# Adaptive Management of Bluetooth Slave/Slave Bridge

Jelena Misic, Vojislav B. Misic

# Adaptive Management of Bluetooth Slave/Slave Bridge

Jelena Mišić and Vojislav B. Mišić

*Abstract*— **This paper discusses the minimization of end-to-end delays in a small Bluetooth scatternet with a Slave/Slave bridge. We assume that both piconets operate under exhaustive scheduling policy, that slaves generate packet bursts, that burst arrivals follow Poisson distribution, and that the size of packet bursts follows geometric distribution. Using the theory of queues with vacations, we investigate the possibility of choosing scatternet parameters so as to minimize the aforementioned delays. We show that the minimization of delay should use the time interval between bridge exchanges in order to minimize inter-piconet traffic, since the intra-piconet traffic delay is less sensitive to the actual value of the time interval between bridge exchanges. Then, we present a load adaptive algorithm that would change this time dynamically, and show how the end-to-end delay is indeed minimized in this manner. Simulation results confirm that the algorithm is able to achieve minimum end-to-end delay for inter-piconet traffic under a wide range of traffic loads.**

*Keywords*— **Bluetooth scatternet, Slave/Slave bridge, performance evaluation, queueing theory, delay minimization**

## I. Introduction

Bluetooth is quickly gaining acceptance as a communication technology of choice for short range ad hoc networking [19]. However, the Bluetooth specification [4] is far from being finished and complete, and a number of important issues have not yet been resolved. Some among them are related to the operation of simple networks or piconets: for example, the scheduling policy to be used when the master polls its slaves. Other, probably more important, issues are related to the operation of Bluetooth scatternets, more complex networks in which two or more piconets share a device: formation of scatternets and scheduling of shared devices, to name just two. In particular, issues related to performance of Bluetooth scatternets are still not investigated in sufficient detail.

Initial attempts to deal with the problem of inter-piconet scheduling have focused on the feasibility and fairness, and paid little attention to performance. For example, in [11] it has been shown that constructing an optimal schedule (i.e., the schedule that will maximize the overall throughput) in a Bluetooth scatternet is an NP-complete problem. A distributed algorithm was proposed that adapts to changes in topology, yet can execute in polynomial time.

The scatternet scheduling algorithm proposed in [3] provides fair bandwidth allocation through a credit-based scheme. However, the algorithm does not scale well when the number of piconets in the scatternet increases, due to the lack of coordination between the nodes. The Maximum Distance Rendezvous Point (MDRP) algorithm described in [12] and further elaborated in [15] introduces the concept of superframe, and strives to distribute rendezvous points in an uniform fashion within the superframe. Both algorithms use the Bluetooth SNIFF mode, which does pose some limitations on their ability to adapt to varying traffic conditions.

J. Mišić and V. B. Mišić are with The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China.

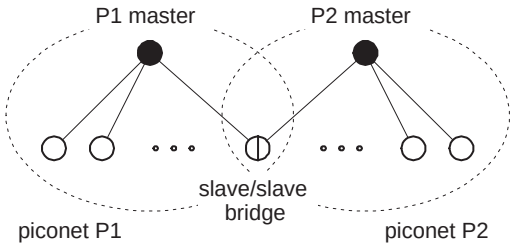



Fig. 1. The Bluetooth scatternet with an SS bridge.

On a different note, the pseudo-random scheme proposed in [23] utilizes randomly chosen checkpoints for possible exchanges between the piconet masters and shared devices (bridges). A similar approach was used in [9], where a new Bluetooth mode specifically geared towards scatternet support has been described. In both cases, the frequency of the checkpoints is adjusted to match the intensity of the traffic, but in a rather coarse manner; again, there are no provisions for coordination between different piconets. Furthermore, delays can be excessive because the algorithms allows checkpoints to be missed.

Performance evaluation of Bluetooth scatternets and scheduling under varying load do not seem to be very popular research issues, since only a handful of papers deal with them. One notable exception is the Load Adaptive Algorithm (LAA) proposed in [8], where the operation of the bridge is adjusted, in each exchange cycle, to match the current traffic, as estimated through a simple algorithm. A detailed analysis has revealed some flaws that seriously affect the performance of this algorithm, but this will be reported in a forthcoming paper [21].

Overall, some results do exist, but the problem is far from being solved in a satisfactory manner and a lot of work remains to be done. In this paper, we focus on the operation of a Bluetooth scatternet with a so-called Slave/Slave (SS) bridge. In this topology, the shared device acts as a slave in both piconets it links, as shown in Fig. 1. We will assume that the piconet masters poll their respective slaves using the scheduling discipline known as exhaustive service, for reasons to be explained below. The operation of such a scatternet will be analyzed using the

theory of queues with vacations [24], based on our previous work on performance evaluation of simple Bluetooth scatternets [20].

We will analyze the impact of scatternet traffic parameters on end-to-end packet delays for both intra- and inter-piconet traffic. These delays can be minimized through proper choice of bridge parameters, in particular the time interval between bridge exchanges. We will then look into the possibility of controlling this time interval in an adaptive manner, so as to keep the inter-piconet traffic delay as low as possible. Such management is indeed possible, and satisfactory performance can be achieved under a wide range of traffic parameter values. The validity of our approach will be confirmed through simulations.

The paper is organized as follows: in Sec. II, we describe the operation of a Bluetooth scatternet with an SS bridge and present the queueing model that will be used for subsequent analyses. Next Section presents a detailed queueing theoretic analysis of relevant delay variables under exhaustive service scheduling. We also present the solutions of the analytical model and discuss the impact of scatternet and traffic parameters on packet delays. Sec. IV discusses possible ways to adjust the values of bridge parameters so that the values of end-to-to end delays are minimized. Finally, Sec. V concludes the paper and outlines some promising avenues for further research.

## II. The system model

In the scatternet topology with an SS bridge, the bridge device acts as slave in both piconets: it spends some time $T_1$ in one of them, then switches to the other one for $T_2$, and back again. Whenever the bridge joins a piconet (a rendezvous event [12]), it starts exchanging packets with the piconet master. Once the exchange is finished, the master returns to local operation (i.e., it services regular slaves in its piconet) and the bridge stays idle until the next exchange. Meanwhile, local operation in the other piconet goes on uninterrupted until the next rendezvous with that piconet. This is shown schematically in Fig. 2.

We will assume that the operation of the scatternet may be analyzed using a simple queueing model, in which each slave device maintains a single uplink queue, whereas each piconet master maintains a number of downlink queues (one per slave), plus a queue for packets to be sent to destinations in the other piconet (bridging queue). The bridge device has two such queues, one for traffic that flows from $P1$ to $P2$, and another one for the traffic flowing in the opposite direction.

