Efficient Polling Schemes for Bluetooth Picocells Revisited

Ka Lok Chan, Vojislav B. Mišić, and Jelena Mišić

Abstract—**The performance of asynchronous data traffic in Bluetooth piconets is very much dependent on the choice of the polling scheme. In this paper we consider the problem of designing efficient and fair, yet simple to implement, polling schemes that would achieve best performance (i.e., the lowest end-to-end packet delay) under TCP traffic. A number of such schemes are described and their performance is compared against some benchmark schemes derived from the known results for polling systems.**

I. INTRODUCTION

Bluetooth is an emerging standard for Wireless Personal Area Networks (WPANs) [1]. Originally, Bluetooth was envisaged as a wireless cable replacement technique. However, the number of possible uses of Bluetooth have increased to include different networking tasks between computers and computer-controlled devices such as PDAs, mobile phones, smart peripherals, and others.

Bluetooth devices must form networks before the actual communication can start [2]. The simplest form is a piconet: a small, centralized network with up to eight active nodes or devices. One of the nodes is designated as the master, while the others are slaves. All communications in the piconet take place under master's control. Each piconet hops through the available RF frequencies in the ISM band around 2.4GHz [2] in a pseudo-random manner. The hopping sequence, which is determined from the Bluetooth device address of the piconet master, is known as the channel [2]. Each channel is divided into time slots of $T = 625\mu s$, which are synchronized to the clock of the piconet master.

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. Data is transmitted in packets, which take one, three, or five slots. When there is no data to send, single-slot packets with zero payload are sent – POLL packets in the downlink, and NULL packets in the uplink direction [2]. As the process of polling the slaves is actually embedded in the data transmission mechanism, we will use the term 'polling' for every downlink transmission from the master to a slave.

By default, all master transmissions start in even-numbered slots, whilst all slave transmissions start in odd-numbered slots. (A downlink packet and the subsequent uplink packet are sometimes referred to as a frame.) Therefore, the master and the addressed slave use the same communication channel, albeit not at the same time. This communication mechanism, known as Time Division Duplex, or TDD for short, is schematically shown in Fig. 1.

K. L. Chan is with The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China. V. B. Mišić, and J. Mišić are with The University of Manitoba, Winnipeg, Manitoba, Canada.

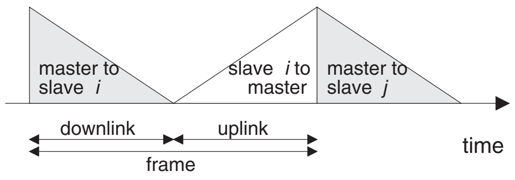


Fig. 1. The TDD communication mechanism in Bluetooth.

As can be seen, the performance of data traffic in the Bluetooth piconet is critically dependent on the manner and sequence 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]. This may not seem to be too big a problem, since optimal polling schemes for a number of similar, single-server multiple-input queueing system are well known [3], [4]. However, communication mechanisms used in Bluetooth are rather specific: namely, all communications are bidirectional, the master polls the slaves using regular packets (possibly without data payload), all slave-slave communications have to be routed through the master, and the master does not know the status of queues at the slaves. It should come as no surprise, then, that a number of polling schemes have been proposed and analyzed [5], [6], [7], [8]. Many of the proposed schemes are simply variations of the well-known limited and exhaustive service scheduling [9], but several improved adaptive schemes have been described as well [7], [10].

In this paper, we will classify and analyze the performance of a number of existing polling schemes for Bluetooth. We use a set of ideal polling schemes—ideal because they cannot be implemented in real Bluetooth networks—as the benchmarks against which the performance of other schemes is to be compared. In order to compare the polling schemes under realistic conditions, our simulation setup tries to mimic as closely as possible the environment in which Bluetooth technology is used to encapsulate data packets from devices and applications that use other protocols such as Ethernet or 802.11 [11]. In such setup, we explicitly model TCP traffic encapsulated in Bluetooth data packets. As will be seen, the performance in this case critically depends on certain TCP parameters. However, the values of these parameters may be chosen in such a way as to minimize the end-to-end packet delays.

The paper is organized as follows: Sec. II introduces different type of pollers for intra-piconet scheduling, both realistic and ideal ones, while Sec. III provides details on the transmission of

TCP traffic over Bluetooth and the traffic models we have used. Our simulation setup and the results of performance assessment for homogeneous and heterogeneous traffic are given in Secs. IV and V, respectively. Sec. VI concludes the paper.

II. Polling in the Bluetooth Piconet

The polling scheme is obviously the main determinant of transmission performance in Bluetooth piconets. As usual, the main performance indicator is the end-to-end packet delay, with lower delays being considered as better performance. But there are at least two other requirements to satisfy. First, the piconet master should try to maintain fairness among the slaves, so that all slaves in the piconet receive equal attention in some shorter or longer time frame. (Of course, their traffic load should be taken into account.) Second, Bluetooth devices are, by default, low power devices, and the polling scheme should be sufficiently simple in terms of computational and memory requirements.

The optimization of performance may be undertaken in different ways, which may be roughly classified along three more or less independent directions.

First, the polling scheme determines the number of frames exchanged during a single visit to the slave. This number may be set beforehand to a fixed or variable value, or it may be dynamically adjusted on the basis of traffic information.

Second, different slaves may receive different portions of the bandwidth; again, the allocation may be done beforehand, or it may be dynamically adapted to varying traffic conditions. The latter approach is probably preferable in Bluetooth piconets, which are ad hoc networks formed by mobile users, and the traffic may exhibit considerable variability. (In fact, due to users' mobility, even the topology of the piconet may change on short notice.) However, the fairness of polling may be more difficult to maintain under dynamic bandwidth allocation.

