Efficient Polling Schemes for Bluetooth picocells

Antonio Capone[1], Mario Gerla[2], Rohit Kapoor[2]
[1]DEI, Politecnico di Milano
[2]University of California, Los Angeles, USA

Abstract- **Bluetooth is a new low-cost wireless technology that is going to play an important role in communications among small electronic devices and the access to wired networking infrastructure. Bluetooth stations that communicate directly form a piconet. In a piconet one station has the role of master and the others are slaves. The access to the medium is based on a TDD (Time Division Duplexing) scheme controlled by the master. The master sends packets to slaves in even-numbered slots triggering a transmission from slaves in the subsequent slot. Slaves are allowed to send packets only in response to a master packet. The way in which the master schedules packets transmission to slaves or polls them determines system performance. In this paper we consider the problem of designing an efficient and simple polling and scheduling scheme for Bluetooth. We propose some practical schemes and compare their performance with some ideal schemes derived from known results for polling systems.**

I. INTRODUCTION

Bluetooth is a new low cost, low power wireless technology [1, 2] that will allow users to interconnect their mobile phones, laptops, palmtops, headsets, and other electronic devices forming a small network usually referred to as the Personal Area Network (PAN). It is also likely that this technology will be a competitor to the traditional WLAN technologies for wireless access to wired IP networks or even for ad-hoc wireless networking [3, 4].

Bluetooth is based on a centralized connection-oriented approach. Bluetooth devices sharing a wireless channel form a piconet. One device in a piconet has the role of the master and controls access to the channel, while the others are slaves. Information can only be exchanged between a master and a slave and any information that must go from one slave to another must go through the master. In a piconet, stations share the radio channel on a time-division basis and transmissions are controlled by the master. The channel is slotted (one slot is $625\mu s$ long) and downlink transmissions (from the master to a slave) and uplink transmission (from a slave to the master) always alternate in a TDD (Time Division Duplexing) fashion. Each transmission occurs on one of the 79 1Mbit/s carriers defined in the ISM (Industrial Scientific Medical) unlicensed radio band. To comply with the utilization rules of the ISM band and to make the system robust to interference caused by other systems using the same bandwidth, transmissions of a piconet hop from one carrier to another according to a pseudo-random hopping sequence.

Both synchronous and asynchronous connection can be setup in a piconet between the master and slaves. Synchronous connections provide a circuit-oriented service with constant bandwidth and are based on a fixed and periodic allocation of slots. Asynchronous connections provide a packet-oriented service and can use different packet formats spanning over 1, 3 or 5 slots. Slot allocation is dynamic and is managed by the master that can decide to send data or signaling packets to slaves in the slots available for downlink transmission. Each slave is only allowed to send packets immediately after the reception of a packet from the master explicitly addressed to that slave. Therefore, slave transmissions are triggered by master transmissions and the way in which the scheduler takes decisions regarding which slave to poll determines the performance of the system. Note that, once a packet is received from the master, if the slave has no packet to transmit it returns a NULL packet without user information and, therefore, a slot is wasted. Similarly, if the master scheduler decides to trigger the transmission of a slave even if there is no packet to send to that slave, it can send a POLL packet without user information and, again, a slot is wasted.

The problem of finding an efficient scheduling algorithm for piconet operation is quite similar to the classical optimization problem of polling schemes for which theoretical results are available in the literature [5, 6, 7]. However, the constraints added by the Bluetooth specific operative mode, the partial information of the queues' status available at the master and the need to keep the algorithm as simple as possible make the problem interesting from a practical and theoretical point of view. The aim of this paper is to evaluate the performance of different Bluetooth polling schemes using simulation and to give some insight into the problem of designing efficient and practical schemes. The analyzed schemes are first evaluated using a simple traffic model and their behavior is then studied by means of a TCP/IP application scenario. In section 2 the polling optimization problem is described and the performance of some ideal and practical polling schemes is evaluated. In section 3 the behavior of TCP/IP connections with the considered schemes is shown using the results obtained with NS-2 (Network Simulator ver. 2). Section 4 concludes the paper.

