Cluster Computing 8, 223–231, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Providing QoS in Bluetooth

RACHID AIT YAIZ* and GEERT HEIJENK
Department of Computer Science, University of Twente

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.

Keywords: Bluetooth polling, intra-piconet scheduling, delay guarantees, predictive fair poller

1. Introduction

Bluetooth [4] is a wireless access technology, which was initially developed as a replacement for cables. However, Bluetooth has evolved to a wireless 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 mechanisms in Bluetooth are studied in [1,3,5,6,12,13,18] (see also Section 3). However, none of the studied pollers is able to guarantee packet delay bounds in its current state. Bluetooth can use SCO channels to transport some types of traffic that require delay bounds. However, SCO channels cannot transport large packets nor do they have retransmission possibilities.

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. Section 4 evaluates the proposed implementations. Finally, Section 5 concludes this paper and mentions future work.

2. The guaranteed service approach

The Guaranteed Service approach [19] (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 [17] 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^B can be computed given a requested fluid model bandwidth R by

$$d^B = \begin{cases} \dfrac{b-M}{R}\dfrac{r^p-R}{r^p-r^t} + \dfrac{M+C_{\text{tot}}}{R} + D_{\text{tot}}, & r^t \le R < r^p, \\[2ex] \dfrac{M+C_{\text{tot}}}{R} + D_{\text{tot}}, & r^t \le r^p \le R. \end{cases} \tag{1}$$

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 r^p, token rate r^t, bucket size b, minimum policed unit m and maximum transfer unit M. Summarizing, if an application specifies its traffic using a token bucket traffic specification, and if the network elements in the GS path export their deviation from the fluid model, then, provided that $D_{\text{tot}} < d^B$, a fluid model service rate R can be requested such that a desired delay bound d^B is achieved.

3. Implementation of the GS 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. On a time division basis, a Bluetooth node can

* Corresponding author.
E-mail: r.aityaiz@utwente.nl

be a master in one piconet and/or a slave in one or more other piconets, 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. This poll can be either implicitly, i.e. by means of a baseband packet containing data, or explicitly, i.e. by means of a baseband packet containing no data (POLL packet).

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 to the service discipline used to serve a slave. For instance, the Fair Exhaustive Poller (FEP) [12] 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 [13] and the Demand-Based poller [18] 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 on 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. Note that such a mechanism has been specified for the Bluetooth link in the so-called Bluetooth logical link control and adaptation protocol (L2CAP). 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 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 ($r_i^t, b_i, r_i^p, m_i, M_i$) of a GS 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 that results in the given poll rate, and to plan polls with a time spacing $\tilde{p}_i$ (poll period) equal to the calculated average inter-poll time (see Figure 1). Each planned poll must complete execution within a relative deadline $\tilde{d}_i$ from its planned time.

In figure 1, $s_{i,j,k(l_{i,j})}$ is 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, u_i is the transmission time following an unsuccessful poll for flow i, where an unsuccessful poll for flow i is a poll, for the node associated with flow i, that

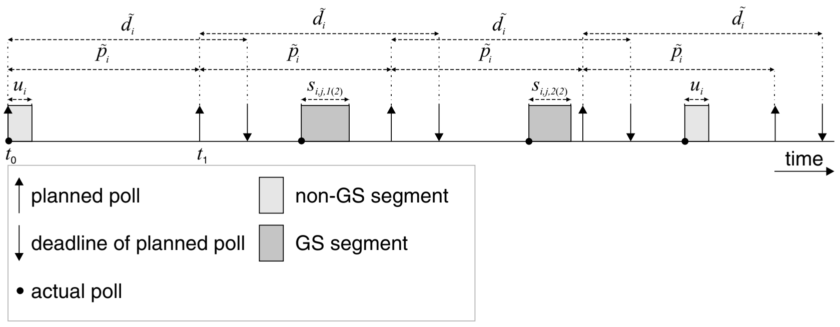


Figure 1. Planning polls with a fixed time interval.

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 Figure 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.

