Analytical Characterisation of the Performance of Bluetooth Piconets Using Serial Port Profile

R. Luque, M.J. Morón and E. Casilari-Perez
Dept. Tecnología Electrónica, University of Málaga, Málaga, Spain
mjmoron,jrlg,ecasilari@uma.es

ABSTRACT
Bluetooth is a key connectivity technology for the deployment of wireless Personal Area Networks as far as it is the most popular low power communication feature incorporated in devices such as laptops or smartphones. This paper proposes an analytical model to predict the delay of the transmissions in Bluetooth piconets employing Serial Port Profile (SPP), which is massively implemented by Bluetooth-enabled equipments. The characterization includes the impact of the overhead and the segmentation imposed by the different protocols involved in the transmission as well as the delay provoked by the polling process that is executed to regulate the activity of the different slaves in the piconet. The model has been empirically evaluated and tested in an actual Bluetooth piconet.

Categories and Subject Descriptors
C.2.1 [Computer Communication Networks]: Network architecture and design

General Terms
Measurement, Performance.

Keywords
Bluetooth piconet, Serial Port Profile, transmission delay

1. INTRODUCTION
Bluetooth (BT) has become a major technology for the development of short range and low power networking applications in PANs (Personal Area Network) and BANs (Body Area Networks). The basic topology of Bluetooth networks is the piconet, a group of up to eight devices which follow a common frequency-hopping radio channel. The access to the shared radio medium is governed by one of the units (the master), which polls the other devices (slaves) in a cyclic way according to a time-division duplex (TDD) mechanism. The way in which this polling process is scheduled and executed dramatically influences on the performance of the BT piconet. Besides, in order to provide vendor interoperability, the BT specifications defines the so-called Serial Port Profile (SPP), which polls the other devices (slaves) in a cyclic way according to a time-division duplex (TDD) mechanism. The way in which this polling process is scheduled and executed dramatically influences on the performance of the BT piconet. Besides, in order to provide vendor interoperability, the BT specifications defines the so-called Serial Port Profile (SPP) (starting point for other BT profiles, such as Dial-Up Networking Profile or DUN) is one of the most popular profiles included in commercial BT devices, such as Blackberry units, Smartphones, keyboards, GPS or wireless medical sensors.

The paper is organized as it follows: Section 2 proposes an analytical model for the estimation of the transmission delay in a generic piconet with several slaves. Section 3 validates the model against extensive and systematic pre-programmed transmissions in a real BT piconet. Conclusions are summarized in Section 4.

2. CHARACTERISATION OF THE DELAY
2.1 Model with a Single Slave
Serial Port Profile specifies the utilization of RFCOMM protocol to emulate RS232 cable communications. RFCOMM conveys the user data (structured in frames) to the lower layers of Bluetooth stack through L2CAP (Logical Link Control & Adaptation Protocol). L2CAP layer is in charge of managing the Bluetooth QoS (Quality of Service), as well as of multiplexing, segmenting and reassembling data flowing from/to the upper layers. The user data are fragmented by RFCOMM into frames. Every RFCOMM frame is encapsulated by L2CAP into a single L2CAP frame. L2CAP is in turn layered over the physical layer (the Bluetooth Baseband implemented in the BT controller). The Bluetooth Baseband (if necessary) splits the RFCOMM/L2CAP frames into a series of Bluetooth packets before sending them to the radio medium.

To estimate the minimum delay ($t_o$) for transmitting $N$ user data bytes under SPP in ideal conditions (i.e.: when no packet retransmission is provoked by any error bit), we have to consider the impact of the protocol overhead introduced by RFCOMM and...
L2CAP together with the fragmentation that L2CAP and Baseband layers perform. Equation (1) computes this delay (see also [4]). The formula takes into account that user data can be segmented in different RFCOMM/L2CAP frames ($n_{ff}$ "not-final" or intermediate frames and one final frame of $L_f$ bytes) so that the reception is considered to be finished after the reception of the last bit of this final frame:

$$t_{\text{tx}}(N) = n_{ff} \cdot t_{\text{ACK}}(L_f + O_h(L_f) + H_f) + t_{\text{tx}}(L_f + O_h(L_f) + H_f)$$  \hspace{1cm} (1)$$

The variables in this equation are defined as follows:

- $L_f$ is the size at which RFCOMM will fragment the user data into a series of RFCOMM/L2CAP frames before delivering them to L2CAP. This size is constrained by both the Maximum Frame Size ($N_f$) of RFCOMM [1] and the Maximum Transfer Unit (MTU) of L2CAP for RFCOMM ($M_f$):

$$L_f = \min(N_f, M_f - O_{\text{max}})$$  \hspace{1cm} (2)$$

where $O_{\text{max}}$ denotes the maximum possible overhead of RFCOMM (5 bytes) so that the difference ($M_f - O_{\text{max}}$) indicates the maximum number of user data that can be transported in a L2CAP frame without surpassing the limit fixed by $M_f$. On the other hand, $N_f$ has a default value of 127 bytes [5], although it can be negotiated by the nodes in the range 23-32767 bytes.

