Wireless Personal Communications **23:** 195–206, 2002.

Polling Best Effort Traffic in Bluetooth

RACHID AIT YAIZ [1] and GEERT HEIJENK [1, 2]

[1] *Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*
E-mail: yaiz@cs.utwente.nl and heijenk@cs.utwente.nl
[2] *Ericsson EuroLab Netherlands, P.O.Box 645, 7500 AP Enschede, The Netherlands*
E-mail: geert.heijenk@eln.ericsson.se

Abstract. Bluetooth [1] is a wireless access technology where polling is used to share bandwidth among the nodes. We have introduced a new poller named Predictive Fair Poller (PFP) in [2, 3]. In this paper we explain the operation of the Predictive Fair Poller and compare it with the conventional Round Robin poller and the Fair Exhaustive Poller (FEP) [4] for two Best Effort traffic scenarios. We show through simulations that the Predictive Fair Poller is able to divide bandwidth in a fair and efficient manner.

Keywords: Bluetooth, polling, scheduling.

1. Introduction

Bluetooth [1] is a low power, short range, and cheap wireless access technology, which can be used ad hoc or in an infrastructure. In each of the scenarios, a Bluetooth node is either a master or a slave, and direct communication only takes place between a master and a slave. One master and up to seven slaves can be affiliated with each other and form a Piconet.

Bluetooth is a time slotted access technology where each second is divided into 1600 time slots. Time slots are either downlink slots, i.e. from the master to a slave or uplink slots, i.e. from the addressed slave to the master. Data is exchanged between the master and a slave using Baseband packets that cover one, three or five time slots. Often, IP will be used over Bluetooth, where IP packets cover one or more Baseband packets.

The traffic in a Piconet is controlled by the master of that Piconet in such a way that a slave is only allowed to transmit data if it was addressed (by the master) in the previous time slot. In other words, the master polls the slaves giving them an opportunity to transmit data. A master polls a slave either explicitly or implicitly, where an explicit poll of a slave means that the master sends a packet with no payload (POLL packet) to that slave when data destined for that slave is not available. Similarly, a slave that has no data destined for the master responds to a poll with a packet with no payload (NULL packet).

2. Goals for a Poller

Polling can be done in many different ways. The difference between the polling mechanisms is related to the order in which the slaves are served and the service discipline used to serve a slave.

In the Bluetooth access technology, the goals for a poller depend on whether Best Effort (BE) traffic is considered or QoS traffic. In both cases it is important that the poller uses the

available time slots as efficient as possible by minimizing the number of transmitted POLL and NULL packets, while maintaining fairness between the different slaves. In the Best Effort case, fairness means that the slaves get the same fraction of their fair share of resources, where a fair share is equal to the share that the slaves would have been given in case a Generalized Processor Sharing (GPS) system was used. In the QoS case, fairness means that the slaves get the same fraction of their negotiated QoS requirements.

3. Related Work

As mentioned before, a master and up to seven slaves can be affiliated with each other and form a so-called Piconet. On the other hand, a Bluetooth node can be member of multiple piconets forming a so-called scatternet. However, as the master's MAC address is used to identify a Piconet, Bluetooth nodes can be the master in only one Piconet. Scheduling the traffic from a master to a slave and vice versa is referred to a Bluetooth polling, Bluetooth MAC scheduling or intra-piconet scheduling, while scheduling the node's participation in different piconets is referred to as inter-piconet scheduling. Bluetooth pollers are studied in [4–10]. However, up to now, mainly two pollers are being used in Bluetooth: the Round Robin (RR) poller and the Fair Exhaustive Poller (FEP) [4].

Round Robin poller with a 1-limited service discipline equally divides the number of available polls between all the slaves and does not take into account the number of polls needed by a slave. As a result, slaves that do not need to be polled will be polled anyhow instead of polling slaves that need these polls, and hence transmission time is wasted. Using an exhaustive service discipline reduces this problem but makes it possible for a slave to consume all the bandwidth.

Fair Exhaustive Poller has been introduced to take away the implications of the Round Robin poller. It divides the number of available polls between the slaves that it considers as being active, while checking regularly whether an inactive slave has become active. For this, it maintains two lists: a list of active slaves and a list of inactive slaves, and the slaves that are member of the list of active slaves are polled in a 1-limited Round Robin manner. The number of successive useless polls (POLL and NULL packet) or the average success rate of polls can be used as a measure of activity of a slave. Hence, the poller is able to detect whether a slave has become inactive and consequently move it to the list of inactive slaves. Furthermore, each slave i can define a maximum inter-poll interval (T_{poll_i}). This maximum inter-poll interval is used by the Fair Exhaustive Poller to poll an inactive slave regularly to check whether it has become active or not. If this slave is polled successfully (data in either direction or both) it will be moved from the list of inactive slaves to the list of active slaves.

