Inter-Piconet Scheduling in Bluetooth Scatternets

Liron Har-Shai, Ronen Kofman, Gil Zussman and Adrian Segall
Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 32000, Israel
E-mail: {liron@comnet, ronen@comnet, gilz@tx, segall@ee}.technion.ac.il
Web: www.comnet.technion.ac.il/{~cn18s01, ~gilz, segall}

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. Efficient scatternet data flow requires design of inter-piconet scheduling algorithms. This paper presents and evaluates a load adaptive scheduling algorithm based on the Bluetooth *hold* mode. An OPNET model of a Bluetooth scatternet has been developed in order to evaluate the performance of the algorithm. We present the model and analytic results, used to verify the performance of the model. Then, we evaluate the performance of various intra-piconet scheduling algorithms. Finally, we present simulation results regarding inter-piconet scheduling and show that the proposed algorithm outperforms scheduling algorithms using the *sniff* mode.

1 Introduction

Recently, much attention has been given to the research and development of Personal Area Networks (PAN). These networks are comprised of personal devices, such as cellular phones, PDAs and laptops, in close proximity to each other. Bluetooth is an emerging PAN technology, which enables portable devices to connect and communicate wirelessly via short-range ad-hoc networks [5],[6],[14]. The basic Bluetooth network topology (referred to as a *piconet*) is a collection of slave devices operating together with one master. A multihop ad-hoc network of piconets in which some of the devices are present in more than one piconet is referred to as a *scatternet* (see for example Figure 1). A device that is a member of more than one piconet (referred to as *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).

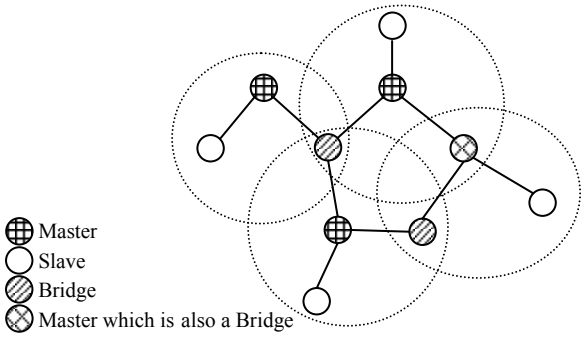

Figure 1: An example of a Bluetooth scatternet

In the Bluetooth specifications [5], the capacity allocation by the master to each link in its piconet is left open. The master schedules the traffic within a *piconet* by means of polling and determines how bandwidth capacity is to be distributed among the slaves. Numerous heuristic *intra-piconet* scheduling algorithms have been proposed and evaluated via simulation (e.g. [7],[8] and references therein).

Efficient *scatternet* operation requires determining the link capacities that should be allocated in each piconet, such that the network performance will be optimized. In [18] and [20], the scatternet capacity assignment problem has been analyzed, and optimal and heuristic algorithms for its solution have been proposed. The required link capacities determined by such an algorithm 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 [14] an overall architecture for handling scheduling in a scatternet and a family of inter-piconet scheduling algorithms have been presented. Recently, several inter-piconet scheduling algorithms have been proposed and evaluated (e.g. [1],[13],[15],[17]). We note that in [12] it has been shown that constructing an optimal link schedule that maximizes the total throughput in a scatternet is an NP hard problem.

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" [14] in which they can switch between piconets. Thus, scheduling algorithms in large-scale scatternets can be based on the Bluetooth low power mode *sniff* (see Section 2), which provides recurring "rendezvous points" (e.g. [1],[13]). It can also be based on new modes that require modifications to the Bluetooth specifications (e.g. the *jump* mode [11]).

Bluetooth, which is a PAN technology, and IEEE 802.11 [10], which is a WLAN technology, are complementary technologies. Therefore, we anticipate that most of the scatternets will not be used as a replacement to WLANs and will be composed of only a few piconets. In such small-scale scatternets, the coordination of the presence of the bridges in the 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 paper we propose an inter-piconet scheduling algorithm intended for small-scale scatternets. The algorithm is based on the low power mode *hold* (see Section 2), which enables a unit to leave a piconet for a short period, and does not require modifications to the Bluetooth specifications.

We will show that for a few intra-piconet scheduling regimes, a piconet can be modeled as a polling system[1]. However, as mentioned in [7], due to the special characteristics of the Bluetooth MAC (see Section 2), the operation model of most scheduling regimes differs from those of classical polling models. Thus, the analysis of intra and inter-piconet scheduling requires the development of simulation models.

Since previous Bluetooth OPNET models ([2],[9]) were not designed to evaluate inter-piconet scheduling schemes, we have developed an OPNET model of a scatternet. In order to verify the performance of the model, we present analytic results regarding intra-piconet scheduling that are based on polling models. We show that the simulation results are very close to the analytic results. Then, we evaluate, via simulation, the performance of various *intra-piconet* scheduling regimes.

Finally, a load adaptive *inter-piconet* scheduling algorithm based on the *hold* mode and an inter-piconet scheduling algorithm based on the *sniff* mode are presented. The performance of the algorithms is compared via simulation and it is shown that the load adaptive algorithm outperforms the algorithm using the sniff mode.

This paper is organized as follows. Section 2 gives a brief introduction to Bluetooth technology and Section 3 presents the OPNET model of a Bluetooth scatternet. In Section 4, we present analytic and simulation results regarding intra-piconet scheduling, and validate the model. Section 5 presents inter-piconet scheduling algorithms and Section 6 evaluates the performance of these algorithms. In Section 7, we summarize the main results and discuss possible extensions.

2 Bluetooth Technology

Bluetooth utilizes a short-range radio link, which operates in the 2.4GHz ISM band. Since the radio link is based on frequency-hop spread spectrum, multiple channels (frequency hopping sequences) can co-exist 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 7 slaves.

Bluetooth channels use a frequency-hop/time-division-duplex (FH/TDD) scheme. The channel is divided into 625-μsec 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-slots *packets*, which are transmitted in consecutive slots. Packets can carry synchronous information (voice link) or asynchronous information (data link).[2] Information can only be exchanged between a master and a slave, i.e. there is no direct communication between slaves.

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 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* (in case it has nothing to send). The master schedules the traffic within a *piconet* according to an *intra-piconet* scheduling algorithm (e.g. round robin).