3.1.1. Admission control

Each traffic specification $(r_i^t, b_i, r_i^p, m_i, M_i)$ and corresponding requested fluid model bandwidth (R_i) is ultimately converted to a poll period $\tilde{p}_i$, a relative deadline $\tilde{d}_i$ and a maximum segment size $s_i^{\max} = \max_{j,k} s_{i,j,k(l_{i,j})}$. Consequently, a GS flow i can be looked at as a periodic task. A periodic task τ_i that corresponds to GS flow i is represented by the tuple (p_i, e_i, d_i), where

- $p_i = \tilde{p}_i$ is the fixed interval (period) between two consecutive instances of a task τ_i
- $e_i = s_i^{\max}$ is the maximum execution time of an instance of task τ_i
- $d_i = \tilde{d}_i$ is the relative deadline of each instance of task τ_i

The decision whether a set of n GS flows can be accepted can be done by deciding whether the corresponding task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ can be accepted (feasibility analysis). This decision can only be made if the scheduling policy is also known. The Bluetooth polling mechanism can be modeled as a non-preemptive scheduling policy, which on its turn can be divided into two classes: the class of idling non-preemptive scheduling policies and the class of non-idling non-preemptive scheduling policies. The idling non-preemptive scheduling policies are allowed to insert idle times even if there are task instances waiting for execution. Inserting idle times makes it sometimes possible to schedule task sets that could otherwise not be scheduled under the class of non-idling scheduling policies.

As the feasibility analysis of an idling non-preemptive schedule is NP-Hard in the strong sense [9], we decided to use a non-idling non-preemptive scheduling policy. It is shown in [7, 11, 14] that non-idling non-preemptive Earliest Deadline First (EDF) is optimal among the class non-idling non-preemptive scheduling policies. This means that if a feasible non-idling non-preemptive scheduling policy exists for a given task set, then non-idling non-preemptive EDF will also be feasible for that task set.

Once decided that the planned polls will be executed according to the non-idling non-preemptive EDF scheduling policy, the admission control of a set of n GS flows can be translated to the feasibility analysis of the corresponding task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ under the non-idling non-preemptive EDF scheduling policy. Zheng et al. stated in [20] that in the presence of non real-time tasks, such a task set is feasible if

1.
$$U = \sum_{i=1}^{n} \frac{e_i}{p_i} \leq 1, \text{ and}$$

2.
$$\forall t \in S, \sum_{d_i \leq t} \left(1 + \left\lfloor \frac{t - d_i}{p_i} \right\rfloor\right) e_i + e^{\max} \leq t,$$

where

$$S = \bigcup_{i=1}^{n} \left\{ d_i + np_i : n = 0, 1, \ldots, \left\lfloor \frac{t_{\max} - d_i}{p_i} \right\rfloor \right\}, \tag{2}$$

and

$$t_{\max} = \max \left\{ d_1, \ldots, d_n, \left(\frac{e^{\max} + \sum_{i=1}^{n} (1 - d_i/p_i) e_i}{1 - U} \right) \right\}, \tag{3}$$

and where $e^{\max}$ is the maximum execution time of any task instance (including the non real-time tasks). The first condition ensures that the maximum utilization does not exceed unity, while the second condition ensures that the deadlines can actually be met. In the absence of non real-time tasks, the feasibility conditions mentioned above are sufficient but not necessary. For the case in which there are no non real-time tasks, sufficient and necessary conditions can be found in [8,10,11,14].

3.1.2. Determining poll period $\tilde{p}_i$

The poll period $\tilde{p}_i$ is determined considering the worst case response time a packet can experience. In Figure 1, consider packet j of flow i, with a size $L_{i,j}$ (in bytes), which 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. As a result, the worst case service time of a packet is $l_{i,j}\tilde{p}_i + \tilde{d}_i$. In order to let the poll period $\tilde{p}_i$ be inversely proportional to the requested fluid model bandwidth R_i it is decided to consider the relative deadline $\tilde{d}_i$ of a planned poll as a deviation from the fluid model service time. The remaining part of the worst case service time of a packet should not be larger than the fluid model service time, i.e.

$$l_{i,j}\tilde{p}_i \le \frac{L_{i,j}}{R_i}, \quad m_i \le L_{i,j} \le M_i, \tag{4}$$

and thus

$$\tilde{p}_i \le \frac{\frac{L_{i,j}}{l_{i,j}}}{R_i}, \quad m_i \le L_{i,j} \le M_i. \tag{5}$$

