Providing Delay Guarantees in Bluetooth

Rachid Ait Yaiz[1] and Geert Heijenk[1,2]

[1]Dept. of Computer Science
University of Twente
P.O.Box 217, 7500 AE
Enschede, The Netherlands

[2]Ericsson EuroLab Netherlands
Business and Science Park 25
P.O.Box 645, 7500 AP
Enschede, The Netherlands

{r.aityaiz,geert.heijenk}@utwente.nl

Abstract

Bluetooth polling, also referred to as Bluetooth MAC scheduling or intra-piconet scheduling, is the mechanism that schedules the traffic between the participants in a Bluetooth network. Hence, this mechanism is highly determining with respect to the delay packets experience in a Bluetooth network. In this paper, we present a polling mechanism that provides delay guarantees in an efficient manner, and we evaluate this polling mechanism by means of simulation. It is shown that this polling mechanism is able to provide delay guarantees while saving as much as possible resources, which can be used for transmission of best effort traffic or for retransmissions.

1. Introduction

Bluetooth [4] is a wireless access technology, which was initially developed as a replacement for cables. However, Bluetooth has evolved to an access technology that can be used in new areas not comprised before. We believe that voice and video will be involved in these new areas, and that applications dealing with voice and video will become available. Applications that deal with voice and video require a network that causes small packet delays or at least bounded packet delays. In order for Bluetooth to be useful to such applications, it must ensure that packet delays are low or at least bounded.

Bluetooth uses a polling mechanism to divide bandwidth among the participants. Together with error recovery, paging, and inquiry this polling scheme is highly determining with respect to the packet delay. Polling mechanism in Bluetooth are studied in [7, 3, 8, 12, 6, 5], and [1] (see also Section 3). However, none of the studied pollers is able to guarantee packet delay bounds in its current state. This paper presents a polling mechanism that is able to guarantee delay bounds in an efficient manner. Section 2 summarizes the Guaranteed Service approach of providing packet delay guarantees. Section 3 shows how the Guaranteed Service approach can be implemented in Bluetooth. Further, section 4 evaluates the proposed implementation. Finally, section 5 concludes this paper and mentions future work.

2. The Guaranteed Service approach

The Guaranteed Service approach [13] (GS) makes use of the concept that packet delay in a network is a function of the arrival pattern of packets, the packet sizes, and the way these packets are served throughout the network. It states that if a flow is described using a token bucket [11] flow specification, and if each network element in the GS path computes and exports parameters that describe the way it provides a requested fluid model bandwidth R, then a delay bound $d_{\max}$ can be computed given a requested fluid model bandwidth R by

$$d_{\max} = \begin{cases} \frac{b-M}{R}\frac{p-R}{p-r} + \frac{M+C_{tot}}{R} + D_{tot}, & r \leq R < p, \\ \frac{M+C_{tot}}{R} + D_{tot}, & r \leq p \leq R. \end{cases} \quad (1)$$

The parameters that each network element exports represent the maximum additional queueing delay a packet will experience compared with the case which a dedicated wire of bandwidth R (fluid model) would have been used in. More precisely, each network element exports the rate-dependent deviation C, and the rate-independent deviation D from the fluid model, while C_{tot} and D_{tot} are the sum of the deviations taken over all network elements in the GS path. Furthermore, the token bucket specification consists of peak rate p, token rate r, bucket size b, minimum policed unit m and maximum transfer unit M. Summarizing, if an application specifies its traffic using a token bucket traffic specifi-

Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)
0-7695-1921-0/03 $17.00 © 2003 IEEE

cation, and if the network elements in the GS path export their deviation from the fluid model, then, provided that $D_{tot} < d_{\max}$, a fluid model service rate R can be requested such that a desired delay bound $d_{\max}$ is achieved.