Multiple piconets in the same geographic area form a *scatternet*. Since Bluetooth uses packet-based communication over slotted links, it is possible to interconnect different piconets in the same scatternet. Hence, 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 (see for example Figure 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.

The Bluetooth specification defines a few low power modes. Two of these modes, which are described below, 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, and 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 many intervals. Thus, hold mode requires repeated negotiations that waste at least a slot pair while sniff mode requires a single negotiation.

The Bluetooth protocol stack includes a few layers. We shall briefly present three layers that are required for the understanding of the simulation model (for more information see [5] and [6]):

- *Baseband* – Controls the physical link through the radio, assembling packets and controlling frequency hopping.
- *Link Manager Protocol (LMP)* – Controls and configures links to other devices (for example negotiates hold timeout).
- *Logical Link Control and Adaptation (L2CAP)* – Multiplexes data from higher layers and converts between different packet sizes.

3 OPNET Bluetooth Model

We have developed an OPNET model of a Bluetooth scatternet. This model enables performance evaluation of various inter- and intra-piconet scheduling schemes. This section describes the model and concentrates on the modules responsible for schedul-

[1] A polling system consists of several queues served by a single server according to a set of rules (polling scheme) [3, p. 200],[16].

[2] In this paper we focus on networks in which only data links are used.

ing and routing. The model consists of 3 kinds of Bluetooth nodes: *master*, *slave* and *bridge*. It is an integration of processes developed for the model (such as Baseband and LMP) and standard OPNET processes (such as Radio Transmitter and Receiver).

The simulation model implements most of the features of the Bluetooth Baseband layer required for evaluation of the scheduling algorithms (time division duplex, multi-slot packets, low power modes, etc.). As we mainly focus on scheduling, the model does not simulate the frequency hopping mechanism and assumes that the channel is error free. In the current model we do not simulate the connection establishment procedures and assume that the scatternet is given (i.e. it is formed by a scatternet formation algorithm [14]). We also assume that the different masters' clocks are synchronized.

Routing protocols for a scatternet with inter-piconet communications have not yet been defined. Hence, in order to simulate inter-piconet traffic we use the model described in [4], according to which the Baseband layer routes Bluetooth data packets (referred to as ACL packets).

3.1 Slave

The node model of a slave, which is not a bridge, is illustrated in Figure 2. A packet transmitted by the master is received by the Radio Receiver, which sends it to the RF Interface. The RF Interface checks if the packet is destined for the slave and if it is, it is passed on to the Route Module. Then, if the buffer is not empty, the first packet from the buffer it is sent (via the RF Interface) to the Radio Transmitter, which transmits it to the master.

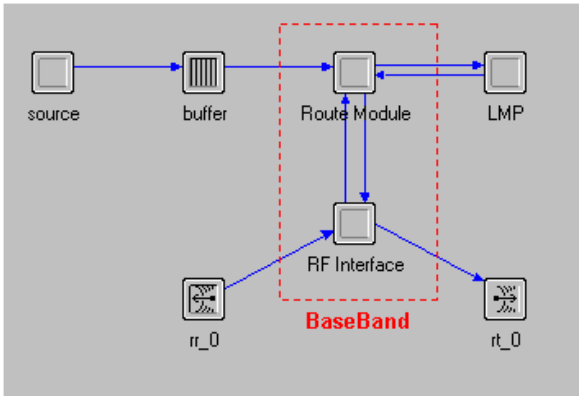

Figure 2: The node model of a slave

The RF Interface is responsible for transferring packets to and from the Route Module. We note that a similar process (RF interface) exists in the master and the bridge. The RF Interface finite state machine (FSM) is presented in Figure 3 and the states are described below.

– *Init* – Initializes all state variables and attributes.
– *Idle* – Waits for a certain interrupt.
– *Rcv UP* – When a packet arrives from the Route Module, it is sent to the RF by the *Process TX* and *Send RF* states.
– *Rcv RF* – When a packet arrives from the radio, it is discarded if it is not addressed for the slave. Otherwise, the packet is sent to the Route Module by the *Process RX* and *Send Up* states.

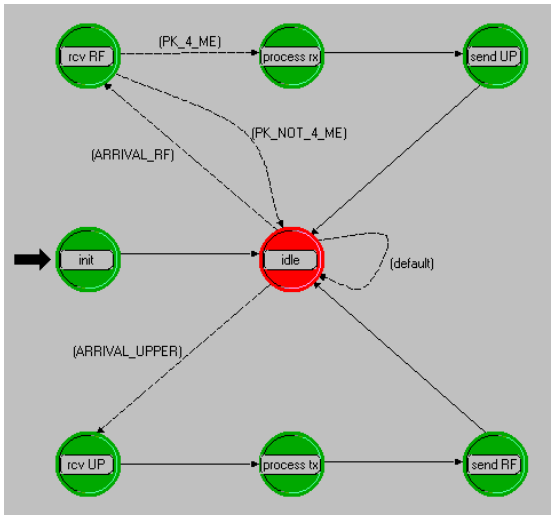

Figure 3: RF Interface finite state machine (FSM)

Another process that exists in every type of node is the Packet Generator Source. It can generate 2 different types of arrival processes:

– Packets generated according to a Poisson arrival process. The distribution of packet lengths (1, 3, and 5-slots) is user defined.
– 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-slots packet such that the generated packets are of the maximal possible size. For example, a file whose length is 14 slots will be segmented into two 5-slots packets, one 3-slots packet, and one single slot packet.

The Packet Generator Source replaces Bluetooth's *L2CAP* layer that generates Baseband packets (ACL) from higher layers packets. It is obvious that the arrival rate to the Baseband from the L2CAP is not Poisson. However, in order to use analytical models we focus at this stage on Poisson arrival. In the future we intend to extend the Packet Generator in order to analyze different types of traffic, such as TCP/IP traffic.

The Route Module in the slave is slightly degenerated since the slave responds only to its master. Thus, we shall discuss this process in the next section.

3.2 Master

The node model of a master, which is not a bridge, is presented in Figure 4. The master has 7 different queues (Q) for outgoing traffic to each of its slaves, and an additional queue (buffer) for packets that it generates.

The Route Module of the master is responsible for the scheduling policy of the master and partially responsible (with the Bridge Route Module) for routing. There are several scheduling algorithms implemented in the master Route Module, such as pure round robin and exhaustive round robin. These algorithms will be presented in Section 4. The finite state machine of the Route Module is presented in Figure 5 and some of the states are described below.