The packet exchange between either piconet master and the bridge device should be done exhaustively, i.e., until all queued packets in both the master and the bridge device are exchanged. The rationale for this is simple: since a piconet master controls all communications (actually, it participates in each of them), no local communication is possible during the period of bridge exchange. Therefore, the bridge exchange should be as short as possible, and that necessitates the use of exhaustive scheduling.

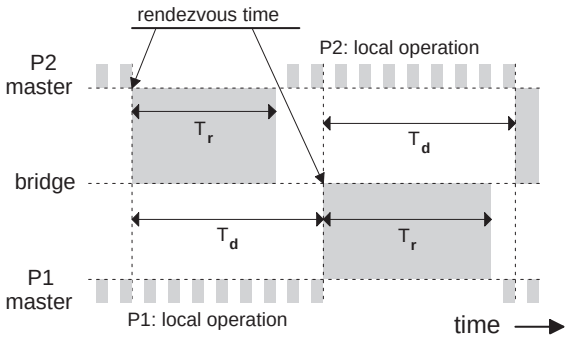



Fig. 2. Operation of the Bluetooth scatternet with an SS bridge.

The exchange ends when there are no more data packets to send, which is detected by a NULL packet received from the bridge in response to a POLL packet sent from the master [4]. Alternatively, since the number of packets in either device is known and will not change during the exchange, the actual number of packets to be exchanged may be determined before the actual packet exchange starts.

The exchange may also end if the bridge has an appointment in the other piconet. In such cases, some packets may be left in the bridge or the current master, or both, to be exchanged in the next cycle. Alternatively, the master of the other piconet may wait until the bridge ends the current exchange, but this is less than desirable because of the performance penalty noted above.

At the end of an exchange, the bridge and the current master must negotiate their next rendezvous. It is desirable that both participants join the exchange at the same time; otherwise, the first one to arrive will have to wait idle. The piconet master should not wait idle because of performance issues mentioned above.

Such negotiation may be implemented using the Bluetooth HOLD mode or, alternatively, SNIFF mode [4]. In both cases, the request can be made by either participant, master or bridge. However, the use of HOLD mode is preferred because the HOLD timeout (i.e., the time interval between successive exchanges) may be changed from one exchange to another, whereas the SNIFF period is fixed, and thus better suited to periodic tasks.

Whenever the bridge switches from one piconet to another, some time may be lost for synchronization: first, the bridge has to re-synchronize its clock to that of the piconet master [4]. The actual duration of the clock sync time required for synchronization may take any value from 0 to $2T$, with mean value equal to $T$ [2]. Furthermore, as Bluetooth packets may be one, three, or five packets long, additional time may be needed to finish the current frame before switching. This time may take any value from 0 to $8T$ (either way) and its mean value is not greater than mean packet length. Therefore, both of these delays are small compared to other delays in the scatternet, and hence will not have any noticeable effect on the results of our analyses.

### A. Intra-piconet scheduling

Another issue that impacts the performance of Bluetooth networks is the scheduling policy within the piconet itself. Although a number of such policies may be used [24], it is well known that optimal performance in a polling system may be obtained using scheduling policy known as Stochastically Largest Queue (SLQ) [18]. However, this policy requires that "the lengths of all queues are known at all times" to the server (i.e., piconet master), and the next queue (i.e., master-slave channel) to be serviced is the one which has the highest sum of master and slave queues [5].

Unfortunately, this policy cannot be easily applied in the Bluetooth environment. The Bluetooth packet structure has no provisions for simple exchange of relevant information: packet headers do not have any fields to carry this information, nor are there any spare bits to be used to that effect [4]. As a consequence, if the information on queue status is to be exchanged between the master of a piconet and its slaves, it will have to be done at the expense of actual packet payload. This may not seem too high a price to pay from the viewpoint of higher layers of the protocol stack, which by default incur some administrative overhead. However, at the Bluetooth MAC level, it may be more practical to focus on policies that do not require the master to have any knowledge of the length of slaves' queues [5].

In that case, best performance for polling systems that are symmetrical (i.e., the same traffic is offered to all queues) and non-idling (i.e., the master does not repeatedly poll the slave with an empty queue) can be obtained by using exhaustive service scheduling [17], [18]. Under this policy, the master stays with one slave as long as there are packets to send either way, and moves on to next slave only when both downlink and uplink queues for the current one are empty, as detected via a POLL-NULL frame. The next slave to be polled is determined through a simple round-robin, or cyclical, algorithm.

Indeed, previous analyses of Bluetooth piconets have demonstrated the superiority of exhaustive service over other scheduling policies [5], [6], [10], [13], [14], [16]. This is the rationale for using the exhaustive service policy for intra-piconet scheduling in the work reported in this paper.

## III. Calculating the Scatternet Delays

Scatternet performance will be mainly determined by queueing delays, since packets are queued at each device they pass through: slave, master, and (possibly) bridge. Our main performance indicators will be the end-to-end packet delays for local (i.e., intra-piconet) and non-local (i.e., inter-piconet) traffic.

- The end-to-end delay for local traffic consists of access delay, the time a data packet has to wait in the uplink queue of the source device before it is serviced, and queueing delay at the master, which stands for the delay a packet has to wait in the appropriate downlink queue before being sent to its destination.
- The end-to-end delay for non-local traffic includes both of the aforementioned components, as well as the queueing delays incurred in the queues to and from the bridge.

Many authors model the traffic using Poisson distributed packet arrivals in their analyses on account of the tractability of such models, even though they are known to be inadequate for real life data traffic [22]. In order to achieve more realistic results, we will assume that packets are generated in bursts or batches that arrive in Poisson distributed time intervals with arrival rate of $\lambda$. This approach corresponds to the case where Bluetooth is used as the data link layer for IP traffic: since IP PDUs are longer than Bluetooth ones, a single IP PDU will have to be segmented into several Bluetooth packets, thus giving rise to a packet burst.

We assume that the arrivals in the downlink and bridge queues follow Poisson distribution as well, and that downlink, uplink, and bridge queues operate independently from one another. Although these assumptions are not entirely correct, they do simplify the analysis without sacrificing accuracy, as the agreement of analytical solutions with simulation results vividly confirms.

The length of the burst follows the probability distribution that may be described with a probability generating function (PGF) $G_b(x) = \sum_{k=0}^{\infty} b_k x^k$, where $b_k$ is the probability that the burst will contain exactly $k$ packets [7]. (We will also need the mean value of the burst length $\overline{B} = G_b'(1)$.)

In this work, we assume that the packet burst length follows a geometric distribution; other, possibly more realistic distributions may easily be incorporated into our analysis framework, provided their first and second moments are known.

All packet generators in a piconet share the same value for traffic locality $P_l$, i.e., the probability that both the source and destination of the burst will be in the same piconet. Of course, all packets within a single burst have the same destination, and all destinations are chosen with equal probability.