The Round Robin poller (with 1-limited service discipline) performs well in the Best Effort case provided that no slave needs more polls than the total available number of polls divided by the number of slaves. The Fair Exhaustive Poller does not have this shortcoming. However, we strongly believe that nor the Round Robin poller nor the Fair Exhaustive Poller is able to provide QoS. Consequently, there is a need for a poller that performs at least as well as the Round Robin poller and the Fair Exhaustive Poller in the Best Effort case while also supporting QoS traffic.

4. The Predictive Fair Poller

We have introduced a poller named Predictive Fair Poller (PFP) [2, 3], which takes both efficiency and fairness into account. It predicts for each slave whether data is available or not and it keeps track of the fairness. Based on these two aspects it decides which slave to poll next. In the Best Effort case, the Predictive Fair Poller estimates the fair share of resources for each slave and keeps track of the fractions of these fair shares that each slave has been given. The Predictive Fair Poller can be used to poll Best Effort traffic in a fair and efficient manner by keeping track of both the fairness based on these fractions of fair share and the predictions. In the QoS case, QoS requirements are negotiated with the slaves and translated to fair QoS treatments. The poller keeps track of the fractions of these fair QoS treatments that each slave has been given. Similar to the Best Effort case, the Predictive Fair Poller can be used to poll QoS traffic such that the QoS requirements are met by keeping track of the fairness based on these fractions of the fair QoS treatments.

First, we show and explain the building blocks of the Predictive Fair Poller. Subsequently, we discuss the implementation of these building blocks for the Best Effort case. For the sake of simplicity we discuss the Predictive Fair Poller for traffic destined from the slaves to the master. However, we will mention how the Predictive Fair Poller can also handle traffic from the master to the slaves.

4.1. Building Blocks of PFP

The selection of the next slave to be polled is performed by the PFP Slave Selector which is shown in Figure 1. It is located in the master and requires knowledge of the results (PR) of its poll decisions. The PFP Slave Selector can be fed with a Traffic Demand (TD_i) for each slave i in order to support QoS traffic. However, if a slave did not make its traffic demand known to the master then the Traffic Demand estimated by Traffic Demand Estimator in the Slave Status Tracker (see Figure 1) will be used instead. In other words, the Traffic Demand (TD'_i) is either the Traffic Demand (TD_i) made known by slave i (e.g. QoS case) or the Traffic Demand estimated by the Traffic Demand Estimator in case slave i did not make its Traffic Demand known to the master (Best Effort case).

The Fair Share Determinator in the PFP Slave Selector uses the Traffic Demands ($TD'_1 \ldots TD'_7$) to determine the Fair Share (fs_i) of bandwidth for each slave i.

Besides a Traffic Demand Estimator the Slave Status Tracker also contains a Fraction of Fair Share Determinator and a Data Availability Predictor. The Fraction of Fair Share Determinator in Slave Status Tracker i uses the Fair Share (fs_i) determined by the Fair Share Determinator and the Poll Results (PR) to determine the Fraction of Fair Share of bandwidth (Ffs_i) that slave i has been given. The Data Availability Predictor in Slave Status Tracker i uses the Traffic Demand (TD'_i) and the Poll Results (PR) to determine the probability (P_{data_i}) of data being available for transmission from slave i to the master.

The probabilities (P_{data_i}) of data being available for transmission from each slave i to the master and Fraction of the Fair Share of bandwidth (Ffs_i) that each slave i has been given are used by the Decision Maker to decide which slave to poll next. The decision rules depend on the requirements on both the efficiency and the fairness.