- $O_h(x)$ is the overhead of RFCOMM in each frame: 5 bytes if the payload ($x$) exceeds 127 bytes and 4 bytes in other case, while $H_f$ is the number of bytes of the L2CAP header (4 in Bluetooth 1.1 and for the basic mode of Bluetooth 1.2).

- $n_{ff}(N)$ is the number of not-final L2CAP frames in which the user data are segmented. It can be calculated as:

$$n_{ff}(N) = \left\lfloor \frac{N}{L_f} - 1 \right\rfloor$$  \hspace{1cm} (3)$$

where $\left\lfloor x \right\rfloor$ indicates the lowest integer higher than $x$.

- $L_f$ represents the number of bytes of the final L2CAP frame, computable as:

$$L_f = \left( (N - 1) \mod L_f \right) + 1$$  \hspace{1cm} (4)$$

The equation (1) contemplates the segmentation that BT executes when the transmission of the $N$ user data bytes (and the corresponding overhead introduced by RFCOMM and L2CAP) requires more than one Baseband packet. Therefore, the formula considers two components, $t_{\text{ACK}}$ and $t_{\text{tx}}$, defined as it follows:

- The term $t_{\text{ACK}}(x)$ represents the time that the BT Baseband requires to send and acknowledge all the BT packets corresponding to any not-final L2CAP frame of $x$ bytes:

$$t_{\text{ACK}}(x) = \begin{cases} 
0 & x = 0 \\
2 \cdot T_t & 0 < x < L_1 \\
4 \cdot T_t & L_1 < x < L_3 \\
6 \cdot T_t & L_3 < x < L_5 \\
6 \cdot T_t + t_{\text{ACK}}(x \mod L_f) & x > L_5 
\end{cases}$$  \hspace{1cm} (5)$$

where $\left\lfloor x \right\rfloor$ denotes the highest integer lower than $x$, $T_t$ is the duration of a Bluetooth slot (625 µs), while $L_1$, $L_3$ and $L_5$ are the maximum sizes of the payload of a 1, 3 and 5-slot Bluetooth packet, respectively. These sizes are 27, 183 and 339 bytes for DH (Data High-Rate) packets and 17, 121 and 224 bytes for DM (Data Medium-Rate) packets. DM-type packets convey less user data as they incorporate an additional overhead to provide FEC protection. The recursive expression in (5) models the time needed to acknowledge all the BT packets into which the L2CAP frames are segmented. The formula assumes the optimal case in which no errors occur in the packets. Thus, any BT packet is always acknowledged in the next slot (see [6] for the case with losses). As a result, there is an invariable delay of 2, 4 or 6 slots for every packet of 1, 3 and 5 slots, respectively.

- The term $t_{\text{tx}}(x)$ describes the time needed to transmit the final L2CAP frame of $x$ bits. As the transmission is finished when the last bit of the final frame is received in the reception point (in our case the BT master), neither the final acknowledgement slot nor the complete final slot of the BT packet are computed. Thus, this time $t_{\text{tx}}(N)$ can be specifically estimated as a function of the number of transmitted bits:

$$t_{\text{tx}}(x) = \left\{ \begin{array}{ll}
& N_{g}(x) \cdot T_g \quad x \leq L_5 \\
& T_{\text{ACK}}(L_f) \cdot \frac{x}{L_f} + t_{\text{tx}}(x \mod L_f) \quad x > L_5
\end{array} \right. $$  \hspace{1cm} (6)$$

where $T_g$ is the time for transmitting 1 bit (1 µs at 1 Mbps) while $N_{g}(x)$ is the number of bits of the final BT packet, which can be computed as:

$$N_{g}(x) = N_{\text{max}} + N_{pl}(x)$$  \hspace{1cm} (7)$$

where $N_{\text{max}}$ (126 bits) is the control information in the Bluetooth packet (54 bits of the packet header and a 72 bit access code), while $N_{pl}(x)$ represents the size in bits of the Bluetooth payload:

$$N_{pl}(x) = \begin{cases} 
& x + H_5 + H_{\text{CRC}} \cdot 8 & \text{DH packets} \\
& \frac{(x + H_5 + H_{\text{CRC}})}{10} \cdot 15 & \text{DM packets}
\end{cases}$$  \hspace{1cm} (8)$$

where $H_{\text{CRC}}$ (2 bytes) describes the CRC (Cyclic Redundancy Check) field while $H_5$ ($H_5=1$ byte for 1 slot and $H_5=2$ for 3 and 5 slot-packets, respectively) represents the payload header. The previous equation considers that for DM packets, for every 10 information bits 5 redundancy bits are added. Consequently, if the number of bits is not a multiple of 10, the packet must be filled with extra bits after the CRC.

Note that the equation (6) also takes into account that if the final L2CAP frame exceeds the size of a 5-slot BT packet, more than one BT packet will be needed. So, as for the case of not final L2CAP frames, the expression also includes the time to send and acknowledge the corresponding intermediate 5-slot BT packets. In addition, the management of QoS in BT establishes a polling mechanism that obliges the master to address the slaves just at regular intervals (before a poll interval, $T_{\text{poll}}$, expires). As a result, when the first data are ready to be transmitted at the application layer in the slave, the slave must wait to be polled by the master with a specific 1-slot POLL packet. Consequently the transmission may still be delayed up to an extra time of $T_{\text{poll}}$. Assuming that this waiting period can be modeled by a uniform distribution, the expected mean of the actual delay ($t'_{g}(x)$) has to incorporate the effect of the polling process by adding an offset of $T_{\text{poll}}/2$ to the previously computed delay:

$$t'_{g}(N) = t_{g}(N) + \frac{T_{\text{poll}}}{2} + T_s$$  \hspace{1cm} (9)$$

The previous equation also includes the slot ($T_s$) required by the master to send the poll packet, so that the slave can start its transmission.
2.2 Model for a BT Piconet with two slaves

The Bluetooth specifications do not define the scheduling mechanism that must be followed to poll the slaves. In this sense, the literature has proposed many polling schemes to optimize the performance of Bluetooth communications, especially under specific circumstances such as asymmetric transmissions (see, for example, [7] [8] or [9] for an overview on this issue). However, in spite of all these existing alternatives, 1-limited (or pure) Round Robin is commonly implemented in most commercial Bluetooth devices (see CSR chipset specification [10]). Although it may offer a very poor performance, this scheduling policy is selected due to its simplicity and low implementation cost.

The scheduling scheme imposes the way in which slaves are ‘visited’ by the master as well as the number of BT packets that can be exchanged between the master and each slave after each polling operation. The simple Pure Round Robin algorithm follows a circular and fixed ordering of the slaves. Similarly the bandwidth is equally distributed among the slave nodes with independence of their actual needs for transmitting data. Thus, under the version of 1-Limited Service polling implemented by BT devices, once any transmission from any slave begins, the master just permits the slave to send one uplink BT packet (of up to 5 slots) before polling the next slave in the sequence. On the other hand, if a slave has no user traffic to send, it will have to respond with a NULL 1-slot packet after being polled (which may lead to a very poor network performance in asymmetric traffic conditions). Therefore, the mere existence of another slave will induce an extra delay in the transmissions of user data requiring more than one BT packet. We propose to extend the model developed in the previous section when two slaves are present in the piconet. In this sense we consider two limit cases that determine the lower and upper bounds of the transmission delay of this scenario (always assuming that no losses occurs so that no packet has to be retransmitted):

1) Lower bound of the delay.

This is the case when the transmissions of the two slaves do not coincide, i.e.: when just one slave is transmitting or the packets from both slaves do not overlap in time. This situation has been illustrated in Figure 1 in which one slave (S1) has to transmit data that must be split in (at least) two 5-slot BT packets while the other slave (S2) has no data to send to the master. As it can be observed from the figure, the TDD (Time Division Duplex) scheme of BT imposes that the master always starts its transmissions in an even-number slot while a slave (S1 or S2), on being polled, must reply in the next odd-numbered slot. The figure also shows that S1 must wait two extra slots before transmitting the second packets as long as S2 must receive and respond the poll packet sent by the master.