The probabilities of packet length being one, three, and five slots are $p_1$, $p_3$, and $p_5$, respectively; of course, $p_1 + p_3 + p_5 = 1$, since other packet lengths are not allowed in Bluetooth [4]. The corresponding PGF is $G_p(x) = p_1 x + p_3 x^3 + p_5 x^5$, and first and second moments of the packet length distribution are $\overline{L} = G_p'(1)$ and $\overline{L^2} = G_p''(1) + G_p'(1)$, respectively.

The queueing theoretic analysis presented in this paper will make use of the equivalent Laplace-Stieltjes transform (LST) of the probability distributions. The LST may be obtained from the corresponding PGF by substituting the variable $x$ with $e^{-s}$ [24]; for example, the LST of the packet burst length PDF is $G_b^*(s) = \sum_{k=0}^{\infty} b_k e^{-ks}$, and the LST of the packet length PDF is $G_p^*(s) = p_1 e^{-s} + p_3 e^{-3s} + p_5 e^{-5s}$.

*A. Access delays*

Let us assume that the two piconets have $m_1$ and $m_2$ members, respectively, which means there will be $m_1 - 2$ and $m_1 - 2$ proper slaves. For simplicity, we assume that only proper slaves generate and receive traffic, but simple adjustments of arrival rates can easily cater for the case in which one or both of the masters generate or receive traffic as well. Then, the burst arrival rates will be $\lambda_{b12} = (m_1-2)\lambda(1-P_l)$ and $\lambda_{b21} = (m_2-2)\lambda(1-P_l)$, for traffic flows from $P1$ to $P2$, and from $P2$ to $P1$, respectively.

The PGF for the length of the exchange time between either piconet master and the bridge may be written as

$$G_r(x) = G_e(x)G_{e0}(x)x^2 \tag{1}$$

and its mean value is $\overline{T_r} = G'_r(1)$. In the last expression, $G_e(x)$ denotes the PGF of the distribution of the number of exchanged data packets:

$$G_e(x) = e^{(\lambda_{b12}+\lambda_{b21})2T_d(G_b(G_p(x))-1)} \tag{2}$$

and $G_{e0}(x)$ denotes the PGF of the distribution of the number of empty (i.e., POLL and NULL) packets sent during the exchange:

$$\begin{aligned} G_{e0}(x) = \sum_{n=0}^{\infty} e^{-(\lambda_{b12}+\lambda_{b21})T_d} \\ \sum_{k=0}^{\infty} \left( \frac{(\lambda_{b21}2T_d)^k}{k!} \frac{(\lambda_{b12}2T_d)^{k+n}}{(k+n)!} \right. \\ \left. + \frac{(\lambda_{b12}2T_d)^k}{k!} \frac{(\lambda_{b21}2T_d)^{k+n}}{(k+n)!} \right) G_b^n(x) \end{aligned} \tag{3}$$

In order to determine the mean access delay, we need the mean duration of piconet service cycle, the time interval for a piconet master to service all of its slaves once. Under exhaustive service scheduling, a single visit to a slave may take several frames, the actual number which is equal to the number of packets in the downlink queue or the number of packets in the corresponding uplink queue, whichever is larger, plus one. Let $X_c$ denotes the service cycle time in a piconet; the PGF of this cycle time is $G_{X_c}(x)$, while its mean value can be obtained as

$$\overline{X_c} = G'_{X_c}(1) \tag{4}$$

Let $X_{ms}$ denote the time to serve a single channel in a piconet, i.e., the time to empty both channel queues for one particular slave; its PGF is:

$$G_{X_{ms}}(x) = G_c(x)G_{c0}(x)x^2 \tag{5}$$

where $G_c(x)$ stands for the PGF of the distribution of the number of data packets exchanged between the master and a slave:

$$G_c(x) = e^{(\lambda_{u1}+\lambda_{d1})G_{X_c}(x)(G_b(G_p(x))-1)} \tag{6}$$

and $G_{c0}(x)$ denotes the PGF of the distribution of the number of empty packets sent during that same time:

$$\begin{aligned} G_{c0}(x) = \sum_{n=0}^{\infty} e^{-(\lambda_{u1}+\lambda_{d1})G_{X_c}(x)} \\ \sum_{k=0}^{\infty} \left( \frac{(\lambda_{u1}G_{X_{c1}}(x))^k}{k!} \frac{(\lambda_{d1}G_{X_c}(x))^{k+n}}{(k+n)!} \right. \\ \left. + \frac{(\lambda_{d1}G_{X_c}(x))^k}{k!} \frac{(\lambda_{u1}G_{X_c}(x))^{k+n}}{(k+n)!} \right) G_b^n(x) \end{aligned} \tag{7}$$

In all these expressions, $\lambda_{u1} = \lambda$ and $\lambda_{d1} = \lambda P_l + \lambda_{b21}/(m_1 - 2)$, while $X_c$ denotes the service cycle time in a piconet.

The piconet cycle time is the sum of channel service times and the bridge exchange time. The PGF of the bridge exchange time (within a single piconet cycle) is equal to:

$$G_{rc}(x) = \frac{\overline{X_c}G_r(x)}{2T_d - \overline{T_r}} + \left(1 - \frac{\overline{X_c}}{2T_d - \overline{T_r}}\right) \tag{8}$$

Then, the PGF for the piconet cycle time is

$$G_{X_c}(x) = G_{X_{ms}}^{m_1-2}(x)G_{rc}(x) \tag{9}$$

By solving the system of equations (5) to (9), we can obtain closed form expression for the PGF of the cycle time.

In queueing theory, the concept of vacation may be applied to a system in which multiple queues are serviced by a single server [24]. When the server services a client according to the chosen service discipline, it then goes on to service other clients, and thus becomes unavailable to that particular client. From the viewpoint of that client, the server takes a *vacation*, which lasts until next visit to the client. (If the client queue is empty at the time of next server visit, the server will immediately start the new vacation.) The vacation time is, then, the time while the server is busy servicing other client queues. The duration of the vacation period $V$ may be described with the following PGF: Then, the PGF for the server vacation time can be obtained as:

$$\begin{aligned} G_V(x) &= x^2 G_{X_{ms}}^{m_1-3}(x)G_{rc}(x) \cdot \\ & \quad e^{\lambda_{d1}G_{X_c}(x)(G_b(G_p(x))-1)} \end{aligned} \tag{10}$$

We will also need to determine the time it takes to service a single packet – the packet service time [24]. In an isolated piconet, the packet service time is proportional to the packet length. In a scatternet, however, this time will be extended due to the fact that local piconet operation is periodically (i.e., once every $2T_d$) interrupted by the bridge exchange. The corresponding PGF then becomes:

$$\begin{aligned} G_{pe}(x) = G_p(x)G_r(x)\frac{\overline{L}}{2T_d - \overline{T_r}} \\ + G_p(x)\left(1 - \frac{\overline{L}}{2T_d - \overline{T_r}}\right) \end{aligned} \tag{11}$$

and its first and second moments are $\overline{L_e} = G'_{pe}(1)$ and $\overline{L_e^2} = G''_{pe}(1) + G'_{pe}(1)$, respectively.

