Load-Adaptive Inter-Piconet Scheduling in Small-Scale Bluetooth Scatternets

Liron Har-Shai, Ronen Kofman, Adrian Segall, and Gil Zussman
Technion — Israel Institute of Technology

ABSTRACT

Bluetooth enables wireless communication via ad hoc networks. The basic topology (*piconet*) is a collection of slaves controlled by a master. A *scatternet* is a multihop network of piconets. We anticipate that most scatternets will be composed of only a few piconets. However, even in small scatternets, efficient data flow requires the design of inter-piconet scheduling algorithms. Thus, this article presents and evaluates a load adaptive scheduling algorithm tailored for small-scale scatternets. The main advantage of this algorithm is the use of the Bluetooth low-power *hold* mode, which allows greater flexibility than other low-power modes. A simulation model has been developed in order to evaluate the performance of the algorithm. We show that the results obtained by the model are very close to the analytic results. Then we evaluate the performance of various intra-piconet scheduling algorithms. Finally, we present simulation results regarding inter-piconet scheduling, and compare the proposed algorithm to algorithms using the *sniff* mode.

INTRODUCTION

Recently, much attention has been given to research and development of personal area networks (PANs). These networks comprise personal devices, such as cellular phones, PDAs, and laptops, in close proximity to each other. Bluetooth is a PAN technology that enables portable devices to connect and communicate wirelessly via short-range ad hoc networks [1]. The basic Bluetooth network topology (referred to as a *piconet*) is a collection of slave devices operating together with one master. A multihop network of piconets in which some of the devices are present in more than one piconet is referred to as a *scatternet* (Fig. 1). A device that is a member of more than one piconet (referred to as a *bridge*) must schedule its presence in all the piconets in which it is a member (it cannot be present in more than one piconet simultaneously).

In the Bluetooth specifications [1], the capacity allocation by the master to each link in its piconet is left open. The master schedules the traffic within a piconet and determines how bandwidth capacity is to be distributed among the slaves. Numerous intra-piconet scheduling algorithms have been proposed and evaluated via simulation (e.g., [2, 3, references therein]).

Efficient scatternet operation requires determining the link capacities that should be allocated in each piconet so that network performance is optimized [4]. The required link capacities should be allocated by *inter-piconet scheduling algorithms*. These algorithms schedule the presence of the bridges in different piconets and should be coordinated with intra-piconet scheduling algorithms. In the past, several inter-piconet scheduling algorithms have been proposed and evaluated (e.g., [5–7]).

Some of the proposed inter-piconet algorithms are intended for large-scale scatternets. Scheduling in such scatternets requires complex coordination mechanisms that enable bridges to establish recurring *rendezvous points* in which they can switch between piconets. Thus, scheduling algorithms in large-scale scatternets can be based on the Bluetooth low-power *sniff* mode, which provides recurring rendezvous points [5, 6]. It can also be based on new modes that require modifications to the Bluetooth specifications (e.g., the *jump* mode [8]).

Bluetooth, which is a PAN technology, and IEEE 802.11, which is a wireless LAN (WLAN) technology, are complementary technologies. Therefore, we anticipate that most scatternets will not be used to replace WLANs and will be composed of only a few piconets. In such small-scale scatternets, the coordination of the presence of bridges in different piconets is easier than in large-scale scatternets. In small-scale scatternets, every time a bridge leaves a piconet it can schedule its next rendezvous point instead of using periodic schedules.

Accordingly, in this article we propose and evaluate an inter-piconet scheduling algorithm tailored for small-scale scatternets. The algorithm is based on the low-power *hold* mode, which enables a device to leave a piconet for a short period and does not require modifications to the Bluetooth specifications.

Zussman *et al.* [9, 10] have shown that for a few intra-piconet scheduling regimes, a piconet can be modeled as a polling system (a polling system consists of several queues served by a single server according to a set of rules). However, as mentioned in [2], due to the special characteristics of Bluetooth medium access control, the operation model of most scheduling regimes differs from those of classical polling models. Thus, analysis of intra- and inter-piconet scheduling requires development of simulation models. Therefore, we have developed a simulation model of a scatternet.

In this article we show that the simulation results regarding intra-piconet scheduling are very close to the analytic results. Then we evaluate, via simulation, the performance of various intra-piconet scheduling regimes. Finally, we focus on scatternets composed of two piconets connected by a bridge that is a slave of the two masters. A load adaptive inter-piconet scheduling algorithm based on *hold* mode and an inter-piconet scheduling algorithm based on *sniff* mode are presented. The performance of the algorithms is compared via simulation, and it is shown that in the considered type of scatternets, the load adaptive algorithm usually yields better results than the algorithm using sniff mode.

The rest of the article is organized as follows. We give a brief introduction to Bluetooth technology and then present the simulation model. We present analytic and simulation results regarding intra-piconet scheduling, and validate the model. We present inter-piconet scheduling algorithms and evaluate the performance of these algorithms. We then summarize the main results and discuss possible extensions.