– *Init* – Initializes all state variables and attributes.
– *Idle* – Waits for a certain interrupt.

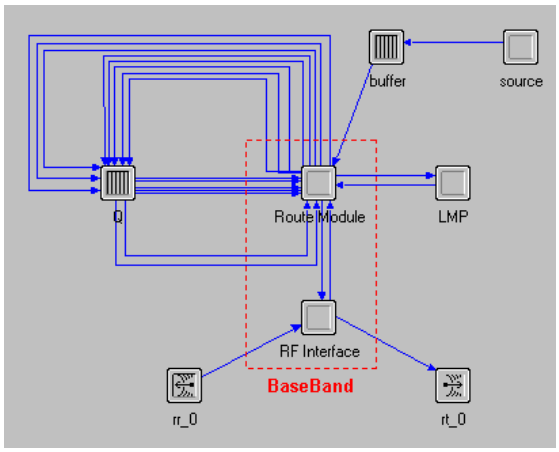

Figure 4: The node model of a master

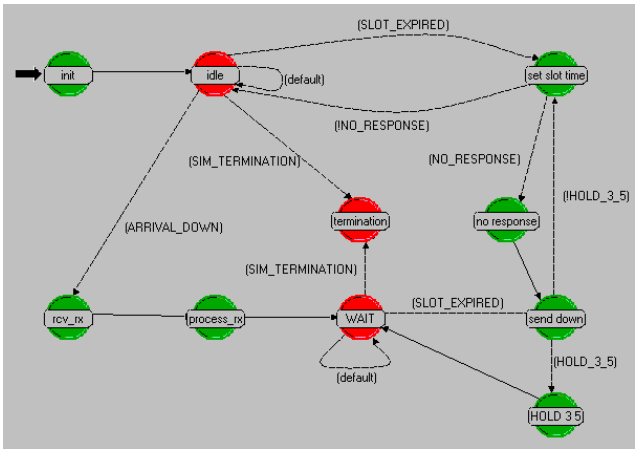

Figure 5: Master Route Module FSM

– *Rcv RX* and *Process RX* – Receive packets from lower layers and analyze them.
– *Wait, Hold, Set Slot Time* – Implement Bluetooth's slot mechanism (TDD).
– *No Response* – Responsible for handling non-acknowledged packets.
– *Send Down* – Selects (according to the scheduling policy) the next slave, which will receive a data or a POLL packet.
– *Termination* – Erases all allocated variables and writes sampled results to a log file.

3.3 Bridge

A piconet whose master is also a bridge is non-active when the master is away. Therefore, scatternets where masters are bridges may result in poor bandwidth utilization [18],[20]. Thus, our bridge model simulates a node that is a slave of a few masters. Implementing a bridge that is also a master requires further work.

The node model of a bridge that is a slave of two masters is described in Figure 6. The bridge has a different queue (Q) for each piconet it is connected to and an additional queue (buffer) for the packets that it generates. It also has 2 pairs of Radio Transmitter-Receiver, one for each piconet. The node model of a bridge connecting a few masters is similar.

The Route Module of the bridge is responsible for scheduling the presence of the bridge in the different piconets, according to

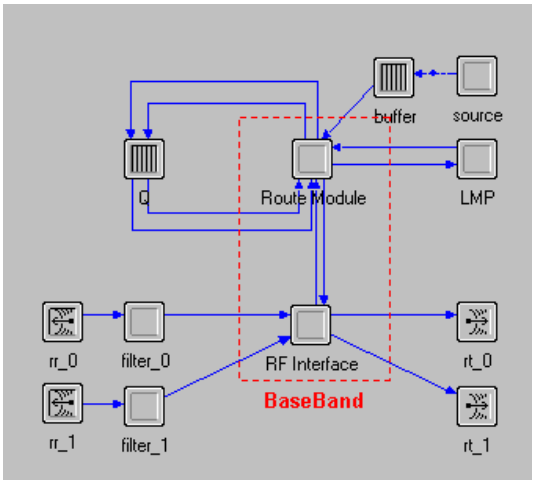

Figure 6: The node model of a bridge

the selected scheduling algorithm. The switch between piconets is implemented by agreeing with the masters on hold or sniff mode. The negotiation regarding the mode is performed by the Link Manager Protocol (LMP) process, using a single slot packet. The Route Module is also responsible for routing packets from different piconets through the bridge.

4 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.

Several intra-piconet scheduling algorithms have been proposed in the past (e.g. [7],[8] and references therein). We have implemented a few algorithms and verified the performance of the model by analytical results that are based on polling models. The implemented algorithms include the following:
– *Round Robin* (RR) – The master communicates with the slaves in a round robin policy. In every round it sends a single packet to every slave.
– *Exhaustive Round Robin* (ERR) – The master communicates with the slaves in a round robin policy. While communicating with a slave, the master sends packets until its queue and the slave's queue are empty.
– *Exhaustive Round Robin according to Slaves Queues* (SERR) – The master communicates with the slaves in a round robin policy. While communicating with a slave, the master exhausts the slave's queue (sends packets until the slave responds with a NULL packet).
– *Exhaustive Round Robin according to Master Queues* (MERR) – The master communicates with the slaves in a round robin policy. While communicating with a slave, the master exhausts its queue.
– *Priority Round Robin* (PRR) – The master polls the slaves according to their priorities, allowing one slave to be polled more frequently than another. The priorities can be changed dynamically.
– *Longest Queue* (LQ) – Every time the master can send a packet, it checks the status of the queues towards the slaves. It polls the slave with the longest queue.

– *Shortest Queue* (SQ) - Every time the master can send a packet, it checks the status of the queues towards the slaves. It polls the slave with the shortest (non-empty) queue.

In order to verify the performance of the model, we have compared analytic result to simulation results in cases there is no inter-slave traffic in the piconet. In this section, we shall present analytic results regarding a few scheduling regimes and compare them to simulation results. It will be shown that the simulation results are very close to the analytical results. We shall also compare the performance of the scheduling algorithms described above in two different scenarios. In the first scenario there is no inter-slave traffic and in the second scenario there is inter-slave traffic as well as master-slave traffic. We shall show that the Exhaustive scheme (ERR) and the Largest Queue scheme (LQ) perform well in both scenarios.

