Chapter 17

Adaptive Cycle-Controlled E-Limited Polling in Bluetooth Piconets

Jelena Mišić and Vojislav B. Mišić

University of Manitoba, Winnipeg, Manitoba, Canada

17.1 Introduction

Bluetooth is an emerging standard for Wireless Personal Area Networks (WPANs) [15]. Bluetooth devices are organized in piconets: small, centralized network with $\mu \leq 8$ active nodes or devices [2]. One of the nodes acts as the master, while the others are slaves, and all communications in the piconet take place under master's control. All slaves listen to downlink transmissions from the master. The slave may reply with an uplink transmission if and only if addressed explicitly by the master, and only immediately after being addressed by the master.

Thus, the performance of data traffic in the Bluetooth piconet is critically dependent on the manner in which the master visits or polls its slaves — the polling or scheduling scheme. The

current Bluetooth specification does not require or prescribe any specific polling scheme [2], and a number of such schemes have been proposed [5]. However, most of these do not any provide support regarding any predefined delay bounds; only recently a scheme has been proposed that can support such bounds, albeit at the expense of efficiency and/or fairness [6]. This observation has motivated us to consider the lightweight scheduling scheme described here.

In the new scheme, each slave is allocated a number of consecutive time slots depending on its data traffic. This number is dynamically determined in each piconet cycle. Slaves which do not use all the allocated slots in one cycle are given less time in the next cycle, while slaves which have not exchanged all packets during the allocated time are given more time in the next cycle. In this manner, the number of wasted (POLL and NULL) packets is minimized, and the scheme is thus able to optimize the efficiency of data transfers, esp. in piconets with highly asymmetric traffic. The allocation of consecutive time slots tends to preserve the integrity of baseband packet bursts and thus leads to reduction of delays for IP packets in the piconet. Furthermore, each slave will be visited at least once during the cycle, which guarantees fairness by making sure that no slave(s) can monopolize the piconet at the expense of others. Finally, the actual algorithm is computationally simple, and the bulk of the calculations are performed only once per piconet cycle.

In cases where the piconet contains slaves with synchronous traffic, the scheme may simply be modified so as to control the duration of the piconet cycle. In this manner, the scheme is able to support piconet with both synchronous and asynchronous traffic. On account of this modification, the scheme will be referred to as the Adaptive Cycle-controlled E-limited polling, or ACE.

We start our discussion with a detailed description of the ACE scheme and related algorithms. In Section 17.3, we outline the queueing model for the master-slave pair of queues under constant maximum bandwidth allocation within the piconet cycle. Due to space limitations, the complete probability distributions for the access and end-to-end packet delays are not derived; they can be found in [10]. We compare the performance of the new scheme to that of other polling schemes and discuss possible modifications to improve its performance. Finally, we summarize our findings and give some concluding remarks.

17.2 An overview of the ACE scheme

The ACE scheme is based on the well-known E-limited scheme [14]. In this scheme, the master stays with a slave for a fixed number M of frames ($M > 1$) or until there are no more packets to exchange, whichever comes first. Packets that arrive during the visit are allowed to enter the uplink queue at the slave and may be serviced – provided the limit of M frames is not exceeded [14]. Boundary values of $M = 1$ and ∞ correspond to the well known 1-limited scheme, also known as (Pure) Round Robin, and exhaustive service [5, 14].

We note that the Bluetooth piconet does indeed resemble a simple polling system in which multiple input queues are serviced by a single server – a well known problem in queueing theory [13]. In such cases, exhaustive service has been shown to perform better than either 1-limited or E-limited service [7].

However, communications in Bluetooth are bidirectional by default, the master polls the slaves using regular packets (possibly empty), and all slave-slave communications have to be routed through the master. As a consequence, existing analytical results on polling schemes do not hold in Bluetooth piconets. Indeed, earlier research has shown that the 1-limited scheme achieves good results at high traffic loads, but tends to waste a lot of bandwidth under low and medium loads; the exhaustive service scheme achieves good results at low to medium loads, but cannot guarantee fairness at high loads, esp. if the traffic loads of individual slaves are not uniform [5]. In such cases, the E-limited scheme is able to achieve good results, which has been confirmed through both queueing theoretic analysis and discrete event simulations [9].

Now, the basic concept of E-limited polling can be extended to provide dynamic bandwidth allocation according to the behavior in previous piconet cycle. To that effect, the ACE scheme (in its simpler form) allocates one of the two possible values of M to each slave. Slaves that have finished their packet exchange, which is detected through the presence of a POLL-NULL sequence as is customary in Bluetooth [4], will get less bandwidth (M_L in the next cycle. Slaves that have undelivered packets for exchange, will get more bandwidth (M_H. This scheme is described with the following pseudocode.

```
procedure ACE polling
do forever
    for all slaves
        poll slave i
        if both uplink and downlink queues are empty
        then set M(i) = M_L
                break (i.e., move on to next slave)
        else if M packets are exchanged
        then set M(i) = M_H
                break (i.e., move on to next slave)
        end if
        loop (i.e., poll the same slave again)
    end for
end do
```