Thus, the component of the delay which will be directly influenced when more than one slave is sharing the piconet is \( t_{\text{ACK}}(N) \). This component, described in eq. (5), has to be redefined to account for these two slots of the polling process:

\[
I_{\text{ACK}}(x) = \begin{cases} 
0 & x = 0 \\
4 \cdot T_s & 0 < x \leq L_s \\
6 \cdot T_s & L_s < x \leq L_s \\
8 \cdot T_s & L_s < x \leq L_s \\
8 \cdot T_s \cdot \frac{x}{L_s} + I_{\text{ACK}}(x \mod L_s) & x > L_s 
\end{cases}
\]

Consequently, the minimum time required \( t_{\text{ACK}}(N) \) to send \( N \) user data bytes in a piconet of two slaves can be computed as the delay \( t_p(N) \) in the case with one slave but substituting \( t_{\text{ACK}}(N) \) by this new term \( t_{\text{ACK}}(N) \) in eq. (1) and (6).

2) Upper bound of the delay.

In opposition to the previous optimistic situation, the worst case in a piconet of two slaves takes place when both slaves are transmitting long frames of user data to the master simultaneously. In this case, sketched in Figure 2, each slave will transmit a 5-slot BT packet after receiving a poll. As a consequence, when compared with a piconet of 1 slave, the acknowledgement of a BT packet sent by a slave may be deferred up to 6 extra slots (the slot needed by the master to address the other slave plus the 5 slots required by the BT packet sent by the neighbor).

\[
I_{\text{ACK}}(x) = \begin{cases} 
0 & x = 0 \\
8 \cdot T_s & 0 < x \leq L_s \\
10 \cdot T_s & L_s < x \leq L_s \\
12 \cdot T_s & L_s < x \leq L_s \\
12 \cdot T_s \cdot \frac{x}{L_s} + I_{\text{ACK}}(x \mod L_s) & x > L_s 
\end{cases}
\]

Finally, as in the piconet with just one slave, when no packets are being transmitted, the master is only compelled to poll the slaves just after a period of \( T_{\text{poll}} \). Therefore, the emission of the initial BT packet of each transmission may wait up to \( T_{\text{poll}} \) to be initiated. Again, equation (12) takes into account the mean case for which an extra delay of \( T_{\text{poll}}/2 \) is added to the lower limit of the delay to define the mean delay in optimal conditions:

\[
t_{\text{rmiss}}(N) = t_{\text{rmiss}}(N) + \frac{T_{\text{poll}}}{2} + T_s
\]

For the computation of the maximum delay case we consider the worst case in which the slave waits for a period \( T_{\text{poll}} \):

\[
t_{\text{rmiss}}(N) = t_{\text{rmiss}}(N) + \Delta T + T_s
\]

being:

\[
\Delta T = \max(6 \cdot T_s, T_{\text{poll}})
\]
As it can be observed, the definition of the increment $\Delta T$ also considers the situation in which the other slave is transmitting a 5-slot packet when the new communication begins and the polling interval established by $T_{poll}$ cannot be satisfied.

### 2.3 Generalization for a piconet of n slaves

For a generic point-to-multipoint communication topology with multiple slaves, the previous model for two slaves can be easily extended. So, for the lower limit of the bound and considering that there are $n_s$ slaves with just one slave transmitting, we have that $t_{\text{ACK}(x)}$ can be redefined as:

$$
    t_{\text{ACK}(x)}(x,n_s) = \begin{cases} 
    0 & x = 0 \\
    (2 + 2 \cdot (n_s - 1)) \cdot T_s & 0 < x \leq L_1 \\
    (4 + 2 \cdot (n_s - 1)) \cdot T_s & L_1 < x \leq L_3 \\
    (6 + 2 \cdot (n_s - 1)) \cdot T_s & L_3 < x \leq L_5 \\
    (6 + 2 \cdot (n_s - 1)) \cdot T_s + \frac{x}{L_5} & x > L_5 \\
    + t_{\text{ACK}(x \mod L_5,n_s)} & 
    \end{cases}
$$

Basically, the new version of the term includes the extra delay of 2 slots per slave demanded by the master to poll the rest of the slaves. On the contrary, the worst case takes place when all the slaves in the piconet transmit series of 5-slot BT packets at the same time. For this situation, $t_{\text{ACK}(x)}$ and $\Delta T$ is computed as:

$$
    t_{\text{ACK}(x,n_s)}(x) = \begin{cases} 
    0 & x = 0 \\
    (2 + 6 \cdot (n_s - 1)) \cdot T_s & 0 < x \leq L_1 \\
    (4 + 6 \cdot (n_s - 1)) \cdot T_s & L_1 < x \leq L_3 \\
    (6 + 6 \cdot (n_s - 1)) \cdot T_s & L_3 < x \leq L_5 \\
    (6 + 6 \cdot (n_s - 1)) \cdot T_s + \frac{x}{L_5} & x > L_5 \\
    + t_{\text{ACK}(x \mod L_5,n_s)} & 
    \end{cases}
$$