Finally, the sequence in which slaves are visited may be set beforehand, or it may change from one piconet cycle to another, depending on the traffic information. In either case, slaves that had no traffic in the previous cycle(s) may be skipped for one or more cycles, but the polling scheme must ensure that the fairness is maintained.

We note that a tradeoff between low delay and high has to be found as well. If we want to keep the delay within some limits, the master should poll each slave frequently, i.e., should not spend too much time with any slave. However, this increases the fraction of wasted polls (when no data is transmitted) to less active slaves, which may decrease the throughput for other slaves that might have data to transmit or receive.

A. Traditional polling schemes

The simplest polling schemes use a fixed ordering of the slaves and fixed bandwidth allocation per slave. The only variable parameter, then, is the duration of master's visit to each slave.

Under *1-Limited Service* polling, the master visits each slave for exactly one frame, and then moves on to the next slave. The sequence of slaves is fixed and does not change. Data packets are sent if there are any, otherwise empty packets (POLL or NULL) are sent. The scheme is sometimes referred to as (Pure) Round Robin [5] or simply limited service.

Under *Exhaustive Service* polling, the master stays with the slave as long as there are packets to exchange in either downlink or uplink direction. The absence of packets is detected by a POLL-NULL frame. The sequence of slaves is again fixed and does not change.

Under the *E-Limited Service* polling, the master stays with a slave until there are no more packets to exchange, or for a fixed number M of frames ($M > 1$), 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 [9]. The sequence of slaves is again fixed and does not change. In fact, 1-limited and exhaustive service polling may be considered as special cases of E-limited service, where the limit M equals 1 and ∞, respectively.

It has been shown that, in traditional polling systems, exhaustive service performs better than either 1-limited or E-limited service [3]. However, Bluetooth piconets are not traditional polling systems—they have bidirectional traffic—and these results do not hold [5], [7], [12].

Furthermore, even better results may be obtained through the so-called Stochastically Largest Queue (SLQ) policy. Under this policy, the server always services the queue with the highest number of packets [4]. Translated into the Bluetooth environment, this means that the master should always poll the slave for which the sum of lengths of uplink and downlink queues is the highest. However, this is not possible in Bluetooth, where there is no way for the master to know the current status of *all* of its slaves' uplink queues.

Nonetheless, SLQ and similar schemes may be useful as benchmarks – reference points against which the other polling schemes are measured. To that end, we have defined two variants of the SLQ policy.

In the scheme referred to as *Benchmark-Longest Queue First Limited Service* (BM LQF Limited), the master always polls the slave with the highest sum of uplink and downlink queues. Each visit lasts for one frame only, and the status of all slaves' queues is re-examined after each frame.

In the scheme referred to as the *Benchmark-Longest Queue First Exhaustive Service* (BM LQF Exhaustive), the master examines the slaves, and polls the slave with the longest sum of uplink and downlink queues. The master stays with this slave until both queues are exhausted, as detected through a POLL-NULL frame. Only then is the status of all slaves' queues re-examined.

In both of these schemes, in particular the second one, fairness may suffer, and it is not difficult to imagine the situation in which two slaves that talk to each other can virtually monopolize the piconet and starve all others. In order to have a fair benchmark polling scheme, we have defined the *Benchmark Round Robin* (BM RR) polling, in which the slaves are polled in fixed order just as in Pure Round Robin, except that the slave(s) that have no traffic in either direction will be polled only when the whole piconet is idle. In this manner, slot wastage is minimized, since POLL-NULL frames occur only when there is no data to send in the entire piconet.

B. Adaptive Polling Schemes

We stress again that none of the three BM schemes can be implemented in the real Bluetooth environment (fortunately, they can be simulated). However, other polling schemes have been proposed that try to improve performance through the use of the other two mechanisms mentioned above.

For example, the polling scheme in which the master will always poll the slave with the longest *downlink* queue may be implemented with ease. We will refer to this scheme as *Longest Downlink Queue First* (LDQF). This scheme makes sense in situations where the piconet master acts as the access point to another network (e.g., Ethernet, as per BNEP profile [11]). In such cases, the downlink traffic may be expected to exceed the uplink one, and the overall performance will be mainly determined by the performance of downlink traffic.

Under the *Exhaustive Pseudo-cyclic Master* (EPM) queue length scheme, proposed in [5], each slave is visited exactly once per cycle. At the beginning of each cycle, the slaves are reordered according to the decreasing length of downlink queues.

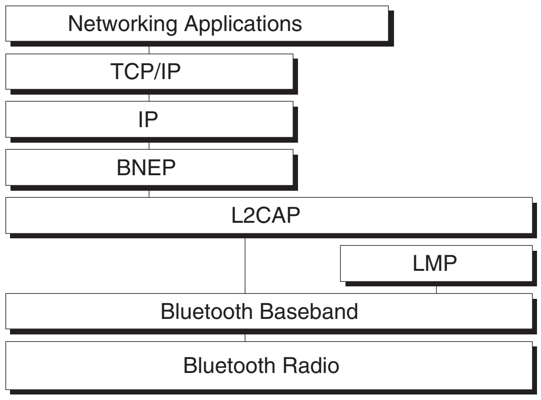
Note that the LDQF and EPM schemes correspond to the exhaustive and 1-limited variants of the BM LQF scheme, except that only the downlink queues are taken into consideration.

