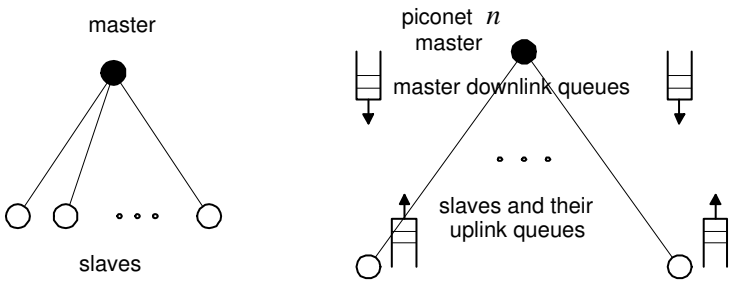
Intra-Piconet Polling Algorithms in Bluetooth

Jelena Mišić and Vojislav B. Mišić *

Bluetooth is an emerging standard for Wireless Personal Area Networks (WPANs): short range, ad hoc wireless networks [1]. Originally, Bluetooth was envisaged as a wireless cable replacement technique, which is why the basic RF range of Bluetooth devices is only about 10 meters [2]. However, the number of possible uses of Bluetooth have increased to include different networking tasks between computers and computer-controlled devices such as PDAs, mobile phones, smart peripherals, and others. However, Bluetooth networks operate in a rather different manner from other wireless networks. Furthermore, many important aspects of communication using Bluetooth networks are not defined by the current Bluetooth specification [2]. Such aspects include the manner in which the master polls its slaves, the scheduling algorithms for Bluetooth scatternets, the preferred scatternet topology (or topologies), and others. Quality of service is supported in a very limited manner, too. All of these aspects, however, play a crucial role in determining the performance of Bluetooth networks, which is ultimately one of the main criteria for their wider acceptance in the marketplace. Performance analysis of Bluetooth networks is an important research topic addressed by many researchers. Yet, because of the differences from other networks, known performance analysis results from other wireless networks cannot be directly applied in the Bluetooth environment. In this chapter, we will review and roughly classify existing algorithms for intra-piconet polling. We will also describe some of the topics that deserve future research attention.

*J. Mišić and V. B. Mišić are with the Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada.



(a) Topology.

(b) Queueing model of a single piconet.

Figure 1: Bluetooth piconet topology.

1 Introduction: on Bluetooth Networks and Bluetooth Communications

As mentioned above, Bluetooth is a communication technology for short range, ad hoc wireless networks formed by fixed or mobile devices. Bluetooth devices must form networks before the actual communication can start [3]. The simplest network is called piconet: a small, centralized network with up to eight active nodes or devices. One of the nodes is designated as the master, while the others are slaves. At most seven slaves can be active at any given time, and up to 255 others can be *parked* but still listening to the communications in the piconet. This topology is shown schematically in Fig. 1(a).

Bluetooth uses a set of RF frequencies (79 or 23, in some countries) in the ISM band at about 2.4GHz. Frequency Hopping Spread Spectrum (FHSS) technique is utilized in order to combat interference. Each piconet hops through the available RF frequencies in a pseudo-random manner. The hopping sequence, which is determined from the Bluetooth device address of the piconet master, is known as the channel [3]. Each channel is divided into time slots of $T = 625\mu s$, which are synchronized to the clock of the piconet master. In each time slot, a different frequency is used.

All communications in the piconet take place under the control of the piconet master. All slaves listen to downlink transmissions from the master. The slave may reply with an uplink transmission if and only if addressed explicitly by the master, and only immediately after being addressed by the

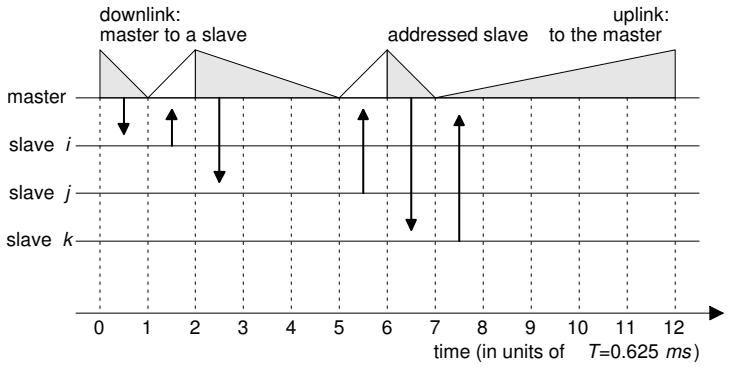


Figure 2: TDD master-slave communication in Bluetooth. Gray triangles denote data packets, white triangles denote empty (POLL and NULL) packets.