3. Implementation of the Guaranteed Service approach in Bluetooth

Bluetooth is a wireless access technology that operates in the 2.4 GHz ISM (Industrial Scientific Medical) band. Bluetooth nodes are either a master or a slave, and communication only takes place between the master and a slave, and never directly between two slaves or two masters. One master and up to seven slaves can be affiliated with each other and form a so-called piconet. A slave can be affiliated with multiple masters, making it possible to interconnect piconets forming a so-called scatternet. 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, while other protocols might be used on top of Bluetooth (e.g. IP over Bluetooth). The traffic within a piconet is controlled by the master of that piconet such that a slave is only allowed to transmit if it was addressed (by the master) in the previous time slot. In other words, the master polls the slaves to allow them to transmit data if available. Bluetooth supports two types of links between a master and a slave: a *Synchronous Connection-Oriented (SCO)* link and an *Asynchronous Connection-Less (ACL)* link. Baseband packets sent over an SCO link (SCO packets) cover one time slot while baseband packets sent over an ACL link can cover one, three, or five time slots. In case of an SCO link between the master and a slave, the master polls that slave at regular intervals. The addressed slave can then respond with an SCO packet. In case of an ACL link, polling can be done in many different ways. The difference between the polling mechanisms is related to the order which slaves are polled in and the service discipline used to serve a slave. For instance, the Fair Exhaustive Poller (FEP) [7] and the Efficient Double Cycle (EDC) poller [3] maintain a polling table in order to avoid polling inactive slaves. The Head-Of-Line priority (HOL priority) poller [8] and the Demand-Based poller [12] deal with polling ACL slaves in the presence of SCO channels. The flow bit based pollers [6] and the sniff based poller [5] make use of existing Bluetooth capabilities to respectively track the activity of a slave and to regulate the poll rate. Finally, the Predictive Fair Poller (PFP) [1] 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. A distinguishing feature of the Predictive Fair Poller is that it explicitly takes fairness into account. By proper definition of fairness with respect to providing a particular type of QoS, this poller can be extended to provide that type of QoS.

Higher layer packets cover one or more baseband packets. The way in which higher layer packets are segmented into baseband packets depends on the segmentation policy and the allowed baseband packet types. For instance, a segmentation policy may require that the largest available baseband packet is used, unless there is a smaller baseband packet available in which the remainder of the higher layer packet fits. Note that the ratio of baseband header size and guard space to baseband packet size decreases for larger baseband packets. Hence, the larger the used baseband packet the higher the net number of bytes per slot. Consequently, a slot rate cannot directly be translated to a bit rate. Furthermore, with respect to the upstream traffic (slave to master), the master lacks knowledge about the availability of data at a slave. As a result, the master sometimes needs to poll a slave more often than needed.

The provisioning of Guaranteed Service in a network requires the source to provide a traffic specification and a desired delay bound, and it requires the receiver to calculate the proper bandwidth request. Furthermore, it requires the network elements to compute and export their deviation from the fluid model, and it requires a mechanism (not necessarily RSVP) that transports all specifications and requests as well as the exported values, between the source, the destination, and the intermediate network elements. Finally, it requires the network elements to perform admission control, and to schedule the Guaranteed Service traffic as promised. In this paper, we focus on the determination of the C and D error terms, on the admission control, and on the scheduling of the Guaranteed Service traffic. As the C and D error terms and the admission control are directly related to the polling mechanism (i.e. scheduling mechanism), they are studied in the context of a polling mechanism. First, we introduce a polling mechanism that plans polls with a fixed interval. Next, we show the shortcomings of this fixed interval poller and introduce a variable interval poller, which is an improved version of the fixed interval poller. Finally, we evaluate the proposed polling mechanisms by means of simulation. Note that we study the implementation of the Guaranteed Service in a single piconet. In this paper, we restrict ourselves to an ideal radio environment where no transmission errors occur and where retransmissions are not needed. We assume that no inquiry or paging procedures take place and thus that all the time slots are available for data transmission. Furthermore, we assume the availability of logical channels where a poll for a QoS (e.g. Guaranteed Service) flow cannot result in BE data to be transmitted, and where BE traffic and QoS traffic are queued separately to prevent BE traffic from interfering