BLUETOOTH TECHNOLOGY

Bluetooth utilizes a short-range radio link that operates in the 2.4 GHz industrial, scientific, and medical (ISM) band. Since the radio link is based on frequency-hop spread spectrum, multiple channels (frequency hopping sequences) can coexist in the same wide band without interfering with each other. Two or more units sharing the same channel form a piconet, where one unit acts as a *master* controlling the communication in the piconet and the others act as slaves. A master can have up to seven slaves.

Bluetooth channels use a frequency-hop/time-division-duplex (FH/TDD) scheme. The channel is divided into 625-µs intervals called *slots*. The master-to-slave transmission starts in even-numbered slots, while the slave-to-master transmission starts in odd-numbered slots. Masters and slaves are allowed to send 1, 3, or 5-slot *packets* transmitted in consecutive slots. Packets can carry synchronous information (voice link) or asynchronous information (data link). Information can only be exchanged between a master and a slave (i.e. there is no direct communication between slaves). Note that we focus on networks in which only data links are used.

A slave is allowed to start transmission in a given slot if the master has addressed it in the preceding slot. The master addresses a slave by

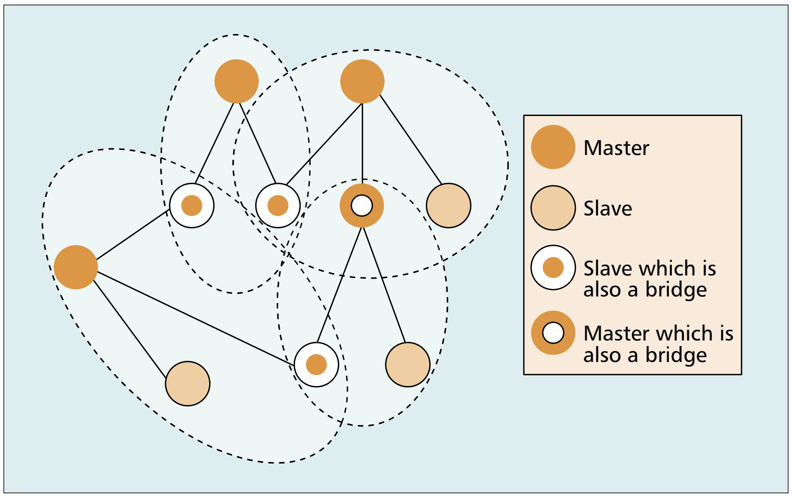


Figure 1. *An example of a Bluetooth scatternet.*

sending a data packet or a 1-slot *POLL packet*. The slave must respond by sending a data packet or a 1-slot *NULL packet* (if it has nothing to send). We shall refer to master-to-slave communication as *downlink* and to slave-to-master communication as *uplink*. The master schedules the traffic within a piconet according to an intra-piconet scheduling algorithm.

Multiple piconets in the same geographic area may form a scatternet. A unit can participate in two or more piconets on a time-sharing basis, and even change its role when moving from one piconet to another (we refer to such a unit as a bridge). For instance, a bridge can be a master in one piconet and a slave in another piconet (Fig. 1). However, a unit cannot be a master in more than one piconet. The presence of bridges in different piconets has to be controlled by an inter-piconet scheduling algorithm.

Two low-power modes defined in the Bluetooth specifications can be used to enable inter-piconet communication:

Hold mode: A slave in this mode is inactive in the piconet for an agreed period. At the end of the period the slave becomes active and can be addressed by the master. The period is called *hold timeout*, and its length is negotiated between the master and the slave.

Sniff mode: A slave in this mode is inactive in the piconet for agreed intervals (*sniff interval*). At the beginning of every interval it becomes active for a few slots (*sniff attempt*) in which the master can address it. If the master addresses it, it becomes active until a timeout (*sniff timeout*) expires. Otherwise, it becomes inactive until the beginning of the next interval. All the above-mentioned time periods are negotiated between the master and the slave.

When a bridge is inactive in a piconet, it can be active in a neighboring piconet; therefore, the hold and sniff modes can be used for inter-piconet communication. The main difference between the two modes is that the duration of the hold period is set every time the slave is placed in hold mode, whereas the parameters of the sniff mode are set once and can be used for a few intervals.

