Round Robin with Look Ahead: A New Scheduling Algorithm for Bluetooth

Daqing Yang, Gouri Nair, Balaji Sivaramakrishnan, Harishkumar Jayakumar and Arunabha Sen
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA
{dyang, ggn, sbv, harishkumar, asen}@asu.edu

Abstract

In this paper we propose two new Media Access Control (MAC) scheduling algorithms for Bluetooth whose objective is to achieve high channel utilization (throughput). Conventional scheduling policies such as Round Robin (RR) in Bluetooth environment results in wastage of slots and hence poor utilization of the network resources. As Bluetooth devices are designed to carry both voice and data, scheduling becomes a complex task, as slots are reserved for voice traf£c at periodic intervals and the data packets are allowed to have variable size. In this paper we view the MAC scheduling problem in Bluetooth as an online Bin Packing problem. The two scheduling policies being proposed in the paper, Look Ahead (LA) and Look Ahead Round Robin (LARR) can be viewed as online bin packing with lookahead. In this paper, we £rst analytically demonstrate that an optimal scheduling policy can have about 66% improvement in throughput over the Round Robin policy. Our extensive simulation shows that both LA and LARR achieves nearly 10% improvement in throughput over RR. As the computational complexity of LARR is lower than that of LA and also LARR avoids the possibility of starvation, we suggest the use of this algorithm over RR.

1 Introduction

The development of the Bluetooth protocol in late 1990s is the result of an effort to standardize a simple and inexpensive network protocol for *personal ad-hoc networks*, comprising of devices such as notebook computers, personal digital assistants, cellular phones, digital still and video cameras. Bluetooth uses an ad-hoc *piconet* structure that uses short range (10 meters) frequency hopping (1600hops/sec) to improve robustness of the link in the 2.4GHz ISM band. A piconet is an ad-hoc network formed by at most eight Bluetooth enabled devices, one of which acts as the *master* and the remaining ones as *slaves*. Bluetooth uses *Time Division Duplex* (TDD) scheme, where communication from *master-to-slave* and *slave-to-master* strictly alternates. The master of the piconet is responsible for *slot* scheduling for communication with the slaves. A *frame* comprises of a set of time-division duplex slots, with each Bluetooth packet occupying 1, 3 or 5 slots. Bluetooth supports both *voice* and *data* traf£c [3, 8]. The members of a piconet can use either *Synchronous Connection Oriented* (SCO) links or *Asynchronous Connectionless Links* (ACL) for communication between them. SCO links are used for voice communication, whereas ACLs are used for data communication. In £gure 1, the interleaving of SCO and ACL slots are shown for a piconet with four members. Since SCO slots carry voice traf£c, they occupy £xed slots assigned apriori by the Master. In £gure 1, between every SCO slot pair (one from master-to-slave and the other for slave-to-master communication) there are four slots that are used by the ACLs for data communication. The frame size is four in this scenario.

It may be observed that as the frame size is £xed, unless the slots in a frame are properly utilized for data communication using ACLs, the slots are wasted, thereby reducing the throughput of the network. Scheduling policy used for £lling up the slots of a frame by the ACLs determine to a large extent the utilization of the slots. Recognizing the important role of the scheduling policy in determining network throughput, several researchers in recent years have focussed their attention on development of more ef£cient scheduling algorithms [2, 5]. In this paper we propose a new scheduling policy that performs signi£cantly better than the Round Robin algorithm, widely used scheduling algorithm in Bluetooth today.