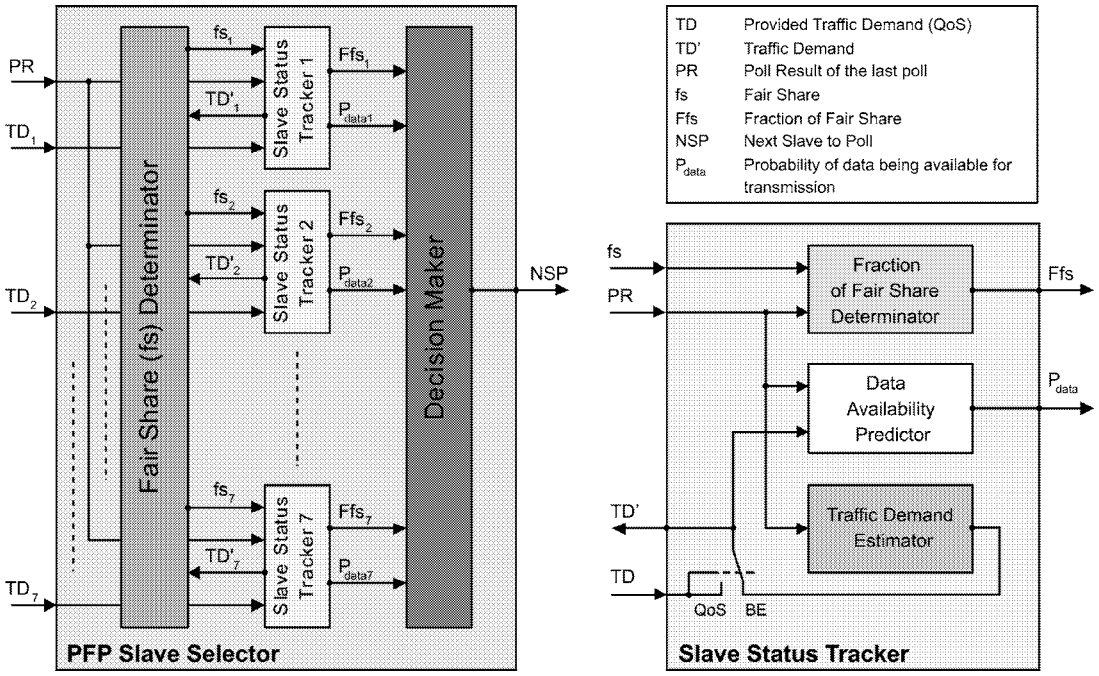

Figure 1. Block diagram of the Predictive Fair Poller.

4.2. Implementation of PFP for the Best Effort Case

In the Best Effort case the poller is not given any information about the offered load. This means that nor the IP packet sizes (and thus the number of Baseband packets belonging to the same IP packet) nor the inter-arrival times of these IP packets are known to the poller. Because the distribution of the inter-arrival times of these IP packets is also unknown, the poller assumes arrivals according to a Poisson process based on the fact that a Poisson process is one of the most unpredictable processes with respect to the arrival times of the arrivals. During the rest of this paper we will use the following terminology:

- Macro success time is the time at which a slave responds to a poll with a first segment of an IP packet.
- Macro poll time is the time at which a slave responds to a poll with a first segment of an IP packet or with a NULL packet.
- Micro poll time is the time at which a slave responds to a poll with a continuation segment of an IP packet.
- Macro success ratio is the success ratio of the Macro polls, i.e. the number of received first segments divided by the sum of the number of received first segments and the number of received NULL packets.

4.2.1. *The Traffic Demand Estimator*

Because of the assumption on the distribution of the inter-arrival times of the IP packets, the Traffic Demand Estimator only needs to estimate the average inter-arrival time of the IP packets. This estimation can be done when the macro success ratio is lower than one. The Traffic Demand Estimator then estimates the inter-arrival time by calculating a moving average of the inter-macro success time. In case the Macro success ratio approaches one, the

Traffic Demand Estimator can only indicate that the average inter-arrival time is less or equal to the average inter-macro success time.

4.2.2. *The Data Availability Predictor*

The Data Availability Predictor calculates the probability P_{data_i} of slave i having a Baseband packet waiting to be transmitted to the master. First, the Data Availability Predictor checks whether an IP packets of which one or more segments are received is completely received. One way of doing this is looking at the payload size of the last received Baseband packet from that slave, where a full Baseband packet serves as an indication that it is likely that a continuation Baseband packet is waiting at the slave. If it is not likely that a continuation Baseband packet is waiting at a slave, the Data Availability Predictor will calculate the probability that either an arrival has occurred at the slave since the last macro poll time or that more than one IP packet were waiting at a slave just before the last macro poll time.

4.2.3. *The Fair Share Determinator*

For the Best Effort case we want the poller to coarsely emulate a Generalized Processor Sharing (GPS) system. With respect to Bluetooth polling this means that on the smallest possible time scale available polls are equally divided among the slaves that have a Baseband packet waiting. Consider each slave i having with probability (P_{data_i}) a Baseband packet available for transmission to the master. Polling a slave i will then result in a Baseband packet being transmitted from slave i with probability P_{data_i}. Taking into account the fact that a slave is allowed to respond to a poll with at most one Baseband packet, a polled slave i responds on average with P_{data_i} Baseband packets. In other words, the probability (P_{data_i}) of a Baseband packet being available for transmission from slave i is equal to the instantaneous offered load of slave i. The instantaneous fair share (fs_i) can be calculated using:

$$fs_i = \frac{P_{data_i}}{\sum_{k=1}^{N} P_{data_k}}, \tag{1}$$

with N the number of slaves.