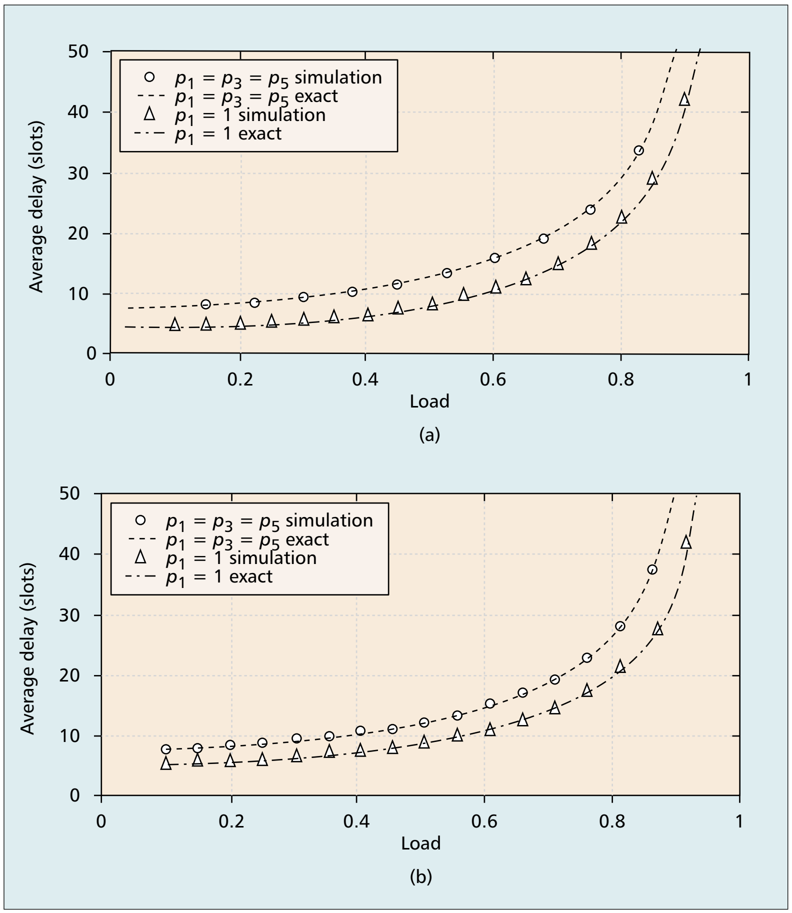


Figure 2. *The average delay, obtained via simulation, and exact mean delay in symmetrical piconets with four slaves in which* $p_1 = 1$ *and* $p_1 = p_3 = p_5$. *The piconets are operated according to a) the RR regime; b) the ERR regime.*

The Simulation Model and Definitions

In order to evaluate the scheduling algorithms, we have developed an OPNET simulation model of a scatternet. It implements most features required for the evaluation of scheduling algorithms. The Bluetooth specifications [1] do not require synchronization of masters' clocks; therefore, a guard time is wasted in the process of moving a bridge from one piconet to another. Since this time is usually negligible related to the intra- and inter-piconet delay, we have assumed that the different masters' clocks are synchronized. Moreover, it has been assumed that the channel is error-free.

We have implemented two different types of arrival processes:

- Packets generated according to a Poisson arrival process with user-defined packet length (1-, 3-, and 5-slot) distribution.
- Files generated according to a Poisson arrival process. The distribution of the file length is exponential. This arrival process is followed by a segmentation of the file to 1-, 3-, and 5-slot packets such that the generated packets are of maximal possible size.

In order to compare simulation results to the exact results obtained in [9, 10], we use a similar notation to the one used in those papers. We denote the number of slaves N. The probabilities of a packet length being 1, 3, or 5 slots are denoted by p_1, p_3, and p_5, respectively. Accordingly, the mean packet length is denoted by $L = p_1 + 3p_3 + 5p_5$. The arrival rate (packets per slot) to each queue is denoted λ. Finally, we define the delay as the time elapsed from packet generation to receipt of the last bit by the destination (the mean delay is denoted D).

Intra-Piconet Scheduling

Intra- and inter-piconet scheduling algorithms should be coordinated in order to achieve efficient scatternet communication. For example, while a bridge is present in a piconet, the intra-piconet scheduler might wish to poll it more frequently than other slaves. Thus, the evaluation of inter-piconet scheduling algorithms requires implementing intra-piconet algorithms.

In this section we present some of the intra-piconet scheduling algorithms implemented in the simulation model. In order to verify the performance of the model, we compare analytic results to simulation results. In all cases we have checked, the simulation and analytic results have been very close. We present analytic and simulation results for two different intra-piconet algorithms. Finally, the performance of the different algorithms is compared.

Scheduling Algorithms

Several intra-piconet scheduling algorithms have been proposed in the past (e.g., [3, references therein]). We have implemented several algorithms in which the master communicates with the slaves according to a fixed cyclic order. These algorithms include:

- *Round Robin* (RR) — At most a single packet is sent in each direction every time a slave is served.
- *Exhaustive Round Robin* (ERR) — The master switches to the next slave only when both the downlink (master-to-slave) and uplink (slave-to-master) queues are empty.
- *Slave Exhaustive Round Robin* (SERR) — The master switches to the next slave only when the uplink queue is empty (i.e., the slave responds with a NULL packet).

We have also implemented the following algorithms in which the order of service is not fixed:

- *Longest Queue* (LQ) — Before the master sends a packet, it checks the lengths of the downlink queues. It polls the slave with the longest queue.
- *Priority Round Robin* — The master polls the slaves according to their priorities.