By substituting $e^{-s}$ in place of $x$ in the previously derived PGFs we obtain the LST-s for the individual components of the access delay. The LST of the overall access delay distribution will, then, be:

$$W^*_{a1}(s) = \frac{1 - V^*(s)}{s\overline{V}} \cdot \frac{1 - G_b(G^*_{pe}(s))}{\overline{B}\,(1 - G^*_{pe}(s))} \cdot \frac{s(1 - \lambda_{u1}\overline{B}\;\overline{L_e})}{s - \lambda_{u1}\left(1 - G_b(G^*_{pe}(s))\right)} \quad (12)$$

Mean access delay can be obtained from (12) as $\overline{W_{a1}} = -(W^*_{a1})'(0)$, which evaluates to:

$$\overline{W_{a1}} = \frac{(\lambda_{u1}\overline{B}^2\;\overline{L_e^2} + \overline{B^{(2)}}\;\overline{L_e}}{2\overline{B}(1 - \lambda_{u1}\overline{B}\;\overline{L_e})} + \frac{\overline{V^2}}{2\overline{V}}$$

where $\overline{B^{(2)}} = E[B(B-1)] = G''_b(1)$ denotes the second factorial moment of burst size distribution [24].

### *B. End-to-end delays*

The calculation of end-to-end delay is slightly more involved, as two distinct cases may be observed. If both the source and destination nodes of a packet are in the same piconet, then its total delay is the sum of access delay at the slave and the queueing delay at the master, and the LST of this delay is $W^*_{11}(s) = W^*_{a1}(s)W^*_{m1}(s)$. The LST for the delay at the master has the same form as for the access delay, $W^*_{a1}(s) = W^*_{m1}(s)$. As the queueing delay in piconet $P2$ may be described with a similar expression, the mean end-to-end delay for local traffic will be

$$\overline{W_{11}} = \overline{W_{a1}} + \overline{W_{m1}} \quad (13)$$

$$\overline{W_{22}} = \overline{W_{a2}} + \overline{W_{m2}} \quad (14)$$

in piconets $P1$ and $P2$, respectively.

Thanks to the exhaustive service intra-piconet scheduling, the burstiness of traffic in the downlink queues is essentially equal to that in the uplink queues. A burst might be interrupted by the bridge exchange, but the transfer will be resumed afterwards, and such burst will actually reappear in the corresponding downlink queue. Therefore, the queueing delay at the master is equal to the access delay at the slave, $W^*_{m1}(s) = W^*_{a1}(s)$.

If the source and destination of a packet reside in different piconets, the packet has to pass through the bridge. In this case, the overall delay consists of no less than four different queueing delays. Since the packets in the bridge are served exhaustively, the LST for the waiting time at the bridging queue of $P1$ master is

$$W^*_{m1b}(s) = \frac{1 - V^*_T(s)}{s\overline{V_T}} \cdot \frac{1 - G_b(G^*_p(s))}{\overline{B}(1 - G^*_p(s))} \cdot \frac{s(1 - \lambda_{b12}\overline{B}\;\overline{L})}{s - \lambda_{b12} + \lambda_{b12}G_b(G^*_p(s))} \quad (15)$$

where $V^*_T(s) = T^*_d(s)^2/G^*_r(s)$ and $\overline{V_T} = 2T_d - \overline{T_r}$; also, $T^*_d(s) = \overline{T_d}/s$.

Packets have to wait in the outgoing queue at the bridge until they are exchanged with the master of the other piconet; the LST for this component of the delay is

$$W^*_{b2m}(s) = \frac{1 - G_b(G^*_p(s))}{\overline{B}(1 - G^*_p(s))} \cdot \frac{1 - V^*_T(s)}{s\overline{V_T}} \cdot \frac{s(1 - \lambda_{b12}\overline{B}\;\overline{L})}{s - \lambda_{b12} + \lambda_{b12}G_b(G^*_p(s))} \quad (16)$$

Finally, the waiting time in the downlink queue at the other master is equal to the access delay.

Corresponding components of the mean queueing delay are:

$$\overline{W_{m1b}} = \frac{\lambda_{b12}\overline{B}^2\overline{L^2} + \overline{B^{(2)}}\;\overline{L}}{2\overline{B}(1 - \lambda_{b12}\overline{B}\;\overline{L})} + \frac{\overline{V_T^2}}{2\overline{V_T}} \quad (17)$$

$$\overline{W_{b2m}} = \frac{\lambda_{b12}\overline{B}^2\overline{L^2} + \overline{B^{(2)}}\;\overline{L}}{2\overline{B}(1 - \lambda_{b12}\overline{B}\;\overline{L})} + \frac{\overline{V_T^2}}{2\overline{V_T}} \quad (18)$$

where $\overline{L^2} = G''_p(1) + G'_p(1)$ denotes the second moment of packet length distribution.

The overall LST for the distribution of end-to-end delay of non-local traffic going from $P1$ to $P2$ is, then, $W^*_{12}(s) = W^*_{a1}(s)W^*_{m1b}(s)W^*_{b2m}(s)W^*_{m2}(s)$. Consequently, the mean end-to-end delay time for packets flowing from $P1$ to $P2$ will be:

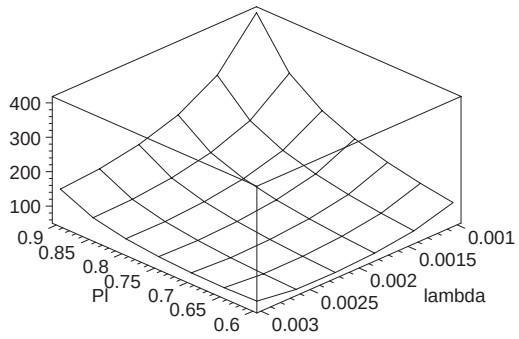
$$\overline{W_{12}} = \overline{W_{a1}} + \overline{W_{m1b}} + \overline{W_{b2m}} + \overline{W_{m2}} \quad (19)$$

An analogous expression may be written for the delay of the packets traveling in the opposite direction.