Another scheme proposed in [5] is *Limited and Weighted Round Robin* (LWRR), which achieves high efficiency by reducing the rate of visits to inactive slaves. Initially, each slave is assigned a weight equal to the so-called maximum priority, or MP. Each slave is polled in E-limited fashion with up to M frames. Whenever there is a data exchange between the slave and the master, the weight of the slave is increased to the value of MP. On the other hand, when a POLL-NULL sequence occurs, the weight for that particular slave is reduced by one. If the slave weight drops to one (which is the lowest value), the slave has to wait a maximum of $MP - 1$ cycles to be polled again.

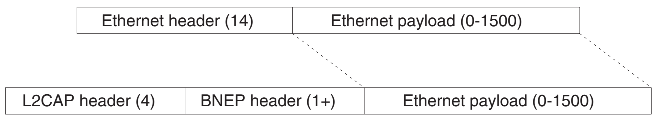
A similar idea is used in the *Fair Exhaustive Polling* (FEP) scheme [7], where a pool of active slaves is maintained by the master. Slaves are polled with one frame per visit, as in 1-limited service. When a POLL-NULL sequence occurs, that slave will be dropped from the pool, which means that it will not be polled for some time. The pool is reset when the last slave in it is to be dropped, or when the master downlink queue that corresponds to that slave contains a packet, or after a predefined timeout. In this manner, the slaves that have more traffic will receive proportionally larger share of the bandwidth, for as long as they have traffic.

The sequence of active slaves in the original FEP is fixed, but we may modify the scheme so that this sequence is dynamically determined in each cycle, according to the decreasing length of downlink queues. This scheme will be referred to as the FEP LDQF scheme.

A slightly different approach—similar to the token bucket scheme in wired networks—may be used to transform the E-limited scheme into an adaptive scheme. In this case, each slave has a separate and variable upper limit of up to M frames to be transferred during a single visit. When the slave is polled and there is a data packet sent in either direction, the current value of M for that slave is decreased by one, until some predefined limit is reached. Otherwise, when a POLL-NULL frame is encountered, the value of M for that slave is reset to the maximum value, but the slave will skip a predefined number of cycles. In this manner, the slave that has been idle for some time, can send more data immediately after becoming active again. If there is continuously backlogged traffic, the service gradually decreases to a fair share of available bandwidth. We will refer to this scheme as adaptive E-polling.x



(a) BNEP Protocol Stack.



(b) Packet Headers.

Fig. 2. BNEP: transmission of TCP/IP traffic over Bluetooth (adapted from [11]).

III. TCP/IP TRAFFIC OVER BLUETOOTH

All of the polling schemes described above focus on the optimization of performance of Bluetooth baseband traffic. However, Bluetooth piconets will actually transport the traffic between applications running on Bluetooth and other devices, hence we should look into the details of packet segmentation and reassembly policies. As most traffic nowadays is based on the TCP family of protocols, it is necessary to examine the ways in which such traffic can be transported over Bluetooth. Fortunately, the Bluetooth Network Encapsulation Protocol, or BNEP, provides a ready solution to these problems [11].

The BNEP protocol is designed to encapsulate and forward Ethernet frames through Bluetooth networks. Multi-hop traffic, including the slave to slave traffic, may be handled by Bluetooth masters and/or bridges acting as store-and-forward switches. In other words, the entire TCP PDU, which consists of a number of Bluetooth PDUs, has to be stored in the device before being repackaged (if necessary) and forwarded to the next stop along the route. Routing in this case is done in the IP layer, transparently to Bluetooth. Fig. 2(a) shows the protocol stack when TCP/IP packets are encapsulated using BNEP. The headers and their typical length are shown in Fig. 2(b). Note that each TCP message generated will require a total of 59 bytes in appropriate headers throughout the protocol stack.

In order to achieve higher efficiency, it is necessary to maintain a large maximum segment size (MSS). Although the default value of MSS is 536 bytes [13], the value of MSS can be automatically set for each TCP connection by the path MTU discovery procedure [14]. In our simulations, MSS was set to 1280 bytes. In this case, the packet header takes only about 5% of the maximum size of a L2CAP packet, which can fit into four DH5 Bluetooth baseband packets. As Bluetooth baseband packets can have either one, three, or five slots each (and the five slot DH5 packet carries the payload of up to 339 bytes), segmentation and reassembly of logical link data packets are done in L2CAP layer. In our simulation, we used a simple segmentation policy which splits L2CAP data packets into the smallest number of baseband packets.

Another important issue is efficiency and reliability of data transfer. Bluetooth baseband layer has simple provisions for both flow control and Automatic Repeat Request (ARQ) [2]. Traditional TCP congestion control mechanisms (e.g. Tahoe and Reno) have been shown to be rather inefficient when transported over reliable wireless data networks [15], and the problem of TCP fairness in Bluetooth has been addressed in [10]. Since the focus of our work is the performance of different polling scheme under TCP traffic, we have effectively disabled the congestion control mechanisms by setting the TCP advertising window to the minimum value of 2MSS [16]. Note that both end hosts are situated in the same piconet, and the volume of traffic in the piconet is not too high. Therefore, setting the advertising window in this way suffices to make the network fully utilized, without compromising the validity of our results.