Piconet cycle control

In cases when the piconet contains both synchronous and asynchronous slaves, the master can modify its bandwidth allocation algorithm so as to control the duration of the piconet cycle. In this case, the available piconet cycle is divided into two parts. The first part is allocated directly to the slaves: those that did not use all of the allocated time in the previous cycle will get a smaller part (analogous to M_L in the previous case), whilst those that did use all the allocated time and still have packets to exchange will get a larger part (analogous to the M_H) in the previous case). The other part is known as the free `pool`, and slaves can use it to accommodate extra traffic they might have in the current cycle. (The percentage of total cycle duration set aside for the `pool` is an adjustable parameter.)

```
procedure ACE polling with cycle limit
do forever
    for all slaves
        poll slave i
        if both uplink and downlink queues are empty
          or slave has used allocation plus current pool
        then record used time t(i)
                return unused time (if any) to the pool
                break (i.e., move on to next slave)
        else loop (i.e., poll the same slave again)
        end if
    end for (i.e., move on to next slave)
    calculate pool, basic per-slave allocation
    for all slaves
        if the slave used more time than allocated
        then allocate twice basic per-slave time
        else allocate basic per-slave time
    end for
    restart the polling cycle
end do
```

In case some slaves have a predefined bandwidth requirement, e.g., on account of synchronous traffic or negotiated QoS, they can be excluded from both dynamic bandwidth allocation and re-ordering, and they can be assigned a fixed position within the piconet cycle – say, at the beginning of each cycle.

The role of the reference slave and slave reordering

The fairness of ACE polling depends on the choice of the reference slave – the slave from which the piconet polling cycle starts. Any of the slaves could be chosen to be the reference slave, and

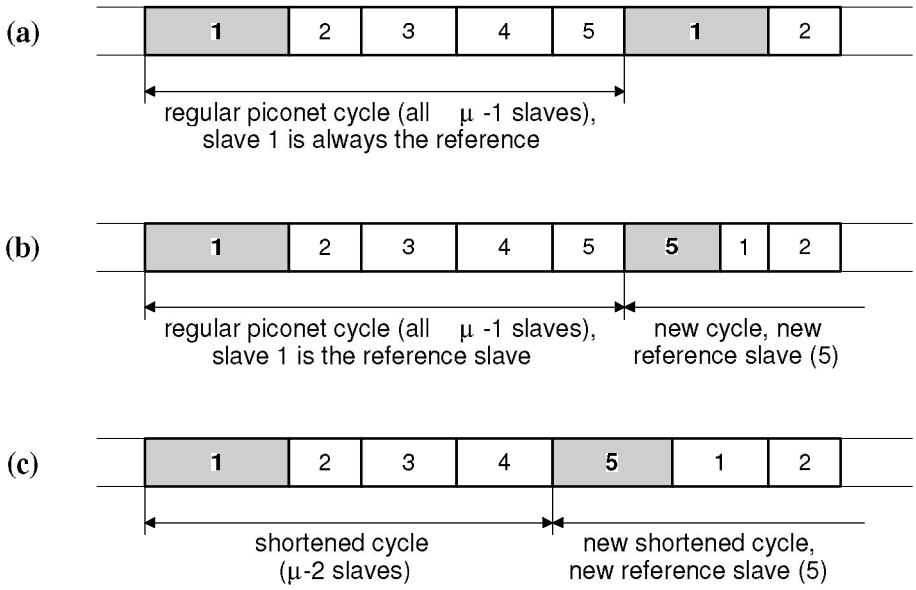


Figure 17.1: Pertaining to the choice of reference slave. (From J. Mišić, V. B. Mišić, and E. W. S. Ko, "Fixed cycles and adaptive bandwidth allocation can coexist in Bluetooth," *Canad. J. Elect. Comput. Eng.* **29**(1/2):135–147), © 2004 CJECE.)

the master could simply poll the slaves in fixed order, as shown in Fig. 117.2 (a). However, this would destroy the fairness of bandwidth allocation to individual slaves. Namely, the first slave in a given cycle may use its direct allocation plus the entire `pool`. After the first slave finishes its transfer, the remaining unused time (if any) is returned to the `pool`. Each subsequent slave can use its direct allocation, plus the current value of the `pool`. In this manner, even though the dynamic allocation is made on the basis of the traffic in the *previous* piconet cycle, slaves with higher values of traffic in the *current* cycle can still get extra time.

However, if the slaves are polled in fixed order, the first slave would always have the entire `pool` at its disposal, whereas the last one will seldom have more than the time obtained through direct allocation. Therefore, the scheme in this form is inherently unfair. Moreover, the worst case cycle time and, consequently, the maximum polling interval will suffer as well.