II. POLLING SCHEMES

Single server polling systems are queue systems in which the server visits a set of queues to serve customers. The polling scheme is the set of rules that determines when the server

0-7803-7097-1/01/$10.00 ©2001 IEEE

switches from one queue to another, which is the next queue to be visited by the server, the number of customers of a queue to be served, etc. Polling schemes have been extensively studied in the literature and applied to many communication networks problems [5, 6]. A polling system model can also be adopted to describe the Bluetooth MAC operation in a picocell where the master station handles the packet transmissions to the slaves and solicits transmissions from the slave stations. Naturally, for the Bluetooth MAC operation we are interested in polling schemes that are able to make efficient use of the scarce radio resource. The efficiency definition depends on the performance parameters and the quality of service constraints considered. In this section we limit our study to models where the arrival processes to queues are independent of network status and the amount of service received, and the queue length is infinite, deferring the study of the system behavior with active traffic sources like TCP connections to the next section. Under these assumptions, the merit figure of the polling scheme is the delay suffered by data packets.

In [5] it is proved that the optimal polling schemes for symmetric systems (same traffic offered to all queues) are exhaustive, i.e. they are schemes that never switch the server from one queue to another when it is at a non-empty queue. Moreover, when all queue lengths are known, the optimal scheme routes the server to the longest queue. On the other hand, when the status of the queues is not known, the cyclic routing scheme is optimal. In [6] a subset of polling schemes with a fairness constraint is considered: each queue must be visited once per cycle but the visit order can be dynamically changed. The optimal polling policy with symmetric systems is the one that enforces a decreasing queue length order at the beginning of each cycle. The optimality property of this scheme holds also with many non-symmetric systems but has not been proved in the general case.

The Bluetooth operation model differs from that of the classical schemes analyzed in the literature mainly for two reasons. After the transmission of a packet from the master to a slave, there is always a chance for the slave to transmit a packet to the master; therefore, the visits of the server to the master-to-slave and the slave-to-master queues are strictly related. Moreover, since the MAC scheduling procedure is managed by the master, the knowledge of the queues' status is only partial. The master knows the status of all master-to-slave queues, but cannot have an updated information on the status of the slave-to-master queues even if an explicit signaling is provided. In [8] and [9] it is assumed that the master knows if a slave queue has packets to be transmitted so that only the non empty master-slave queues are considered by the MAC scheduler. This approach prevents wastage of radio slots and gives interesting results; but such a scheme assumes an ideal scenario where the master has an updated knowledge of slave queues at all times. In a real scenario, the slaves can communicate the queue status only at the time they transmit packets. In particular, if the slave queue is empty and no packets arrive into the master queue, the scheduler does not poll the slave, and the slave cannot inform the scheduler when new packets arrive and its queue status changes.

However, since the performance evaluation of ideal cases is important for comparison reasons, we define some benchmark Bluetooth polling schemes for which we assume that the scheduler knows the length of all queues. The first benchmark scheme (B1) we consider is exhaustive, i.e. the server does not switch to another master-slave pair until both the master and the slave queues are empty, and the next master-slave pair to be served is the one which has the highest sum of the master and slave queue lengths. Note that this scheme could be easily proven to be optimal following the approach in [6], if the Bluetooth operations did not waste time slots when one of the queue pairs is empty and the other is not; in this case, in fact, a packet with no user information (POLL packet from master to slave or a NULL packet from slave to master) must be transmitted. We start with comparing the delay performance of the B1 scheme with three practical schemes which do not require the knowledge of slave queues and, therefore, can be implemented without adding to Bluetooth new signaling messages from the slave to the master:

- Pure Round Robin (PRR): a fixed cyclic order is defined and a single chance to transmit is given to each master-slave queue pair according to the cyclic order; note that this scheme is not exhaustive.
- Exhaustive Round Robin (ERR): as with PRR a fixed order is defined but the scheme is exhaustive and the server does not switch to the next pair until both the master and the slave queues are empty.
- Exhaustive Pseudo-cyclic Master queue length (EPM): a dynamic cycle order is defined at the beginning of each cycle (each master-slave pair is visited exactly once per cycle) according to a decreasing master to slave queue length order.

To compare the delay performance of these schemes we use a C++ discrete event simulator. The simulation scenario consists of a single Bluetooth piconet with 7 slaves. Independent statistically identical traffic generators are considered for each of the 14 queues. Traffic sources generate information files according to a Poisson arrival process. The length of files is exponentially distributed and the average value is given in number of packets, i.e. information units that can be transmitted in a Bluetooth time slot. Once a queue is granted the permission to send, it sends a maximum of 1, 3 or 5 time slots according to the number of packets in the queue (the effect of the different percentages of overhead in the 1, 3 or 5 slots Bluetooth packets is ignored in the simulation results presented in this section). The mean delay curves obtained with an average file length of 8 packets are shown in Figure 1.