$$
\Delta T(n_s) = \max((n_s - 1), 6), T_{pol}
$$

### 3. EMPIRICAL MODEL VALIDATION

We have evaluated the correctness of the proposed model by measuring the end-to-end delay in the communications of an actual Bluetooth piconet utilizing SPP. The employed measurement testbed is outlined in Figure 3. As it can be appreciated from the figure, the deployed piconets consisted of multiple slaves, the previous model for two slaves can be easily extended. So, for the lower limit of the bound and considering that there are $n_s$ slaves with just one slave transmitting, we have that $t_{\text{ACK}(x)}$ can be redefined as:

$$
    t_{\text{ACK}(x)}(x,n_s) = \begin{cases} 
    0 & x = 0 \\
    (2 + 2 \cdot (n_s - 1)) \cdot T_s & 0 < x \leq L_1 \\
    (4 + 2 \cdot (n_s - 1)) \cdot T_s & L_1 < x \leq L_3 \\
    (6 + 2 \cdot (n_s - 1)) \cdot T_s & L_3 < x \leq L_5 \\
    (6 + 2 \cdot (n_s - 1)) \cdot T_s + \frac{x}{L_5} & x > L_5 \\
    + t_{\text{ACK}(x \mod L_5,n_s)} & 
    \end{cases}
$$

$$
\Delta T(n_s) = \max((n_s - 1), 6), T_{pol}
$$

For a better evaluation of the upper bound and the mean lower bound of the delay the experiments were replicated in two scenarios. In the first scenario just one slave transmits so mean delay can be calculated assuming the formula for the lower bound. In the second scenario both nodes sends user data to the master so that these transmissions periodically coincide. This synchronization of the transmissions of both slaves induces delay peaks that should be fitted by the proposed formula for the upper delay bound. DH packets were utilized for the shown experiments although a similar fitting of the model is obtained with DM packets. The Poll interval ($T_{poll}$) is selected to be 10 ms, the minimum value that the CSR BT module can guarantee.

Figure 4 compares the performed measurements with the results of applying the proposed analytical models to compute the upper and mean lower bounds. In particular the performed measurements include 1) the mean estimated delay for the case in which just one slaves is transmitting 2) the 99% percentile of the estimated delay when both slaves send packets. This second case practically describes the maximum expectable transmission delay. The absolute maximum of the measurements was not considered to avoid the impact of very specific samples with an unexpectedly high value of the delay induced by external factors such as an event in the operating system. For the whole considered range of the user data size, the figure shows the capability of the models to fit the actual behavior of the delay bounds. Additionally, for comparison purposes the figure also includes the results of the analytical model for a piconet with a single slave. The figure obviously evidences that the delay introduced by the polling policy suffers an abrupt increase whenever a new BT packet is required to transport the user data. On the other hand, if the data fits in just one BT packet (data sizes below 330 bytes) the results for the optimal case in the piconet of two slaves are equal to those of a piconet with one slave as long as long as the transmitting slave will not have to wait any poll to the other slave. Finally, to prove the validity of the extended model for a piconet with several slaves, we aggregated a third slave to the network (by connecting a fourth BT dongle to the PC) and repeated the previous experiments. The results of this piconet of 3 slaves are displayed in Figure 5. The figure proves again the accuracy of the model as...
well as the increasing impact of the polling process on the delay when the size of the user data augments.

4. CONCLUSIONS
This work has presented and validated an analytical model to compute the delay of the transmissions of ACL traffic in Bluetooth piconets under Serial Port Profile. In contrast with other studies, the model includes the effect of the overhead and the segmentation introduced by the different protocols (RFCOMM, L2CAP, Baseband) in the Bluetooth stack as well as the delay provoked by the polling policy when more than one slave is active in the network. The model permits to predict the minimum and maximum expected delay of BT transmissions, given the size of the data user and the number of slaves in the piconet. The model presumes optimal radio conditions so it assumes that no packet retransmission occurs. Thus, the model, which has been empirically validated in an actual piconet, could be employed to assess the limits of Bluetooth applicability in piconets trying to support ACL best effort traffic applications. The model has been developed for the version 1.1 of the Bluetooth standard but it can be easily extended to the recent 2.0 and 2.1 versions.

5. ACKNOWLEDGMENTS
This work was partially supported with public funds by the FEDER/MCINN National Project No. TEC2009-13763-C02-01.

6. REFERENCES