2 Bluetooth Scheduling and Online Bin Packing

Consider a piconet comprising of a master and two slaves. The master has two queues MQ_1 and MQ_2 where

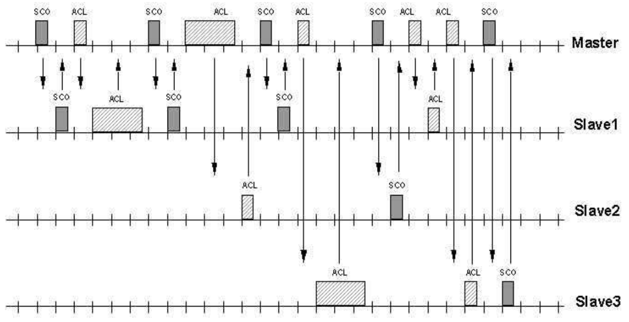

Figure 1. Interleaved SCO and ACL slots for communication between the Master and the Slaves

the packets to be delivered to two slaves are stored. Similarly the slaves S_1 and S_2, maintains two separate queues SQ_1 and SQ_2 where the packets to be delivered to the master are stored. As indicated earlier, the packets can have only three sizes 1, 3 and 5. Suppose the content of MQ_1 is (5, 5, 5, 5, 5) and the content of MQ_2 is (1, 1, 1, 1, 1). Similarly, the content of SQ_1 is (5, 5, 5, 5, 5) and the content of SQ_2 is (1, 1, 1, 1, 1). Since master-slave communication takes place in pairs, we can think of two combined master-slave queues $CMSQ_1$ and $CMSQ_2$ whose content will be the sum of the contents of MQ_1, SQ_1 and MQ_2, SQ_2 respectively. Thus the content of the hypothetical (or distributed) queues $CMSQ_1$ and $CMSQ_2$ will be (10, 10, 10, 10, 10) and (2, 2, 2, 2, 2) respectively. Suppose in our scenario the number of slots between every SCO slot pair (one from master-to-slave and the other for slave-to-master communication) is 10. These 10 slots can be used by the ACLs for data communication. It can easily be veri£ed that if Round Robin scheduling is used, it will take 10 frames to transfer all data between the master and the slaves. However, using a different scheduling policy all data could have been transferred using only 6 frames.

Bin Packing Problem: In a bin packing problem we have a set of objects $L = \{a_1, \ldots, a_n\}$ with a size $s(a_i)$ associated with each object a_i. We also have a set of bins with a £xed capacity B. The objective is to pack the objects $a_i, 1 \leq i \leq n$, into as few bins as possible, subject to the constraint that sum of the sizes of the objects placed in a bin does not exceed the bin capacity B.

Since the number of slots between every SCO slot pair is £xed (say 10), we can view this as a bin with capacity 10. The packets in the combined master-slave queues can be viewed a objects that we need to place within the bins (frames). A good scheduling algorithm would attempt to put the packets in as few frames as possible, because it will increase the network throughput.

In the *off-line* version of the bin packing problem, the packing algorithm has the complete knowledge of all the elements $\{a_1, \ldots, a_n\}$ and their sizes, and this knowledge can be used by the packing algorithm to ef£ciently pack the objects in the bins. However, in the *online* [7] version of the bin packing problem, the packing algorithm does not have the complete knowledge of all the elements $\{a_1, \ldots, a_n\}$ and their sizes. Instead, objects arrive in a sequence $a_1, a_2, \ldots, a_n$, and the scheduling algorithm has to place the $i - th$ object a_i in the bin as soon as it arrives. At this time the scheduling algorithm has already placed the objects $a_1, \ldots, a_{i-1}$ in some bins, which cannot be rearranged, and has no knowledge of the sizes of the objects $a_{i+1}, \ldots, a_n$ that are yet to arrive. The scheduling algorithm for Bluetooth can be viewed as online bin packing problem with a limited amount of future knowledge. This is so, beacuse if there are x slaves present in the piconet, the scheduling algorithm can examine the packets at the front of the queues $CMSQ_1, \ldots, CMS_x$ (*head-of-the-line (HOL) packets*), and decide which packet to place in the current frame.

Bin packing is known as an NP-complete problem [6]. There are a number of approximation algorithms for bin packing, of which the *First-Fit (FF)*, the *Best-Fit (BF)* and the *Next-Fit (NF)* are best known [1, 4]. The FF rule places each successive object into the £rst bin in the sequence $B_1, B_2, \ldots$, where it will £t. The NF rule £lls the bins in sequence, i.e., if an object a_i is placed in a bin B_j no object $a_k, k \geq i$ can be placed in any bin B_l with $l \leq j$. The NF scenario describes the Bluetooth scheduling appropriately, because the frames leave the master or the slave device and is not available for further packet placement at a later time.