Several possibilities exist to restore the fairness. In the deterministic approach, the ACE scheme could be modified by rotating the role of the reference slave amongs all the slaves. For example, the role of the reference slave could be assigned to the last slave from the previous cycle, as shown in Fig. 17.2 (b). In this case, the last slave will get *two* chances to exchange packets with the master,

but this role is taken by each slave in turn. The last slave in one cycle will become the reference slave in the next one, so it will get slightly more time to exchange packets with the master. This slave will probably get only a short time allocation at first, and much more time immediately after that, so that any leftover traffic will be taken care of immediately. In the worst case, when there is no data to or from this slave, two frames will be wasted instead of one.

Note that with the deterministic approach the slave which is polled last in the current cycle becomes the slave which is polled first in the coming cycle. Therefore the polling cycle may be shortened to $(\mu - 2)$ slaves, as shown in Fig. 17.2 (c). In this case, the **`new_cycle()`** procedure allocates bandwidth to the current set of $\mu - 2$ slaves, with the `C` parameter suitably modified; the `queues` value for the missing slave could be taken from the cycle before the last. Each slave will thus get equal attention in a super-cycle of $\mu - 1$ shorter cycles, or $\mu - 2$ normal piconet cycles. Therefore under equally loaded slaves average number of slots per cycle devoted for polling one slave is equal among the slaves.

In the probabilistic approach, the fairness is achieved by choosing the next slave in the cycle with the equal probability. For example, first slave in the cycle will be chosen with probability $1/(\mu - 2)$, second slave with probability $1/(\mu - 2)$ and so on. However, this approach is computationally more complex than the deterministic one.

The third solution – which we have chosen to implement – is a variation of a well known concept from queueing theory. Namely, it has been shown that single-server, multiple-queue polling systems achieve best performance if the server always services the queue with the highest number of packets; this policy is known as Stochastically-Largest Queue, or SLQ [8]. Again, this policy cannot be followed in Bluetooth because the master cannot know the status of *all* the slaves' queues at any given time. The best that can be done is to rearrange the polling order of the slaves in each cycle according to the length of the corresponding downlink queue; these queues are maintained by the piconet master and thus their lengths are known to the master at any given time. We will refer to this policy as Longest Downlink Queue First (LDQF); as will be seen, it consistently provides a small but noticeable improvements in piconet performance. We note that this modification is particularly well suited to the situation where the piconet master acts as the access point to another network (e.g., Ethernet, as per BNEP profile [1]). In such cases, the downlink traffic

may be expected to exceed the uplink one, possibly by an order of magnitude or more, and overall performance will be mainly determined by the performance of downlink traffic.

With this modification, the ACE algorithm may be described with the following pseudocode.

```
procedure ACE polling with cycle limit and LDQF
do forever
    for all slaves according to the current sequence
        poll next slave in the current sequence
        if both uplink and downlink queues are empty
          or slave has used allocation plus current pool
        then record used time t(i)
              return unused time (if any) to the pool
              break (i.e., move on to next slave)
        else loop (i.e., poll the same slave again)
        end if
    end for
    calculate pool, basic per-slave allocation
    for all slaves
        if the slave used more time than allocated
        then allocate twice basic per-slave time
        else allocate basic per-slave time
    end for
    reorder slaves per LDQF policy
end do
```

17.3 The queueing model of the ACE scheme

The performance of the ACE scheme can be assessed using a queueing theoretic approach. Consider an isolated piconet with the master and $\mu - 1$ active ACL slaves, as shown in Fig. 17.2. The

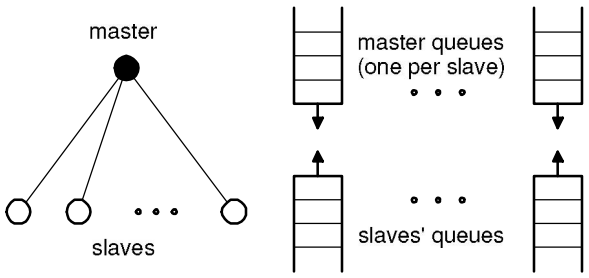


Figure 17.2: A single piconet and its queueing model.

operation of the piconet may be described with a queueing model in which each slave maintains an uplink queue, while the master maintains a number of downlink queues, one per each active slave. All queues are assumed to have buffers of infinite size; as data traffic will be generated by the applications running on the devices themselves, packet arrival rates will be low and buffer overflows will be sufficiently rare.

Each slave generates packets in bursts, which correspond to the application packets [1]. Application packet arrivals to the uplink queue of slave i follow a Poisson distribution, which has been shown to be a satisfactory approximation for the traffic of many Internet applications [11]. If the application packets themselves arrive in bursts, the probability-generating function (PGF) of the burst size of application packets should be integrated into the corresponding PGF for the baseband burst size.

The length of the burst of baseband packets is geometrically distributed, but other distributions can be easily accommodated in our model, provided that the corresponding probability distribution (i.e., the PGF) is known. Assuming that all slaves use the same segmentation/reassembly mechanism (which is reasonable, as this mechanism is commonly implemented in firmware), the burst length distribution will be the same for all slaves.