Let us introduce the poll efficiency $\epsilon_{\mathrm{p}_{i,j}}$, which is the average number of bytes per poll that is associated with packet j of flow i. The poll efficiency $\epsilon_{\mathrm{p}_{i,j}}$ is a result of the size $L_{i,j}$ of packet j of flow i, the segmentation policy that is followed, and 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 $m_i \le L_{i,j} \le M_i$) is

$$\epsilon_{\mathrm{p}_i}^{\min} = \min_{m_i \le L_{i,j} \le M_i} \frac{L_{i,j}}{l_{i,j}}. \tag{6}$$

Consequently, the maximum poll period that always satisfies (5) is

$$\tilde{p}_i = \frac{\epsilon_{\mathrm{p}_i}^{\min}}{R_i}. \tag{7}$$

3.1.3. Determining relative deadline $\tilde{d}_i$

As will be seen in Section 3.1.4, the lower the relative deadline $\tilde{d}_i$, the lower the C and D error terms, and thus the lower the requested fluid model bandwidth (see also (1)). However, while $s^{\max}$ is the size of the largest possible segment in the piconet, it is shown in [2] that if $\tilde{d}_i \ge \tilde{p}_i + s^{\max}$ for each GS flow i, then the set of GS flows is schedulable using EDF, if and only if condition 1 of the feasibility analysis holds. Increasing the relative deadline $\tilde{d}_i$ of a GS flow i beyond $\tilde{p}_i + s^{\max}$ further decreases the poll period $\tilde{p}_i$. According to condition 1 of the feasibility analysis, this means that the number of flows that can be accepted also decreases. Furthermore, it is shown in [2] that, with respect to the feasibility of a set of GS flows that flow i will be part of, the effect of decreasing the relative deadline $\tilde{d}_i$ below $\tilde{p}_i + s^{\max}$ cannot be determined on forehand. As the relative deadline $\tilde{d}_i$ of a flow i will be used by the receiver of that GS flow for the determination of the fluid model service rate R_i, the relative deadline $\tilde{d}_i$ should not increase during the lifetime of flow i. Hence, we decided to set the relative deadline of a flow i at

$$\tilde{d}_i = \tilde{p}_i + s^{\max}. \tag{8}$$

Note that an additional advantage of this value for the relative deadline $\tilde{d}_i$ is that the feasibility analysis (admission control) becomes simple as the first condition of the feasibility analysis becomes a necessary and sufficient condition.

3.1.4. Exporting C and D error terms

As mentioned in Section 3.1.2, it is decided to consider the relative deadline $\tilde{d}$ as the C and D error terms, i.e.

$$\begin{aligned}\frac{C_i}{R_i} + D_i &= \tilde{d}_i \\ &= \tilde{p}_i + s^{\max} \\ &= \frac{\epsilon_{\mathrm{p}_i}^{\min}}{R_i} + s^{\max}.\end{aligned} \tag{9}$$

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. From (9), it follows that $C_i = \epsilon_{\mathrm{p}_i}^{\min}$ and $D_i = s^{\max}$. Note that if the Bluetooth hop is not the first hop in the GS path, then C_i should be increased by M_i in order to account for packetization.

3.2. Improvement of the polling mechanism

The fixed interval poller of Section 3.1 plans polls for a GS flow i with a fixed interval $\tilde{p}_i$. The poll interval $\tilde{p}_i$ 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 the following drawbacks:

(a) The range of packet sizes may comprise several packet sizes (i.e if $M_i > m_i$). In that case, interval $\tilde{p}_i$ is too small when other packet sizes than L_{p_i} are used, and GS flow i is then polled more often than necessary.

(b) If a planned poll for GS flow i is executed, the next poll for GS flow i will be planned for $\tilde{p}_i$ after the last time a poll for GS flow i was planned for, even if that poll did not result in a GS segment of flow i.