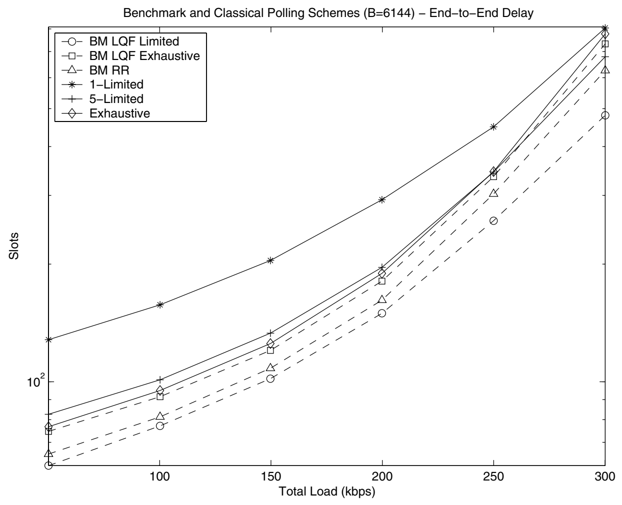
IV. Performance measurements

In order to evaluate and compare the performance of different polling schemes, we have built a Bluetooth simulator using the object-oriented Petri net engine Artifex [17]. Our setup includes the master with seven active slaves. Each slave pair has an active TCP connection via the piconet master. Simulated applications communicate using these connections, with BNEP-based segmentation and reassembly. For simplicity, we disregard the packet processing time, including segmentation and reassembly. Application message arrivals follow a Poisson distribution with arrival rate λ, and the message length is geometrically distributed with mean length of $\overline{B}$. In all simulations, the default value of $\overline{B}$ was 6KB, which corresponds to a typical message size of web traffic, and the total application traffic rate was in the range of 50Kbps to 300Kbps. Simulations were run for five full minutes, after a one minute warm-up delay, and all measured delays are expressed in Bluetooth time slots $T = 625\mu s$.

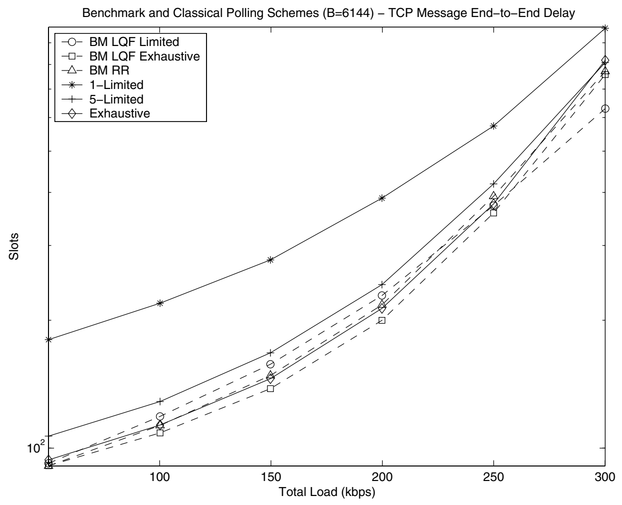
A. Traditional Polling Schemes

Delay performance of the traditional polling schemes, including the three benchmark schemes is shown Fig. 3; E-limited service used $M = 5$ frames per visit. For clarity, delays are shown using a logarithmic scale. Baseband delay corresponds to the mean delay of Bluetooth baseband packets contained in the TCP message, while TCP message delay is measured from the time the TCP message is assembled to the time it is received in its entirety at the destination.

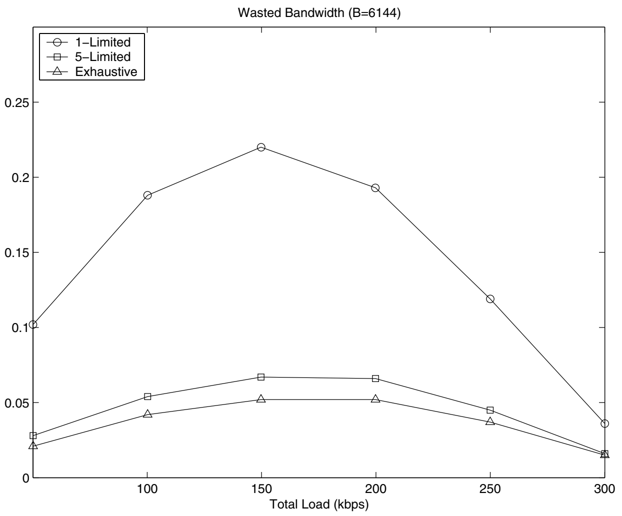
Under the definition of max-min fairness, the BM RR polling scheme is the most fair scheme among all benchmark polling schemes [18], while the BM LQF polling performs the best in term of mean end-to-end packet delay.



(a) Baseband Packet End-to-End Delay.



(b) TCP Message End-to-End Delay.



(c) Wasted Bandwidth (selected polling schemes only).

Fig. 3. Performance of traditional polling schemes.

Note that 1-limited polling is the best in terms of fairness – each slave is guaranteed to be visited once every seven frames. However, this scheme is not work conserving, i.e., it will poll slaves without any data even when there are slaves that have data. The POLL-NULL frames that occur in this process consume bandwidth, and thus lead to an increase in mean delays. This effect is even more pronounced under TCP traffic: under a given offered load, the message inter-arrival time increases when the mean burst size increases. Consequently, when a flow is active, it will remain active until the whole burst of packets are serviced; but only a few flows will be active at any given time. Therefore, under limited service, a large portion of the available bandwidth will be wasted by the POLL-NULL sequences.

On the contrary, exhaustive service tries to be as work conserving as possible regardless of fairness.

Note that the BM RR polling scheme may be considered as a work-conserving variation of the 1-limited scheme. Its delay performance is nearly optimal. In contrast, the BM Exhaustive polling scheme does not perform well even though it is work conserving.