4.1 Round Robin with 1-slot Packets

In [19] we show that a piconet in which the master operates in a *round robin* regime (RR) and only 1-slot packets are used can be analyzed as a TDMA system [3, p. 194].

Accordingly, for a Poisson arrival process of 1-slot packets, the expected delay (in slots) on a link from slave i to the master is:

$$D_i = 1 + \frac{n}{1 - 2n\lambda_i}, \qquad (1)$$

where n is the number of slaves and λ_i is the arrival rate to slave i (packets/slot). Similarly, (1) also describes the delay on the link from the master to slave i. In this case, λ_i should denote the arrival rate to the master's queue of packets intended to slave i. Thus, in a symmetrical system in which the arrival rates to all master and slaves queues are equal, the expected delay is:

$$D = 1 + \frac{n}{1 - 2n\lambda}, \qquad (2)$$

where λ is the arrival rate to a queue. In such a system the load is $2n\lambda$.

Figure 7 compares the analytic and simulation results of the average delay for various values of load. The considered piconet includes 4 slaves, it is symmetrical (all the arrival rates are equal), and the 1-slot packets are generated according to a Poisson arrival process.

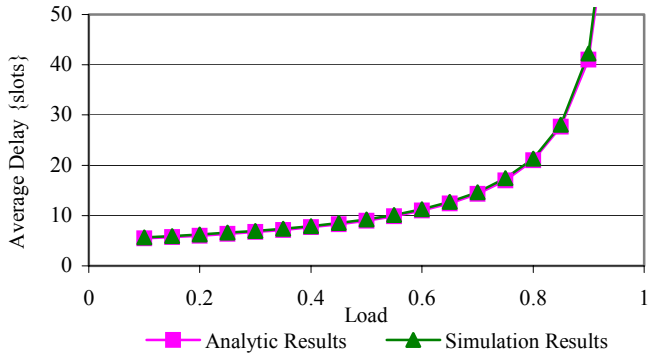

Figure 7: Analytic and simulation results for the average delay in a symmetrical piconet with 4 slaves and 1-slot packets operated in a round robin regime

4.2 Round Robin with 1,3 and 5-slot Packets

In [19] we show that a piconet in which the master operates in a *round robin* regime (RR) and 1,3 and 5-slot packets are allowed can be analyzed as a *1-limited polling system* [3, p. 201],[16]. Since the analysis of 1-limited polling systems is difficult, we consider symmetrical systems in which the arrival rates to all queues are equal.

Accordingly, for a Poisson arrival process of 1,3 and 5-slot packets, the expected delay (in slots) is:

$$D = 2P_3 + 4P_5 + \frac{1 + n\{1 + 2\lambda(P_3 + 6P_5 - 1)\}}{1 - 2n\lambda(1 + 2P_3 + 4P_5)}, \qquad (3)$$

where P_3 and P_5 are the probabilities of a 3-slot and a 5-slot packet (respectively), n is the number of slaves, and λ is the arrival rate (packets/slot) to a queue (master to slave or slave to master). In such a system the load is $2n\lambda\,(1 + 2P_3 + 4P_5)$.

Figure 8 compares the analytic and simulation results of the average delay for various values of load. The considered piconet includes 4 slaves and the arrival process is symmetrical. The packets are generated according to Poisson arrival process and the distribution of the arriving packets is uniform (i.e. the probability of a 1,3 and 5-slot packet is equal).

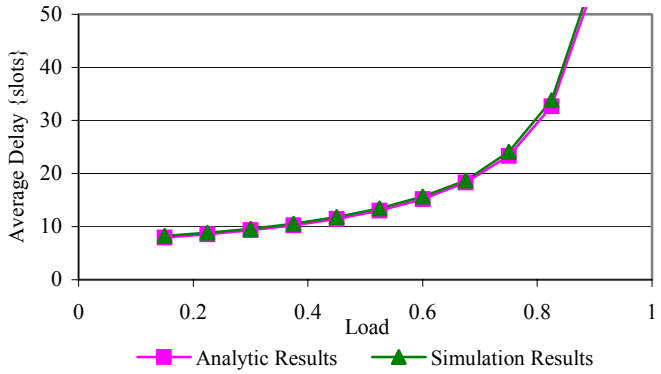

Figure 8: Analytic and simulation results for the average delay in a symmetrical piconet with 4 slaves and 1, 3 and 5-slots packets (P_1=P_3=P_5=1/3) operated in a round robin regime

4.3 Exhaustive Round Robin with 1-slot Packets

Consider a piconet in which there is only slave to master traffic and only 1-slot packets are allowed. In [19] we show that in such a piconet, if a master operates in an e*xhaustive round robin* regime (ERR), the piconet can be modeled as an *exhaustive polling system* [3, p. 200],[16] and approximate analysis is possible.

In symmetrical scenarios in which the arrival rates to all slaves are equal, the approximate expected delay (in slots) for Poisson arrival process of 1-slot packets is:

$$D = 1 + \frac{n}{1 - 2n\lambda}, \qquad (4)$$

where n is the number of slaves and λ is the arrival rate (packets/slot) to a slave-to-master queue. Accordingly, the load in the system is $2n\lambda$.

Figure 9 compares analytic and simulation results in a symmetrical piconet with 4 slaves. In this piconet, 1-slot packets are generated at the slaves according to a Poisson arrival process.

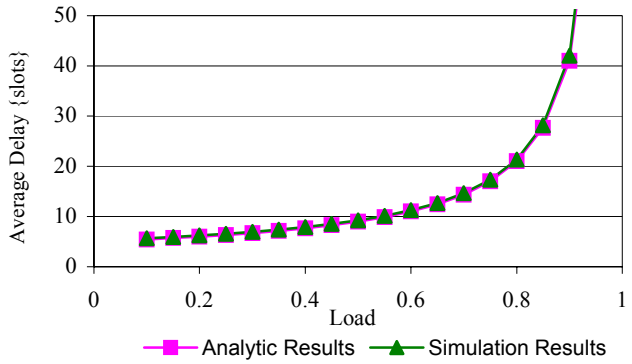

Figure 9: Analytic and simulation results for the average delay in a symmetrical piconet with 4 slaves and arrival process of 1-slot packets to the slaves. The piconet is operated in an exhaustive round robin regime (ERR)

Notice that it seems that for 1-slot packets the delay in the round robin regime (2) is equal to the delay in the exhaustive round robin regime (4). However, unlike equation (2), equation (4) is computed for a piconet without master-to-slave traffic. In case there is such traffic, the delay in the exhaustive regime is different (its computation is subject to further research).