master. Data is transmitted in packets, which take one, three, or five slots; link management packets also take one slot each. The RF frequency does not change during the transmission of the packet. However, once the packet is sent, the transmission in the next time slot uses the next frequency from the original hopping sequence (i.e., the two or four frequencies from the original sequence are simply skipped). By default, all master transmissions start in even-numbered slots, whilst all slave transmissions start in odd-numbered slots. A downlink packet and the subsequent uplink packet are commonly referred to as a frame. Therefore, the master and the addressed slave use the same communication channel, albeit not at the same time. This communication mechanism, known as Time Division Duplex, or TDD for short, is schematically shown in Fig. 2.

Because of the TDD communication mechanism, all communications in the piconet must be routed through the master. Each slave will maintain (operate) a queue where the packets to be sent out are stored. The master, on the other hand, operates several such queues, one for each active slave in the piconet. The corresponding queueing model is shown in Fig. 1(b). We note that these queues may not physically exist, e.g., all downlink packets might be stored in a single queue; but the queueing model provides a convenient modeling framework which facilitates the performance analysis of Bluetooth networks.

The master polls the slave by sending the data packet from the head of

the corresponding downlink queue. The slave responds by sending the data packet from the head of its uplink queue. When there is no data packet to be sent, single-slot packets with zero payload are sent – POLL packets in the downlink, and NULL packets in the uplink direction [3]. As the process of polling the slaves is actually embedded in the data transmission mechanism, we will use the term 'polling' for every downlink transmission from the master to a slave.

Since packets must wait at the slave and/or at the master before they can be delivered to their destinations, the delays they experience are mainly queueing delays. Therefore, the performance of the data traffic will be mostly dependent on the choice of the polling scheme used by the master to poll the active slaves in the piconet.

2 Intra-piconet polling schemes

The polling scheme is obviously the main determinant of performance of Bluetooth piconets, and one of the main determinants of performance of Bluetooth scatternets. As usual, the main performance indicator is the end-to-end packet delay, with lower delays being considered as better performance. There are, however, at least two other requirements to satisfy. First, the piconet master should try to maintain fairness among the slaves, so that all slaves in the piconet receive equal attention in some shorter or longer time frame. (Of course, their traffic load should be taken into account.) Second, Bluetooth devices are, by default, low power devices, and the polling scheme should be sufficiently simple in terms of computational and memory requirements.

As noted above, the current Bluetooth specification does not specifically require or prescribe any specific polling scheme [2]. This may not seem to be too big a problem, since optimal polling schemes for a number of similar single-server, multiple-input queueing system are well known [4, 5]. However, the communication mechanisms used in Bluetooth are rather specific, because

- all communications are bidirectional (i.e., there cannot exist a downlink packet without an uplink packet, or vice versa),
- the master polls the slaves using regular packets, possibly without data payload (i.e., all polls and responses thereto take at least one slot each),

- all slave-slave communications have to be routed through the master (i.e., there can be no direct slave-to-slave communication), and
- the master does not know the status of queues at the slaves, because there are no provisions for exchange of such information in the Bluetooth packet structure.

As the consequence, the existing results cannot be applied, and the performance of different polling schemes has to be re-assessed, taking the aforementioned characteristics of the Bluetooth communication mechanisms. It should come as no surprise, then, that a number of polling schemes have been proposed and analyzed [6, 7, 8, 9]. Many of the proposed schemes are simply variations of the well-known limited and exhaustive service scheduling [10], but several improved adaptive schemes have been described as well [8, 11].

In the discussion that follows, we will present a rough classification of those polling schemes, using the following criteria. First, the polling scheme determines the number of frames exchanged during a single visit to the slave. This number may be set beforehand to a fixed value, or it may be dynamically adjusted on the basis of current and historical traffic information.

Second, different slaves may receive different portions of the bandwidth; again, the allocation may be done beforehand, or it may be dynamically adapted to varying traffic conditions. The latter approach is probably preferable in Bluetooth piconets, which are ad hoc networks formed by mobile users, and the traffic may exhibit considerable variability. In fact, due to users' mobility, even the topology of the piconet may change on short notice. At the same time, the fairness of polling may be more difficult to maintain under dynamic bandwidth allocation.