The conclusions presented above may be confirmed by looking at the diagram of wasted bandwidth of Fig. 3(c). Wasted bandwidth is obtained by counting the POLL-NULL frames that occur during the visit to one slave at the time some other uplink or downlink queue is non-empty. The wasted bandwidth starts increasing with the load until the offered load reaches about 150Kbps. In this region the offered load is small, and most of the time the system is actually idle when a traffic burst arrives. As the master cannot known whether a given slave has any traffic, polling most slaves is effectively a waste of bandwidth. Above the load of 250Kbps or so, both uplink and downlink queues start to grow, and the number of wasted POLL-NULL frames decreases – but the delays exhibit a rapid increase, as shown in Figs. 3(a) and 3(b).

These results confirm our previous observations, the 1-limited scheduler wasted the most bandwidth when the burst size increases. With a mean message length of 6KB, the 1-limited service wasted about 12% of the bandwidth, as opposed to only about 4% for the exhaustive service (under a load of 250Kbps). Under pure Poisson baseband packet arrivals—which is not a realistic model, as shown in [19], and hence is not shown here—the 1-Limited scheduler performed very well, wasting only a small portion of the bandwidth.

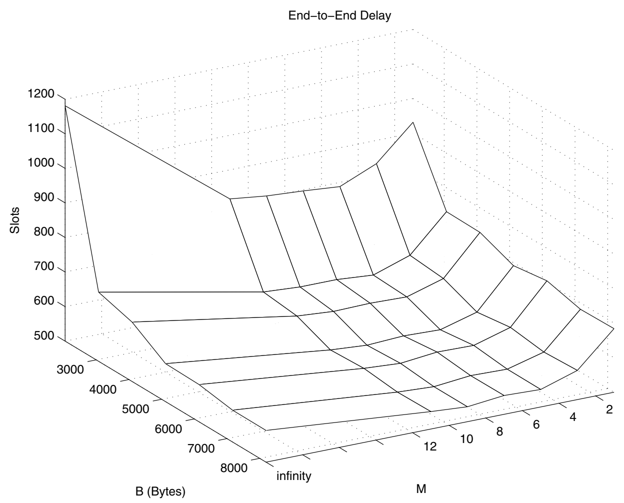
The E-limited service tries to be more work conserving at the expense of fairness; its end-to-end packet delays are shown in Fig. 4. The value of the parameter M should be chosen so as to minimize the number of wasted POLL-NULL frames while the system is not idle. The optimal value of M actually depends on the mean message length, which effectively determines the burst size for the baseband traffic. (Similar observations, albeit in a slightly different context, were made in [12].) We have found that the simple E-limited polling scheme with $M = 5$ works efficiently in a wide range of mean message lengths.

B. Adaptive Polling Schemes

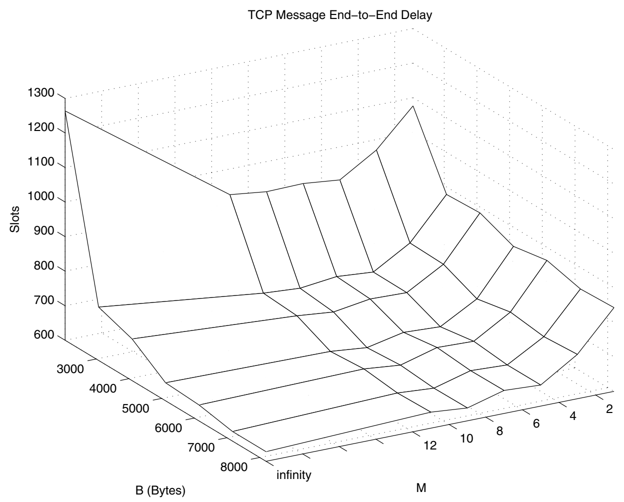
As noted above, both the EPM and FEP LDQF schemes make use of the LQF discipline. Comparing the performance of BM RR and BM LQF Limited schemes of the previous section, the LQF discipline apparently helps in minimizing the mean packet delay. The EPM scheduler performs very well in minimizing the TCP message delay, which is not quite unexpected, given the performance of the BM LQF Exhaustive scheme. EPM always tries to service the longest queue that might produce the largest delays. Under TCP traffic, the TCP flow control effectively eliminates the risk of starvation, and the overall performance is very good. However, the problem of fairness and its performance under bursty traffic are still not addressed, and we will discuss these issues in the next section.

The benchmark schemes will always work better under high load, as they will generally minimize the bandwidth wasted by the POLL-NULL frames.

The FEP LDQF scheme does not perform too well, probably because it lacks the cyclic sequence and because the acknowledgment packets cannot be serviced with low delay. Also, its limited property makes the transmission of a single TCP message take several rounds which also leads to inordinately high

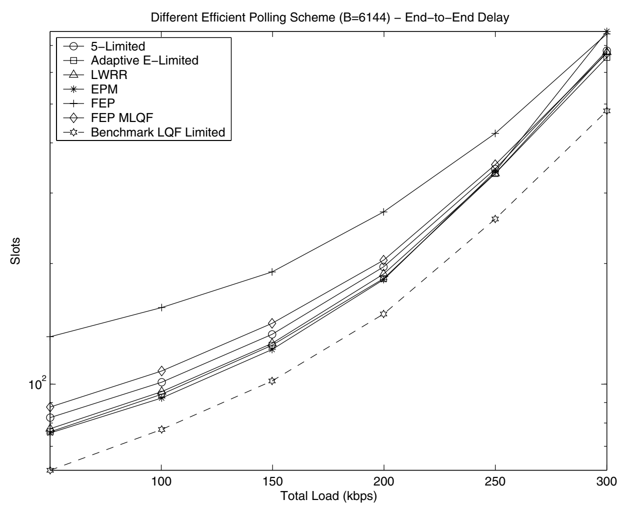


(a) Baseband packet delay.