The quality of the approximate solutions like FF, BF and NF, is usually measured by their worst case performance bound of the form $A(L) \leq \alpha OPT(L)$, where $A(L)$ and $OPT(L)$ are the number of bins required to pack the set of objects L by algorithm A. The bound α for FF and BF is 1.7 and for NF it is 2. It is well known that if the objects in the set are arranged into a *nonincreasing* order of the object size, the performance bound can be further improved. The corresponding versions of the FF, BF and NF techniques are known as FFD, BFD and NFD respectively. The bound α for FFD and BFD is 1.22 and for NF it is 1.691 [1].

In the Bluetooth scheduling environment, we can deploy a version of the NFD, where only the HOL packets are arranged in nonincreasing order before they are scheduled in frames. This will essentially be an online version of the NF algorithm with a limited amount of *lookahead*.

3 Potential for Improvement over Round Robin

For any scheduling algorithm A in Bluetooth, let list $L = (a_1, a_2, \cdots, a_n)$ denote the packets being scheduled by algorithm A, let $A(L)$ denote the number of bins used by packing list L, and let $OPT(L)$ denote the minimum number of bins required to pack list L, the asymptotic worst-case performance ratio is de£ned as

$$R_A = \lim_{n\to\infty} \max_{OPT(L)\,=\,n} \frac{A(L)}{OPT(L)}.$$

We have the following theorem:

Theorem 1 *For the Round Robin scheduling algorithm A in Bluetooth, $R_A = \frac{5}{3}$.*

Proof: Notice the Bluetooth scheduling algorithm has bin sizes 10, and item sizes can only be 2, 4, 6, 8, 10. Therefore, the Round Robin algorithm A guaranteed that any neighboring pair of bins packed item sizes at least 12. If $OPT(L) = n$, the packets need to be packed have size at most $10n$, therefore, $A(L) \leq \lceil \frac{20n}{12} \rceil$. This shows $R_A = \frac{5}{3}$.

Roughly speaking, this theorem shows that the Round Robin algorithm A compared with any scheduling algorithm B in Bluetooth, A is at most $\frac{5}{3}$ worse than B.

The following example shows that Round Robin algorithm compared with Round Robin with 1-look-ahead can be as bad as $\frac{5}{3}$.

Example 3.1
Queue 1: $L_1 = (10, 10, 10, 10, 10, \cdots)$
Queue 2: $L_2 = (2, 2, 2, 2, 2, \cdots)$

In this example, if the number of items in L_1 and L_2 is $5n$, then the Round Robin algorithm needs $10n$ bins, while Round Robin with 1-look-ahead needs only $6n$ bins.

4 Scheduling Algorithms

In this section, two new algorithms proposed in this work are presented.

4.1 Look Ahead

In this scheme, the master examines the HOL packets in the combined master-slave queues, $CMSQ_1, \ldots, CMSQ_x$ (assuming that there are x slaves) and selects the largest packet that will £t into the current frame. This policy attempts to mimic the NFD policy for bin packing discussed in the previous section.

This algorithm schedules packet as follows:

Step 1. Examine the (HOL) packets $CMSQ_1, \ldots, CMSQ_x$ for all the active slaves in the piconet.

Step 2. Prioritize slaves in order of non-increasing size of their HOL packets in CMSQ (the slave with the largest HOL packet has the highest priority).

Step 3. If possible, schedule the highest priority slave in the current frame.

Step 4. If HOL packet of the highest priority does not £t, search through HOL priority list for highest priority that will £t the current frame.

Step 5. If none £ts, wait for the start of a new frame and put the HOL packet in the new frame. Repeat steps 1 through 5.