Finally, the sequence in which slaves are visited may be set beforehand, or it may change from one piconet cycle to another, depending on the traffic information. Slaves that had more traffic in the previous cycle(s) may receive a larger portion of the available bandwidth. Slave that had no traffic may receive less bandwidth, or they may even be ignored for one or more piconet cycles. Again, the main difficulty with such schemes is to ensure that the fairness is maintained.

2.1 Traditional polling schemes

The simplest polling schemes use a fixed ordering of the slaves and fixed bandwidth allocation per slave. The only variable parameter, then, is the

duration of master's visit to each slave.

Under *1-limited service* polling, the master visits each slave for exactly one frame, and then moves on to the next slave [10]. Data packets are sent if there are any, otherwise empty packets (POLL or NULL) are sent. The scheme is sometimes referred to as (Pure) Round Robin [6] or simply limited service.

Under *exhaustive service* polling, the master stays with the slave as long as there are packets to exchange in either downlink or uplink direction [10]. The absence of packets is detected by a POLL-NULL frame.

Under the *E-limited service* polling, the master stays with a slave until there are no more packets to exchange, or for a fixed number M of frames ($M > 1$), whichever comes first [10]. Packets that arrive during the visit are allowed to enter the uplink queue at the slave and may be serviced – provided the limit of M frames is not exceeded [10]. This scheme is also referred to as Limited Round Robin [6, 11].

In fact, 1-limited and exhaustive service polling may be considered as special cases of E-limited service, where the limit M equals 1 and ∞, respectively. In all three cases, the sequence of slaves is fixed and does not change.

In traditional polling systems, exhaustive service performs better than either 1-limited or E-limited service [4]. As Bluetooth piconets are not traditional polling systems (for reasons outlined above), hence this result does not hold. Several authors have found that 1-limited performs better than exhaustive service under high load [6, 12]. Furthermore, E-limited service has been found to offer better performance than either limited or exhaustive service, and the value of M may be chosen to achieve minimum delays for given traffic burstiness [13].

2.2 Dynamic reordering of slaves

In fact, even better results may be obtained through the so-called Stochastically Largest Queue (SLQ) policy. Under this policy, the server always services the queue with the highest number of packets [5]. In other words, the master should always poll the slave for which the sum of lengths of uplink and downlink queues is the highest. However, this is not possible in Bluetooth, as there is no way for the master to know the current status of *all* of its slaves' uplink queues. What is possible, though, is to dynamically reorder the slave polling sequence according to the length of the corresponding

Table 1: An overview of traditional polling algorithms.

polling algorithm	performance aspect			
	access delay	end-to-end delay	adaptivity	fairness
1-limited	average	good (high loads)	none	inherent
exhaustive	best	good (low and medium loads)	none	can be unfair
E-limited	good	best	average	inherent (longer time scale)
EPM [6]	average	good (high loads)	downlink	can be unfair

downlink queues at the master. This reordering may be done once in each piconet cycle, or the master may always poll the slave with the longest downlink queue. In either case, polling may be performed according to limited, exhaustive, or even E-limited scheme.

One scheme that uses dynamic reordering is the *Exhaustive Pseudo-cyclic Master* (EPM) queue length scheme, proposed in [6], where each slave is visited exactly once per cycle. At the beginning of each cycle, however, the slaves are reordered according to the decreasing length of downlink queues.

One problem with dynamic reordering is that fairness among the slaves cannot be guaranteed when if reordering is done for every poll. For example, two slaves that talk to each other might easily monopolize the network and starve all other slaves. Fairness is easier to achieve when reordering is done on a per cycle basis. Still, some polling schemes (such as the exhaustive scheme) present more of a challenge than others.

Table 1 summarizes the characteristics of traditional polling schemes.

2.3 Adaptive bandwidth allocation

The duration of the visit to each slave may be adjusted according to the current or historical traffic information. This can be done in two ways: by

rewarding slaves that have more traffic, or by penalizing slaves that have less traffic. In the former case, the master is allowed to stay longer, and thus exchange more frames, with the slave that has some data to send and/or receive. In the latter case, the slave that had less traffic or no traffic at all, or the slave which is expected to have no traffic, will simply be ignored for a certain number of piconet cycles.