4.4 Performance Evaluation

We shall now evaluate the performance of the scheduling schemes, mentioned above, in two different scenarios. The first scenario is based on the scenario described in [7]. In this scenario there is only master-to-slave and slave-to-master traffic. The considered piconet includes 7 slaves and information files are generated at each of the 14 queues according to a Poisson distribution. The length of the files is exponentially distributed with an average file length of 8 packets. The arrival process is followed by a segmentation of the file to 1, 3, and 5-slots packet (see Section 3.1).

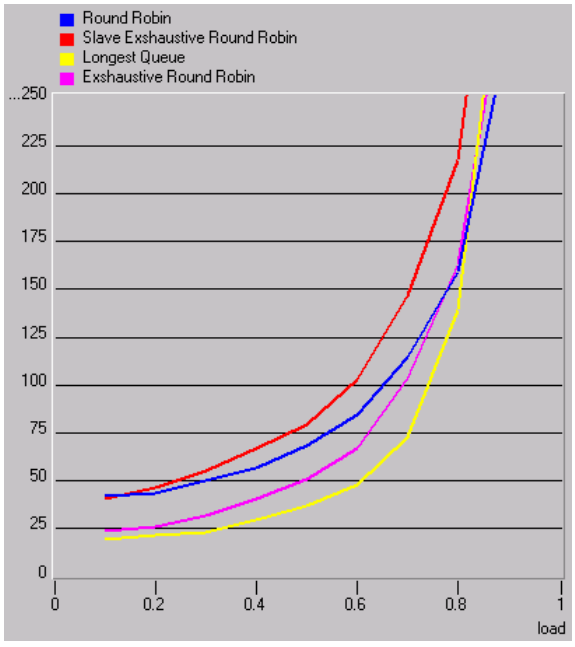

Figure 10: Average delay (in slots) of different piconet scheduling algorithms in a piconet with 7 slaves. Files, with average length of 8 slots, arrive to all queues according to a symmetrical Poisson arrival process

Figure 10 describes the average delay curves of the different schemes. It can be seen that for all realistic load values the *longest queue* regime (LQ) provides the best performance. The *exhaustive round robin* regime (ERR) also performs well up to a load of about 0.8. From that point on, the *round robin* regime (RR) performs better. We note that the main drawback of using the LQ regime is that the channel can be captured by few of the links (recall that this regime does not require the master to serve the slaves in a round robin manner). A similar problem occurs in ERR regime. We note that a few solutions to this problem have been suggested in the past (see for example [7] and references therein).

In the second scenario, we have considered inter-slave traffic (routed through the master). The considered piconet includes 7 slaves and information files are generated *only* at the slaves' queues according to a Poisson distribution. The file length distribution and the segmentation process are as described above. 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 packets.

Figure 11 describes the average delay curves of the different schemes in this scenario. It can be seen that the *round robin* regime (RR) and the *exhaustive round robin* regime (ERR) perform better than the other regimes. However, for low values of load all the regimes have similar performance. We note that unlike the first scenario, the *longest queue* regime (LQ) does not outperform the other regimes. Similarly, the other regimes have different performance curves in the two scenarios.

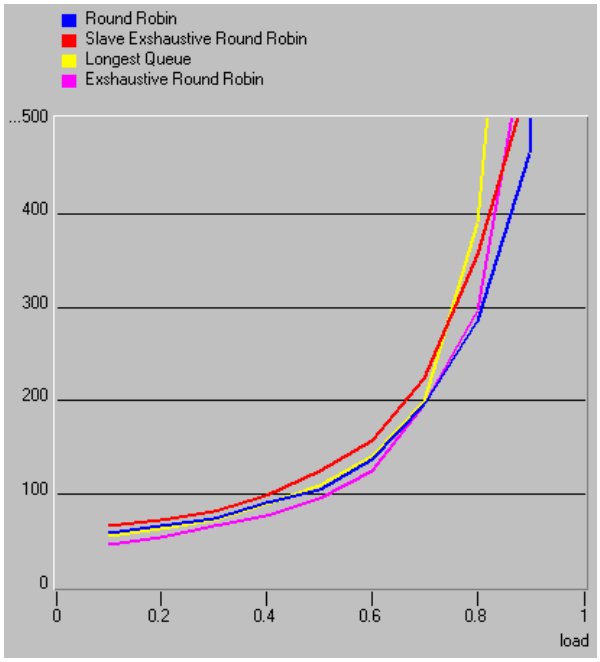

Figure 11: Average delay (in slots) of different piconet scheduling algorithms in a piconet composed of 7 slaves with inter-slave traffic. Files, with average length of 8 slots, arrive to all queues according to a symmetrical Poisson arrival process and the distribution of the destination of a file originating at a slave is uniform

5 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 (described in Section 2) 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. [1],[13]), we also describe an algorithm utilizing this mode. In the next section, this algorithm will serve a benchmark for evaluating the Load Adaptive Algorithm.

5.1 Load Adaptive Algorithm (LAA)

The *Load Adaptive Algorithm* manages the scheduling mechanism of the bridge. It determines the duration of the 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 its queues to the different masters and simple information transferred by the masters during the operation of the TDD protocol.

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

- *Idle State* (IS) – Wasting time by the bridge affects the delay of packets requiring inter-piconet routing. Thus, whenever the connection has exhausted the bridge should try to switch piconets. The bridge is in Idle State if: *the queue to the current piconet is empty* and *received 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 has exhausted. The *Max Queue Size* is the parameter used in order to make the decision. If the queue size is bigger than MQS the bridge should try to switch piconets.
- *Time Commitment* (TC) - The bridge sends this variable before a switch and it indicates the minimum interval that the bridge will spend outside the piconet. It is calculated according to the sizes of the bride's queues to the other piconets. It allows a master not to address the bridge throughout the interval and to readdress the bridge once the Time Commitment expires.
- *Predictability Factor* (β) – The Time Commitment (TC) is calculated according to the outgoing queue size. The number of slots which is required to exhaust the outgoing queue depends on the nature of the traffic originating from the other piconet. The *Predictability Factor* (β) is used in order to estimate the average packet length of this traffic and to 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 the Max Queue Size may cause a long delay to the packets intended for that piconet. In contrast, in cases of heavy traffic the queue sizes may be huge, and therefore, the Time Commitments 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 the *Max Time-Share* (MTS).