A Comparison of Analytic and Simulation Results

In [10] it was shown that a piconet operated according to the RR regime is equivalent to a *1-limited polling system*. Accordingly, in a symmetrical piconet (in which the arrival rates to all queues are equal) with a Poisson arrival process and no interslave traffic, the mean delay (in slots) is

$$D = L + \frac{1+N(1+2\lambda(p_3+6p_5-1))}{1-2N\lambda L} - 1. \quad (1)$$

The load in such a piconet is defined as $2N\lambda L$.

In [9] it was shown that a piconet with only uplink traffic, in which the master operates according to an ERR regime, is equivalent to an *exhaustive polling system*. Accordingly, in a symmetrical piconet with Poisson arrival process, the mean delay (in slots) is

$$D = L + \frac{N(1+4\lambda(p_3+3p_5))}{1-N\lambda(L+1)}. \quad (2)$$

The load in such a piconet is defined as $N\lambda(L + 1)$.

Figure 2 compares analytic and simulation results for various values of load. The considered piconets include four slaves and are operated according to the RR and ERR scheduling regimes (when the piconet is operated according to the ERR regime there is only uplink traffic). The packets are generated according to a symmetrical Poisson arrival process, and there is no interslave traffic. The figures present the delay when all packets are 1 slot long and when the packet length distribution is uniform (i.e., p_1, p_3, and p_5 are equal). For each load value the results have been computed after at least 230,000 slots.

PERFORMANCE EVALUATION

We now evaluate the performance of the scheduling schemes, mentioned above, in two different scenarios. In the first scenario (based on the scenario described in [2]) there is no interslave traffic. The considered piconet includes seven slaves and files are generated at each of the 14 uplink and downlink queues according to a Poisson arrival process. The length of the files is exponentially distributed with an average file length of eight packets. The arrival process is followed by segmentation of the file to 1-, 3-, and 5-slot packets. In the second scenario, we have considered interslave traffic (routed through the master). Files are generated only at the uplink queues. The destination of a file originating at a slave can be any other slave (equal probability for each slave). The master is used as a relay and does not generate traffic. The number of slaves, arrival process, file length distribution, and segmentation process are as described above.

Figure 3 describes the average delay curves of the different schemes in these two scenarios. When there is no interslave traffic, the LQ regime provides the best performance for all realistic load values. The ERR regime also performs well up to a load of about 0.8. From that point on, the RR regime performs better. We note that the main drawback of using the LQ regime is that the channel can be captured by a few links. A similar problem occurs in the ERR regime. A few solutions to this problem have been suggested in the past (e.g., [2]). On the other hand, when there is inter-slave traffic, the RR regime and the ERR regime perform better than the other regimes. However, for low values of load all the regimes have similar performance.

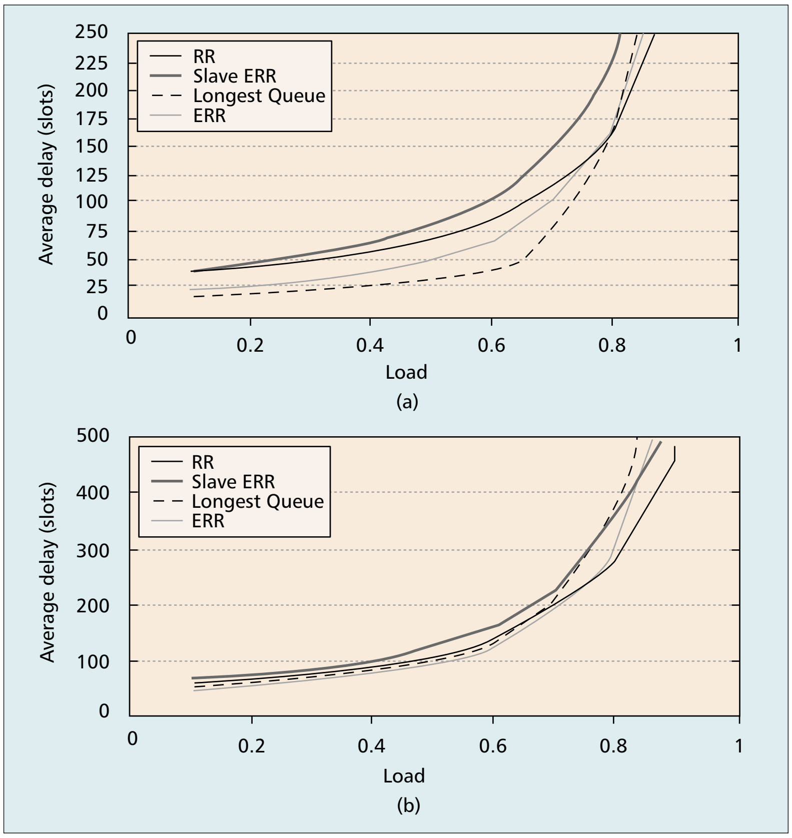


Figure 3. *Average delay (in slots) of different scheduling algorithms in piconets with seven slaves in which files with average length of 8 slots, arrive to the queues according to a symmetrical Poisson arrival process. The piconets include a) no interslave traffic; b) interslave traffic in which the distribution of the files' destinations is uniform.*

INTER-PICONET SCHEDULING

In this section we propose a *Load Adaptive Algorithm* (LAA) for inter-piconet scheduling intended for small-scale scatternets. This algorithm utilizes the hold mode, and its implementation does not require modifications to the Bluetooth specifications. Since a few scheduling algorithms presented in the past utilize the sniff mode (e.g., [5, 6]), we also describe an algorithm utilizing this mode. In the next section this algorithm will serve as a benchmark for evaluating the LAA. We note that scatternets where masters are also bridges may result in poor bandwidth utilization [4]. Thus, we focus on bridges that are slaves of two masters.