(c) Planned polls are executed even if it is known that no GS traffic is available. As the master has only knowledge about the availability of traffic that is directed from the master to a slave, this drawback only applies to GS flows from the master to a 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 (see figure 2):

(a) If a poll for GS flow i resulted in a last segment of a packet j with size $L_{i,j}$, then plan the next poll a time $\frac{L_{i,j}}{R_i}$ after the

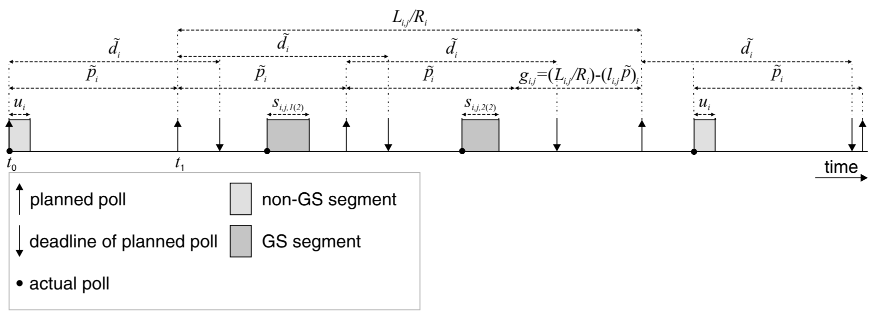


Figure 2. Planning polls with a variable time interval.

planned time of the poll that resulted in the first segment of packet j of GS flow i. Hence, postpone the next poll a time $g_{i,j}$, where (see also (4))

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

(b) 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 $\tilde{p}_i$ after the actual time of the last poll for flow i rather than after its planned time.

(c) If at a planned poll time t_n the poller finds out that there is no GS traffic to serve by that poll, then that poll is skipped, and the next poll is planned at $t_n + \tilde{p}_i$. 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 (not shown in Figure 2).

If the source of flow i offers its data using the packet size that leads to the minimum poll efficiency ($\epsilon_{\mathrm{p}_i}^{\min}$), then the next poll after each last segment is planned for exactly $\tilde{p}_i$ after the last time a poll was planned for (i.e. $g_{i,j} = 0$). The determination of $\tilde{p}_i$, $\tilde{d}_i$, C_i and D_i should take this worst case into account, hence they are the same as for the poller presented in Section 3.1. Furthermore, the task set corresponding to the set of GS flows becomes a sporadic task set. The feasibility analysis for such a set is the same as for a periodic task set, except that the minimum period between instances of a task is now taken into account. As the minimum period is the same as the fixed period determined in Section 3.1, the admission control is the same as described in Section 3.1.1.

3.3. Improvement of the admission control

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 of the same logical channel in the opposite direction. In other words, each GS poll of a slave implies an opportunity to transmit GS traffic of the same logical channel in both directions. Taking this fact into account, we improve the Admission Control 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 particular slave, where $\tilde{p}_k \leq \tilde{p}_l$. Furthermore, the relative deadlines are determined according to Section 3.1.3. If the two GS flows use separate logical channels, then GS flow k and GS flow l have a maximum segment size of $s_k^{\max}$ and $s_l^{\max}$ respectively, which are not necessarily equal, and each GS flows is polled independent of the other. However, if we let two oppositely directed flows that involve the same slave share the same logical channel, then a poll for GS flow k implies a poll for GS flow l and vice versa. Furthermore the resulting maximum segment size will be

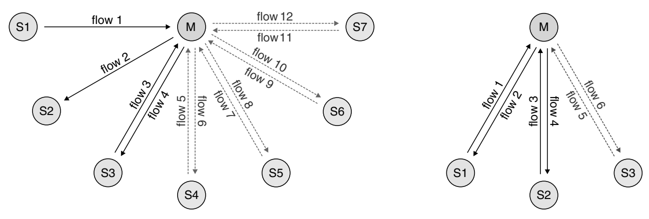
$$s_k'^{\max} = s_l'^{\max} = s_k^{\max} + s_l^{\max} - \frac{2}{1600}, \tag{11}$$

where the two slots were accounting for the empty baseband packets. Whenever the particular slave is polled, the next poll is planned no earlier than $\tilde{p}_k$ after the planned time of the last poll, i.e. the minimum poll interval is $\tilde{p}_k$. By definition, both flows have the same maximum segment size, i.e. $s_k'^{\max} = s_l'^{\max}$. Knowing that flow l will piggyback on flow k, the admission control should take into account only the request from flow k with maximum segment size $s_k'^{\max}$ ($= s_l'^{\max}$). In other words, if two oppositely directed GS flows exist between the master and a particular slave, then the real-time task representing the GS flow with the highest value of $\tilde{p}$ should not be included in the feasibility check, and the two GS flows should share the same logical channel.