We see that with traffic loads up to 0.85, the lowest delay is offered by the B1 scheme. The difference between the ERR and the EPM curves is negligible and the curve of the ERR is even slightly lower. This suggests that the use of a pseudo-cyclic dynamic scheme based on partial information of the queue status' available at the master is not necessary since a simple fixed cycle scheme gives a good performance. As expected the PRR scheme has the worst performance with loads up to 0.85, but the unexpected result is that it has the lowest delay with higher loads. The main reason behind this behavior is that all the other schemes are exhaustive schemes and waste slots when one queue of the pair is empty and the other is

0-7803-7097-1/01/$10.00 ©2001 IEEE

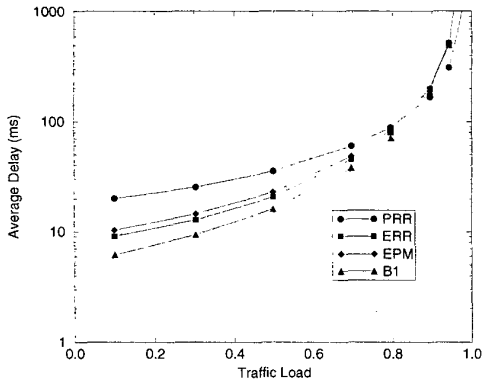

Figure 1: Average delay of PRR, ERR , EPM and B1 polling schemes, with average file length equal to 8 packets

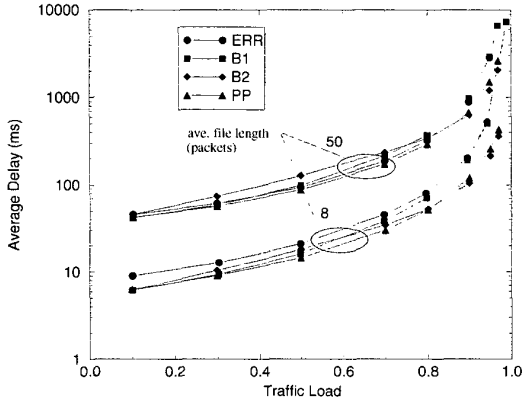

Figure 2: Average delay of ERR, B1, B2 and PP (P=4) schemes, with average file length equal to 8 and 50 packets

not. At high loads this effect reduces the number of slots available for transmission and increases the delay. To explain this effect better, we consider another benchmark scheme (B2). This scheme is in a way partially exhaustive since it serves a master-slave pair only until both the queues have packets to send. The next pair to be served is chosen according to the same criterion as in B1, but the choice is restricted to the subset of pairs in which both queues are non-empty, if any.

The results obtained with an average file length equal to 8 and 50 packets are presented in Figure 2. The B2 scheme has a lower delay than the B1 scheme at high loads due to the reduced number of wasted slots, but the delay at low loads is slightly higher. In the same figure, the curve of the Priority Policy (PP) scheme proposed in [8] (see also [9] for a description of the priority policy) is presented. This scheme is a dynamic weighted round-robin scheme in which the weights of the master-slave pairs depend on the queue status. If both queues have packets to be transmitted the weight is P (P=4 in the simulations), if only one queue has packets the weight is 1, and if both queues are empty, the weight is zero and the pair is not polled. The performance of the PP scheme is quite good and its delay at high loads is slightly higher than that of the B2 scheme. Note, however, that the B2, B1 and PP schemes are all ideal since they assume that the scheduler knows the status of all queues. The most important result of Figure 2 is that the performance of a simple and practical scheme like ERR is very close to that of the other schemes. With an average packet length equal to 50 packets the difference is negligible at moderate load, while it increases at high loads.

The results presented so far suggest the use of simple schemes like ERR in symmetric scenarios. However, the main drawback of using an exhaustive scheme is that the channel can be captured by stations generating a traffic higher than the system capacity. Such a behavior can be observed either over a short-term or a long-term depending on the traffic characteristics and it can be said that the system is unfair. The ERR polling scheme, in fact, visits each queue exactly once per cycle, but, the cycle length can vary without bound and according to the traffic characteristics. This "capture effect" is similar to that shown by 802.3 and 802.11 LAN's where it is due to the access scheme. In Bluetooth, however, we can exploit the logical star topology to design a fairer scheme.