THE LOAD ADAPTIVE ALGORITHM

The LAA manages the scheduling mechanism of the bridge. It determines the duration of bridge activity in the different piconets such that the delay incurred by packets requiring inter-piconet routing is reduced. The algorithm adapts to varying values of load by using information regarding the bridge's queues to the different masters.

In order to determine the time instants when the bridge should switch piconets, the algorithm takes into account a few decision variables and parameters, described below.

Idle state (IS): Time wasted by the bridge affects the delay of packets requiring inter-piconet routing. Thus, whenever the connection

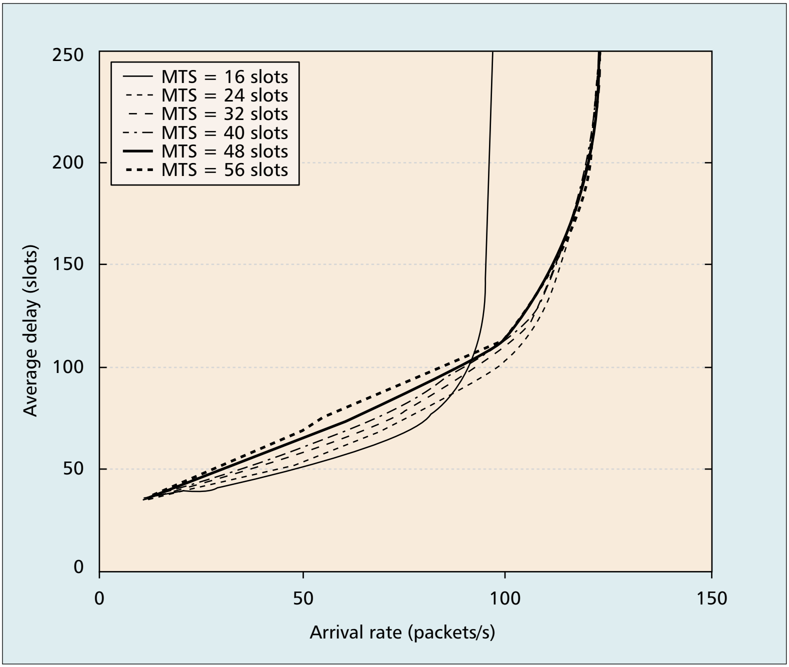


Figure 4. *LAA performance: average delay for different values of MTS as a function of the arrival rate.*

```
1  if (time in piconet A > MTS) or
   (TC expired and
   (queue size to piconet B > MQS or IS))
2      set TC = min (β * queue size to piconet B,
       MTS)
3      switch to piconet B
```

is exhausted, the bridge should try to switch piconets. The bridge enters IS if the queue to the current master is empty and it receives a NULL (non-data) packet.

Max queue size (MQS): Since the traffic can be asymmetric or bursty, the bridge uses the sizes of the queues intended for the other piconets in order to decide whether it should leave before the connection is exhausted. MQS is the parameter used in order to make the decision. If the queue size is larger than MQS, the bridge should try to switch piconets.

Time commitment (TC): The bridge sends this variable before a switch; it indicates the minimum interval the bridge will spend outside the piconet. It is calculated according to the sizes of the bridge's queue to the other piconet. It allows a master not to address the bridge throughout the interval and to readdress the bridge once TC expires.

Predictability factor (β): TC is calculated according to the outgoing queue size. The number of slots required to exhaust the outgoing queue depends on the nature of the traffic originating from the other piconet. β is used in order to estimate the average packet length of this traffic and compute the value of TC.

Max time-share (MTS): In some cases, the traffic rate intended for one of the piconets might be low. In such cases, postponing the switch until the queue size is bigger than MQS may cause a long delay of the packets intended for that piconet. In contrast, in cases of heavy traffic the queue sizes may be huge, and therefore TC derived from them may be too long. Thus, the maximum time a bridge spends in a piconet has to be bounded. We refer to this bound as MTS.

In this article we focus on bridges that connect two masters. Accordingly, in the shaded box on this page we present the pseudocode of the algorithm of a bridge connecting two piconets (referred to as piconets A and B) when it operates in piconet A. We note that MQS, β, and MTS are parameters of the algorithm.