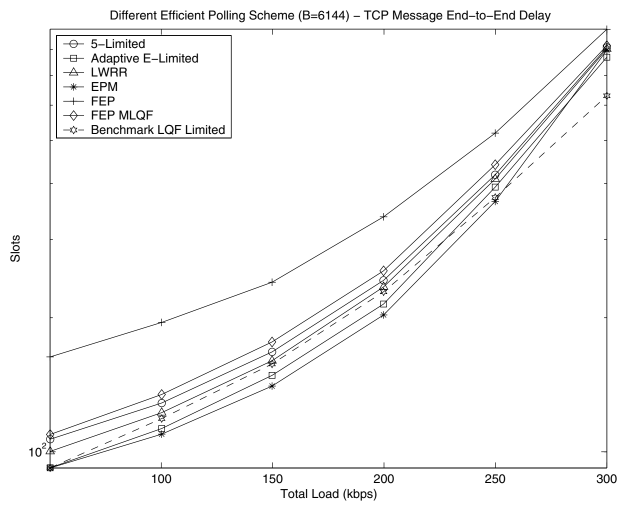


(b) TCP message delay.

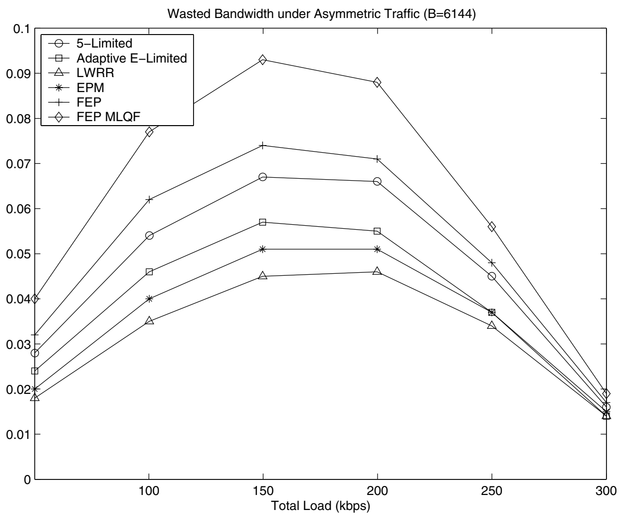
Fig. 4. End-to-end packet delays under E-Limited service, as a function of the parameter M and mean message length $\overline{B}$, at 300Kbps load.



(a) Baseband Packet End-to-End Delay.



(b) TCP Message End-to-End Delay.



(c) Wasted Bandwidth.

Fig. 5. Performance of Adaptive Polling Schemes.

delays. Note that the TCP message delay for BM LQF scheduling scheme is even larger than that of the EPM and adaptive E-limited schemes under low load. Again, the schemes based on the LQF scheme try to minimize the mean delay for baseband packets only. A TCP message, which is contained in a number of consecutive baseband packets, would prefer the polling scheme which stays longer with the slave under low load. At the same time, the acknowledgment packets require smaller service access time (and, consequently, smaller cycle time) in order for TCP flow control mechanism to function effectively.

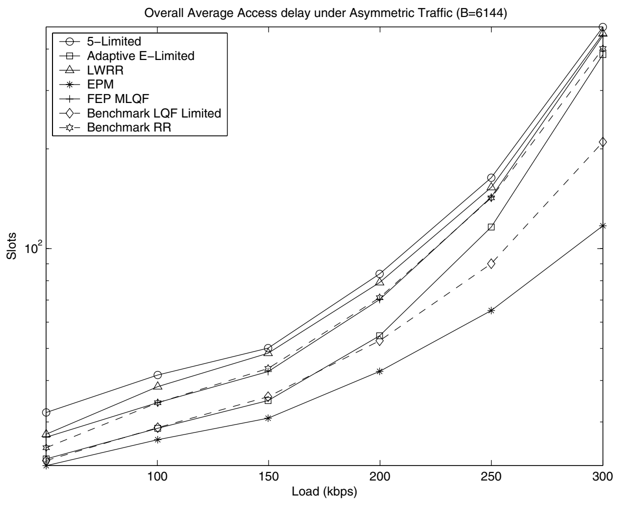
Both the Adaptive E-limited and LWRR schemes perform very well in terms of end-to-end delay and TCP message delay alike. As shown in Fig. 5(c), both schemes actually waste less bandwidth than the original E-limited service, and LWRR is even better than EPM in this sense. It should be noted that both schemes can be implemented without putting an undue burden, computation- and memory-wise, on the power-limited Bluetooth devices.

It may be interesting to observe the behavior of the FEP scheme, which aims to work as an exhaustive scheme in low load and a round robin scheme in high load [7]. Under pure Poisson traffic, FEP works as advertised, and in fact it outperforms other polling schemes in terms of baseband packet end-to-end delays. However, under TCP traffic the delay of the FEP scheme increases, and under bursty traffic the FEP scheme is the worst adaptive scheme in our test set. This is why this scheme will not be included in the discussion that follows in the next section.