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 shares of resources. In the QoS case, this poller can additionally apply an EDF scheme while planning polls according to the descriptions in Sections 3.1 and 3.2.

We evaluate the PFP implementations of the fixed interval poller and the variable interval poller by means of simulations in two Guaranteed Service scenarios. In the first simulation



a. Scenario I b. Scenario II

Figure 3. Simulation setup.

scenario we show the impact of the polling mechanism improvements on the performance of both the guaranteed service flows and the best effort flows. In the second simulation scenario, we compare the PFP implementation of the variable interval poller with an SCO channel. The simulation tool we used is Network Simulator (ns2) [16] with Bluetooth extensions [15] from Ericsson Switchlab, together with our ns2 implementation of PFP.

4.1. Scenario I: Comparison between the fixed interval poller and the variable interval poller

4.1.1. Description of the simulation scenario

In this scenario, simulations are performed using the simulation setup of Figure 3(a). Seven slaves and a master form a piconet, while flows are set up as depicted in the figure. Flows 1 to 4 are GS flows, which the same delay bound is requested for. Furthermore, flows 5 to 12 are BE flows (background traffic). 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 time between two consecutive packet generations of the same GS flow equals the size of the first packet divided by a data rate of 8 kbytes/s (64 kbps). The resulting average time interval between two packet generations of the same GS flow is 20 ms. The sources of the BE flows generate packets with fixed intervals that depend on the BE load. In the first part of simulation scenario I, the delay requirement is set at a fixed value and the sources of the BE flows generate traffic at an equal rate, while simulations are performed at different total BE loads. In the second part of this simulation scenario, the sources of flows 5/6, 7/8, 9/10 and 11/12 generate BE traffic at a data rate of 42.4 kbps, 48 kbps, 53.6 kbps and 59.2 kbps respectively, while simulations are performed at different delay requirements. 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 packet size distribution and the corresponding inter-generation time of packets, the remaining parameters of the token bucket specification are

$$r_i^p = r_i^t = 8 \text{ kbytes/s}, \quad i \in \{1, 2, 3, 4\}, \tag{12}$$

and

$$b_i \geq M_i, \quad i \in \{1, 2, 3, 4\}. \tag{13}$$

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 $\epsilon_{\mathrm{p}_i}^{\min}$ 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 $C_i = \epsilon_{\mathrm{p}_i}^{\min} = 144$ bytes for each GS flow i. 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.

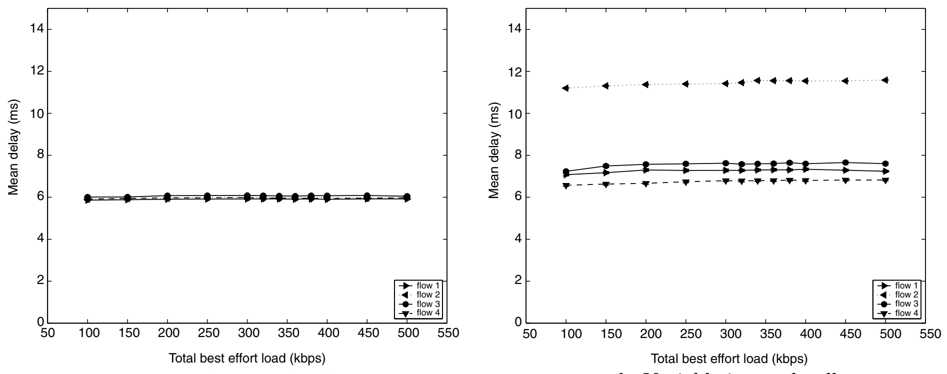
According to Section 3, the GS flows 1 to 4 can be looked at as a set of three periodic or sporadic tasks dependent on whether the fixed interval poller or the variable interval poller is considered. In both cases, each task i is described by a tuple $(p_i, e_i, d_i) = (\frac{C_i}{R_i}, s_i^{\max}, \frac{C_i}{R_i} + D_i)$. All the GS flows are described by equal traffic specifications (token bucket specification), while sharing the same piconet and thus the same maximum possible segment size ($s^{\max}$). As each GS flow is also requesting the same delay bound, the tuple which each GS flow i is described by can be simplified to $(\frac{144}{R}, \frac{4}{1600}, \frac{144}{R} + \frac{6}{1600})$ for each of GS flows 1 and 2, and $(\frac{144}{R}, \frac{6}{1600}, \frac{144}{R} + \frac{6}{1600})$ for the pair of GS flows 3 and 4. Considering the feasibility analysis of Section 3.1.1, the GS flows can be admitted as long as