The former approach is exploited in the *Limited and Weighted Round Robin* (LWRR) [6], which tries to increase efficiency by reducing the rate of visits to inactive slaves. Initially, each slave is assigned a weight equal to the so-called maximum priority, or MP. Each slave is polled in E-limited fashion with up to M frames. Whenever there is a data exchange between the slave and the master, the weight of the slave is increased to the value of MP. On the other hand, when a POLL-NULL sequence occurs, the weight for that particular slave is reduced by one. If the slave weight drops to one (which is the lowest value), the slave has to wait a maximum of $MP - 1$ cycles to be polled again.

A variation of this scheme, labeled *Pseudo-Random Cyclic Limited slot-Weighted Round Robin* [11], uses both slave reordering and poll rate reduction. The sequence in which slaves will be polled is determined in a pseudo-random fashion at the beginning of every cycle, and less active slaves are not polled for a certain number of slots (not cycles). In addition, the maximum number of frames that may be exchanged during a single visit to any slave is limited.

Poll rate reduction is also utilized in the *Fair Exhaustive Polling* (FEP) scheme [8], where a pool of 'active' slaves is maintained by the master. Slaves are polled with one frame per visit, as in 1-limited service. When a POLL-NULL frame occurs, that slave will be dropped from the pool and will not be polled for some time. The 'inactive' slave may be restored to the pool when the master downlink queue that corresponds to that slave contains a packet, or when the entire pool is reset to its original state. The pool is reset when the last slave in the pool is to be dropped, or after a predefined timeout. In this manner, the slaves that have more traffic will receive proportionally larger share of the bandwidth as long as they have traffic to send or receive.

A slightly more sophisticated approach to poll rate reduction is employed in the scheme known as *Adaptive E-Limited Polling* [14]. In this case, each slave is serviced for up to M frames during a single visit. However, the limits are assigned to each slave separately, and they are made variable between predefined bounds. Initially, all slaves are assigned the value of M equal to

Table 2: An overview of adaptive polling algorithms.

polling algorithm	performance aspect			
	access delay	end-to-end delay	adaptivity	fairness
LWRR [6]	average	good (high loads)	good (penalizes inactivity)	inherent
FEP [8]	good	bad (under bursty traffic)	good (penalizes inactivity)	can be unfair
ACLS [15]	good	good	excellent	inherent
FPQ [16]	good	good	good (but computationally intensive)	adjustable

the upper bound of M^+. When the slave is polled and there is a data packet sent in either direction, the current value of M for that slave is decreased by one, until the lower bound M^- is reached. When a POLL-NULL frame is encountered, the value of M for that slave is reset to the maximum value, but the slave will skip a certain number of cycles. In this manner, the slave that has been idle for some time can send more data immediately after becoming active again. In case there is continuously backlogged traffic, the service gradually decreases to allocate a fair share of available bandwidth to each slave.

The *Adaptive Cycle-Limited Service* scheme [15] strives to keep the duration of the piconet cycle as close to a predefined value as possible. Bandwidth is allocated dynamically, partly on the basis of historical data (i.e., the amount of traffic in the previous piconet cycle), and partly on the basis of current traffic (i.e., whether they have some data to exchange or not). However, each of the m slaves is guaranteed a fair share of the available bandwidth over the period of $m - 1$ piconet cycles, plus a certain minimum bandwidth in each piconet cycle. This scheme appears well-suited for piconets in which some of the slaves have tight bandwidth and latency constraints, as is often the case with multimedia traffic.

We also mention the Efficient Double-Cycle polling [17], an adaptive algorithm which tries to optimize performance for uplink and downlink traffic

separately; and FPQ, a predictive algorithm with rudimentary QoS support [16].

Table 2 summarizes the characteristics of adaptive polling schemes.

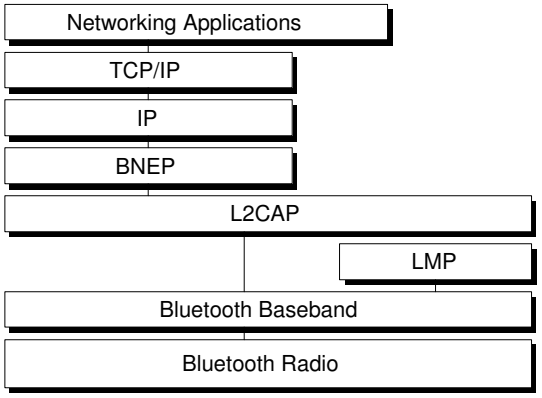
3 BNEP and segmentation/reassembly policies

All of the polling schemes described above focus on the optimization of performance of Bluetooth baseband traffic. As the traffic to be transported originates from the applications running on Bluetooth and other devices, we should also look into the details of packet segmentation and reassembly policies. As most traffic nowadays is based on the TCP family of protocols, it is necessary to examine the ways in which such traffic can be transported over Bluetooth. Fortunately, the Bluetooth Network Encapsulation Protocol, or BNEP, provides a ready solution to these problems [18].