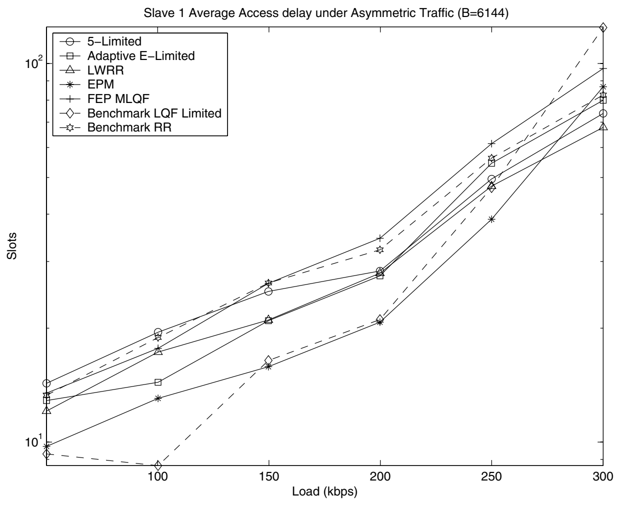
V. Performance under heterogeneous traffic

All measurements described in the previous Section have been conducted under the assumption that the traffic is homogeneous with respect to the data rates of the slaves, i.e., that all slaves have equal data rates. However, this may not be sufficient and general enough to assess the performance of different polling schemes. To that end, we have also conducted simulations with heterogeneous traffic, in which different slaves have differnet data rates, using the following setup. (Note that the mean aggregate uplink and downlink rates are still equal.) Let the piconet operate under homogeneous traffic with the offered load of ρ, with equal portions of $\rho_s = \rho/m$ contributed by each of the m slaves. For heterogeneous traffic, the total offered load is still ρ, but the contributions of individual slaves are uniformly distributed in the range of $0.2\rho_s$ to $1.8\rho_s$. In both cases, all slaves generate traffic with the same mean burst/message size.

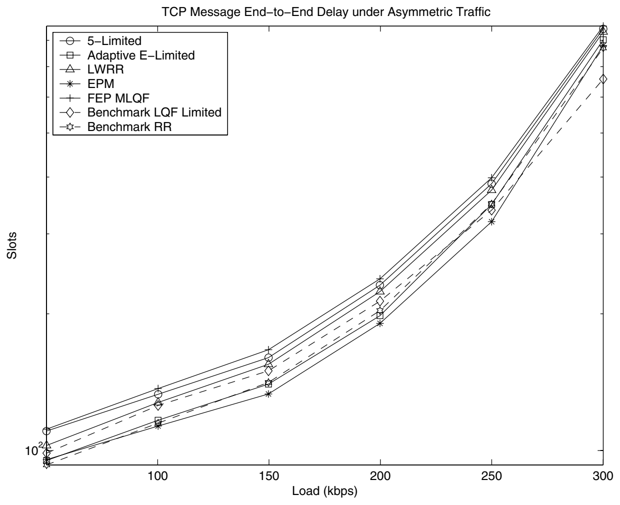
Mean access delays for all the packets in the piconet, and only for the packets generated by the slave 1 (which contributes $0.2\rho_s$ load), are shown in Fig. 6. As can be seen, the adaptive E-limited service outperforms both traditional E-limited and LWRR in terms of overall TCP message delays and access delays. This is further confirmed by the diagram of wasted bandwidth shown in Fig. 7. The EPM, LWRR and Adaptive E-limited schemes actually waste less bandwidth under high loads than the other schemes, but note how the corresponding curves converge to one another under very high loads. Interestingly enough, the FEP LDQF scheme does not work well under heterogeneous traffic, while the BM LQF schemes still perform the best in terms of overall end-to-end delays.



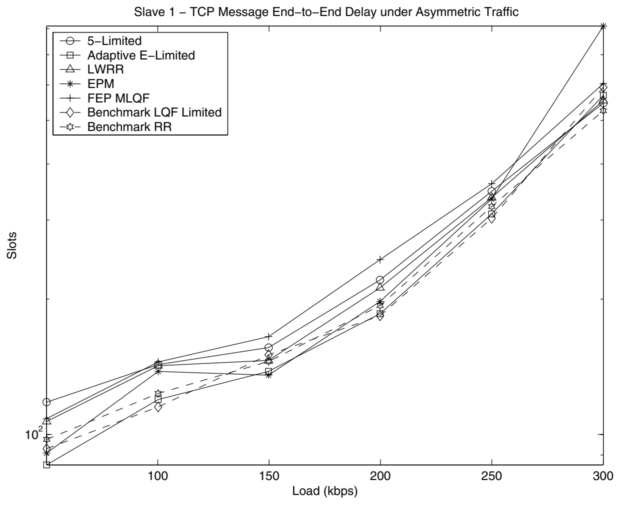
(a) Overall Access Delay.



(b) Slave 1 Access Delay.



(c) Overall TCP Message Delay.



(d) Slave 1 TCP Message Delay.

Fig. 6. Overall and Slave 1 mean baseband packet delays under heterogeneous traffic.

To address the problem of fairness, let us focus on the delay performances of a particular slave (slave 1), which contributed the load of $0.2\rho_s$. Since this is the smallest contribution to the total load, this slave consumes the smallest amount of bandwidth in an environment with a large amount of background loading. By the definition of max-min fairness, if the scheduling scheme is fair, every packet burst should be served with at least a fair share of the system service rate. Consequently, such a slave should not suffer too large delays compared to others, even when the background loading is very high.

Note that under small packet burst arrival rate, the uplink queue with not build up quickly and the access delay will remain small. Notice that the access delay could be more meaningful in assessing fairness for non-exhaustive scheduling schemes. As the packet generated by slave 1 will have different destinations among other slaves with equal probabilities, the end-to-end delay of a packet generated by slave 1 consists of both the access delay and delay contributed by the downlink queues that correspond to the destination slaves.