$$U = 2\frac{\frac{4}{1600}R}{144} + \frac{\frac{6}{1600}R}{144} \leq 1. \tag{14}$$

Consequently, the four GS flows can be admitted as long as $R \leq 16.457$ kbytes/s. This implies that the minimum delay bound that can be requested is $\check{d}^B \approx 23.2$ ms (see (1)). On the other hand, the requested fluid model bandwidth R_i of a GS flow i should never be lower than its token rate r_i^t. Substituting $R_i = r_i^t$ in (1), leads to the delay bound that will never be exceeded, i.e. $\hat{d}_i^B = 43.75$ ms for each GS flow i (see knee at delay requirement of 43.75 ms in figure 5(a)[1]).

4.2. Simulation results

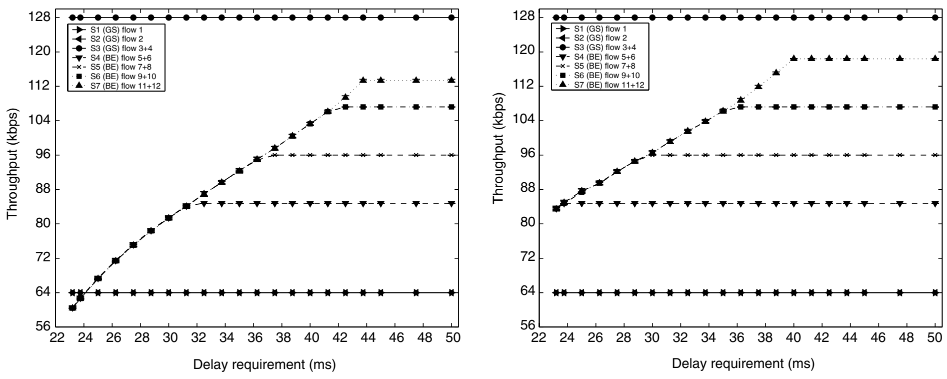
As mentioned in Section 3.2, the fixed interval poller plans polls more often than necessary. This has advantageous impact on the mean delay of the GS flows. As can be seen in figure 4(a), all the GS flows experience the same low mean delay. The reason for this is that the fixed interval poller polls all the GS flows with the same fixed interval. The variable interval poller polls GS flows only when it assumes that it is needed. For instance, flow 2 is a GS flow that is not involved in piggybacking and that is directed from the master to the slave. This flow will only be polled when GS traffic is actually available and when it should be transmitted according to the fluid model. This can be seen in Figure 4(b), where flow 2 experiences a mean delay higher than the ones experienced by the remaining GS flows. Furthermore, these remaining GS flows experience mean delays higher than the ones experienced under the fixed interval poller. The reason for this is that the GS



a. Fixed interval poller

b. Variable interval poller

Figure 4. Scenario I: Mean delay as a function of the total best effort load ($d^B = 23.2$ ms).



a. Fixed interval poller

b. Variable interval poller

Figure 5. Scenario I: Throughput as a function of the delay requirement.

flows are polled less often, depending on the packet sizes they transmit.

