the mean of previous inter-arrival times on both the nodes (the master and the slave concerned), are compared and the minimum of the two is taken to be the polling interval in the *sniff* mode. As soon as the master or the slave receive more than two packets at their queues, the slave is switched back to *active* mode when the master polls the slave again.

#### B. Last Inter-Burst Time: LIBT

In this policy, the inter-arrival time between the last two packet bursts is used as the predicted value for the inter-arrival time between last burst and next burst. The assumption is that inter-arrival times are highly correlated with the

previous ones. As in the Mean Policy, the polling interval for the *sniff* mode is decided to be the minimum of the two expected inter-arrival times calculated at the two nodes (the master and the slave). The criteria for switching back to *active* mode is same as in the Mean Policy.

### C. Queue Status based Polling Interval: QSPI

In this policy, the present state of the queue at the connections' incoming packet buffer is used by the master for fixing the next polling interval. 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. In the *sniff* mode, a slave can be in two states, defined by the polling interval of the slave. The polling interval of a slave in state II is double that of the polling interval in state I. When a slave is switched to *sniff* mode, it is in state I. When the master polls a slave in state I, it doubles the polling interval if it finds no data in the queue i.e., the slave moves to state II, if there is one packet in the queue, the slave remains in state I and if there are more than one packets, the slave is moved back to *active* mode. When a slave in state II is polled and there are two packets in the queue, the slave is moved back to state I, if there is one packet, then the slave remains in state II and for the case of more than two packets, the slave is switched back to *active* mode.

## V. SIMULATION RESULTS

A simulator based on the Berkeley network simulator (ns) [12] was developed for the Bluetooth network. Various traffic models and traces from actual runs were used for comparing the performance of the proposed policies. The following types of traffics were imposed at different Bluetooth slaves.

1. FTP running over TCP Traffic : Trace from <http://ee.lbl.gov>
2. HTTP running over TCP Traffic : Trace generated using Network Simulator, Berkeley.
3. REALPLAY Audio (16 Kbps) traffic trace collected from an actual run.

The values of the parameters were taken to be as follows

:

- Power consumed by device in standby mode : 0.05 units per slot
- Power consumed by device in standby mode : 0.5 units per slot
- Power consumed by device in standby mode : 1 unit per slot
- Overhead of switching states :  $2 \times P_{TRANSMIT} + 2 \times P_{RECEIVE} = 3$  power units

The observations of power consumption for all the policies have been normalized with respect to the Offline Opti-

um Policy, as it gives a lower bound on power consumed in the devices. The simulation was run for 100,000 Bluetooth slots. The graphs showing the power consumed by different policies normalized with respect to the offline optimum are shown in the figures below. The Offline Policy has prior information about data arrival and schedules accordingly.

The ratio of control packets and *null* packets [1] (extra packets) with respect to the total packets sent in a bluetooth piconet has been shown in Table 1. The piconet has connections including the traffics types mentioned above. *Control Packets* are used for the decision of link parameters like power mode, polling interval etc. *null* packets are sent when a node has no data to send while polling. This ratio is a better metric than simple throughput for our model because low power modes can only be used in case the traffic rate is not high. If throughput had been shown as our metric the low values of throughput because of low traffic would have led to a misleading comparison. We will call this Ratio as Extra Packets to Total Packets Ratio (EPTPR). The lower the value EPTPR the better is the policy with regards to link usage. If EPTPR is low then the bandwidth utilization for data transfer (*goodput*) is also high as less control packets are sent. This translates into an efficient usage of the throughput of the connection.

EPTPR should be compared with respect to the case when all the slaves are in *active* mode (we do not use any low power mode in this case). We will be calling the policy in which the slaves can only be in the *active* mode as the Always Active policy, and we have used Earliest Deadline First (EDF) scheduling in this policy.