In this paper, we focus on bridges that connect two masters. Accordingly, in Figure 12 we present the pseudocode of the algorithm of a bridge connecting two masters. We note that MQS, β, and MTS are parameters of the algorithm.

```
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
```

Figure 12: The pseudocode of the Load Adaptive Algorithm executed by a bridge connecting piconets A and B, when it operates in piconet A

In cases of high inter-piconet traffic the bridge becomes a bottleneck. Therefore, the master has to serve it in 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. However, the master can serve the bridge in any other regime. We note that in our simulation experiments, the exhaustive regime was used in order to serve the bridge.

The algorithm complies with the Bluetooth specification in the following way. When the bridge switches to the other piconet, it enters the hold mode in the first piconet and sets the hold timeout to the value of the Time Commitment (TC). Once the Time Commitment 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 Time Commitment expires, the value of $T_{supervision}$ should be set to such a value that will not create false connection drops.[3]

Notice that the overhead incurred by entering the hold mode is at least 2 time slots. However, we will show that this overhead is negligible in comparison to the performance improvement due to the adaptability of the algorithm.

5.2 Sniff Mode Algorithm

A benchmark algorithm is required for evaluating the performance of the Load Adaptive Algorithm. Since a few scheduling algorithms, presented in the past, utilize the sniff mode (described in Section 2), we have implemented a benchmark algorithm that utilizes this mode. We refer to this algorithm as the *Sniff Mode Algorithm*. It has been designed such that it will 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 (Figure 13 demonstrates some of these 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.

[3] $T_{supervision}$ is the time frame after which the master decides that the slave has been disconnected.

– 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 a round robin policy and communicates with all its slaves including the bridge.
– When the bridge is not present in the piconet, the master conducts a round robin 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 s*niff timeout* is higher than the number of slots required for communicating with all the slaves in a round robin policy.

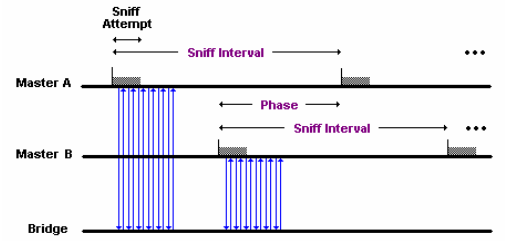

Figure 13: An example of the Sniff Mode Algorithm operation

6 Performance Evaluation

The performance of the Load Adaptive Algorithm and the Sniff Mode Algorithm has been evaluated via simulation. In this section, we present results obtained for the scatternet illustrated in Figure 14. This scatternet consists of two piconets connected by a bridge (referred to as slave 4). 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).

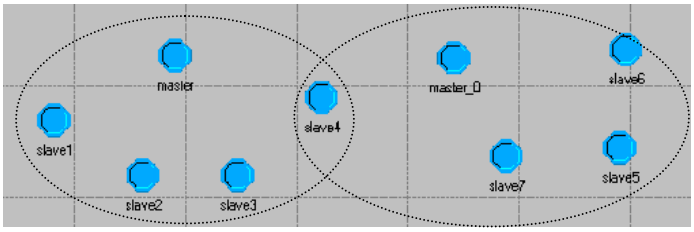

Figure 14: Scatternet consisting of 2 piconets

6.1 Load Adaptive Algorithm

The simulation experiments of the Load Adaptive Algorithm in the scatternet presented above were conducted for different values of MTS (Maximum Time Share – defined in Section 5.1). The values of the parameters β and MQS were set to 2 and 10 (respectively). Figure 15 presents the average delay as a function of the arrival rate[4] for various values of MTS. We note that a brief analysis of the reasons resulting in a piconet switch of the bridge can be found in the Appendix.

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

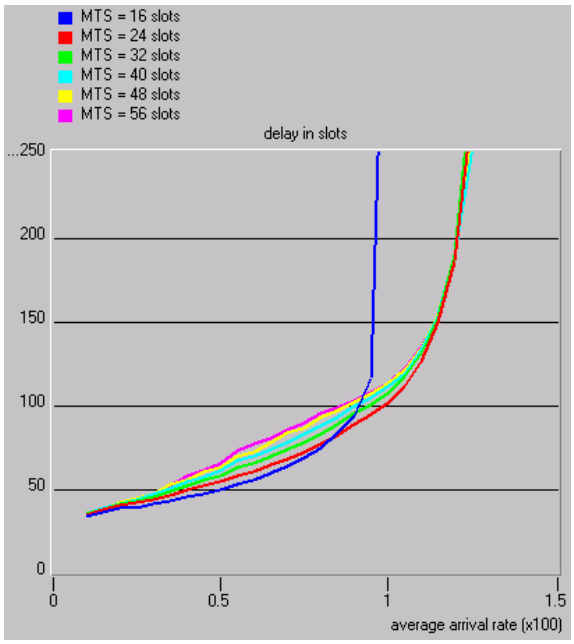

Figure 15: Load Adaptive Algorithm performance – Average delay for different values of MTS as a function of the arrival rates

this parameter may not be fixed and should adapt to the load in the scatternet. However, incorporating in the algorithm a mechanism for changing the MTS requires further research. 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 high correlation between the delay and the sniff interval.

6.2 Sniff Mode Algorithm

The simulation experiments of the Sniff Mode Algorithm in the scatternet presented above were conducted for different sniff interval lengths. Figure 16 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 in 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 Load Adaptive Algorithm. However, as we will show in Section 6.3, in most cases the delay resulting from the Sniff Mode Algorithm is higher than the delay resulting from the Load Adaptive Algorithm. Moreover, notice that a packet generated in one piconet and intended to the other piconet must wait until the end of the sniff interval in order to be sent to the neighboring piconet. Accordingly, unlike the Load Adaptive Algorithm in the Sniff Mode Algorithm, long sniff intervals result in long delay even for low load.

[4] Since the load differs in different parts of the scatternet, the arrival rate was not transformed to load.