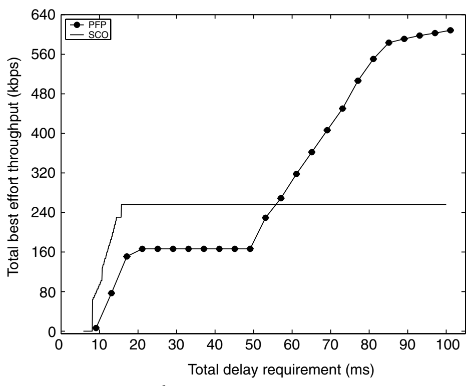
Although the mean delay of the GS flows is higher under the variable interval poller than under the fixed interval poller, delay bounds are never exceeded. Moreover, the variable interval poller consumes less resources, saving resources that can be used for retransmissions (in a non-ideal radio environment) or for the transmission of best effort traffic. Figures 5(a) and (b) show the sum of the upstream and downstream throughput of the different slaves as a function of the delay requirement. As could be expected, the GS flows always achieve their maximum throughput as long as they are admitted by the admission control. The throughput of the best effort flows 5 to 12 depends on the requested delay bound of the GS flows as well as on the BE loads (fairness). It can be seen from figure 5(a) and (b) that the best effort flows achieve a higher throughput when served by the variable interval poller.

4.3. Scenario II: Comparison between the variable interval poller and an SCO channel

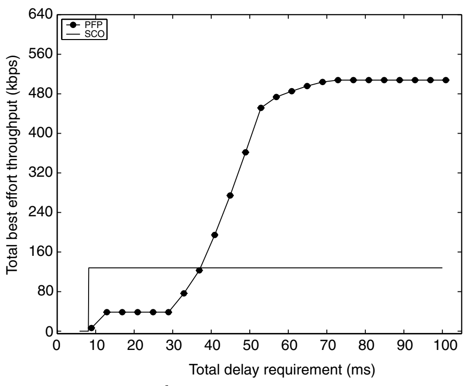
4.3.1. Description of the simulation scenario

In this scenario we compare the variable interval poller with an SCO channel by means of simulations, while considering the simulation setup of figure 3(b). Three slaves and a master form a piconet. Flows 1 to 4 are GS flows, which the same delay bound is requested for. Furthermore, flows 5 and 6 are BE flows generating 1 Mbps of background traffic. The sources of GS flows 1 to 4 are sample based codecs that generate samples (1 sample = 1 byte) at a fixed data rate r^d. In this scenario the total delay that a packet experiences includes the time needed to collect and packetize samples (packetization delay), the queueing delay, and the transmission delay. Taking into account only packets that fit in a single baseband packet, the larger the packet size, the fewer baseband packets are needed to obtain a certain data rate, but also the higher the packetization delay of a packet. Furthermore, the larger the baseband packet, the higher its transmission delay.

The sources of both the GS flows and the BE flows generate packets of the same fixed size L. Assuming fixed packet sizes, we choose the packet sizes such that the total delay requirement is met while maximizing the BE throughput. Note that given a total delay requirement, the GS flows may use a different packet size in the two compared cases (one using an ACL channel with PFP polling, and the other using an SCO channel). The allowed ACL baseband packet type is DH1 unless the chosen packet size exceeds 23 bytes. In that case, DH3 baseband packets will also be allowed. The allowed SCO baseband packet type is HV3, with a maximum payload of 30 bytes. The segmentation policy requires that the largest allowable baseband packet is used, unless the remainder of the packet fits into a smaller baseband packet.



a. $r^{\mathrm{d}} = 4$ kbytes/s (32 kbps)



b. $r^{\mathrm{d}} = 8$ kbytes/s (64 kbps)

Figure 6. Scenario II: Total best effort throughput as a function of the total delay requirement (BE load of 1 Mbps).

Because of the simulation assumptions, the token bucket specification is given by

$$r_i^p = r_i^t = r_i^d, \quad i \in \{1, 2, 3, 4\}, \tag{15}$$

and

$$b_i \geq M_i = m_i = L, \quad i \in \{1, 2, 3, 4\}. \tag{16}$$

4.3.2. Simulation results

Figures 6(a) and (b) show the best effort throughput as a function of the total delay requirement. The PFP line shows the simulated values of the variable interval poller, while the SCO line shows the theoretical value of an SCO channel. It can be seen that the variable interval poller can achieve delay bounds that approach the delay bounds than can be achieved using an SCO channel. Furthermore, we see in case of low data rates and/or low delay bounds that the variable interval poller saves less resources than an SCO channel does. However, the variable interval poller can use the saved resources for retransmissions, which is not possible with an SCO channel. With looser delay requirements, the variable interval poller saves significantly more bandwidth compared to the SCO channel.