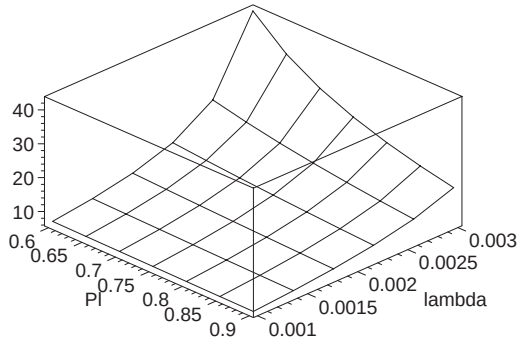
### *C. Minimizing the end-to-end packet delays*

It is worth investigating whether it would be possible to choose the values of different traffic and scatternet parameters so as to minimize one or the other end-to-end delay (or, preferably, both). First, we note that the number of adjustable parameters is rather small. Some of them, such as packet burst arrival rate, burst length distribution, or traffic locality, are determined by the applications that communicate using Bluetooth, which is beyond control (at the MAC level). Other traffic parameters, such as mean burst length and packet length distribution, might be more amenable to control through appropriate segmentation and reassembly policies. These policies are handled at the higher layers of the Bluetooth protocol stack, and probably cannot be changed too frequently, let alone dynamically. Finally, the size of the scatternet may be controlled indirectly, through the scatternet formation algorithm(s). Once the scatternet is formed, the time to re-structure it is prohibitively long [19], and higher transmission delays might, in fact, be acceptable – as a lesser evil, though.

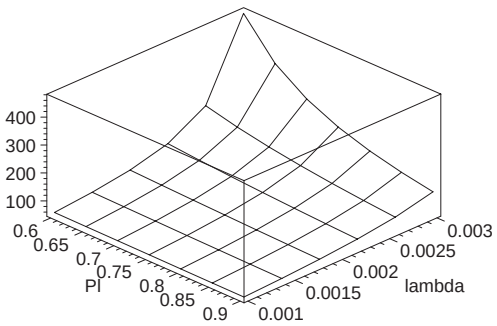
This leaves us with the time between successive bridge exchanges, $T_d$, as probably the only scatternet parameter that can be changed as often as necessary. Our objective is, then to investigate (a) whether it is possible to modify



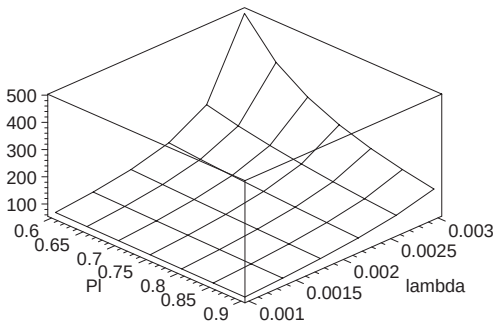

(a) Loci of $T_d$ values that minimize end-to-end delay for local traffic. (Note that the orientations of both $\lambda$ and $P_l$ axes have been changed for easier viewing.)



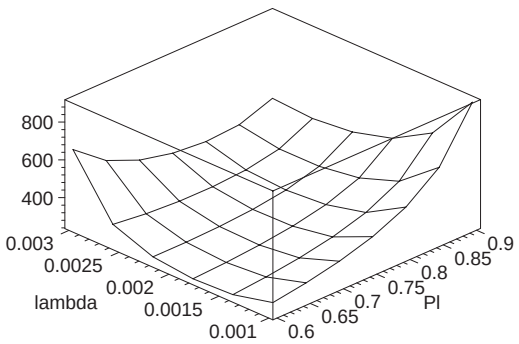

(b) Loci of $T_d$ values that minimize end-to-end delay for non-local traffic.



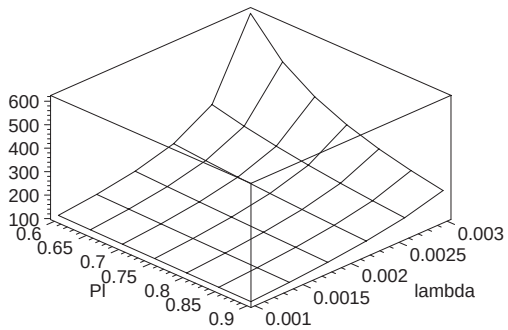

(c) Minimum end-to-end delay for local traffic.




(d) End-to-end delay for local traffic when non-local delay is minimized.




(e) End-to-end delay for non-local traffic when local delay is minimized. (Note that the orientations of both $\lambda$ and $P_l$ axes have been changed for easier viewing.)



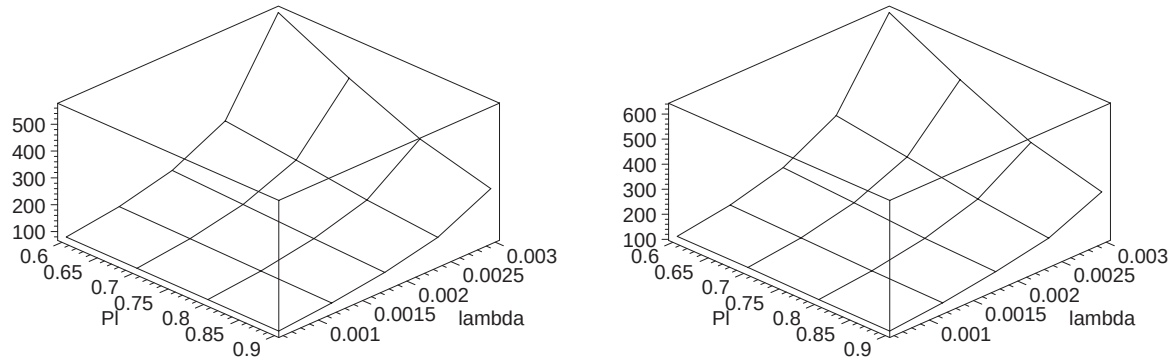

(f) Minimum end-to-end delay for non-local traffic.

Fig. 3. Minimizing end-to-end packet delay for local traffic (left-side) and non-local traffic (right-hand side) in the scatternet with an SS bridge.

the time interval $T_d$ in order to reduce end-to-end packet delays, and (b) which of the delays we should strive to minimize, local or non-local, or maybe a combination of the two.

The answer to the first question is affirmative: the minimum of the local end-to-end delay may be found from $\partial\overline{W_{11}}/\partial T_d = 0$, while keeping $\lambda$, $\overline{B}$, and $P_l$ as parameters. The resulting equation can be solved using numerical methods, and the solutions are shown in the left-hand column of Fig. 3. If, on the other hand, we want to minimize the non-local end-to-end delay, we should solve the equation $\partial\overline{W_{12}}/\partial T_d = 0$; the results are shown in the right-hand column of Fig. 3. All diagrams were obtained in a scatternet with two piconets, each of which had $m_1 = m_2 = 8$ members (i.e., there were six packet-generating slaves). We have assumed mean burst length of $\overline{B} = 5$ and mean packet length of $\overline{L} = 3$ (with $p_1 = p_2 = p_3 = 1/3$). Traffic locality was $P_l = 0.9$, with equal probability of all destinations within either piconet. All delays are expressed in units of the Bluetooth clock time slot $T = 0.625ms$.

Two important observations can be made from the diagrams. First, values of $T_d$ in the low range of 10 to $40T$ generally minimize non-local delay, while values in the range of 50 to $400T$ lead to minimal local delays. The explanation is simple: local delay will be smaller if the




(a) Local delay variation.