The simplest way to achieve this aim is to modify the ERR scheme by limiting the number t of transmissions (tokens) that can be performed by each pair per cycle. Naturally for $t = 1$ this new Limited Round Robin (LRR) scheme becomes the PRR scheme. The maximum number of transmission per cycle allows to get a limit on the cycle length and to avoid the capture effect.

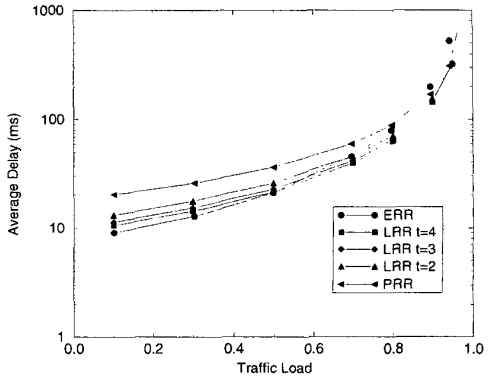

Figure 3: Average delay of ERR and LRR schemes with average file length equal to 8 packets

In Figures 3 and 4 the mean delay curves for the LRR scheme with various values of t are presented for the 8 and 50 packets average file length cases respectively. At low loads the delays of the LRR scheme are higher than that of the ERR one, but for $t = 4$ the difference is small. At high loads, the LRR scheme can even achieve a lower delay due to the wasting of slots by the ERR scheme (as explained earlier). To show that the LRR scheme has a fair behavior, we need a non-homogeneous scenario. We consider one master-slave traffic source pair generating files with an average length equal to 500 packets and 6 other source pairs generating files with an average length of 8 packets. The traffic load generated by each source is equal to 0.06 resulting in a total load equal to 0.84. The mean waiting delays obtained (in ms) are shown in Table 1. To evaluate these results, we can compare them with those

0-7803-7097-1/01/$10.00 ©2001 IEEE

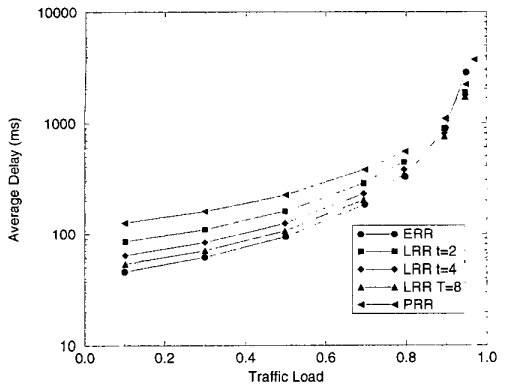

Figure 4: Average delay of ERR and LRR schemes with average file length equal to 50 packets

of two homogeneous scenarios with the same total load (= 0.84) and where each source generates 8 packet or 500 packet files. With the homogenous scenarios, the mean waiting delay is 78 ms for the 8 packet case, and 3051 ms for the 500 packet case. We observe in Table 1 that, with the non-homogeneous scenario and the ERR scheme, the two types of packet flows experience the same waiting delay. This delay is much higher than the corresponding delay in the homogeneous scenario for the 8 packet files, while it is much lower for the 500 packet files. On the contrary, with the LRR scheme the waiting delay experienced by the two traffic types is different and closer to the delay of the corresponding homogeneous scenarios.

	ERR	LRR t=6	LRR t=10
l=500	547	4690	3988
l=8	545	88	97

Table 1: Delay with non-homogeneous and homogeneous scenarios with traffic load equals to 0.84