4.2 Look Ahead Round Robin

There are two problems associated with the Look Ahead algorithm. First, every time a packet is scheduled, the HOL packet with the next largest size has to be identi£ed. If their are N slaves, using a priority queue data structure, it can be computed in $O(log\ N)$ amount of computation. However, in Bluetooth environment, this amount of computational overhead may be unacceptably high. Second, as the algorithm always attempts to schedule the packet with the largest size, it may unduly favor some slaves at the expense of others. In the extreme case, this may lead to *starvation* for some of the slaves.

The Look Ahead Round Robin (LARR) was developed to overcome the drawbacks of the Look Ahead algorithm. It combines the simplicity of the Round Robin algorithm with the *look-ahead* feature of the Look Ahead algorithm. To avoid the possibility of starvation and also to reduce the computational complexity, it no longer attempts to £nd the largest HOL packet. Instead the slaves are serviced in a round robin fashion and if the HOL packet of the combined master-slave queue (CMSQ) currently being examined £ts in the currently open frame, it is scheduled. The only difference from RR occurs when the current packet does not £t. In this situation, in order to ensure maximum slot usage, the algorithm *looks ahead* and attempts to schedule a packet from next slave in line to receive service in the round robin fashion.

The slaves are all serviced in the order in which they enter the active piconet. This algorithm schedules packet slots in the following way:

Step 1. The algorithm will £ll the current frame with the HOL packets in a Round Robin fashion, if the HOL packet can £t in the frame.

Step 2. If the HOL packet of the queue being serviced is too large to £t in the current frame, following Round Robin fashion, £nd the queue whose HOL packet will £t in the current frame. If no such queue exists, wait for the start of a new frame and schedule the HOL packet in the new frame. Repeat steps 1 and 2.

It may be noted that in order to schedule a packet from $CMSQ_i, 1 \leq i \leq x$, the Master needs to know the size of the HOL packet in the slaves. This information may be conveyed to the master by slaves using the unused £elds in the header of the immediately preceding packet.

5 Advantages of Look Ahead Round Robin

The Look Ahead Round Robin algorithm is based on the conventional Round Robin scheduling with a *look-ahead* feature.

In this algorithm if the HOL packet of the queue being serviced is too large to £t in the current frame, the algorithm will *look ahead* to see if their are some queues (which will be serviced in future) with HOL packets that will £t in the current frame. In *k-look-ahead* scheme the algorithm will examine the HOL packets of the next k queues, and £ll the current frame with the £rst HOL packet that can £t in the frame.

Clearly, the new algorithm maintains the computational simplicity Round Robin algorithm. It runs in linear time and uses constant space (with respect to the constant of k as in the *k-look-ahead*). Since the algorithm is based on Round Robin, it will avoid starvation.

In this section, we examine some interesting scheduling cases that shows Round-Robin with *k-look-ahead* may signi£cantly reduce the number of frames necessary to transmit all the packets. In all the following examples, we suppose the number of packets in each queue is n.

Example 5.1

Queue 1: $L_1 = (6, 6, 6, 6, 6, \cdots)$
Queue 2: $L_2 = (8, 8, 8, 8, 8, \cdots)$
Queue 3: $L_3 = (4, 4, 4, 4, 4, \cdots)$
Queue 4: $L_4 = (2, 2, 2, 2, 2, \cdots)$

In example 5.1, the Round Robin algorithm needs $3n$ frames, and the Round Robin with 1-look-ahead algorithm needs $2n$ frames, with performance comparison ratio $\frac{3}{2}$.