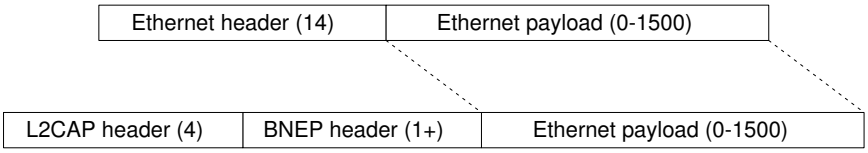
The BNEP protocol is designed to encapsulate and forward Ethernet frames through Bluetooth networks. Multi-hop traffic, including the slave to slave traffic, may be handled by Bluetooth masters and/or bridges acting as store-and-forward switches. In other words, the entire TCP PDU, which consists of a number of Bluetooth PDUs, has to be stored in the device before being repackaged (if necessary) and forwarded to the next stop along the route. Routing in this case is done in the IP layer, transparently to Bluetooth. Fig. 3(a) shows the protocol stack when TCP/IP packets are encapsulated using BNEP. The headers and their typical length are shown in Fig. 3(b). Note that each TCP message generated will require a total of 59 bytes in appropriate headers throughout the protocol stack.

As Bluetooth baseband packets can have either one, three, or five slots each, with varying payload of up to 339 bytes, as shown in Table 3, L2CAP packets obviously have to be segmented into a number of baseband packets. Upon reception, the payload has to be extracted from the baseband packets, and the L2CAP packet has to be reassembled. Again, the Bluetooth specification offers little guidance in that respects, and several policies for packet segmentation and reassembly have been described [8, 9].

We note that the noise and interference levels may play a critical role in choosing the segmentation policy. The Bluetooth frequency hopping sequence has been shown to be fairly efficient, and a large number of Bluetooth



(a) BNEP Protocol Stack.



(b) BNEP with an Ethernet Packet payload sent using L2CAP (all field sizes are expressed in bytes).

Figure 3: BNEP: transmission of TCP/IP traffic over Bluetooth (adapted from [18]).

Table 3: Packet types for communication over an ACL link.

Type	Slot(s)	Payload (bytes)	FEC	Asymmetric data rate (kbps total)
DM1	1	17	2/3	217.6
DH1	1	27	none	341.6
DM3	3	121	2/3	516.2
DH3	3	183	none	692.0
DM5	5	224	2/3	514.1
DH5	5	339	none	780.8

piconets may coexist in the same radio range [19]. Still, packets can be damaged by noise and interference, in which case retransmission will be needed. We note that in a noisy environment,

- DM-type packets, which include 2/3 Forward Error Correction (FEC), might be preferred over DH-type packets, which have no such provisions; and
- shorter (DH3 and DM3) packets might be preferred over longer ones, despite their smaller payload, because they are less susceptible to damage.

There are many issues related to the transmission of TCP/IP (and other types of) traffic over Bluetooth that ought to be investigated in more detail, in particular reliability, fairness, admission control, and congestion control. An overview of the performance of difference polling algorithms in the presence of TCP/IP traffic is given in [14].

4 Open problems and directions for further research

This chapter presents a brief overview and classification of intra-piconet polling schemes for Bluetooth piconets. Some additional issues, in particular the transmission of TCP/IP traffic over Bluetooth, and segmentation and reassembly policies, are discussed as well.

The aspects of Bluetooth networking not covered here include piconet admission control and QoS support. Also, issues related to scatternet formation, operation, and maintenance, are not mentioned here. We have also not addressed the issues related to the Bluetooth radio layer, where problems such as noise, interference, and packet loss, may have a significant impact on performance.

Future work might also analyze the performance of Bluetooth networks under different traffic models, preferably those that try to mimic real life traffic patterns for devices, such as PDAs, that are most likely candidates for communication using Bluetooth technology. Issues related to stability, fairness, reliable data transfer, and others, could also be addressed. Finally, the practical implications of performance analyses on scatternet formation and restructuring, should be examined.