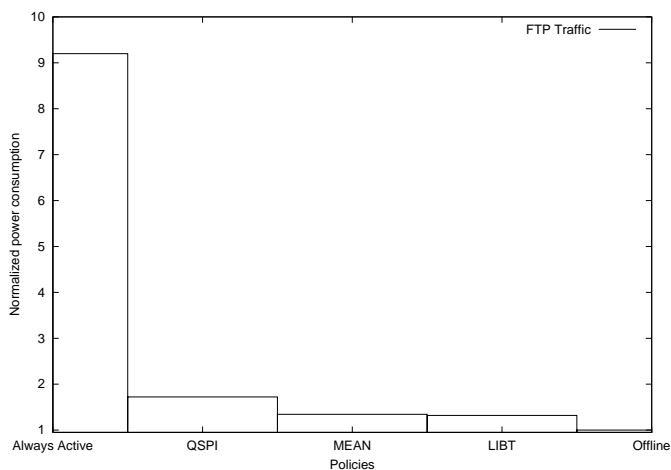


Fig. 2. Power Consumption for a Slave with FTP Traffic

Figures 2, 3 and 4 show the performance of various policies with respect to the power consumption of the device. All the policies are compared for equal number of packet transfers. Always Active Policy clearly wastes a lot of

Policies	EPTPR
AAM	0.79
QSPI	0.70
MEAN	0.33
LIBT	0.34
Offline	0.25

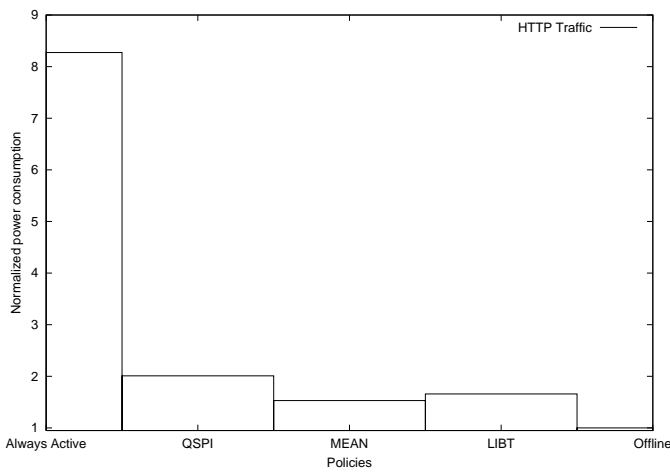


Fig. 3. Power Consumption for a Slave with HTTP traffic

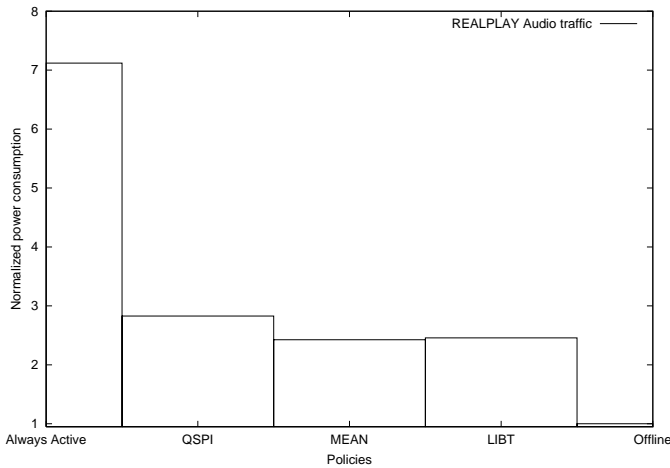


Fig. 4. Power Consumption for a Slave with REALPLAY Audio Traffic

power because Always Active Policy keeps sending idle packets if no data is enqueued. If data is available, packets containing data are sent in place of *null* packets. Thus Always Active Policy is suitable only in the case of very high traffic rate. However even the other policies work well in that case also. So Always Active Policy has no advantage over other policies proposed in this paper.

As seen in Table 1, Always Active policy has a high EPTPR (bandwidth share in which actual data is sent to service connections) as well. This is because in this policy most of the bandwidth is wasted for sending *null* packets for polling when the data queues are empty.

Table 1. Ratio of Extra Packets to the Total Packets sent for various Policies

The power consumed in a device for QSPI is approximately 170% of the Offline Optimum Policy for FTP traffic and 200% of Offline Optimum Policy for HTTP traffic, as observed from Figures 2, 3. The power consumption for REALPLAY Audio is 300% of the Offline Optimum Policy, as observed in Figure 4. The corresponding figures are much higher for Always Active Policy. Also, the EPTPR value (see Table 1) for QSPI is lower than Always Active Policy as it sends less number of *null* packets.