The results show that the adaptive E-limited scheme performs very well in overall access delays, slave 1 access delays, overall TCP message delays and slave 1 TCP message delays. The token bucket-like robustness of this scheme makes it work well under heterogeneous traffic. It can provide fairness among different slaves with different offered load, which other schemes, including LWRR, do not provide.

The EPM scheme, on the other hand, even though it provides the lowest overall access and TCP message delays, does suffer from fairness-related problems when the piconet load increases. namely, all the delays, including the TCP message delay, increase quite a lot. This increase is observed even for slaves that contribute only a small amount of load to the piconet. As a result, the adaptive E-limited scheme should be the most appropriate choice under heterogeneous traffic.

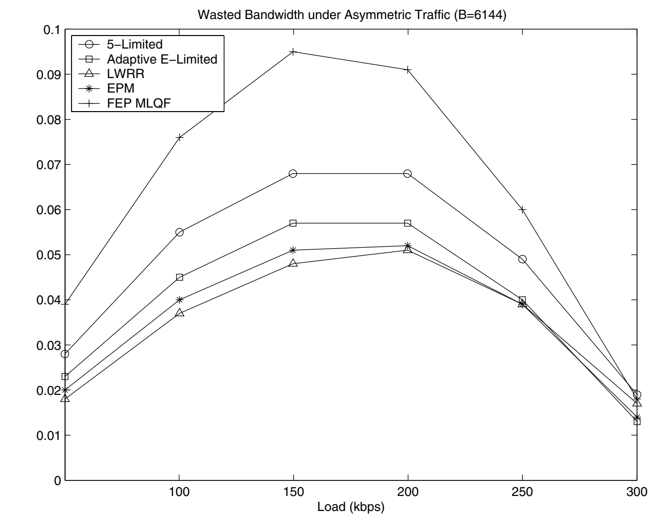


Fig. 7. Wasted Bandwidth under heterogeneous traffic

VI. Conclusions

In this paper, we have evaluated a number of traditional and adaptive polling schemes under TCP traffic, with Poisson arrivals of application messages. We have considered both homogeneous traffic, when all slaves have equal traffic arrival rates, and heterogeneous traffic, where the arrival rates for individual slaves differ. Our simulations have shown that the schemes known as Adaptive E-limited service, and Limited and Weighted Round Robin, perform best, with the adaptive E-limited polling scheme performing slight better under heterogeneous traffic.

References

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

[2] Bluetooth SIG, *Specification of the Bluetooth System*. Feb. 2001.

[3] H. Levy, M. Sidi, and O. J. Boxma, "Dominance relations in polling systems," *Queueing Systems Theory and Applications*, vol. 6, no. 2, pp. 155–171, 1990.

[4] Z. Liu, P. Nain, and D. Towsley, "On optimal polling policies," *Queueing Systems Theory and Applications*, vol. 11, no. 1–2, pp. 59–83, 1992.

[5] A. Capone, R. Kapoor, and M. Gerla, "Efficient polling schemes for Bluetooth picocells," in *Proceedings of IEEE International Conference on Communications ICC 2001*, vol. 7, (Helsinki, Finland), pp. 1990–1994, June 2001.

[6] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, "Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network," in *Proceedings Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2001*, vol. 1, (Anchorage, AK), pp. 591–600, Apr. 2001.

[7] N. Johansson, U. Körner, and P. Johansson, "Performance evaluation of scheduling algorithms for Bluetooth," in *Proceedings of BC'99 IFIP TC 6 Fifth International Conference on Broadband Communications*, (Hong Kong), pp. 139–150, Nov. 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 Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)*, (San Diego, CA), pp. 384–388, Nov. 1999.

[9] H. Takagi, *Queueing Analysis*, vol. 1: Vacation and Priority Systems. Amsterdam, The Netherlands: North-Holland, 1991.

[10] Y.-Z. Lee, R. Kapoor, and M. Gerla, "An efficient and fair polling scheme for Bluetooth," in *Proceedings MILCOM 2002*, vol. 2, pp. 1062–1068, 2002.

[11] Bluetooth SIG, "Bluetooth Network Encapsulation Protocol (BNEP) Specification," tech. rep., Revision 0.95a, June 2001.

[12] J. Mišić and V. B. Mišić, "Modeling Bluetooth piconet performance," *IEEE Communication Letters*, vol. 7, pp. 18–20, Jan. 2002.

[13] R. T. Braden, "Requirements for Internet hosts — communication layers," Internet standard RFC 1122, IETF, Oct. 1989.

[14] J. C. Mogul and S. E. Deering, "Path MTU discovery," draft Internet standard RFC 1191, IETF, Nov. 1990.

[15] R. Ludwig, B. Rathonyi, A. Konrad, K. Oden, and A. Joseph, "Multi-layer tracing of TCP over a reliable wireless link," in *Proceedings of the 1999 ACM SIGMETRICS*, pp. 144–154, 1999.

[16] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," draft Internet standard RFC 2581, IETF, Apr. 1999.

[17] RSoft Design, Inc., *Artifex v.4.4.2*. 2003.

[18] E. L. Hahne, "Round-roin scheduling for max-min fairness in data networks," *IEEE Journal on Special Areas in Communications – Wireless Series*, vol. 9, pp. 1024–1039, Sept. 1991.

[19] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *ACM/IEEE Transactions on Networking*, vol. 3, pp. 226–244, June 1995.