We assume that the traffic goes from slaves to other slaves only, which simplifies the calculations without undue loss of generality. (Any traffic generated or received by the master can be easily modeled by increasing the packet arrival rates in the downlink queues without any other changes of the analytical model.) All packets within the given burst will have the same destination node, and the distribution of destinations is assumed to be uniform. Of course, the case where the destination probabilities for individual slaves differ can be dealt with by simply recalculating the

resulting downlink arrival rates may be calculated from the matrix of probabilities for slave-to-slave communication, without changing the model itself.

Data packets can last for one, three, or five slots of the Bluetooth clock (with length $T = 625\mu s$) [2], with probabilities of p_1, p_3 and $p_5 = 1 - p_1 - p_3$, respectively. For simplicity, we will assume that $p_1 = p_2 = p_3$ and, consequently, $\overline{L} = G'_p(1) = 3$, although other distributions may be accommodated without difficulty. The sequence of the one downlink and following uplink packet transmission will be denoted as a frame.

Using the theory of queues with vacations, we are able to derive the probability distributions for the following variables:

- Piconet cycle time, which is the time for the master to visit all the slaves in the piconet exactly once (note that the visit may last for several frames).
- Slave channel service time, which is the time the time from the moment when master polls the slave for the first time, until either an empty frame has been encountered, or the maximum number of frames have been exchanged. (Note that this maximum number of frames is variable.)
- Vacation time from the standpoint of one slave. When the concept of vacation [14] is applied to Bluetooth networks, the piconet master acts as the server, while the client corresponds to a slave or, rather, the pair of uplink queue at the slave and the corresponding downlink queue at the master. From the viewpoint of a given client, the vacation time is the time during which the server is busy servicing other client queues and, thus, unavailable to service that particular client.
- Frame length time, which is not equal to the packet length because of the presence of empty (POLL and NULL) packets.

Using these variables, we can derive the probability distributions for the packet access delay – the time the data packet has to wait in the uplink queue of a slave before being serviced by the master – and the packet end-to-end delay – the time that elapses between the data packet enters the uplink queue at the source slave and the time it is received by the destination slave. The derivations

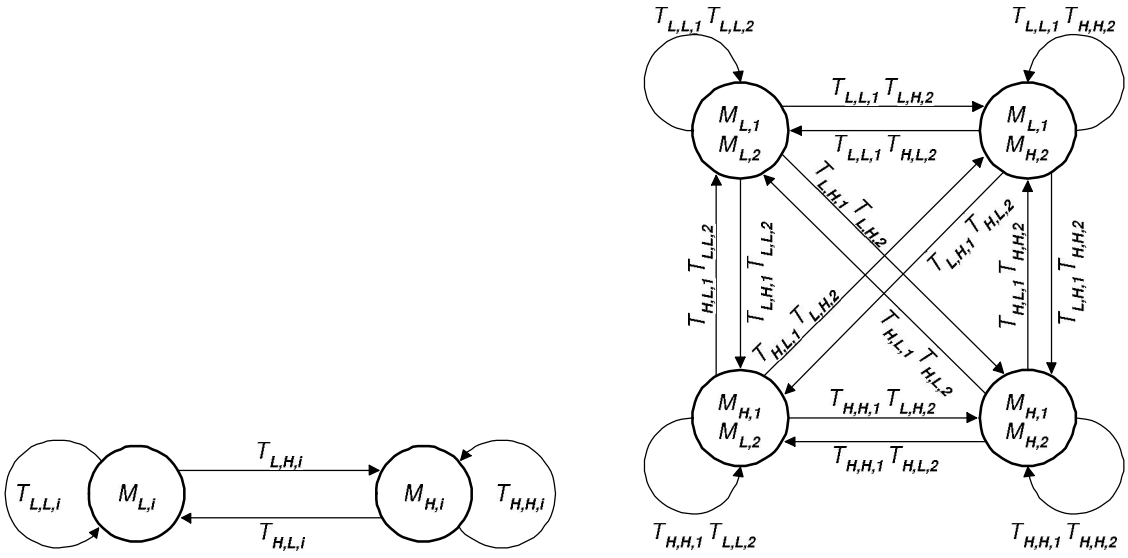
are rather involved and thus are omitted here; the interested reader can consult an earlier paper [9] for details.

A similar approach, based on the so-called imbedded Markov points that correspond to the end of individual piconet cycles. Any given slave i can be in the state of low or high bandwidth allocation during any given piconet cycle, depending on their usage of allocated bandwidth during the previous piconet cycle:

- The slaves that did not use all of the allocated slots in the previous cycle will be given low bandwidth of $M_{L,i}$ frames.
- The slaves that have used all of the allocated slots will be given high bandwidth of $M_{H,i}$ frames.