Mean Policy also works well for FTP and HTTP traffic at approximately 140% and 160% power consumption as compared to the Offline Optimum Policy (see Figures 2, 3). The performance for REALPLAY Audio (Figure 4) is not as good as for FTP/HTTP because the traffic is more in the case of REALPLAY Audio. MEAN works very well as far as EPTPR is concerned (see Table 1). Overall, MEAN works better than Always Active and QSPI both in terms of power and bandwidth wastage (EPTPR).

LIBT has higher efficiency than QSPI and similar to MEAN in terms of Power Consumption for FTP/HTTP. This is because TCP, which is at the transport layer for the aforesaid applications, has a high correlation between the immediate preceding packet and the next packet, as the calculation of round-trip-time and window size for the traffic depends on the feedback obtained from the network. However for UDP traffic like Real Audio which is an open system with no feedback, the arrival times of bursts have no correlation. Hence LIBT does not perform well for Real Audio Traffic (See Figures 2, 3 and 4).

LIBT also performs well in terms of EPTPR as it calculates the polling interval properly for applications running over TCP layer (see Table 1). Thus LIBT works well for traffic types with correlation between packets.

## VI. CONCLUSION

In this paper we have proposed policies for scheduling and switching states of Bluetooth devices for increasing throughput (i.e., decreasing the bandwidth wasted in sending non-data packets) and decreasing power consumption. All the proposed schemes, along with a policy where all the devices are always in *active* mode, and a policy with previous information of packet arrival times (for getting

lower bounds on power) were compared for different traffic models and real traffic traces. The policies were found to perform well in terms of power savings and throughput enhancement. The proposed policies decrease power consumption to nearly 16% to 20% of the Always Active policy while showing a decrease of 50% to 75% in the ratio of extra packets to the total packets sent (EPTPR).

## VII. FUTURE WORK

The effect of various scheduling policies over end-to-end packet delay and jitter has to be considered. A scheduling policy which optimizes the power consumption and reduces the delays to an acceptable value should be devised. The policy should be easily implementable in hardware. The scheduling policy should also take advantage of correlation between data bursts in the traffic. However, if no correlation is present, then the policy should avoid sending extra packets in the connection while trying adjust to fluctuating traffic. The authors are working over these issues.

## REFERENCES

- [1] Bluetooth Special Interest Group. <http://www.bluetooth.com>
- [2] <http://www.bluetooth.net>
- [3] S.Keshav, An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network, Addison Wesley, 1999.
- [4] W. Mangione-Smith and P.S. Ghang, "A low power medium access control protocol for portable multi-media systems", *3rd International Workshop on Mobile Multimedia Communications*, September 25-27, 1996
- [5] M. Stemm and P. Gauthier and D.Harada, "Reducing power consumption of network interfaces in hand-held devices", *3rd International Workshop on Mobile Multimedia Communications*, September 25-27, 1996
- [6] John M. Rulnick and N. Bambos, "Mobile power management for maximum battery life in wireless communication networks", *Proc. of IEEE INFOCOM*, 1996.
- [7] Jyh-Cheng Chen, Krishna M. Sivalingam, Prathima Agrawal, Shalinee Kishore, "A Comparison of MAC Protocols for Wireless Local Networks Based on Battery Power Consumption", *Proc. of IEEE INFOCOM*, 1998.
- [8] [http://www.networks.digital.com/npb/html/products\\_guide/roamwir2.html](http://www.networks.digital.com/npb/html/products_guide/roamwir2.html), Jan 14, 1998.
- [9] Hong, J.; Tan, X.; Towsley, D.; "A performance analysis of minimum laxity and earliest deadline scheduling in a real-time system", *IEEE Transactions on Computers*, Volume 38, Issue 12, Dec, 1989
- [10] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication" *Proc. of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1998.
- [11] Chetto, H.; Chetto, M.; "Some results of the Earliest Deadline Scheduling Algorithm", *IEEE Transactions on Software Engineering*, Volume 15, Issue 10, Oct, 1989
- [12] <http://www-mash.cs.s.berkeley.edu/ns/>