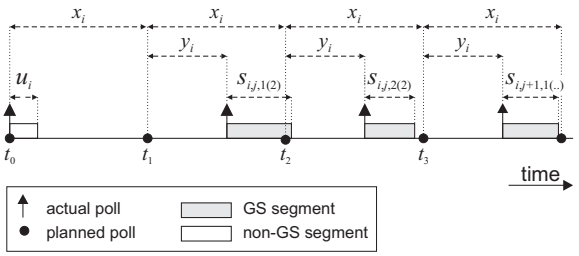

Figure 1. Planning polls with a fixed time interval.

with QoS traffic within a node.

3.1. Planning polls with a fixed time interval

Given the requested bandwidth (R_i) and the token bucket specification (r_i,b_i,p_i,m_i,M_i) of a flow i, the poll rate that must be supported can be computed. An obvious way to poll at a given poll rate is to calculate the average inter-poll time x_i that results in the given poll rate, and to plan polls with a time spacing equal to the calculated average inter-poll time. This is shown in Fig. 1.

Let $s_{i,j,k(l_{i,j})}$ be the transmission time (duration of both upstream and downstream baseband packet) of the k-th segment out of $l_{i,j}$ segments of the j-th packet that belongs to flow i. Furthermore, let u_i be the transmission time following an unsuccessful poll, where an unsuccessful poll is a poll that did not result in data belonging to flow i. The time which a poll takes place at corresponds to the time which a master to slave transmission starts at. In Bluetooth, the slave starts its transmission at least one time slot (0.625 ms) after the master started its transmission, dependent on whether the master transmitted one, three or five slots to the slave. Consequently, for that poll to results in data to be transmitted from the slave to the master, that slave does not necessarily has to have its data available for transmission at the time the master starts its transmission to that slave. However, we require in our study the data to be available at the time the master start its transmission. For instance, in Fig. 1, data that becomes available at t_0^+ ($= t_0 + \delta$, where $\delta \downarrow 0$) will not be served as a result of the poll at t_0, but has to wait for the next poll. Once the planned time for a poll arrives, the poller can be busy polling another slave and also multiple polls can be planned for the same time. Consequently, a planned poll can be delayed by a period of at most y_i. We will discuss the determination of y_i in Section 3.1.2. It can be seen from Fig. 1 that a packet can experience an initial delay of $x_i + y_i$ before the transmission of its first segment begins. The poller may clear away this initial delay by reserving a poll rate that is higher than the poll rate that corresponds to the requested fluid model bandwidth (R_i). However, the Guaranteed Service provides the means to communicate this deviation to the requester (receiver). Hence, we accept this initial delay and propagate it to the exported error terms, which may result in a higher fluid model bandwidth (R_i) request.

3.1.1. Determining the value of x_i

For the determination of x_i we use Fig. 1, which shows a worst case with respect to the delay a packet experiences. Consider packet j of flow i with a size $L_{i,j}$ (in bytes) that will be broken up into $l_{i,j}$ segments. If this packet becomes available at t_0^+, then it will not be served during the poll at t_0, but it will be served during the next poll, which is planned for t_1. The poll planned for t_1 can be delayed for at most a time period y_i, resulting in an initial delay of $x_i + y_i$. As mentioned before, we decided to accept this initial delay and to ensure that all the segments are served within a time period $\frac{L_{i,j}}{R_i}$ after the first actual poll (at $t_0 + x_i + y_i$). By means of proper determination of y_i (see Section 3.1.2) and by means of admission control (see Section 3.1.4), we will ensure that the transmission of a packet j of flow i will be completed at most a time period $x_i + y_i + l_{i,j}x_i$ after the moment it becomes available for transmission. In other words, after experiencing an initial delay of $x_i + y_i$, a packet j of flow i will be served within a time period $l_{i,j}x_i$. Hence, to ensure that, after an initial delay of $x_i + y_i$, all segments of a packet j of flow i are served within a time period $\frac{L_{i,j}}{R_i}$, the following must hold