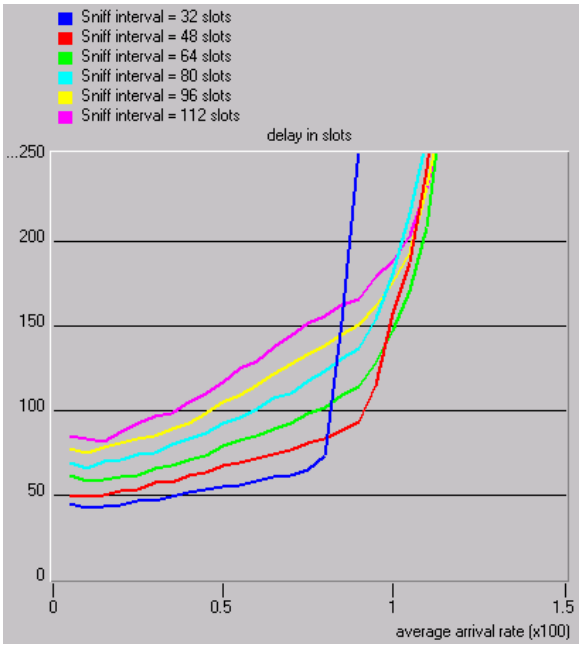

Figure 16: Sniff Mode Algorithm performance - Average delay for different sniff intervals as a function of the arrival rates

6.3 LAA vs. Sniff Mode Algorithm

The performance of the Load Adaptive Algorithm (LAA) has been compared to the performance of the Sniff Mode Algorithm. Recall that the Sniff Mode Algorithm was designed such that it will 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 Load Adaptive Algorithm 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. Accordingly, Figures 17 and 18 describe the results of the Load Adaptive Algorithm and Sniff Mode Algorithm for different values of MTS and sniff intervals.

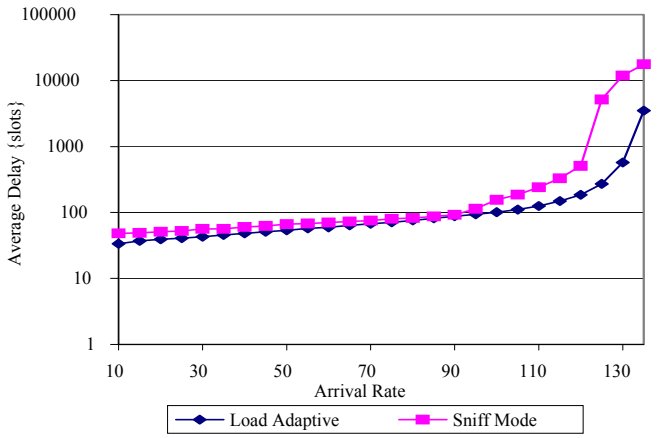

Figure 17: Comparison of the Load Adaptive and the Sniff Mode algorithms for MTS = 24 slots and sniff interval of 48 slots

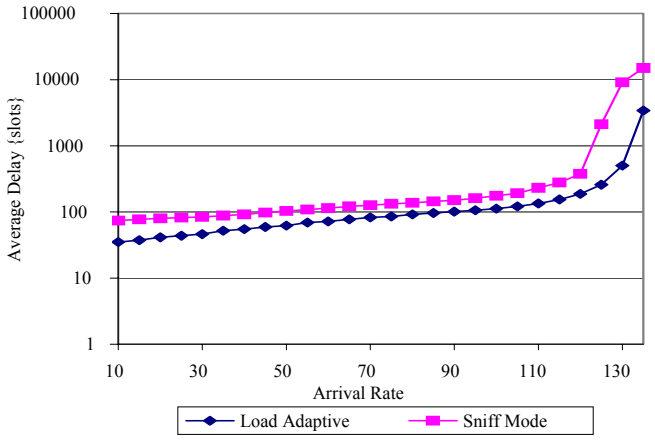

Figure 18: Comparison of the Load Adaptive and the Sniff Mode algorithms for MTS = 48 slots and sniff interval of 96 slots

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

7 Conclusions and Future Study

This paper presents a Load Adaptive Algorithm for inter-piconet scheduling in Bluetooth 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 an OPNET model of a scatternet. We have shown that the simulation results are very close to the analytical results and that the simulation model is very reliable. We have also presented results regarding the performance of intra-piconet scheduling algorithms.

Then, we have compared the performance of the Load Adaptive Algorithm to the performance of a scheduling algorithm based on the sniff mode. We have shown that the Load Adaptive Algorithm outperforms the Sniff Mode Algorithm. Thus, this algorithm is an excellent candidate for inter-piconet scheduling in small-scale scatternets.

We intend to extend the OPNET model presented in this paper in order to incorporate other important aspects of the Bluetooth technology such as mobility, topology construction and routing. Such an extended model is required for evaluating various protocols designed for Bluetooth scatternets. In addition, future study will focus on enhancing the Load Adaptive Algorithm and examining its adaptability to larger scatternets.

Finally, we note that 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 to interact with scatternet formation and routing protocols.

Acknowledgements

We would like to thank Prof. Uri Yechiali for helpful discussions regarding Bluetooth and Polling. We would like to thank Yoram Or-Chen, Yoram Yihyie, Hai Vortman, and the Computer Networks Laboratory staff for their technical support.

Appendix – LAA Behavior

The following graphs demonstrate the behavior of the Load Adaptive Algorithm. The graphs present the fraction of the piconet switches, which occurred due to each parameter (IS, MQS, MTS), as a function of the arrival rate. The different lines designate the different values of MTS.

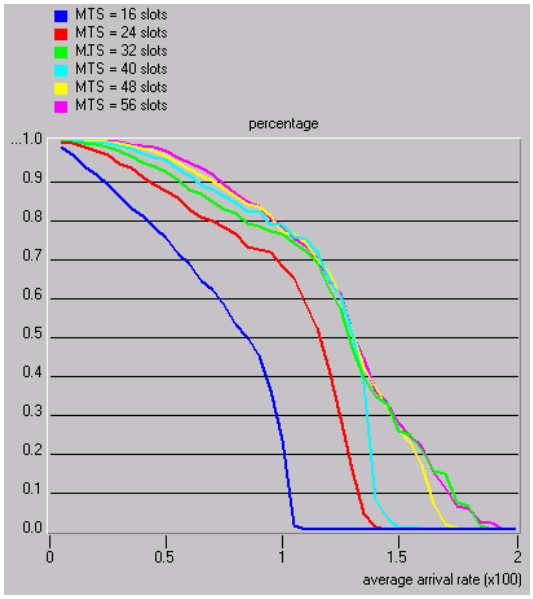

Figure 19 - The fraction of the switches that occurred due to bridge idleness (IS) as a function of the arrival rate, for different values of MTS