4.2.4. *Fraction of Fair Share Determinator*

The Fraction of Fair Share Determinator must first calculate the share s_i that slave i has been given. Share s_i is defined as the reciprocal of the number of polls since the last poll to slave i (including the poll to slave i). Using the Share s_i and the Fair Share fs_i, the Fraction of Fair Share is defined as:

$$Ffs_i = \begin{cases} \frac{s_i}{fs_i}, & \text{if } s_i < fs_i, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

4.2.5. *Decision Maker*

For each slave i, a probability P_{data_i} of a Baseband packet being available for transmission to the master and a Fraction of Fair Share Ffs_i is applied to the Decision Maker. Based on these inputs the Decision Maker decides which slave to poll next. It is clear that it is urgent to poll a slave with $P_{data} = 1$ and $Ffs = 0$ while it is not needed to poll a slave with $P_{data} = 0$ and $Ffs = 1$. The decision to make in the area between these two extremes depends on the policy.

We define a variable U_i for each slave i. We name it poll Urgency and define it as:

$$U_i = \alpha P_{data_i} + (1-\alpha)(1 - Ffs_i), \quad 0 \leq \alpha \leq 1\,. \tag{3}$$

Most of the time, making a poller extremely efficient results in the poller being not fair (refer to Round Robin poller with exhaustive service discipline), while basing poll decisions only on the fairness will lead to a poller that is not necessarily efficient. As a result, both the probabilities P_{data_i} and the Fractions of Fair Share Ffs_i should have impact on the polling decisions. Therefore, the tuning variable α is introduced to tune the impact of the probability P_{data_i} and the Fraction of Fair Share Ffs_i on the urgency. Furthermore, the Decision Maker selects the slave i with the highest Urgency value U_i.

We conjecture that polling the slave with the highest probability each time will lead to an efficient and fair distribution of bandwidth among the slaves as long as there is at most one slave i with $P_{data_i} = 1$ each poll moment. If it is predicted that more than one slave definitely have a Baseband packet available for transmission to the master then the fairness must also be considered. On the other hand considering only the fairness while making a decision will lead to a polling scheme that is not necessarily efficient. As a result, the tuning variable α should be greater than zero and less than one. Through simulations we found that $\alpha = 0.1$ leads to low response times (sum of waiting times and service times of IP packets) in case IP packets are generated by Poisson processes.

4.2.6. *Extension to QoS Traffic Handling*

In case of Best Effort traffic a Fair Share Determinator and a Fraction of Fair Share Determinator are used to determine the fair share of bandwidth for each slave and to keep track of the fractions of these fair shares that each slave has been given. In the QoS case, the Fair Share Determinator and the Fraction of Fair Share Determinator should be replaced by a Fair QoS Treatment Determinator and a Fraction of Fair QoS Treatment Determinator. The Fair QoS Treatment Determinator should be fed with a Traffic Demand TD_i and a QoS Request qr_i for each slave i. Using this input, the Fair QoS Treatment Determinator determines for each QoS slave how to handle traffic from that slave, e.g. how often slave i should be polled. Using QoS Treatment qt_i and Fair QoS Treatment fqt_i, the Fraction of Fair QoS Treatment is defined as:

$$Ffqt_i = \begin{cases} \frac{qt_i}{fqt_i}, & \text{if } qt_i < fqt_i, \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

For instance, $Ffqt_i$ is the number of polls for slave i in a time period divided by the number of polls that slave i should get in the same time period.

4.2.7. *Extension to Duplex Traffic Handling*

In order to make duplex traffic handling possible, two Slave Status Trackers are implemented for each master-slave pair, one for traffic from master to slave and one for traffic from slave to master. The major difference between both slave status trackers is the Predictor. In case of traffic from the master to the slaves, the poller knows whether data is available for transmission to a slave or not. Thus, the probability is either zero or one. The implementation of the other building blocks remains as described before. This includes the Decision Maker which still selects the slave with the highest Urgency level (with respect to traffic from master to slave or traffic from slave to master) regardless of the Urgency level with respect to traffic in the opposite direction.

5. Simulation Results

We present simulation results of the Predictive Fair Poller, the 1-limited Round Robin poller and the Fair Exhaustive Poller in two Best Effort traffic scenarios. The simulation tool we used is Network Simulator (ns2) [11] with Bluetooth extensions [12] from Ericsson Switchlab together with our ns2 implementation of both the Fair Exhaustive Poller and the Predictive Fair Poller.