$$l_{i,j}x_i \leq \frac{L_{i,j}}{R_i}, \quad m_i \leq L_{i,j} \leq M_i, \tag{2}$$

and thus

$$x_i \leq \frac{\frac{L_{i,j}}{l_{i,j}}}{R_i}, \quad m_i \leq L_{i,j} \leq M_i. \tag{3}$$

Let us introduce the poll efficiency $\epsilon_{p_{i,j}}$, which is the average number of bytes per poll that is associated with packet j of flow i. The poll efficiency $\epsilon_{p_{i,j}}$ is also a result of the segmentation policy that is followed as well as the set of baseband packet types that is allowed to be used. The minimum poll efficiency of a flow i taken over all possible packet sizes (i.e. for $m_i \leq L_{i,j} \leq M_i$) is

$$\epsilon_{p_i}^{\min} = \min_{m_i \leq L_{i,j} \leq M_i} \frac{L_{i,j}}{l_{i,j}}. \tag{4}$$

Consequently, the maximum poll interval that always satisfies (3) is

$$x_i = \frac{\epsilon_{p_i}^{\min}}{R_i}. \tag{5}$$

a. Let $s^{\max} = \max_{n,m,k} s_{n,m,k(l_{n,m})}$ be the maximum transmission time of a segment in the Bluetooth piconet

b. Let $y_i = s^{\max}$

c. Let the accumulative service time be $\mu_{acc} = s^{\max} + \sum_{n=1}^{i-1} \text{ceil}(\frac{y_i}{x_n}) s_n^{\max}$, where $s_n^{\max} = \max_{m,k} s_{n,m,k(l_{n,m})}$ is the maximum transmission time of a segment belonging to flow n

d. If $\mu_{acc} \leq y_i$, then goto step h

e. Let $y_i = \mu_{acc}$

f. If $y_i > x_i$ (avoid infinite loop), then goto step h

g. Goto step c

h. End

Figure 2. Determination of y_i.

3.1.2. Determining the value of y_i

Each planned poll has to wait for an ongoing poll to finish, which can also be a poll for the same flow. Furthermore, polls for different flows can be planned for the same moment. If at random one of the planned polls is chosen and executed, all the flows should take into account the fact that they may have to wait for all the other flows, and hence each flow i will experience the same large value of y_i. On the other hand, if each flow is assigned a priority and if polls for a particular flow only have to wait for polls for flows that are assigned a higher priority, the maximum time that the planned polls have to wait for is minimized. Consequently, we decided to assign each flow a priority, and to execute the planned polls for flows with the highest priority first.
Consider the flow number being the priority value of a flow, where a lower flow number corresponds to a higher priority. In order to determine the value of y_i for flow i, the algorithm from Fig. 2 can be followed. In that algorithm, the value of y_i is given an initial value of $s^{\max}$, which corresponds to the maximum transmission time of a segment in the Bluetooth piconet. The reason for this is that ongoing transmissions cannot be interrupted. If higher priority flows are present, the value of y_i is increased until $s^{\max}$ together with the transmission time of the segments that should be drained from these higher priority flows during a period y_i fit into the same period y_i.

Flows can be given a flow number in the order in which they have been accepted. However, maintaining another order may lead to another y_i for a flow i. In Section 3.1.4 we will show how the Admission Control can take advantage of this fact.