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 fixed interval poller and the variable interval poller divide bandwidth among the slaves such that the delay which packets experience is bounded. Furthermore, the variable interval poller 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 and/or for retransmission of QoS traffic. A comparison with an SCO channel showed that the variable interval poller 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.

Note

1. The simulation times are chosen such that each (two-sided) 95% confidence interval is less than 2% of its corresponding determined average.

References

[1] R. Ait Yaiz and G. Heijenk, Polling best effort traffic in bluetooth. Wireless Personal Communications 23(1) (2002) 195–206.

[2] R. Ait Yaiz, Intra-Piconet Scheduling in Bluetooth, PhD thesis, University of Twente, July 2004.

[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*, Rome, Italy (2001) pp. 43–49.

[4] BT, 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*, Hyderabad, India (2000) pp. 90–94.

[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 (2001).

[7] L. George, P. Muhlethaler and N. Rivierre, Optimality and non-preemptive real-time scheduling revisited. Technical Report 2516, Institut National de Researche en Informatique et Automatique (1995).

[8] L. George, N. Rivierre and M. Spuri, Preemptive and non-preemptive real-time uniprocessor scheduling, Technical Report 2966, Institut National de Researche en Informatique et Automatique (1996).

[9] R. Howell and M. Venkatrao, On non-preemptive scheduling of recurring tasks using inserted idle time, Information and Computation Journal 117(1) (1995).

[10] R. Jansen and R. Laan, The stack resource protocol based on real-time transactions, IEE Proceedings-Software 146(2) (1999) 112–119.

[11] K. Jeffay, D. Stanat and C. Martel, On non-preemptive scheduling of periodic and sporadic tasks, in: *Proceedings of the Twelfth IEEE Real-Time Systems Symposium*, San Antonio (1991) pp. 129–139.

[12] 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).

[13] M. Kalia, D. Bansal and R. Shorey, MAC scheduling and SAR policies for Bluetooth: A master driven TDD pico-cellular wireless system, in: *Proceedings of the Sixth International Workshop on Mobile Multimedia Communications*, San Diego, California (1999) pp. 384–388.

[14] K. Kim and M. Naghibdadeh, Prevention of task overruns in real-time non-preemptive multiprogramming systems, in: *Proceedings of Performance* (1980) pp. 267–276.

[15] J. Nielsen, IP routing performance in bluetooth scatternets: A simulation study, Master's thesis, Department of Computer Systems (DoCS), Uppsala University, Uppsala (2000).

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

[17] C. Partridge, *Gigabit Networking*, 2nd edn, Addison-Wesley, (1993).

[18] 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).

[19] C.P.S. Shenker and R. Guerin, Specification of guaranteed quality of service, *RFC 2212, IETF* (1997).

[20] Q. Zheng and K. Shin, On the ability of establishing real-time channels in point-to-point packet-switched networks, IEEE Transactions on Communications 42(2/3/4), 1096–1105 (1994).

Rachid Ait Yaiz (1974) received his BS in Electrical Engineering from the Technische Hogeschool Arnhem, the Netherlands, in 1996 and his MSc in Electrical Engineering from the University of Twente, the Netherlands, in 1999. He received his Ph.D. in Telecommunications from the same university in 2004. Currently, he works for TNO Telecom. 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.
E-mail: r.aityaiz@telecom.tno.nl

Geert Heijenk (1965) received his MSc in Computer Science from University of Twente, the Netherlands, in 1988. He worked as a research staff member at the same university and received his Ph.D. in Telecommunications in 1995. He has also held a part-time position as researcher at KPN research, the Netherlands, from 1989 until 1991. From 1995 until 2003, he was with Ericsson EuroLab Netherlands, first as a senior strategic engineer, and since 1999 as a research department manager. From 1998 until 2003 he was also a part-time senior researcher at the University of Twente. Currently, he is a full-time associate professor at the same university. His research interests include mobile and wireless networks, resource management, and quality of service.
E-mail: geert.heijenk@utwente.nl