5.1. The Poisson Scenario

In this scenario we simulated under the following assumptions:

- There is only upstream traffic, i.e. from the slaves to the master.
- The available Bluetooth Baseband packet types are DH1, DH3 and DH5 with a payload of 27 bytes, 183 bytes and 339 bytes respectively.
- The IP packet size distribution used is trimodal [13] with 30% of 40-byte IP packets, 12% of 1500-byte IP packets and 58% of IP packets with a size in the range of 300 to 600 bytes. This results in an average of $\overline{d} = 8.30$ data slots and an average of $\overline{p} = 1.99$ polls per IP packet.
- Seven slaves (S1...S7) generate IP packets according to Poisson processes with arrival rates $\lambda_1 \ldots \lambda_7$, while:

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = \lambda_a \,, \tag{5}$$

and:

$$\lambda_6 = \lambda_7 = \lambda_b \geq \lambda_a \,. \tag{6}$$

- The coefficient of variation of the arrival rates is defined as:

$$C.O.V. = \frac{\sqrt{\frac{\sum_{i=1}^{7} \lambda_i^2 - \frac{1}{7}(\sum_{i=1}^{7} \lambda_i)^2}{6}}}{\frac{1}{7}\sum_{i=1}^{7} \lambda_i} \,, \tag{7}$$

where $C.O.V. = 0$ implies that all arrival rates are equal, i.e. $\lambda_a = \lambda_b$.

- The utilization of the Piconet is defined as:

$$\rho = \frac{\sum_{i=1}^{7} \lambda_i \overline{d}}{C_{tdd}} \,, \tag{8}$$

with $C_{tdd} = 1600$ slots/s the total number of available time slots per second and $\overline{d}$ the average number of data slots per IP packet.

Due to the assumption on the arrival rates ($\lambda_1 \ldots \lambda_7$) each combination of the coefficient of variation of the arrival rates and the utilization of the Piconet gives a unique solution for the arrival rates and thus makes it possible to use *C.O.V.* and ρ as input for the simulations.

Pollers can be compared based on different performance aspects. We will compare the Predictive Fair Poller, the Fair Exhaustive Poller, and the Round Robin poller taking the following performance aspects into account:

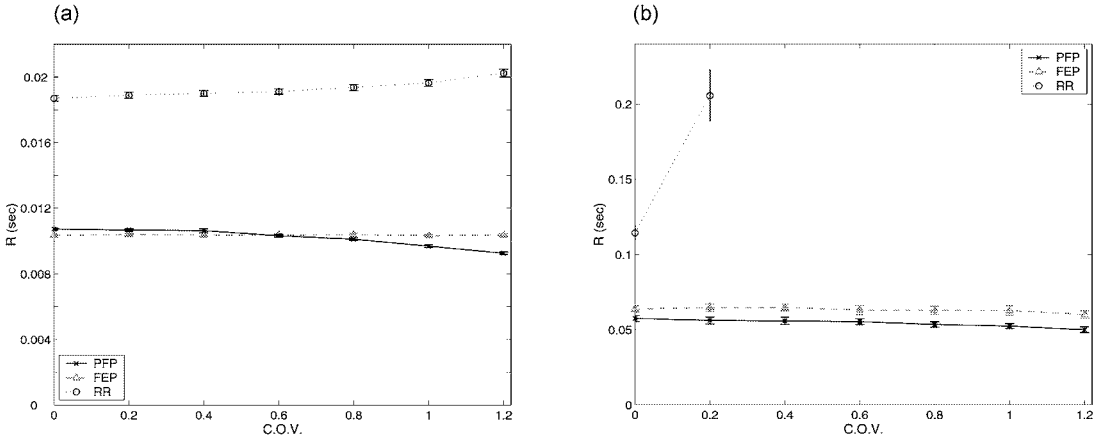

Figure 2. Response time (R). (a) Lowly loaded Piconet ($\rho = 0.1$). (b) Highly loaded Piconet ($\rho = 0.7$).

- Efficiency (η), which is defined as the number of data slots divided by the total number of slots and is at most equal to the utilization ρ. Because the total load is less than the load that can be handled in a Bluetooth Piconet, a maximum efficiency ($\eta = \rho$) means that every slave transmits the data that it needs to transmit, meaning that the poller is fair. On the other hand, a non maximum efficiency ($\eta < \rho$) means that the poller is not fair.
- Mean response time R (sum of waiting time and service time) taken over all packets received by the master.