We end this discussion by citing a few among the papers that contain more detailed comparative analyses of different polling schemes. In particular, we note an early report by Johansson *et al.* [8], a detailed and influential paper by Capone *et al.* [6], and a more recent overview paper by Chan *et al.* [14]. It may also be interesting to note that the majority of authors have relied on discrete event simulation to obtain their results. Analytical techniques have been used only recently [12, 20], possibly because the well-known results in the area of polling systems [5, 10] cannot be directly applied in Bluetooth.

References

[1] "Wireless PAN medium access control MAC and physical layer PHY specification," IEEE standard 802.15, IEEE, New York, NY, 2002.

[2] Bluetooth SIG, *Specification of the Bluetooth System – Architecture & Terminology Overview*, vol. 1. Nov. 2003.

[3] Bluetooth SIG, *Specification of the Bluetooth System.* Feb. 2001.

[4] H. Levy, M. Sidi, and O. J. Boxma, "Dominance relations in polling systems," *Queueing Systems Theory and Applications*, vol. 6, no. 2, pp. 155–171, 1990.

[5] Z. Liu, P. Nain, and D. Towsley, "On optimal polling policies," *Queueing Systems Theory and Applications*, vol. 11, no. 1–2, pp. 59–83, 1992.

[6] A. Capone, R. Kapoor, and M. Gerla, "Efficient polling schemes for Bluetooth picocells," in *Proceedings of IEEE International Conference on Communications ICC 2001*, vol. 7, (Helsinki, Finland), pp. 1990–1994, June 2001.

[7] 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*, vol. 1, (Anchorage, AK), pp. 591–600, Apr. 2001.

[8] N. Johansson, U. Körner, and P. Johansson, "Performance evaluation of scheduling algorithms for Bluetooth," in *Proceedings of BC'99 IFIP TC*

6 Fifth International Conference on Broadband Communications, (Hong Kong), pp. 139–150, Nov. 1999.

[9] 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)*, (San Diego, CA), pp. 384–388, Nov. 1999.

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

[11] Y.-Z. Lee, R. Kapoor, and M. Gerla, "An efficient and fair polling scheme for Bluetooth," in *Proceedings MILCOM 2002*, vol. 2, pp. 1062–1068, 2002.

[12] J. Mišić and V. B. Mišić, "Modeling Bluetooth piconet performance," *IEEE Communication Letters*, vol. 7, pp. 18–20, Jan. 2003.

[13] J. Mišić, K. L. Chan, and V. B. Mišić, "Performance of Bluetooth piconets under E-limited scheduling," Tech. report TR 03/03, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, May 2003.

[14] K. L. Chan, V. B. Mišić, and J. Mišić, "Efficient polling schemes for bluetooth picocells revisited," in *HICSS-37 Minitrack on Wireless Personal Area Networks*, (Big Island, Hawaii), Jan. 2004.

[15] V. B. Mišić, E. W. S. Ko, and J. Mišić, "Adaptive cycle-limited scheduling scheme for Bluetooth piconets," in *Proceedings 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC'2003*, vol. 2, (Beijing, China), pp. 1064–1068, Sept. 2003.

[16] J.-B. Lapeyrie and T. Turletti, "FPQ: a fair and efficient polling algorithm with QoS support for Bluetooth piconet," in *Proceedings Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2003*, vol. 2, (New York, NY), pp. 1322–1332, Apr. 2003.

[17] R. Bruno, M. Conti, and E. Gregori, "Wireless access to internet via Bluetooth: performance evaluation of the EDC scheduling algorithm,"

in *Proceedings of the first workshop on Wireless Mobile Internet*, (Rome, Italy), pp. 43–49, July 2001.

[18] Bluetooth SIG, "Bluetooth Network Encapsulation Protocol (BNEP) Specification," tech. rep., Revision 0.95a, June 2001.

[19] S. Zürbes, "Considerations on link and system throughput of Bluetooth networks," in *Proceedings of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2000*, vol. 2, (London, UK), pp. 1315–1319, Sept. 2000.

[20] D. Miorandi, C. Caimi, and A. Zanella, "Performance characterization of a Bluetooth piconet with multi-slot packets," in *Proceedings WiOpt'03*, (Sophia-Antipolis, France), Mar. 2003.