More generally,

Example 5.2

Queue 1: $L_1 = (6, 6, 6, 6, 6, \cdots)$
Queue 2: $L_2 = (6, 6, 6, 6, 6, \cdots)$
......
Queue 2k-1: $L_{2k-1} = (6, 6, 6, 6, 6, \cdots)$
Queue 2k: $L_{2k} = (8, 8, 8, 8, 8, \cdots)$
Queue 2k+1: $L_{2k+1} = (4, 4, 4, 4, 4, \cdots)$
Queue 2k+2: $L_{2k+2} = (4, 4, 4, 4, 4, \cdots)$
......
Queue 4k-1: $L_{4k-1} = (4, 4, 4, 4, 4, \cdots)$
Queue 4k: $L_{4k} = (2, 2, 2, 2, 2, \cdots)$

In example 5.2, the Round Robin algorithm needs $3kn$ frames, and the Round Robin with $(2k-1)$-look-ahead algorithm needs $2kn$ frames, with performance comparison ratio $\frac{3}{2}$.

Example 5.3

Queue 1: $L_1 = (6, 6, 6, 6, 6, \cdots)$
Queue 2: $L_2 = (6, 6, 6, 6, 6, \cdots)$
......
Queue 2k: $L_{2k} = (6, 6, 6, 6, 6, \cdots)$
Queue 2k+1: $L_{2k+1} = (8, 8, 8, 8, 8, \cdots)$
Queue 2k+2: $L_{2k+2} = (4, 4, 4, 4, 4, \cdots)$
Queue 2k+3: $L_{2k+3} = (4, 4, 4, 4, 4, \cdots)$
......
Queue 4k+1: $L_{4k+1} = (4, 4, 4, 4, 4, \cdots)$
Queue 4k+2: $L_{4k+1} = (2, 2, 2, 2, 2, \cdots)$

In example 5.3, the Round Robin algorithm needs $(3k+1)n$ frames, and the Round Robin with 2k-look-ahead needs $(2k+1)n$ frames, with performance comparison ratio $\frac{3k+1}{2k+1}$.

The above examples demonstrate that with Round Robin with *k-look-ahead* can reduce the number of frames by 50% over just Round Robin.

6 Simulation Environment

In the previous sections we analytically showed that the Round Robin with Look Ahead algorithm can signi£cantly reduce the number of frames necessary to carry out data transfer between the master and the slaves in a Bluetooth environment. However, this was established by choosing some speci£c situations, where the master and slave queues had packets of speci£c size. In general, the master and slave queues may not have packets with those speci£c sizes. We wanted to £nd out the performance of our new algorithm in a more general setting. We turned to simulation for this purpose.

Our simulation environment was developed in Java, using threads to create pseudo-parallel packet generation and scheduling. We assumed that packet arrival at the master and the slaves followed a Poisson process. The size of the packets were restricted to 1, 3, or 5 slots and was drawn from a uniform distribution. Packet generation was carried out in pairs, one for the master and other for the slave. Once generated, the packets are placed in the appropriate queues. The exact same set of packets were used for the evaluation of three different algorithms: (i) Round Robin, (ii) Look

Ahead and (iii) Look Ahead Round Robin. One packet generator thread was initialized for every slave included in the simulation. The algorithms were implemented as separate threads to promote pseudo-parallel processing. This more accurately captures the packet scheduling process with dynamically changing queue sizes.

The variables in our experiment were (i) the number of slaves in the piconet, and (ii) packet arrival rate. In the £rst set of simulation experiments, the packet arrival rate was £xed and the number of slaves in the piconet were varied. In the second set of experiments, the number of slaves in a piconet were £xed and the arrival rate varied. The arrival rate parameter λ was varied from 0.1 to 1.0 in increments of 0.1. The simulation was carried out for 1000 units of time. The number of frames needed to transfer the entire data set generated within this time period was computed for the three algorithms: Round Robin, Look Ahead and Look Ahead Round Robin. The number of frames needed to transfer data was largest for the Round Robin algorithm. Using the number of frames need by the Look Ahead and Look Ahead Round Robin algorithms, the savings in the number of frames by these two algorithms over the Round Robin algorithm was computed. With the number of slaves and the packet arrival rate £xed, we computed the savings of the LA and LARR algorithm over RR. Each experiment was repeated 500 times to compute average frame savings. We assumed that the number of slaves did not change during the execution of an experiment.