(b) Non-local delay variation.

Fig. 4. End-to-end packet delays in the SS bridge topology under fixed $T_d$ scheduling: hand-picked minima.

local operation in the piconet goes on uninterrupted for as long as possible. In other words, bridge exchanges should be performed as infrequently as possible. However, this causes very high delays for non-local traffic. Therefore, by minimizing the delay for local traffic we actually *penalize* non-local traffic – thus defeating the purpose with which the scatternet was formed in the first place.

On the other hand, shortening the time interval between bridge exchanges means that data packets with destinations in the other piconet are able to get through very quickly, thus reducing non-local delays. Frequent bridge exchanges do affect the performance of local traffic—the piconet master has to stop servicing its own slaves—but not much: the increase in local delays is small (less than 10%, in most cases) and should be perfectly acceptable.

Second, no single value for $T_d$ can be found that works well under the range of burst arrival rates and traffic locality. In other words, fixing $T_d$ to any given numerical value will result in non-optimal performance as soon as the traffic intensity changes. (Remember that Bluetooth scatternets are ad hoc networks, and the traffic flow may vary considerably, even during short time intervals.) Consequently, we should try to devise an adaptive algorithm that would adjust $T_d$ to the current traffic conditions, in order to keep the packet delays—in particular, the end-to-end-delay for non-local traffic—as close to the theoretical minimum values as possible.

In order to validate the analytical solutions, we have built a Bluetooth piconet simulator, using the object-oriented Petri Net-based simulation engine Artifex by Artis Software Inc. [1], running on a Linux platform. The simulator operates at the MAC level, and it contains separate classes for slave devices (with bursty packet generator), piconet masters, bridge device with two distinct interfaces and bridge logic, integrated through a top-level scatternet class that provides measurement capabilities. In order to eliminate possible transient effects at system start-up, all measurements were taken after an initial warm-up delay needed to bring the system to a steady state. The warm-up period lasted for one minute of equivalent time, and the actual measurements took five minutes (i.e., 480,000 Bluetooth time slots $T$).

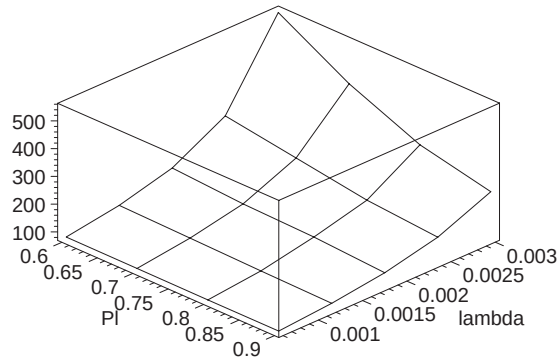
Our initial simulations have shown reasonable, but not very good agreement with analytical solutions. Upon close inspection, it was found that the discrepancy was due to the fact that all exchanges were allowed to run to completion. As explained in Sec. II above, this means that the master of the other piconet will have to wait for the rendezvous with the bridge, effectively freezing all communication in that piconet, and causing excessive delays for both local and non-local traffic. Once the simulator was modified to include provisions for terminating the exchange per force as soon as the rendezvous time comes, the agreement between the simulation results and analytical solutions has considerably improved.

The dependency of minimum end-to-end delays on packet burst arrival rate and traffic locality under such conditions is shown in Fig. 4. The values were obtained by running a series of simulation runs, each with a different value of $T_d$ (which was kept fixed during a single run), and hand-picking the value of $T_d$ that resulted in minimum end-to-end delay for non-local traffic. As can be seen, the delays are very close to the analytically obtained minima in Figs. 3(d) and 3(f).

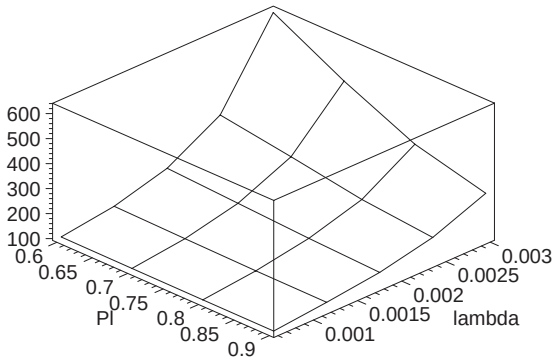
## IV. Adaptive Scheduling of Bridge Operation

In the previous Section, we have established—both analytically and through simulations—that (a) performance of the bridge can indeed be optimized, and (b) we should strive to minimize the delay for non-local traffic. It remains to find out whether this type of minimization can be accomplished in practice. An obvious solution would be to use the data on the minima from the previous Section, and determine the optimal values of $T_d$, using the dependencies of Fig. 3 in tabular or algorithmic form. These values could, then, be forwarded to the bridge by the respective masters. Of course, the estimates, as well as corresponding values for time intervals, should be periodically updated.

Albeit conceptually simple, this solution suffers from shortcomings at different levels. The minima are functions of scatternet, as well as of traffic, parameters . While scatternet parameters are known to the respective piconet




(a) End-to-end delay for local traffic.




(b) End-to-end delay for non-local traffic.

Fig. 5. End-to-end packet delays as functions of burst arrival rate and traffic locality under LASS scheduling (simulation results).

masters, traffic parameters are not directly available: at best, they would have to be estimated by the interested parties.

However, traffic statistics have to be collected by the piconet masters and forwarded to the bridge so that it can perform the estimation. (Alternatively, estimates could be made at the masters and forwarded to the bridge.) Either way, the master(s) need to negotiate the changes with the bridge; the time for such negotiations must come at the expense of actual data transfers. Since the piconet traffic is inherently bursty, such estimates would be inaccurate most of the time, certainly not a desirable feature.

A more attractive solution would be to enable the bridge device to schedule its own operation locally. Scheduling decisions should be made with as little input (from the respective piconet masters) as possible, and with as little negotiation (with those masters) as possible. In other words, the bridge should use some local performance measure to make its own scheduling decisions.

As might be expected, the main problem with this approach is to find a suitable measure. Packet delays—which we want to minimize in the first place—are not really appropriate, for a number of reasons. First, all delay measurements require averaging of some sort, which leads to nontrivial processing overhead. Second, queueing delays at the bridge are but a part of the total packet delay; other components of the total delay would have to be measured by the piconet masters and sent periodically to the bridge, which would incur even more delays.

Packet arrival rates at the bridge are readily available, but cannot easily be related to the performance. Furthermore, these arrival rates are actually aggregate values, and there is no simple way to extract the values of its individual components.