For both states, we can determine the joint queue length distributions for the uplink and downlink queues corresponding to the slave i, $Q_i(z,w)$ and $\Pi_{i,k}(z,w)$, $k = 1 \,..\, M_{L,i}$ for low state and $k = 1 \,..\, M_{H,i}$ for the high state. (The index i has been previously omitted for simplicity.) The transition probabilities between the states are determined according to the corresponding value of `queues[i]`:

$$
\begin{aligned}
T_{L,L,i} &= \text{Prob}(\texttt{queues[i]} = 0 \text{ or } 1) | \text{low state} \\
&= Q_i(1,0) + Q_i(0,1) \\
&\quad + \sum_{m=1}^{M_{L,i}} (\Pi_{i,m}(1,0) + \Pi_{i,m}(0,1)) && \text{(1-a)} \\
T_{L,H,i} &= \text{Prob}(\texttt{queues[i]} = 2) | \text{low state} \\
&= 1 - T_{L,L,i} && \text{(1-b)} \\
T_{H,L,i} &= \text{Prob}(\texttt{queues[i]} = 0 \text{ or } 1) | \text{high state} \\
&= Q_i(1,0) + Q_i(0,1) \\
&\quad + \sum_{m=1}^{M_{H,i}} (\Pi_{i,m}(1,0) + \Pi_{i,m}(0,1)) && \text{(1-c)} \\
T_{H,H,i} &= \text{Prob}(\texttt{queues[i]} = 2) | \text{high state} \\
&= 1 - T_{H,L,i} && \text{(1-d)}
\end{aligned}
$$



(a) Markov chain for one slave.

(b) Markov chain for two slaves.

Figure 17.3: Markov chains that describe adaptive bandwidth allocation. (From J. Mišić, V. B. Mišić, and E. W. S. Ko, "Fixed cycles and adaptive bandwidth allocation can coexist in Bluetooth," *Canad. J. Elect. Comput. Eng.* **29**(1/2):135–147), © 2004 CJECE.)

In the presence of k slaves in the piconet, the number of states of the Markov chain is 2^k. One state of the chain is described by the k-tuple of states corresponding to each slave. Components of the transition probabilities between the states are given by (1). For every state of the chain the slot/frame allocations have to be determined. The Markov chains for the piconets with one and two slaves are shown in Figs. 17.3(a) and 17.3(b), respectively.

We note that actual bandwidth allocations for low- and high bandwidth states for one slave are not equal across the states of the Markov chain. For example, let us assume that we have two slaves, and the cycle is defined as 36 slots (which corresponds to 6 frames, or 12 packets with the mean length $\overline{L} = 3$). Also note that the reordering equally distributes the remaining bandwidth (i.e., `pool`) among the slaves. Now, $M_{L,2}$ is 12 slots (for both `queues[2]` = 0 or 1), and $M_{H,1} = 24$. However, for the state $M_{H,1}, M_{H,2}$, both slaves will get 18 slots.

When the actual bandwidth allocations are found for each slave in each state of the Markov chain, the probability generating functions for each slave, in each state, have to be found, and the

transition probabilities for the Markov chain have to be calculated. From these, we can calculate the queue length probability distribution, and the resulting distributions for the access and end-to-end packet delays.

Again, the calculations are rather involved and are thus omitted; full derivations can be found in [10].

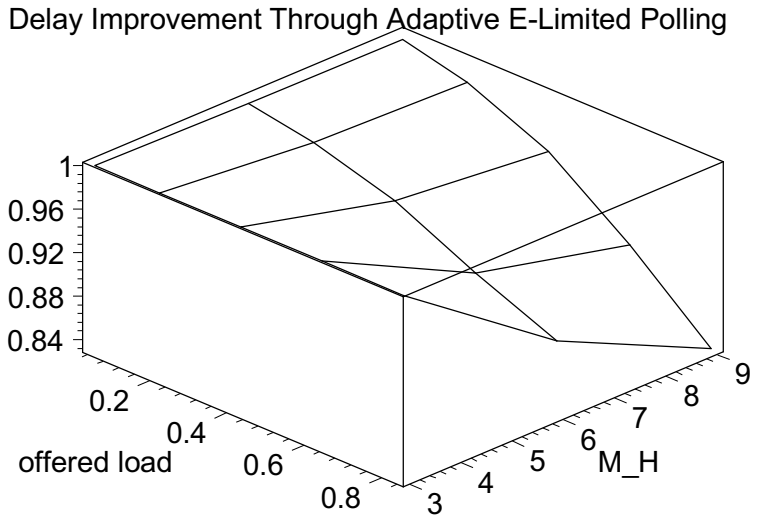
17.4 Performance of the new scheme

In order to evaluate the performance of the ACE scheme, we have built a simulator of the Bluetooth piconet, using the Artifex [12] object-oriented Petri Net simulation engine. The simulator operates at the MAC level, consisting of the master and up to seven slaves. The simulator operates under the assumptions listed in the beginning of Section 17.3; in order to avoid transition effects, the simulation results were measured after an initial warm-up period. All delays are expressed in time slots of the Bluetooth device clock, $T = 625\mu s$.