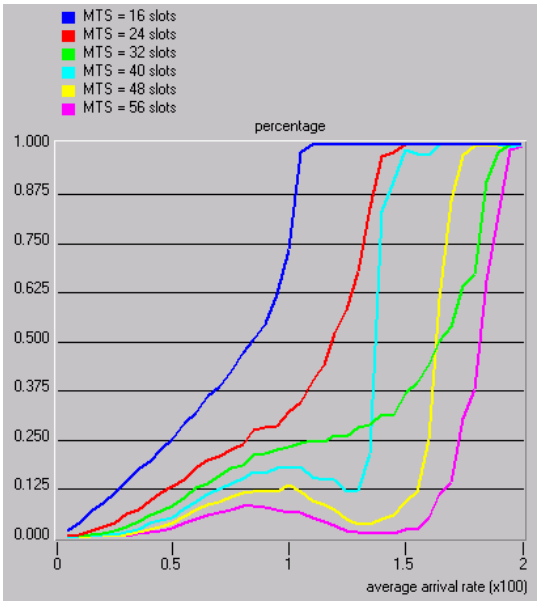

Figure 20 - The fraction of the switches that occurred due to time-share expiry (MTS) as a function of the arrival rate, for different values of MTS

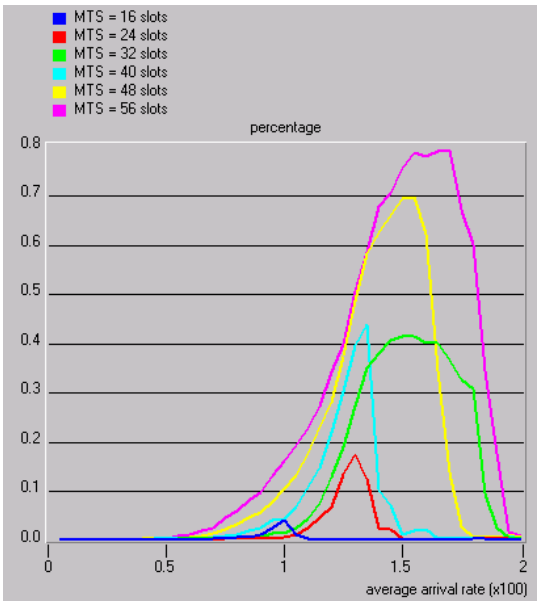

Figure 21 – The fraction of the switches that occurred due to queue size (MQS) as a function of the arrival rate, for different values of MTS

It is clear that for low values of load, the main reason for switching is idleness. For high values of load, the main reason is the time-share bound which is the MTS. The queue size parameter is effective in the relatively high values of MTS and has only minor impact for low values of MTS. We see that the impact is mainly in the middle range of the load. Thus, the algorithm adapts differently in every section of the load scale.

References

[1] S. Baatz, M. Frank, C. Kühl, P. Martini and C. Scholz, "Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme", *Proc. IEEE INFOCOM'02*, June 2002.

[2] E. Balatti and L. Marzegalli, "Increasing TCP/IP Performance over Home Wireless Networks (Bluetooth)", *Proc. OPNETWORK 2001*, Aug. 2001.

[3] D. P. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall Inc., New Jersey, 1992.

[4] P. Bhagwat and A. Segall, "A Routing Vector Method (RVM) for Routing in Multiple Bluetooth Networks", *Proc. IEEE MOMUC'99*, Nov. 1999.

[5] Bluetooth Special Interest Group, *Specification of the Bluetooth System – Version 1.1*, Feb. 2001.

[6] J. Bray and C. Sturman, *Bluetooth 1.1 connect without cables*, Prentice Hall, 2001.

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

[8] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, "Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless Ad-hoc Network", *Proc. IEEE INFOCOM'01*, Apr. 2001.

[9] N. Golmie and F. Mouveaux, "Modeling and Simulation of MAC protocols for Wireless Devices Coexistence Performance Evaluation", *Proc. OPNETWORK 2000*, Aug. 2000.

[10] IEEE 802.11 Standard, Documentation at URL: http://grouper.ieee.org/groups/802/11, June 2002.

[11] N. Johansson, F. Alriksson and U. Jonsson, “JUMP Mode – A Dynamic Window-based Scheduling Framework for Bluetooth Scatternets”, *Proc. ACM MOBIHOC’01*, Oct. 2001.

[12] N. Johansson, U. Korner and L. Tassiualas, “A Distributed Scheduling Algorithm for Bluetooth Scatternet”, *Proc. ITC’17*, Dec. 2001.

[13] P. Johansson, R. Kapoor, M. Kazantzidis and M. Gerla, “Rendezvous Scheduling in Bluetooth Scatternets”, *Proc. IEEE ICC’02*, Apr. 2002.

[14] P. Johansson, M. Kazantzidis, R. Kapoor and M. Gerla, “Bluetooth: An Enabler for Personal Area Networking”, *IEEE Network*, Vol. 15, pp. 28-37, Sep./Oct. 2001.

[15] A. Racz, G. Miklos, F. Kubinszky and A. Valko, “A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets”, *Proc. ACM MOBIHOC’01*, Oct. 2001.

[16] U. Yechiali, “Analysis and Control of Polling Systems”, in *Performance Evaluation of Computer and Communication Systems (eds: L. Donatiello and R. Nelson)*, Springer-Verlag, pp. 630-650, 1993.

[17] W. Zhang and G. Cao, “A Flexible Scatternet-wide Scheduling Algorithm for Bluetooth Networks”, *Proc. IEEE IPCCC’02*, Apr. 2002.

[18] G. Zussman and A. Segall, “Capacity Assignment in Bluetooth Scatternets – Analysis and Algorithms”, *Proc. IFIP-TC6 Networking 2002*, *LNCS 2345 (eds: E. Gregori et al.)*, Springer-Verlag, May 2002.

[19] G. Zussman and A. Segall, “Bluetooth TDD – Analysis as a Polling System”, *working paper*, Aug. 2002.

[20] G. Zussman and A. Segall, “Capacity Assignment in Bluetooth Scatternets – Optimal and Heuristic Algorithms”, *To appear in ACM/Kluwer Mobile Networks and Applications (MONET)*, Special Issue on Advances in Research of WPAN and Bluetooth Enabled Networks.