### *A. The LASS algorithm*

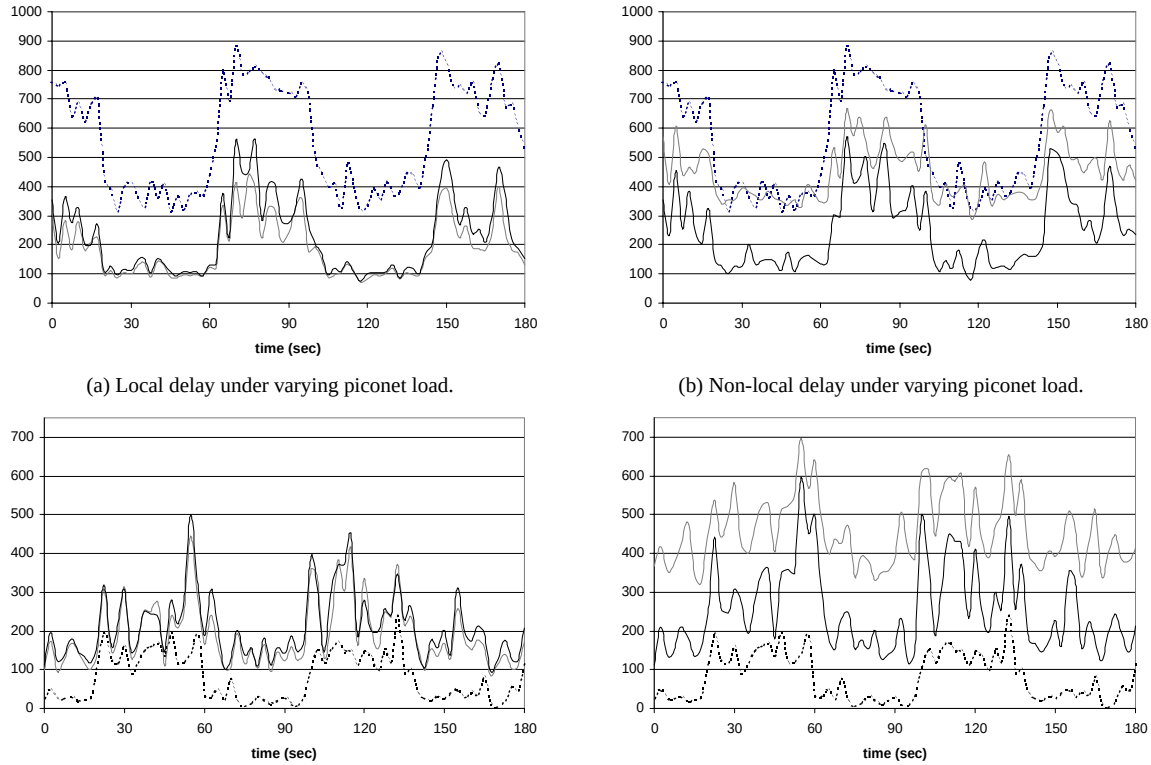
Fortunately, a simple solution can be found. As noted in Sec. II, excessive delays for both local and non-local traffic are caused by periods in which a piconet master is waiting for the bridge to join a pending rendezvous appointment. On the other hand, when a bridge exchange is terminated per force, some packets are left waiting in the corresponding queues, which increases the delays for non-local traffic. Both types of extra delays will be minimized by trying to set the scheduled duration of bridge exchanges (i.e., the time to next rendezvous) to their actual duration.

The manner in which the exchange terminates may be used, then, as a simple sensor variable to guide the adjustment of the rendezvous time for a simple scheduling algorithm.

- If all the packets have been exchanged and the rendezvous time has not been reached yet, the exchange terminates normally. This means that the allocated exchange time is sufficient, in fact more than sufficient, to handle the current (instantaneous) volume of inter-piconet traffic, and it can be shortened.
- Otherwise, if the rendezvous time is reached and some packets are still left in the corresponding queue(s), the exchange must be terminated per force. This means that the time interval between exchanges is insufficient to cater for the instantaneous volume of data traffic. Therefore, the next rendezvous should be scheduled farther away in future, and the time $T_d$ must be lengthened.

The algorithm, which will be referred to as LASS (Load Adaptive Slave/slave Scheduling), is outlined below in pseudocode. It is executed at the bridge, after each packet received from the current piconet master.

```
if rendezvous time reached or exchange complete
then if rendezvous time reached
        then if f = 1 then δ = δ11 else δ = δ10
             f' = 1
        else // exchange complete
             if f = 1 then δ = δ01 else δ = δ00
             f' = 0
        end if
        T'd = Td + 2δ
        if T'd < Tdmin then T'd = Tdmin
        // request HOLD with hold time T'd
        // current master acknowledges HOLD
        // switch to other piconet
else // continue with exchange
end if
```




(a) Local delay under varying piconet load.

(b) Non-local delay under varying piconet load.

(c) Local delay under varying bridge load.

(d) Non-local delay under varying bridge load.

Fig. 6. Large load variations in the SS bridge topology: comparing LASS with fixed $T_d$ scheduling.

As the next exchange takes place in the *other* piconet, adjustment of the time interval should apply to the subsequent exchange with the current piconet, i.e., the exchange after the next one. Therefore, the decision is based on the outcome of *two* last exchanges, not just one, and we need four increment values (i.e., $\delta_{xy}$) instead of just two. In our simulations, we found that the best results are obtained with $\delta_{00} = -2$, $\delta_{01} = 0$, $\delta_{10} = 1$, and $\delta_{11} = 3$.

Furthermore, there should be a lower limit for $T_d$: the efficiency drops sharply when $T_d$ approaches the sum of mean clock and frame synchronization times. In our simulations we used the value of $T_{dmin} = 12T$. No similar upper bound is necessary, because even after repeated bursts a period with no bridge traffic will eventually occur, and LASS is able to reduce $T_d$ quickly.

Fig. 5 shows the end-to-end delays for local and non-local traffic, respectively, as functions of packet burst arrival rate $\lambda$ and traffic locality $P_l$. The mean delays are virtually equal to the hand-selected minima presented in Fig. 4 – but the adaptive algorithm requires no manual tuning in a wide range of traffic parameters. Furthermore, the delays obtained under LASS scheduling are very close to the analytically obtained minima from the right-hand column of Fig. 3.

To demonstrate the ability of the LASS scheduling algorithm to adjust to load variations, we have plotted the delays averaged at $2.5s$ ($4000T$) in Fig. 6. End-to-end delays for local and non-local traffic under a periodic change of scatternet load in the range 1:2 are shown in Figs. 6(a) and 6(b), respectively. The change in scatternet load was obtained by varying the packet burst arrival rate for half of the slaves in each piconet.

Moreover, we have plotted the end-to-end delays for local and non-local traffic under a periodic change of bridge load in the range 1:5 in Figs. 6(c) and 6(d), respectively. The change in bridge load was obtained by varying the traffic locality for some of the packet-generating slaves, while keeping the packet burst arrival rate (and, henceforth, the total scatternet load) constant. In all diagrams, dotted lines denote the number of packets (i.e., the generated during that period of time, gray lines denote the average delays obtained under fixed $T_d$ scheduling with $T_d = 50T$, and solid black lines denote the average delays obtained under LASS scheduling.