3.1.3. Exporting C and D terms

The C error term is the rate-dependent deviation from the fluid model, while the D error term is the rate-independent deviation from the fluid model. It is shown in [2] that the the deviation δF_i of the polling mechanism from the fluid model with respect to serving flow i obeys

$$\delta F_i < x_i + s_i^{\max}. \tag{6}$$

Substituting x_i from (5) in (6) gives

$$\delta F_i < \frac{\epsilon_{p_i}^{\min}}{R_i} + s_i^{\max}. \tag{7}$$

The first term of (7) is rate-dependent and thus $C_i = \epsilon_{p_i}^{\min}$. The last term of (7) is rate-independent and thus $D_i = s_i^{\max}$.

3.1.4. Admission Control

In section 3.1.2, we mentioned that a planned poll can be delayed by an already ongoing poll, while this ongoing poll may be serving the same flow. Furthermore, a planned poll will always be delayed by waiting polls for flows with a higher priority, where a waiting poll is a poll of which the planned time has arrived, but which is waiting for execution. However, a planned poll should never be delayed by waiting polls for slaves with a lower priority, neither should it be delayed by waiting polls for the same flow. The latter can be achieved by ensuring that $y_i < x_i$. Substituting x_i from (5) results in

$$y_i < \frac{\epsilon_{p_i}^{\min}}{R_i}, \tag{8}$$

and thus

$$R_i < \frac{\epsilon_{p_i}^{\min}}{y_i}. \tag{9}$$

From (9) we see that the allowed fluid model bandwidth (R_i) is inversely proportional to y_i, and thus the lower y_i the higher the maximum fluid model bandwidth that can be accepted. Furthermore, the higher the priority of a flow i the lower the value of its y_i. If we were to assign a new GS flow a lower priority than the GS flows that are already accepted, that new flow will have the highest y value, which will possibly make it impossible to accept the new flow. Furthermore, some already accepted GS flows may accept an y value higher than the one they currently experience. Hence, the Admission Control should reassign the priorities such that all the GS flows, including the new one, comply with (9). We assume the availability of logical channels distinguishing between QoS traffic and Best Effort (BE) traffic, and that QoS traffic always has priority over BE traffic.

Consequently, a poll for a GS flow in one direction also gives the opportunity to transmit GS traffic in the opposite direction. In other words, each GS poll of a slave implies an opportunity to transmit GS data in either direction. Taking this fact into account, we improve the Admission Control routine in order to be able to accept more flows.

Consider a GS flow k in one direction and a GS flow l in the opposite direction, which are set up between the master and a slave Sn, where $x_k \leq x_l$. Whenever slave Sn is polled, the next poll is planned no earlier than x_k after the planned time of the last poll, i.e. the minimum poll interval is x_k and, by definition, both flows have the same maximum segment size. Knowing that flow l will piggyback on flow k, the admission control should take into account only the request from flow k.

Consider $i-1$ accepted GS flows and a new request for a GS flow. We propose the method pointed out in Fig. 3 of reassigning the priorities and applying admission control. Note that the number of GS request for which the value of y will be changed can be minimized. For this, if a GS flow in the opposite direction of the new GS flow for which a request has just been made exists, we assign the new GS flow an initial priority value that equals the priority value that is already assigned to its counterpart (GS flow in opposite direction). Otherwise, we assign the new GS flow an initial priority value that equals the highest priority value until then increased by one. Furthermore, let the search action in step e of Fig. 3 be performed in descending order of initial priority value.

a. Store the current priorities

b. Let Q be a set consisting of both the already accepted GS flows and the new GS flow for which a request has just been made, where card$(Q) = i$

c. Let W be a set consisting of GS flows, for which it is assumed that no request has been made. Initially, we set $W = \emptyset$

d. In Q, for each two oppositely directed GS flows in which the same slave is involved, the GS flow with the largest value of x is moved from Q to W

e. If in Q there is a GS flow that if assigned priority card(Q), still complies with (9) then assign priority card(Q) to that flow and to its counterpart in W (if any) and remove the GS flow from Q. Otherwise, reject the new flow and goto step g

f. If card$(Q) > 0$ goto step c. Otherwise, accept the new flow and goto step h

g. Restore the old priorities

h. End

Figure 3. Admission Control routine taking piggybacking into account.