7 Simulation Results

The results of some of our simulation experiments are presented in £gures 2 to 4. In £gure 2, average frame savings by the Look Ahead and the Look Ahead Round Robin over Round Robin is plotted. It can be seen that LARR initially out performs LA, but LA eventually has greater savings, when the number of slaves is greater than 5. This is quite expected because the LA algorithm in this situation has a larger number of queues to examine to £ll the current frame.

In £gures 3, 4 and 5 average savings of the LA and LARR is plotted against the RR algorithm. It may be observed that average savings ranged from 5.5% (LA with 3 slaves) to 12% (LA with 7 slaves). The average savings for LA was less than that of the LARR when the number of slaves was small (3). When the number of slaves were increased LA performed at least as well as LARR and with seven slaves, it outperformed LARR. However, the average gain of the LARR algorithm over the RR algorithm was not signi£cantly less than that of the LA algorithm. With the number of slaves greater than 5 and arrival rate greater than 0.5 both LA and LARR algorithm had approximately 10% savings over the RR algorithm. We believe that this savings is quite substantial for a device like Bluetooth.

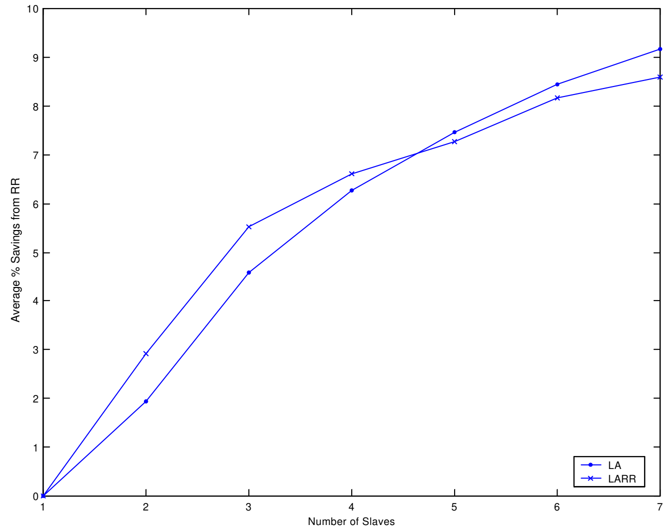

Figure 2. Variation of performance with number of slaves

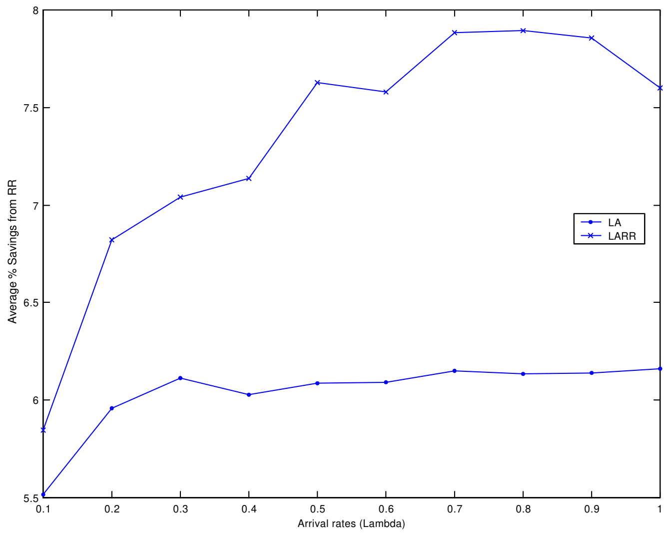

Figure 3. Variation of percentage savings with arrival rates, 3 slaves in piconet

8 Conclusion

In this paper we have presented two scheduling algorithms for Bluetooth. Through analysis and simulation we have shown that these algorithms perform signi£cantly better than the Round Robin algorithm used for scheduling in