As can be seen, the delays generally follow the instantaneous load, although the curves are not quite smooth. This is partly due to the chosen averaging interval of $2.5s$, but the main reason is actually traffic burstiness: packet burst arrivals follow a Poisson distribution, while burst lengths follow a geometric distribution. The local delay obtained with the LASS algorithm is below the one obtained with fixed $T_d$ scheduling most of the time, but the difference is rather small. However, the non-local delay obtained with the LASS algorithm is substantially smaller in a wide range of piconet and bridge loads, which vividly confirms the validity of the LASS approach.

## V. Summary and Directions for Further Research

In this paper we have analyzed the operation of the Bluetooth scatternet topology in which two piconets are linked via a common slave—the so-called Slave/Slave bridge—using the theory of queues with vacations. We have obtained the probability distribution of end-to-end delay times for both local (intra-piconet) and non-local (inter-piconet) traffic. We have analyzed the possibility of choosing bridge parameters, in particular the time interval between bridge exchanges, so as to minimize the end-to-end packet delay. Our analysis shows that minimization is indeed possible, and that the main criteria for minimization should be the end-to-end delay for inter-piconet traffic, rather than the corresponding delay for local traffic.

Then, we have designed an adaptive algorithm that adjusts the time interval between successive bridge exchanges to the current traffic parameters, and we have shown that it can indeed achieve delays that are as low as the minimum values obtained through queueing theoretic analysis. The algorithm is based on a simple sensor variable available to the bridge device, and it does not require extensive processing or communication overhead.

Future research might investigate whether more improvements in performance can be obtained through modifications to the algorithm. It could also address the behavior of more complex Bluetooth scatternets, which might include more than two piconets, as well as two or more bridges per piconet, and investigate whether the approach that works well in small scatternets can be utilized in more complex one networks as well.

## References


[1] Artis Software Inc. *Artifex v.4.2.1*. San Jose, CA, 2001.

[2] S. Baatz, M. Frank, C. Kühl, P. Martini, and S. Christoph. Adaptive scatternet support for Bluetooth using sniff mode. In *Proceedings of the 26th Annual Conference on Local Computer Networks LCN 2001*, Tampa, FL, Nov. 2001.

[3] S. Baatz, M. Frank, C. Kühl, P. Martini, and S. Christoph. Bluetooth scatternets: An enhanced adaptive scheduling scheme. In *Proceedings Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2002.*, pages 782–790, New York, June 2002.

[4] Bluetooth SIG. *Specification of the Bluetooth System*. Version 1.1, Feb. 2001.

[5] A. Capone, R. Kapoor, and M. Gerla. Efficient polling schemes for Bluetooth picocells. In *Proceedings of IEEE International Conference on Communications ICC 2001*, volume 7, pages 1990–1994, Helsinki, Finland, June 2001.

[6] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey. Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network. In *Proceedings Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2001.*, volume 1, pages 591–600, Anchorage, AK, Apr. 2001.

[7] G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, UK, 2nd edition, 1992.

[8] L. Har-Shai, R. Kofman, G. Zussman, and A. Segall. Inter-piconet scheduling in Bluetooth scatternets. In *Proc. OPNETWORK 2002 Conference*, Aug. 2002.

[9] N. Johansson, F. Alriksson, and U. Jönsson. JUMP mode – a dynamic window-based scheduling framework for Bluetooth scatternets. In *Proceedings 2001 ACM International Symposium on Mobile ad hoc networking & computing*, pages 204–211, Long Beach, CA, Oct. 2001.

[10] N. Johansson, U. Körner, and P. Johansson. Performance evaluation of scheduling algorithms for Bluetooth. In D. H. K. Tsang and P. J. Kuhn, editors, *Proceedings of BC'99 IFIP TC 6 Fifth International Conference on Broadband Communications*, pages 139–150, Hong Kong, Nov. 1999.

[11] N. Johansson, U. Körner, and L. Tassiulas. A distributed scheduling algorithm for a Bluetooth scatternet. In *Proceedings of the International Teletraffic Congress – ITC-17*, pages 61–72, Salvador de Bahia, Brazil, Sept. 2001.

[12] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla. Rendezvous scheduling in Bluetooth scatternets. In *Proceedings of IEEE International Conference on Communications ICC 2002*, pages 318–324, New York, Apr. 2002.

[13] M. Kalia, D. Bansal, and R. Shorey. MAC scheduling and SAR policies for Bluetooth: A master driven TDD pico-cellular wireless system. In *Proceedings Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)*, pages 384–388, San Diego, CA, Nov. 1999.

[14] M. Kalia, D. Bansal, and R. Shorey. Data scheduling and SAR for Bluetooth MAC. In *Proceedings VTC2000-Spring IEEE 51st Vehicular Technology Conference*, volume 2, pages 716–720, Tokyo, Japan, May 2000.

[15] M. Kazantzidis and M. Gerla. On the impact of inter-piconet scheduling in Bluetooth scatternets. In *The 3rd International Conference on Internet Computing IC 2002*, Las Vegas, NV, June 2002.

[16] A. Kumar, L. Ramachandran, and R. Shorey. Performance of network formation and scheduling algorithms in the Bluetooth wireless ad-hoc network. *Journal of High Speed Networks*, 10:59–76, Oct. 2001.

[17] H. Levy, M. Sidi, and O. J. Boxma. Dominance relations in polling systems. *Queueing Systems Theory and Applications*, 6(2):155–171, 1990.

[18] Z. Liu, P. Nain, and D. Towsley. On optimal polling policies. *Queueing Systems Theory and Applications*, 11(1–2):59–83, 1992.

[19] B. A. Miller and C. Bisdikian. *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications*. Prentice-Hall, Upper Saddle River, NJ, 2000.

[20] J. Mišić and V. B. Mišić. Bridges of Bluetooth county: topologies, scheduling, and performance. *to appear in IEEE JSAC – Wireless series*, 2003.

[21] V. B. Mišić, J. Mišić, and K. L. Chan. Performance of adaptive inter-piconet scheduling in a scatternet with a slave/slave bridge. *submitted to Wiley Wireless Communications and Mobile Computing Journal, special issue on Performance Evaluation of Wireless Networks*, 2003.

[22] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *ACM/IEEE Transactions on Networking*, 3(3):226–244, June 1995.

[23] A. Rácz, G. Miklós, F. Kubinszky, and A. Valkó. A pseudo random coordinated scheduling algorithm for Bluetooth scatternets. In *Proceedings 2001 ACM International Symposium on Mobile ad hoc networking & computing*, pages 193–203, Long Beach, CA, Oct. 2001.

[24] H. Takagi. *Queueing Analysis*, volume 1: Vacation and Priority Systems. North-Holland, Amsterdam, The Netherlands, 1991.