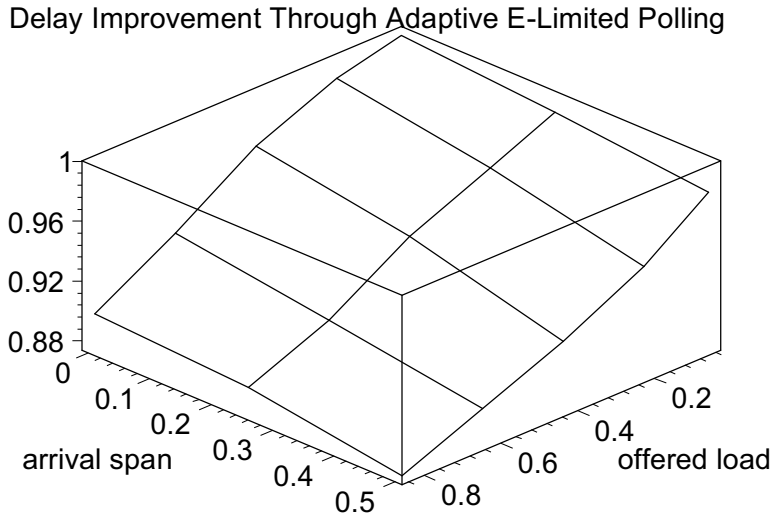
While the detailed results can be found in [9], we will only repeat the main observations here. First, under traditional E-limited polling, which is used as the reference against which the improvements offered by the ACE scheme can be measured, end-to-end packet delays generally increase with both offered load and traffic burstiness (expressed through mean burst size $\overline{B}$. At high volumes of very bursty traffic, the delays tend to increase to values well over $400T$ (which corresponds to about $0.25s$), which signal that the uplink and downlink queues become longer and longer. As Bluetooth devices are mobile, and thus are expected to run for long time on battery power, the size of device queues will tend to be small, and buffer overflows may occur at such ranges of load and burstiness. Therefore, Bluetooth devices should probably not be operated under such high loads.

Another observation is that the cycle time is rather independent of mean burst size – it mainly depends on the offered load. This is not altogether unexpected, since longer bursts at constant load mean that the bursts will come in longer intervals. However, the variance of piconet cycle time will increase, because the bursts will be handled in fewer piconet cycles.

The improvements obtained through the use of adaptive E-limited polling may be seen in



(a) Performance under varying upper limit M_H.



(b) Performance under asymmetric traffic.

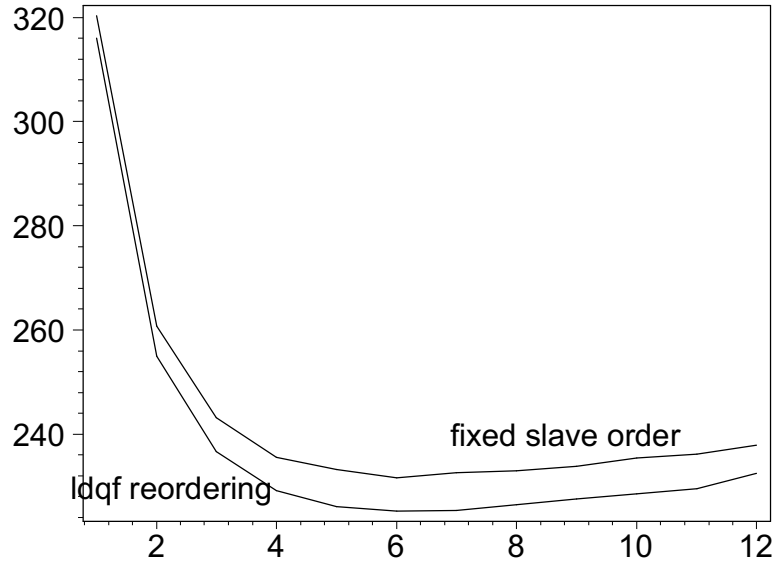
Figure 17.4: Delay improvement due to adaptability, as compared to the simple E-limited polling with $M = 3$.

Fig. 17.4. The upper diagram shows the ratio of end-to-end packet delay obtained under adaptive E-limited polling (with lower limit $M_L = 3$) to the similar delay obtained under the original E-limited polling with $M = 3$ (this essentially corresponds to ACE scheme in which $M_L = M_H = 3$). As the upper limit increases, the delays decrease. This decrease is more pronounced for higher offered loads, where the adaptivity of the ACE scheme improves performance significantly. (Of course, the absolute values of end-to-end packet delay is still high.) The lower diagram shows the delay ratio under asymmetric loads, where $M_L = 3$ and $M_H = 6$. Again, the ACE scheme performs better at higher loads, and it performs better as the asymmetry between the traffic of individual slaves becomes more pronounced.