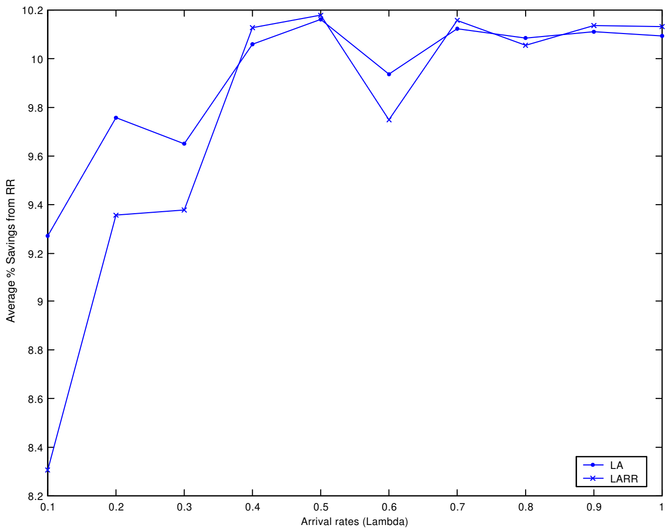

Figure 4. Variation of percentage savings with arrival rates, 5 slaves in piconet

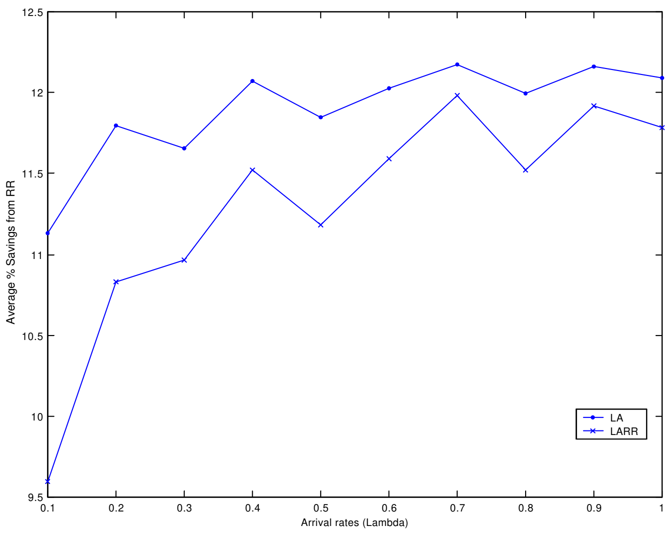

Figure 5. Variation of percentage savings with arrival rates, 7 slaves in piconet

Bluetooth today. Since the computational complexity of the Look Ahead Round Robin is smaller than that of the Look Ahead algorithm, we propose its use in Bluetooth. In addition, the LARR algorithm avoids starvation. Currently we are examining the ef£cacy of these two algorithms from the power consumption point of view.

References

[1] B.S. Baker and E.G. Coffman, A tight asymptotic bound for Next-Fit-Decreasing Bin Packing, in SIAM Journal on Alg. Disc. Meth, vol. 2, no. 2, June 1981 (pg. 147-152).

[2] D. Bansal, M. Kalia, R. Shorey, MAC Scheduling and SAR policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System. in IEEE Mobile Multimedia Conference (Momuc'99) (pg. 384-388), San Diego, California, November 99.

[3] "Bluetooth Special Interest Group",Speci£cation of the Bluetooth System","http: www.bluetooth.com", 2001.

[4] E. G. Coffman,Jr, M. R. Garey, D.S.Johnson, Approximation Algorithms For Bin Packing: A Survey. In Approximation Algorithms for NP-hard Problems, D. Hochbuam, Ed. PWS Publishing Company, 1997, Ch. 2.

[5] A. Das, A. Ghose, A. Razdan, H. Saran and R. Shorey, Enhancing performance of asynchronous data traf£c over the Bluetooth Wireless ad-hoc network. in Proceedings of IEEE INFOCOM'01, (pg. 591-600), 2001.

[6] M. R. Garey, D.S.Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco, 1979.

[7] E. F. Grove, Online bin packing with lookahead, In Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 430-436, San Francisco, California, 22-24 January 1995.

[8] B. A. Miller and C. Bisdikian, "Bluetooth Revealed", Prentice Hall, 2000.