3.2. Polling mechanism improvement

The fixed interval poller of Section 3.1 plans polls for a flow i with a fixed interval x_i. The poll interval x_i is determined taking into account the packet size L_{p_i} that is associated with the least number of bytes per poll (minimum poll efficiency). This leads to three drawbacks. First, the range of packet sizes may comprise more packet sizes (i.e if $M_i > m_i$). In that case, interval x_i is too small when other packet sizes than L_{p_i} are used, and flow i is then polled more often than necessary. Second, if a planned poll for flow i is delayed, the next poll will be planned for x_i after the last time a poll for flow i was planned for, even if that poll did not result in a GS segment of flow i. Finally, planned polls are executed even if it is known that no GS data is available (GS flow from master to slave).

These drawbacks do not adversely affect the performance of the GS flows. On the contrary, polling a GS flow more often than necessary will decrease the average delay of its packets. However, polling the GS flows more often than needed consumes the resources that could otherwise be used for retransmissions (in a non-ideal radio environment) and/or for transmission of BE traffic. We propose three improvements to eliminate these drawbacks. First, if a poll for flow i resulted in a last segment of a packet with size $L_{i,j}$, then plan the next poll a time $\frac{L_{i,j}}{R_i}$ after the time the first poll for packet j of flow i was planned for. Hence, postpone the next poll a time $g_{i,j}$, where (see also (2))

$$g_{i,j} = \frac{L_{i,j}}{R_i} - l_{i,j}x_i \geq 0. \tag{10}$$

Second, if a poll for flow i did not result in a GS segment of flow i, then obviously no GS segment of flow i was available before that actual poll time. As a result, plan the next poll a time period x_i after the actual time of the last poll for flow i. Finally, if at a planned poll time the poller finds out that there is no GS traffic to serve by that poll, then that poll is skipped. The poller has only knowledge of traffic from the master to the slave, hence this improvement only applies to GS flows that are directed from the master to the slave.

If the source of flow i offers its data using the packet size that leads to the minimum poll efficiency ($\epsilon_{p_i}^{\min}$), then the next poll after each last segment is planned for exactly x_i after the last time a poll was planned for (i.e. $g_{i,j} = 0$). The determination of x_i and y_i, the exportation of the C and D error terms and the admission control should take this worst

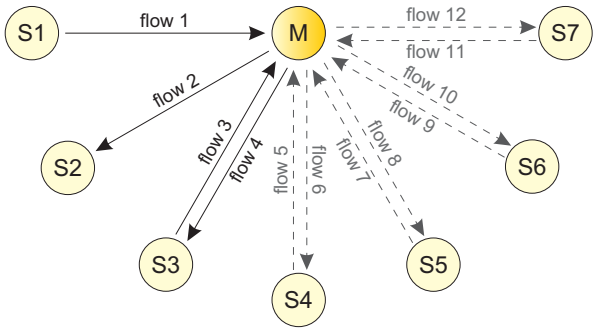

Figure 4. Simulation setup.

case into account, hence they are the same as for the poller presented in Section 3.1.

4. Evaluation

We introduced a poller named Predictive Fair Poller (PFP) in [1]. This poller predicts the availability of data for each slave, and it keeps track of fairness. Based on these two aspects, it decides which slave to poll next. In the BE case, a fair share of resources is determined for each slave, and the fairness is based on the fractions of these fair share of resources. In the QoS case, the QoS requirements are translated to fair QoS treatments, and fairness is based on the fractions of these fair QoS treatments. PFP can be used to poll QoS traffic such that QoS requirements are met by proper translation of the QoS requirements into fair QoS treatments and by keeping track of the fairness based on the fractions of these fair QoS treatments.
We evaluate the PFP implementation of the variable interval poller (see Section 3.2) by means of simulations in a Guaranteed Service scenario. The simulation tool we used is Network Simulator (ns2) [10] with Bluetooth extensions [9] from Ericsson Switchlab, together with our ns2 implementation of PFP. We show that PFP is saving bandwidth, which can be used for transmission of BE traffic and/or for retransmissions. Furthermore, we show that taking piggybacking of GS flows into account makes it possible to accept more GS flows. Finally, we show that PFP fairly divides the remaining bandwidth among the slaves that need it.