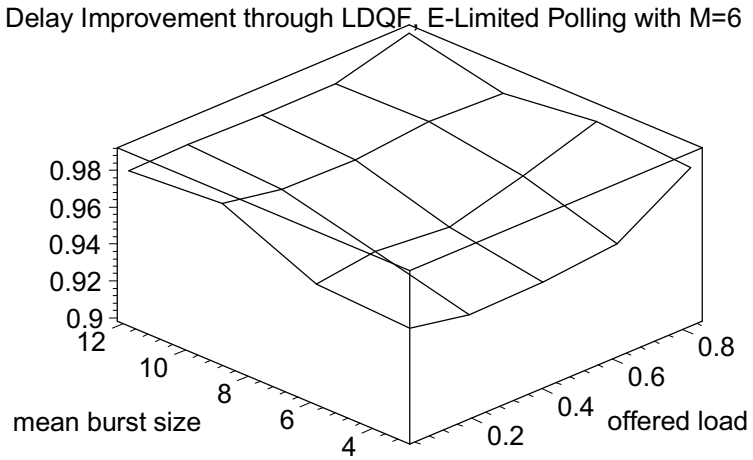
The improvement in end-to-end packet delay due to slave reordering in each cycle using the LDQF policy is shown in Fig. 17.5. The first diagram shows the delay as a function of the maximum number of packets transferred in each visit M. (Note that the value of $M = 1$ corresponds to 1-limited polling.) As can be seen, LDQF reordering reduces the delay, but also makes the minimum of the curve (which, in the case of fixed slave order, corresponds roughly to the value of $M = \overline{B}$) slightly wider. Therefore, the use of LDQF reordering not only improves the delay, but it also makes the minimum delay less dependent on the mean burst size.

Finally, Fig. 17.6 shows the effects of introducing the fixed cycle control. For simplicity, we have plotted the distribution of cycle durations in a sample run for both the original ACE scheme and its fixed cycle variant. The piconet had seven active ACL slaves, and the cycle time limit has been set to $C = 80T$. As can be seen, the fixed cycle control is much more effective in controlling the piconet cycle: the cycle time exhibits a noticeable peak close to the preset cycle limit, and the maximum cycle value (which may exceed the limit due to the presence of the free pool) is about $130T$. On the contrary, under the original adaptive E-limited scheme the cycle times are spread over a much wider range (as there may be cycles in which most slaves get maximum bandwidth allocation), and there is no discernible peak. Note that the smaller peak at about $14T$ corresponds to the cycles in which no slave has any traffic, hence each visit of the master takes only two time slots T.

At the same time, fixed cycle control does lead to some deterioration of delays, esp. at higher loads. One should bear in mind, though, that the main goal of this scheme is to provide guaranteed

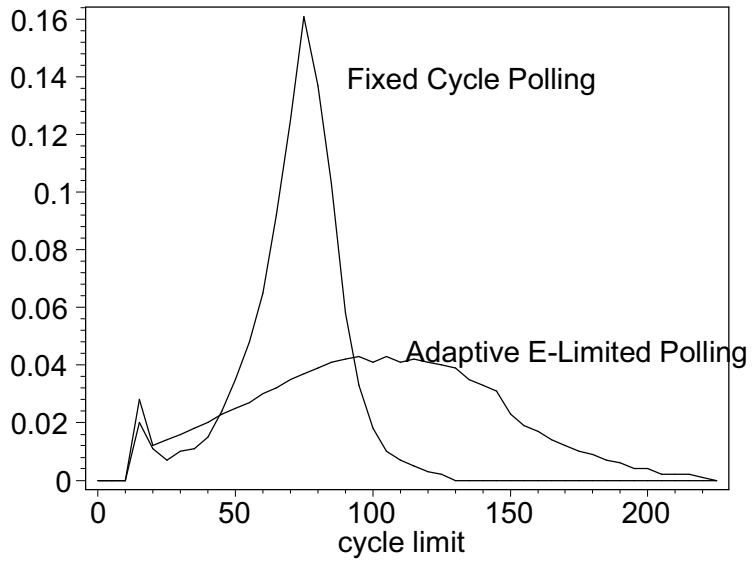


(a) End-to-end packet delays with and without LDQF reordering as functions of M, for fixed mean burst size $\overline{B} = 6$.

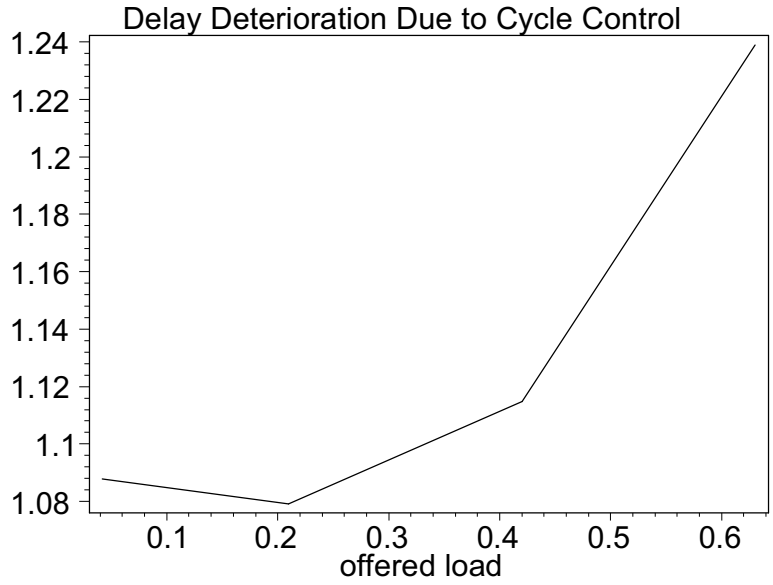


(b) Ratio of end-to-end packet delays with and without LDQF slave reordering.