In high inter-piconet traffic the bridge becomes a bottleneck. Therefore, the master has to serve it with the highest quality available. This might cause performance degradation to the intra-piconet traffic, but it improves the performance of inter-piconet traffic. The master can provide the highest quality of service by using an exhaustive regime in which it empties the bridge's queue (this regime was used in our simulation experiments). However, the master can serve the bridge in any other regime. In general, the master's behavior has a major impact on the performance of the algorithm.

The algorithm complies with the Bluetooth specifications in the following way. When the bridge switches to the other piconet, it enters hold mode in the first piconet and sets the hold timeout to the value of TC. Once the TC expires, the master polls the bridge every few slots, according to its polling scheme. After the bridge returns to the piconet, the master should poll it with higher priority. Since the bridge might *not* return immediately after the TC expires, the value of $T_{supervision}$ (the time period after which the master decides the slave has been disconnected) should be set to a value that will not create false connection drops.

Note that the overhead incurred by entering the hold mode is at least two time slots. However, we show that this overhead is negligible in comparison to the performance improvement due to the adaptability of the algorithm. These two slots also ensure the stability of the algorithm in low loads.

The Sniff Mode Algorithm

A benchmark algorithm is required for evaluating the performance of the LAA. Since a few scheduling algorithms presented in the past utilize sniff mode, we have implemented a benchmark algorithm that utilizes this mode. We refer to this algorithm as the *Sniff Mode Algorithm*. It has been designed to yield excellent results for scatternets composed of two piconets with a symmetrical arrival process. Thus, for the implementation of the algorithm, we have made the following assumptions:

- The *sniff interval* of the two masters is identical.
- The interval of one master begins at the middle of the interval of the other master.
- The bridge switches to a piconet at the beginning of the piconet's sniff interval.
- When the master connects to the bridge, it provides the highest quality of service to the bridge by using an exhaustive regime in which it empties the bridge's queue.

- After emptying the bridge's queue, the master conducts an RR policy and communicates with all its slaves including the bridge.
- When the bridge is not present in the piconet, the master conducts an RR policy with the other slaves.
- To ensure that (when possible) the bridge will continue participating in the piconet after the end of the exhaustive stage, the value of sniff timeout is higher than the number of slots required for communicating with all the slaves in an RR policy.

Note that both algorithms should handle traffic generated in different independent locations. Thus, we deliberately refrained from forcing exact timing restrictions. Namely, the master will serve the bridge at the highest possible quality. However, if the master serves a different slave when the rendezvous point occurs (i.e., when the bridge returns to the piconet), the bridge might have to wait for a few slots until it is served by the master.

Performance Evaluation

The performance of the LAA and Sniff Mode Algorithm has been evaluated via simulation. In this section we present results obtained for a scatternet that consists of two piconets connected by a bridge. Each of the piconets includes three slaves. At each node, 1-slot packets are generated according to a Poisson arrival process (the arrival rates in all the nodes are equal). For every packet, the destination can be any other node in the scatternet (equal probabilities for each of the other nodes).

The Load Adaptive Algorithm

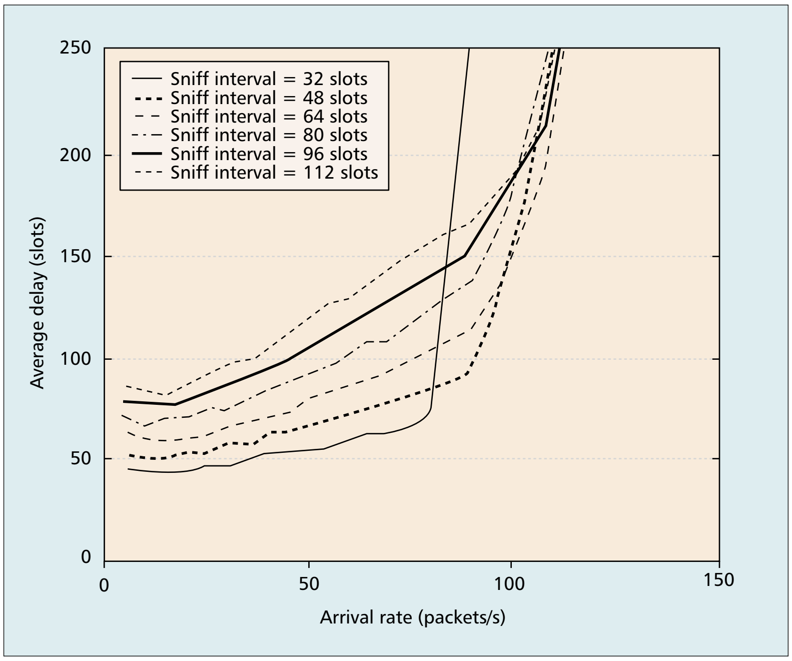
The simulation experiments of the LAA in the scatternet presented above were conducted for different values of MTS (defined earlier). The values of the parameters β and MQS were set to 2 and 10, respectively. Figure 4 presents the average delay as a function of the arrival rate for various values of MTS.