4.1. Description of the simulation scenario

We consider the simulation setup of Fig. 4, which seven slaves and a master form a piconet in. Flows 1 to 4 are GS flows, which the same delay bound is requested for. Furthermore, flows 5 to 12 are BE flows. For the GS flows, the packet sizes are uniformly distributed with a minimum size of 144 bytes, and a maximum size of 176 bytes, i.e. $m_i = 144$ bytes and $M_i = 176$ bytes for each GS flow i. For the BE flows the packet sizes are of a fixed size of 176 bytes. The sources of the GS flows generate packets with fixed intervals of 20 ms resulting each in a data rate of 64 kbps. The sources of the BE flow generate packets with fixed intervals that depend on the BE load. The sources of flows 5/6, 7/8, 9/10 and 11/12 generate BE traffic at a data rate of 41.6 kbps, 47.2 kbps, 52.8 kbps and 58.4 kbps respectively. The allowed baseband packet types are DH1 and DH3, with a maximum payload of 27 bytes and 183 bytes respectively. Furthermore, the segmentation policy requires that the DH3 baseband packet is used, unless the remainder of the packet fits in the DH1 baseband packet.
Because of the fixed intervals between packet generations, and because of the packet size distribution, the remaining parameters of the token bucket specification are

$$p_i = r_i = \frac{M_i}{20 \text{ ms}} = 8.8 \text{ kbytes/s}, \quad i \in \{1, 2, 3, 4\}, \quad (11)$$

and

$$b_i \geq M_i, \quad i \in \{1, 2, 3, 4\}. \quad (12)$$

Because of the packet sizes the source of each GS flow i can use, and because of the allowed baseband packet types, the minimum poll efficiency $\epsilon_{p_i}^{\min}$ is achieved by a packet size of 144 bytes, which is sent using one DH3 baseband packet. Hence, the C error term for these flows is given by $C_i = \epsilon_{p_i}^{\min} = 144$ bytes. As all the nodes are allowed to use DH3 baseband packets, the possibility must be taken into account that both the master and the addressed slave transmit a DH3 packet. Consequently, the D error term is given by $D_i = 2\frac{3}{1600} = 3.75$ ms for each GS flow i.
The algorithm used to determine y_i (see Fig. 2) gives $y_1 = 3.75$ ms, $y_2 = 7.5$ ms and $y_3 = y_4 = 11.25$ ms. According to (9), a requested service rate R can be supported for each of the GS flows as long as $R < \frac{\epsilon_{p_3}^{\min}}{y_3} = 12.8$ kbytes/s. Substitution in (1) shows that a requested delay bound $d_{\max}$ can be supported for each of the GS flows as long as $d_{\max} > 28.75$ ms. As the requested service rate R_i should always be greater than or equal to the token rate r_i, the delay bound $\hat{d}_{\max_i}$ that will never be exceeded can be found by substituting $R_i = r_i$ in (1), i.e. $\hat{d}_{\max_i} \approx 40.11$ ms for each GS flow i.