Using the LRR scheme as a base, we add another feature to the scheduler in the master which enables it to distinguish between slaves on the basis of their "activeness" [10], i.e. according to the status of the queues observed in the last few polling cycles. This is actually a simple attempt to exploit a result in [5] where it is proved that the optimal routing should route the server to the queue which has the highest probability to be the longest one on the basis of available knowledge. We try to achieve a performance gain by reducing the rate of visit to queues which have been found empty in the last visits and hence, should have a lower probability to be the longer queues. Moreover, this should also reduce the number of slots wasted in polling "idle" slaves (i.e., slaves for which both slave and master queues are empty). In this context, we extend the LRR scheme by defining a new LWRR (Limited and Weighted Round Robin) scheme. The LWRR adopts a weighted round robin algorithm with weights dynamically changed according to the observed queue status. Each slave is assigned a weight equals to MP (Maximum Priority) at the beginning. Each time a slave is polled and no data is exchanged between master and slave, the weight of the slave is reduced by 1. The lowest value that the weight of a slave can attain is equal to 1, in which case the slave has to wait a maximum of MP-1 cycles to get a chance to send data. Anytime there is a data exchange between the slave and the master, the weight of the slave is increased to the MP value. Note, however, that the LWRR scheme or any scheme trying to predict the queue status on the basis of the previous visits cannot show any gain if the traffic model is Poisson due to the non-memory characteristics of this traffic. Therefore, in the next section, active TCP traffic generators are used to compare LWRR with other schemes under various simulation scenarios.

III. TCP PERFORMANCE

In this section, we extend our investigations to include TCP traffic, whose behavior depends upon the amount of service received. We study only real schemes and do not consider the performance of the ideal schemes of the previous section. In particular, the Pure Round Robin (PRR), Exhaustive Round Robin (ERR), Limited Round Robin (LRR) and LWRR polling schemes are evaluated under various traffic conditions. In the simulations, the value of t for LRR and LWRR is taken to be 3 and the value of MP for LWRR is taken to be 4. All the simulations were performed using a Bluetooth model developed under the NS-2 simulator. The packet formats adopted in the simulations are the DH1, DH3 and DH5 [11] which correspond to a payload length of 224, 1464 and 2712 bits respectively. In the first set of experiments, we consider a single piconet scenario where there is a Non-Persistent ON/OFF TCP connection[1] between the master and each slave, and the number of slaves is varied from 1 to 7. The connections have exponentially distributed ON-OFF times and different ratios of the ON/OFF time are considered. The average ON time is set to 2 s while the average OFF time is varied according to the ON/OFF ratio. The TCP packet size used is 500 bytes and all simulations have been run for 2 minutes of simulated time. The metric used for comparison is the total throughput obtained by all the TCP connections. Note that the time used to calculate the throughput is the total simulated time, which also includes the OFF time of the TCP sources.

Figures 5, 6 and 7 show the throughput for the TCP connections for ON/OFF time ratios of 1:2, 1:6 and 1:10 respectively. We see that ERR and LWRR always performs better than the other schemes. The LWRR performs almost as well as ERR and even outperforms it for a low value of ON/OFF time ratio(1:2). PRR gives the poorest performance, as it has to poll the idle slaves with as much frequency as the slaves which have data to exchange with the master. In addition, the results confirm the intuitive notion that LWRR shows a greater improvement over LRR and PRR for smaller values for ON/OFF time since the idle time of slaves is greater.

In the second set of experiments, we ran an FTP source over TCP from the master to each slave. The FTP sources produced files of size 80Kb (20 TCP packets of size 500 bytes each) and the inter-arrival time of these files was exponentially distributed. The system load was varied by varying the inter-arrival time. The metric studied here was the end-to-end Ap-

[1] a non-persistent ON/OFF TCP source is reset to slow start at the beginning of every new ON period"