It can be seen that the optimal MTS changes according to the load (initially it is 16 and then it becomes 24). This implies that this parameter may not be fixed and should adapt to the load in the scatternet. We note that for low values of load, the delay for different values of MTS is almost equal. This property results from the adaptability of the algorithm to changing values of load. We shall see later that it is not possible to obtain such results in algorithms based on the sniff mode and that in such algorithms, there is a high correlation between the delay and the sniff interval.

The Sniff Mode Algorithm

The simulation experiments of the Sniff Mode Algorithm in the scatternet presented above were conducted for different sniff interval lengths. Figure 5 presents the average delay as a function of the arrival rate for various sniff intervals.

It can be seen that for low load, short sniff intervals (e.g., 32 slots) result in better performance, while for high load, long intervals provide better performance. The performance of the different sniff intervals resembles the relations between the different values of MTS in the LAA. However, as we will show later, in most cases the delay resulting from the Sniff Mode Algorithm is higher than that resulting from the LAA. Moreover, notice that a packet generated in one piconet and intended for the other piconet must wait until the end of the sniff interval in order to be sent to the neighboring piconet. This imposes a minimum delay that could not be avoided. Accordingly, unlike the LAA in the Sniff Mode Algorithm, long sniff intervals result in long delay for low load.



■ **Figure 5.** *Sniff Mode Algorithm performance: average delay for different sniff intervals as a function of the arrival rate (packets/s).*

LAA vs. Sniff Mode Algorithm

The performance of the LAA has been compared to the performance of the Sniff Mode Algorithm. Recall that the Sniff Mode Algorithm was designed to yield good results for scatternets composed of two piconets with a symmetrical arrival process.

The maximum time the bridge dedicates to each master in the LAA is MTS. Similarly, in the Sniff Mode Algorithm the maximum time dedicated to each master is half of the sniff interval. Thus, we have compared the performance of the algorithms when MTS is equal to half of the sniff interval. We selected MTS and sniff interval values that obtain relatively good results for all load values. Accordingly, Fig. 6 describes the average delay in scatternets using the LAA and Sniff Mode Algorithm.

The Sniff Mode Algorithm imposes strict timing constraints on the bridge, whereas the LAA allows more flexibility to the bridge. For example, in the LAA an idle bridge will usually switch piconets. On the other hand, in the Sniff Mode Algorithm an idle bridge must wait until the beginning of the next sniff interval. Therefore, as we can see from Fig. 6, the LAA usually yields better results than the Sniff Mode Algorithm.

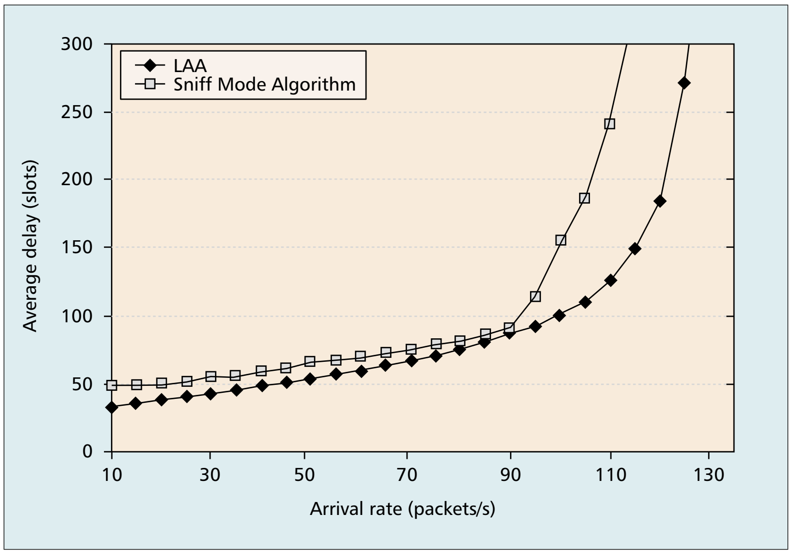


Figure 6. *Comparison of the LAA and the Sniff Mode algorithms for MTS = 24 slots and sniff interval of 48 slots.*

Conclusions and Future Study

We anticipate that unlike initial expectations, Bluetooth scatternets will be relatively small. Hence, this article presents the Load Adaptive Algorithm for inter-piconet scheduling in small-scale scatternets. The algorithm is based on the hold mode and does not require any modifications to the Bluetooth specifications.

In order to evaluate the performance of the algorithm, we have developed a simulation model of a scatternet. We have shown that the simulation results are very close to the analytical results. We have also presented results regarding the performance of intra-piconet scheduling algorithms.

Then we have compared the performance of the LAA to the performance of a benchmark scheduling algorithm based on the sniff mode. We have shown that the LAA usually yields better results than the Sniff Mode Algorithm. Thus, this algorithm is a good candidate for inter-piconet scheduling in small-scale scatternets.

Future study will focus on expanding the adaptability of the LAA. Moreover, a major future research direction is the development of scheduling algorithms that will be able to deal with various quality-of-service requirements as well as interact with scatternet formation and routing protocols.