Based on the simulation results we make the following observations:
In a lowly loaded Piconet ($\rho = 0.1$):

- The three pollers perform equally well with respect to the Efficiency (η) as function of the coefficient of variation of the arrival rates, i.e. $\eta = \rho$. This also means that the three pollers are fair with respect to the long term Fraction of Fair Share.
- Both the Predictive Fair Poller and the Fair Exhaustive Poller perform better than the Round Robin poller with respect to the response time (R) (see Figure 2(a)).

In a highly loaded Piconet ($\rho = 0.7$):

- The Round Robin poller becomes inefficient for increasing values of *C.O.V.*, while the Predictive Fair Poller and the Fair Exhaustive Poller achieve maximum efficiency (see Figure 3). This also means that the Round Robin poller becomes unfair based on the long term Fractions of Fair Share for increasing values of *C.O.V.*, while the Predictive Fair Poller and the Fair Exhaustive Poller achieve maximum fairness.
- The system served by the Round Robin poller becomes unstable for increasing values *C.O.V.*, while the Predictive Fair Poller causes lower response times (R) than the Fair Exhaustive Poller does (see Figure 2(b)). In this scenario it can be shown that using a Round Robin poller causes the two highly loaded slaves to get less polls than they need when:

$$C.O.V. > \frac{2}{5}\sqrt{\frac{35}{12}}\frac{\overline{d}+\overline{p}}{2\overline{p}}\left(\frac{\overline{d}}{\rho(\overline{d}+\overline{p})}-1\right), \tag{9}$$

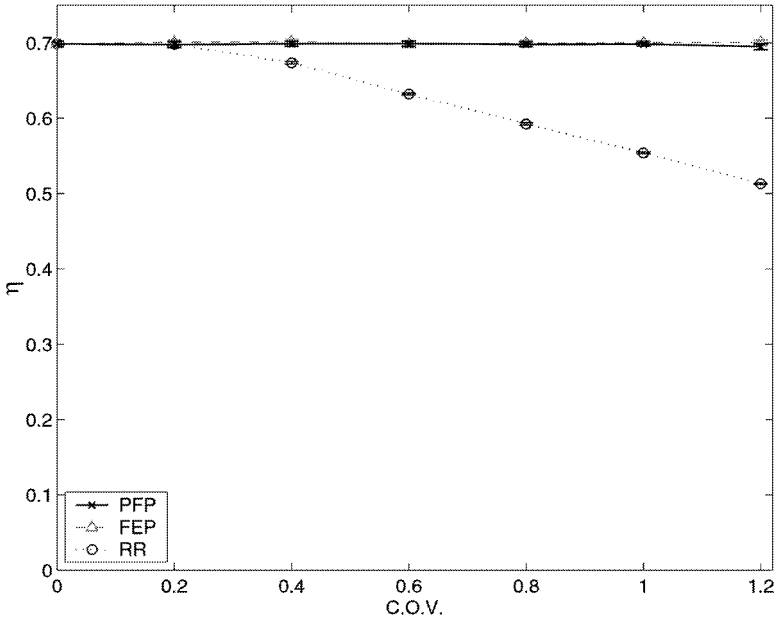

Figure 3. Efficiency (η) in a highly loaded Piconet ($\rho = 0.7$).

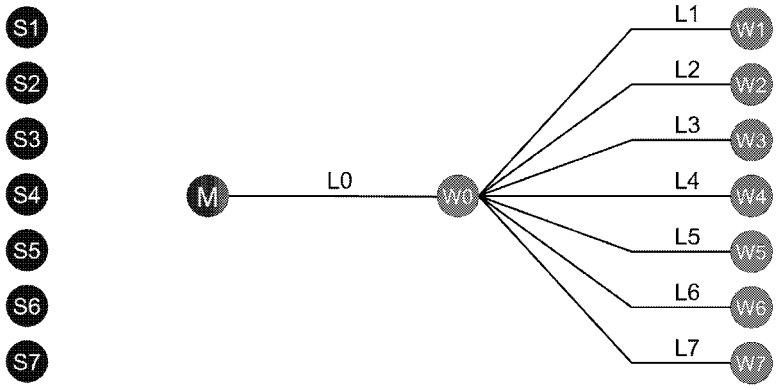

Figure 4. Network setup for simulation of the FTP/TCP scenario.

i.e. when $C.O.V. > 0.27$ and thus $\lambda_a < 16.24$ which implies $\lambda_b > 27.87$ (see Figures 2(b) and 3).