Figure 17.5: Pertaining to LDQF slave reordering.



(a) Cycle time distribution.



(b) Ratio of end-to-end packet delays: ACE with soft cycle control vs. plain ACE.

Figure 17.6: Performance of fixed cycle variant compared to adaptive E-limited polling.

polling interval for slaves with synchronous traffic, rather than optimum performance under high volume of asynchronous traffic. For values of offered load that do not exceed $\rho = 0.4$, the deterioration of delay is below 12%, which appears to be an acceptable price to pay for the improvement in cycle control.

17.5 Summary and possible enhancements

In this paper we have described a lightweight adaptive polling scheme with soft cycle control which is suitable for piconets with asymmetric traffic. We have also developed a complete queueing theoretic model of the piconet operation under dynamic bandwidth allocation within a limited piconet cycle. The new scheme is shown to perform well under a wide range of traffic loads in the piconet, whilst being much simpler than other comparable schemes.

The ACE scheme can be extended to support slaves with predefined QoS parameters such as bandwidth and maximum polling interval. Such requirements may stem from the presence of synchronous, CBR traffic, but also from the more common asynchronous traffic. (It should be noted that the negotiated maximum polling interval is the only mechanism to support QoS which is provided by the current Bluetooth specification [3].) Slaves with QoS requirements can be excluded from the ACE adaptive bandwidth allocation mechanism in two ways: they can be polled using a fixed number of frames, and they can be excluded from slave reordering in each cycle. As the piconet cycle is not fixed, slaves with CBR traffic may have less traffic in some cycles, which can easily be dealt with by ending the exchange when a POLL/NULL frame is encountered, as is common in Bluetooth.

Further extension of the ACE scheme could introduce admission control, in which slaves can be admited in the piconet, or rejected, depending on the ability of the piconet to support the QoS requirements. These requirements can be imposed by the slaves requesting admission, but also by the slaves that are already admitted in the piconet. Some initial results in that direction, but using plain E-limited polling only, are reported in [9].

References

[1] Bluetooth SIG. Bluetooth Network Encapsulation Protocol (BNEP) Specification. Technical report, Revision 0.95a, June 2001.

[2] Bluetooth SIG. *Specification of the Bluetooth System*. Version 1.1, February 2001.

[3] Bluetooth SIG. *Specification of the Bluetooth System – Architecture & Terminology Overview*, volume 1. Version 1.2, November 2003.

[4] Bluetooth SIG. *Specification of the Bluetooth System – Core System Package [Controller volume]*, volume 2. Version 1.2, November 2003.

[5] Antonio Capone, Rohit Kapoor, and Mario Gerla. Efficient polling schemes for Bluetooth picocells. In *Proc. IEEE Int. Conf. on Communications ICC 2001*, volume 7, pages 1990–1994, Helsinki, Finland, June 2001.

[6] Jean-Baptiste Lapeyrie and Thierry Turletti. FPQ: a fair and efficient polling algorithm with QoS support for Bluetooth piconet. In *Proceedings Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2003*, volume 2, pages 1322–1332, New York, NY, April 2003.

[7] Hanoch Levy, Moshe Sidi, and Onno J. Boxma. Dominance relations in polling systems. *Queueing Systems Theory and Applications*, 6(2):155–171, 1990.

[8] Zhen Liu, Philippe Nain, and Don Towsley. On optimal polling policies. *Queueing Systems Theory and Applications*, 11(1–2):59–83, 1992.

[9] Jelena Mišić, Ka Lok Chan, and Vojislav B. Mišić. Admission control in Bluetooth piconets. *IEEE Transactions on Vehicular Technology*, 53(3):890–911, May 2004.

[10] Jelena Mišić, Vojislav B. Mišić, and Eric Wai Sun Ko. Fixed cycles and adaptive bandwidth allocation can coexist in Bluetooth. *Canadian Journal of Electrical and Computer Engineering*, 29(1–2):135–147, January/April 2004.

[11] V. Paxson and Sally Floyd. Wide area traffic: the failure of Poisson modeling. *ACM/IEEE Transactions on Networking*, 3(3):226–244, June 1995.

[12] RSoft Design, Inc. *Artifex v.4.4.2*. San Jose, CA, 2003.

[13] Hideaki Takagi. *Analysis of Polling Systems*. The MIT Press, Cambridge, MA, 1986.

[14] Hideaki Takagi. *Queueing Analysis*, volume 1: Vacation and Priority Systems. North-Holland, Amsterdam, The Netherlands, 1991.

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