4.2. Simulation results

We showed that requested delay bounds can be achieved if the master polls slaves according to the methods described in Section 3.1 and Section 3.2. Simulation runs, each of a simulation time of 530 seconds (>25000 samples of each GS flow), showed that the requested delay bound is not exceeded. Depending on the delay requirement, the

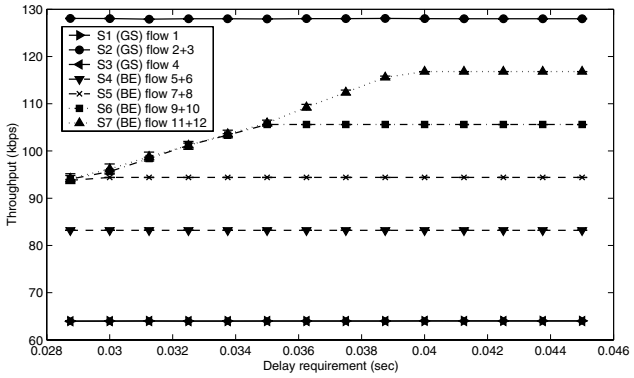

Figure 5. Throughput vs. delay requirement.

poller saves an amount of bandwidth that can be used for retransmissions in a non-ideal radio environment and/or for transmission of BE traffic. Fig. 5 shows the throughput of each slave as a function of the GS delay requirement. It can be seen that, independent of the delay requirement, each GS flow achieves a throughput of 64 kbps. Furthermore, it can be seen that the BE flows achieve their maximum throughput when large delay bounds are requested. For decreasing delay bounds, the remaining bandwidth is fairly divided among the BE flows, which explains why some BE flows achieve their maximum throughput as opposed to other BE flows. Note that in this simulation scenario, a total maximum throughput of 656 kbps (including 256 kbps of GS traffic) is achieved.

5. Conclusions

Bluetooth is an access technology where a master uses a polling mechanism to divide bandwidth among the slaves. This polling mechanism is highly determining with respect to the delay that packets experience in a piconet. The PFP implementation of the variable interval poller divides bandwidth among the slaves such that the delay which packets experience is bounded. Furthermore, PFP polls such that a minimum amount of slots is consumed while polling the GS flows, saving bandwidth that can be used for transmission of BE traffic or for retransmission of the QoS traffic. More simulation results can be found in [2]. A comparison with an SCO channel showed that PFP is able to achieve delay bounds that approach the delay bounds that can be achieved using an SCO channel. As opposed to an SCO channel, PFP can use the saved bandwidth for retransmissions. This property can be exploited to avoid the link quality problems of SCO channels in difficult radio environments, while keeping up QoS. Note that the introduced polling mechanisms can also be used outside the context of the Guaranteed Service approach, such that no error terms are exported. A service rate can be requested without the need for a guaranteed delay bound.

Future work includes the evaluation of the proposed polling mechanisms in a non-ideal radio environment, where transmission errors may occur and where retransmissions are needed. Furthermore, the introduced polling mechanisms must be extended with policies that decide which retransmissions to use the saved bandwidth for.

References

[1] R. Ait Yaiz and G. Heijenk. Polling Best Effort Traffic in Bluetooth. *Wireless Personal Communications*, 23(1):195–206, October 2002.

[2] R. Ait Yaiz and G. Heijenk. Providing QoS in Bluetooth. Technical Report TR-CTIT-03-04, Centre for Telematics and Information Technology, April 2003.

[3] 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*, pages 43–49, Rome, Italy, July 2001.

[4] Specification of the Bluetooth System; the Bluetooth Consortium, version 1.0b. http://www.bluetooth.com.

[5] 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*, pages 90–94, Hyderabad, India, December 2000.

[6] 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, April 2001.

[7] 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, November 1999.

[8] 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*, pages 384–388, San Diego, California, November 1999.

[9] J. Nielsen. IP Routing Performance in Bluetooth Scatternets: a Simulation Study. Master's thesis, Department of Computer Systems (DoCS), Uppsala University, Uppsala, 2000.

[10] The Network Simulator (ns2). Software and documentation available from http://www.isi.edu/nsnam/ns.

[11] C. Partridge. *Gigabit Networking*. Addison-Wesley, second edition, December 1993.

[12] 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, September 2001.

[13] C. P. S. Shenker and R. Guerin. Specification of Guaranteed Quality of Service. RFC 2212, IETF, September 1997.