5.2. The FTP/TCP Scenario

In this scenario we simulated using the network setup pointed out in Figure 4. We simulated under the following assumptions:

- Each slave Si uploads data to wired node Wi through master M and wired node W0 using FTP/TCP, while capacity C_0 of the wired duplex link L0 is much higher than the total capacity in a Piconet.
- The available Bluetooth Baseband packet types are DH1, DH3 and DH5 with a payload of 27 bytes, 183 bytes and 339 bytes respectively.
- The MTU size is 1500 bytes.

– The capacities ($C_1 \ldots C_7$) of the wired duplex links (L1...L7) obey:

$$C_1 = C_2 = C_3 = C_4 = C_5 = C_a, \tag{10}$$

and:

$$C_6 = C_7 = C_b \geq C_a\,, \tag{11}$$

while:

$$C_{tot} = \sum_{i=1}^{7} C_i \leq C_{Bluetooth}\,. \tag{12}$$

– The coefficient of variation of the link capacities is defined as:

$$C.O.V. = \frac{\sqrt{\frac{\sum_{i=1}^{7} C_i^2 - \frac{1}{7}(\sum_{i=1}^{7} C_i)^2}{6}}}{\frac{1}{7}\sum_{i=1}^{7} C_i}\,, \tag{13}$$

where $C.O.V. = 0$ implies that the link capacities $C_1 \ldots C_7$ are equal, i.e. $C_a = C_b$.

Due to the assumption on the link capacities ($C_1 \ldots C_7$) each combination of the coefficient of variation of the link capacities and the sum of the link capacities (C_{tot}) gives a unique solution for the link capacities $C_1 \ldots C_7$ and thus makes it possible to use *C.O.V.* and C_{tot} as input for the simulations.

Because of the different link capacities and the use of TCP, the slaves will generate data at different rates. Hence, the poller must adapt to these different rates by polling some slaves more often than other slaves in order to maximize the throughput T (total data upload rate) while being fair. We will compare the Predictive Fair Poller, the Fair Exhaustive Poller, and the Round Robin poller taking the following performance aspects into account:

– Throughput (T), which is defined as the average number of bits per second received by the master from the slaves.
– Efficiency (η), which is defined as the number of data slots divided by the total number of slots.

Based on the simulation results we make the observation that in a lowly loaded Piconet ($C_{tot} = 100000$ bps) the three pollers perform equally well with respect to the throughput T as function of the coefficient of variation (*C.O.V.*) of the link capacities, i.e. $T \rightarrow C_{tot}$. However, in a highly loaded Piconet ($C_{tot} = 600000$ bps, $\rho \approx 0.8125$) the Round Robin poller achieves a lower throughput (see Figure 5(a)) than both the Predictive Fair Poller and the Fair Exhaustive Poller for increasing values of *C.O.V.*. This means that the Round Robin poller often polls slaves that have no data instead of polling slaves that have data available for transmission, which leads to a less efficient (see Figure 5(b)) and less fair poller ($\eta < \rho$).

6. Conclusions

Polling in Bluetooth is highly determining with respect to performance, especially in a highly loaded Piconet with different traffic demands. We explained how the Predictive Fair Poller works and pointed out one way of extending it to be suitable to handle QoS traffic. Through simulations we compared the Predictive Fair Poller with the Fair Exhaustive Poller and the

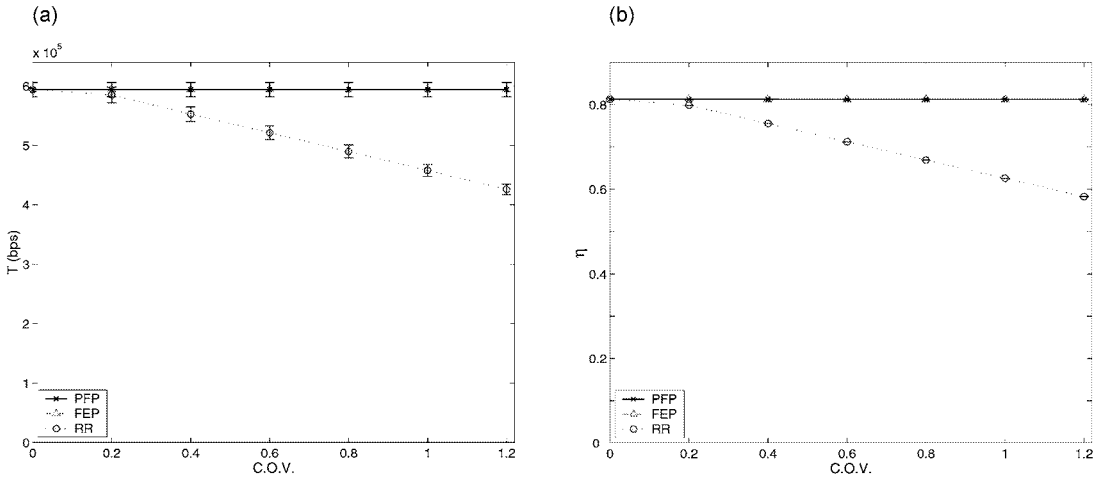

Figure 5. Throughput (T) and Efficiency (η) in a highly loaded Piconet ($C_{tot} = 600000$ bps). (a) Throughput (T). (b) Efficiency (η).