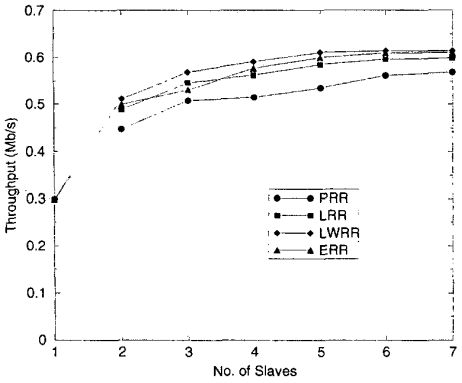

Figure 5: Throughput for PRR, LRR, LWRR and ERR for ON/OFF ratio of 1:2

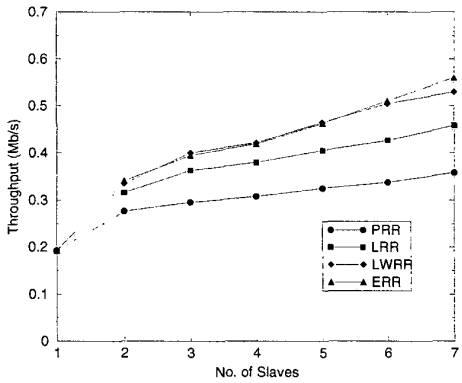

Figure 6: Throughput for ON:OFF ratio of 1:6

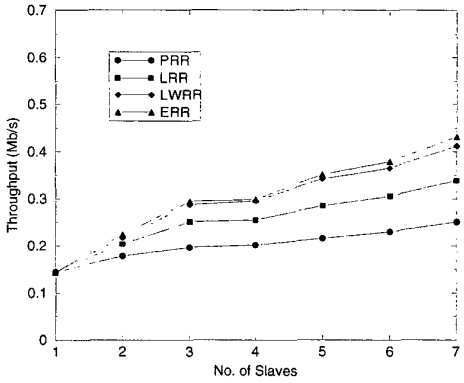

Figure 7: Throughput for ON:OFF ratio of 1:10

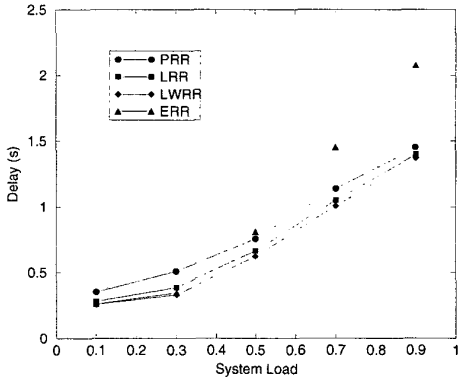

Figure 8: Average End-to-End FTP delays for PRR, LRR, LWRR, and ERR

plication delay and the results are shown in Figure 8. It is seen that LWRR always does better than the other schemes. Also, ERR starts off doing as well as LWRR, but its performance degrades sharply for higher load, as some TCP connections time-out and packets have to be retransmitted. This kind of behavior is expected in ERR as some slaves may not be polled for a long time and TCP connections may get starved.

IV. CONCLUSION

In this paper we have considered the problem of designing efficient and simple polling schemes for Bluetooth picocells. We have compared the performance of different polling schemes and proposed a simple scheme based on a weighted round robin algorithm with dynamic weights and limited service. The results have been obtained using a passive traffic model and active TCP connections. We have shown that the performance of the proposed LWRR scheme is good in all conditions and usually very close to that of the ERR scheme, which itself suffers from the problem of channel capture.

REFERENCES

[1] J.C. Haartsen, Bluetooth: a New Radio Interface Providing Ubiquitous Connectivity, IEEE VTC 2000-Spring, pp. 107–111.

[2] J.C. Haartsen, The Bluetooth radio system, IEEE Personal Communications, vol. 7, no. 1 , Feb. 2000, pp. 28 -36.

[3] S. Zilrbes, W. Stahl, K. Matheus, J. Haartsen, Radio Network Performance of Bluetooth, IEEE ICC 2000, vol. 3 , pp. 1563 -1567.

[4] P. Johansson, N. Johansson, U. Korner, J. Elg, G. Svennarp, Short range radio based ad-hoc networking: performance and properties, IEEE ICC '99, vol. 3, pp. 1414-1420.

[5] Z. Liu, P. Nain, D. Towsley, On Optimal Polling Policies, Queuing Systems, Vol. 11, pp. 59-83, 1992.

[6] O. Fabian, H. Levy, Polling System Optimization Through Dynamic Routing Policies, IEEE INFOCOM '93, vol. 1, pp. 194 -200.

[7] H. Takagi, Analysis of Polling Systems, MIT Press, 1986.

[8] M. Kalia, D. Bansal, R. Shorey, Data scheduling and SAR for bluetooth MAC, IEEE VTC 2000-Spring Tokyo, pp. 716 -720.

[9] S. Garg, M. Kalia, R. Shorey, MAC Scheduling Policies and SAR Policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System, MoMuC 99, pp. 384-387.

[10] N. Johansson, U. Korner, P. Johansson, Performance Evaluation of Scheduling Algorithms for Bluetooth, IFIP TC Fifth International Conference on Broadband Communications '99, Hong Kong.

[11] Specification of the Bluetooth System, version 1.0, July 1999, http://www.bluetooth.com/