Acknowledgments

This research (no. 148/03) was supported by the Israel Science Foundation, by a grant from the Ministry of Science, Israel, and by Intel. We would like to thank Guy Bale, Alon Mahl, and Professor Uri Yechiali.

References

[1] Bluetooth SIG, "Specification of the Bluetooth System — Version 1.2," Nov. 2003.

[2] A. Capone, M. Gerla and R. Kapoor, "Efficient Polling Schemes for Bluetooth Picocells," *Proc. IEEE ICC '01*, June 2001.

[3] A. Das *et al.*, "Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless Ad Hoc Network," *Proc. IEEE INFOCOM '01*, Apr. 2001.

[4] G. Zussman and A. Segall, "Capacity Assignment in Bluetooth Scatternets — Optimal and Heuristic Algorithms," *ACM/Kluwer Mobile Nets. and Apps.*, vol. 9, no. 1, Feb. 2004, pp. 49–61.

[5] S. Baatz *et al.*, "Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme," *Proc. IEEE INFOCOM '02*, June 2002.

[6] P. Johansson *et al.*, "Rendezvous Scheduling in Bluetooth Scatternets," *Proc. IEEE ICC '02*, Apr. 2002.

[7] A. Racz *et al.*, "A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets," *Proc. ACM MOBIHOC '01*, Oct. 2001.

[8] N. Johansson, F. Alriksson, and U. Jonsson, "JUMP Mode — A Dynamic Window-based Scheduling Framework for Bluetooth Scatternets," *Proc. ACM MOBIHOC '01*, Oct. 2001.

[9] G. Zussman, A. Segall, and U. Yechiali, "Bluetooth Time Division Duplex — Exact Analysis as a Polling System," CCIT Report #414, Technion. Dept. of Electrical Engineering; http://www.comnet.technion.ac.il/~gilz/pub_files/ccit_414.pdf, Feb. 2003.

[10] G. Zussman, U. Yechiali, and A. Segall, "Exact Probabilistic Analysis of the Limited Scheduling Algorithm for Symmetrical Bluetooth Piconets," *Proc. IFIP TC6 Pers. Wireless Commun.*, *LNCS*, vol. 2775, Springer, Sept. 2003.

Biographies

Liron Har-Shai (liron_har_shai@hotmail.com) received his B.Sc. degree (cum laude) in electrical engineering from the Technion, Israel Institute of Technology in 2002. He is currently an R&D engineer in the Israel Defense Forces and studying toward an M.Sc. degree in biomedical engineering at Tel-Aviv University. Between 1999 and 2002 he was a member of the Mobile Product Group (MPG) chip design team of Intel Corporation that developed the Centrino™ mobile technology. He was a recipient of the Best Paper Award at the OPNETWORK 2002 Conference.

Ronen Kofman (ronen.kofman@intel.com) received his B.Sc. degree (cum laude) in computer and software engineering from the Technion, Israel Institute of Technology in 2002. He is currently employed by Intel Corporation as a member of the mobile platforms group (MPG) chip design team centered in Haifa, Israel. His professional interests are in the areas of CPU architecture and wireless communications. He is currently involved in the development of the next-generation CPU for mobile computers. He was a recipient of the Best Paper Award at the OPNETWORK 2002 Conference.

Adrian Segall [F] (segall@ee.technion.ac.il) received B.Sc. and M.Sc. degrees in electrical engineering from the Technion, Israel Institute of Technology in 1965 and 1971, respectively, and a Ph.D. degree in electrical engineering with a minor in statistics from Stanford University in 1973. He is presently Benjamin Professor of Computer-Communication Networks in the Department of Electrical Engineering, Technion, Israel Institute of Technology. He has held visiting positions with IBM, AT&T and Lucent Bell Labs. His current research interests are in the area of optical networks, and wireless, sensor and ad hoc networks. He has served in the past as Editor for Computer Communication Theory of *IEEE Transactions on Communications* and Editor for *IEEE Information Theory Society Newsletter*. He is the recipient of the 1981 Miriam and Ray Klein Award for Outstanding Research and of the 1990 Taub Award in Computer Science. He is presently a Senior Editor of *IEEE Journal on Selected Areas in Communications*.

Gil Zussman (gilz@tx.technion.ac.il) received a B.Sc. degree in industrial engineering and management and a B.A. degree in economics (both summa cum laude) from the Technion, Israel Institute of Technology in 1995. He received an M.Sc. degree (summa cum laude) in operations research from Tel-Aviv University in 1999. Between 1995 and 1998 he served as an engineer in the Israel Defense Forces. He is currently working toward a Ph.D. degree in the Department of Electrical Engineering at the Technion. His current research interests are in the area of ad hoc and sensor networks. He is a recipient of the Knesset (Israeli Parliament) Award for distinguished students, the Best Paper Award at the OPNETWORK 2002 Conference, and the Best Student Paper Award at the IFIP-TC6 Networking 2002 Conference, and the Fulbright Post-Doctorate Fellowship.