Round Robin poller in a Poisson scenario and in an FTP/TCP scenario. Simulation results pointed out that the Predictive Fair Poller outperforms the Round Robin poller with respect to all studied performance metrics and that it performs at least as well as the Fair Exhaustive Poller.

In this paper, we have analyzed the essentials of the PFP behavior with respect to Best Effort traffic. Future work includes the definition of the QoS capabilities of the Predictive Fair Poller in more detail, and the analysis of its performance behavior under various circumstances.

References

1. "Specification of the Bluetooth System; The Bluetooth Consortium, Version 1.0B", http://www.bluetooth.com.
2. G. Heijenk and R. Ait Yaiz, "Predictive Fair Polling in a Wireless Access Scheme, Application for United States Patent, No. USPTO 09/954,780", filed September 17, 2001.
3. R. Ait Yaiz and G. Heijenk, "Polling in Bluetooth, a Simplified Best Effort Case", in *Proceedings of the 7th annual CTIT Workshop*, Enschede, the Netherlands, 2001, pp. 91–96.
4. N.J. Johansson, U. Körner and P. Johansson, "Performance Evaluation of Scheduling Algorithms for Bluetooth", in *Proceedings of IFIP TC6 Fifth International Conference on Broadband Communications '99*, Hong-Kong, 1999.
5. M. Kalia, D. Bansal and R. Shorey, "MAC Scheduling and SAR Policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System", in *Proceedings of the Sixth International Workshop on Mobile Multimedia Communications*, San Diego, California, 1999, pp. 384–388.
6. I. Chakraborty, A. Kashyap, A. Rastogi, H. Saran, R. Shorey and A. Kumar, "Policies for Increasing Throughput and Decreasing Power Consumption in Bluetooth MAC", in *Proceedings of the IEEE International Conference on Personal Wireless Communications 2000*, Hyderabad, India, 2000, pp. 90–94.
7. A. Das, A. Ghose, A. Razdan, H. Saran and R. Shorey, "Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless As-hoc Network", in *Proceedings of IEEE Infocom*, Anchorage, Alaska, 2001.
8. 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, 2001, pp. 43–49.

9. R. Rao, O. Baux and G. Kesidis, "Demand-based Bluetooth Scheduling", in *Proceedings of the Third IEEE Workshop on Wireless Local Area Networks*, Boston, Massachusetts, 2001.
10. A. Capone, M. Gerla and R. Kapoor, 'Efficient Polling Schemes for Bluetooth Picocells", in *Proceedings of the IEEE International Conference on Communications 2001*, Helsinki, Finland, 2001.
11. "The Network Simulator (ns2)", software and documentation available from http://www.isi.edu/nsnam/ns.
12. J. Nielsen, "IP Routing Performance in Bluetooth Scatternets: a Simulation Study".
13. R. Epsilon, J. Ke and C. Williamson, "Analysis of ISP IP/ATM Network Traffic Measurements", *Performance Evaluation Review*, Vol. 27, No. 2, 15–24, 1999.

Rachid Ait Yaiz (1974) received the B.Sc. degree in electrical engineering from the Technische Hogeschool Arnhem, the Netherlands, in 1996 and the M.Sc. degree in electrical engineering from the University of Twente, the Netherlands, in 1999. Currently, he is working towards a Ph.D. degree at the Department of Computer Science at the University of Twente. His research interests include mobile and wireless networks and he is particularly interested in the area of quality of service over mobile and wireless networks.

Geert Heijenk (1965) received the M.Sc. degree in computer science from University of Twente, the Netherlands, in 1988. He worked as a research staff member at the same university and received the Ph.D. degree in telecommunications in 1995. He has also held a part-time position as researcher at KPN research, the Netherlands, from 1989 until 1991. In 1995, he joined Ericsson, where he is currently working as a research department manager. Since 1998 he is also a part-time senior researcher at the University of Twente. His research interests include mobile and wireless networks, resource management, and quality of service.