TCP Performance in Bluetooth Piconets

Jelena Mišić, Vojislav B. Mišić and Ka Lok Chan
Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada and
The Hong Kong University of Science and Technology, Hong Kong, China

Recent updates of the Bluetooth specification have introduced significant changes in the Bluetooth protocol stack, including optional flow control. When the Bluetooth piconet is used to carry TCP traffic, complex interactions between TCP congestion control mechanisms and data link layer controls of Bluetooth will occur. In this work, we model the performance of the piconet with TCP traffic, expressed through segment loss probability, round-trip time, and goodput, through both probabilistic analysis and discrete-event simulations. We show that satisfactory performance for TCP traffic may be obtained through proper dimensioning of the Bluetooth architecture parameters.

Bluetooth, Piconet, TCP protocol, Token bucket, Probabilistic analysis

I. INTRODUCTION

Bluetooth is an emerging standard for Wireless Personal Area Networks (WPANs)[2], originally intended as a simple cable replacement. Most performance analyses of data traffic in Bluetooth were focusing on scheduling techniques, and a number of proposals have been made[3–6], usually under the assumption that the traffic consists of individual UDP datagrams, which is unrealistic. To the best of our knowledge, none of these proposals offers any quality of service (QoS) guarantees; only recently a polling scheme that can support negotiated delay bounds (but at the expense of efficiency) has been proposed[7].

However, the number of possible uses of Bluetooth has been steadily growing to include various networking tasks between computers and computer-controlled devices such as PDAs, mobile phones, smart peripherals, and the like. As a consequence, the majority of the traffic over Bluetooth networks will belong to different flavors of the ubiquitous TCP/IP family of protocols. In order to cater to such applications, the recently adopted version 1.2 of the Bluetooth specification allows each L2CAP channel to operate in Basic L2CAP mode (essentially the L2CAP mode supported by the previous version of the specification[8]), Flow Control mode, or Retransmission mode[9]. All three modes offer segmentation, but only the latter two control buffering through protocol data unit (PDU) sequence numbers, and control the flow rate by limiting the required buffer space. Additionally, the Retransmission Mode uses a go-back-n repeat mechanism with an appropriate timer to retransmit missing or damaged PDUs as necessary. The architectural blocks of the L2CAP layer are schematically shown in Fig. I.

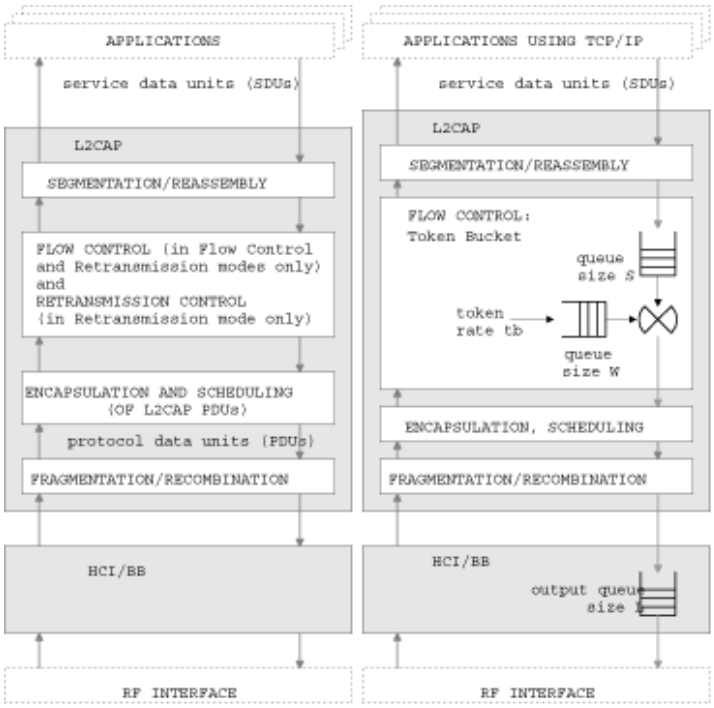

FIG. 1: Architectural blocks of the Bluetooth L2CAP layer (control paths not shown for clarity)((a) Original specification from[9]. (b) Implementation of the Flow Control mode.).

It is clear that complex interactions between the TCP congestion control and the L2CAP flow control and baseband scheduling mechanisms may be expected in Bluetooth networks carrying TCP traffic. In this chapter, we will investigate those interactions, both analytically and through simulations. We will model the segment loss probability, probability distribution of TCP round trip time (including the L2CAP round trip time) and TCP sending rate. We also discuss the dimensioning of various parameters of the architecture with regards to the tradeoff between end-to-end delay and achievable throughput under multiple TCP connections from different slave devices.

The chapter is organized as follows. Section II describes the system model and the basic assumptions about the piconet operation under TCP traffic, and discusses some recent related work in the area of performance modeling and analysis of TCP traffic. Section III discusses the segment loss probability and the probability distribution of the congestion window size, and presents the analytical results. Section IV presents simulation results for the performance of piconet with TCP connections running on slaves. Section VI concludes the chapter. Detailed derivations of the probability density functions for the TCP segment and acknowledgment delay, as well as the blocking probability through the token bucket filter and the output queue serviced by the E-limited scheduler, are presented in Appendices A and B.

II. SYSTEM MODEL AND RELATED WORK

A. The System Model

Bluetooth devices are organized in small centralized networks, or piconets, with $\mu \leq 8$ active nodes or devices[9–11]. One of the nodes acts as the master, while the others are slaves. All communications in the piconet take place under master's control: a slave can talk only when addressed by the master, and only immediately after being addressed by the master. As slaves can only talk to the master, hence all communications in the piconet, including slave-to-slave ones, must be routed through the master.

We consider a piconet in which the slaves create TCP connections with each other, so that each TCP connection will traverse two hops in the network. There are no TCP connections starting or ending at the master. We focus on TCP Reno[?] , which is probably the most widely used variant of TCP as of now. We assume that the application layer at slave i (where $i = 1..\mu-1$) sends messages of 1460 bytes at a rate of λ_i. This message will be sent within a single TCP segment, provided the TCP congestion window is not full. The TCP segment will be encapsulated in an IP packet with the appropriate header; this packet is then passed on to the L2CAP layer. We assume that the segmentation algorithm produces the minimum number of Bluetooth baseband packets[12], i.e., four DH5 packets and one DH3 packet per TCP segment[11]. We also assume that each TCP segment will be acknowledged with a single, empty TCP segment carrying only the TCP ACK bit; such acknowledgment requires one DH3 baseband packet. Therefore, the total throughput per piconet can reach a theoretical maximum of $0.5\frac{1460\cdot 8}{(4\cdot 5+2\cdot 3)\cdot 625\mu s} \approx 360$ kbps. As we consider the case with $\mu - 1 \leq 7$ slaves with identical traffic, each slave can achieve a goodput of only about $360/i$ kbps.

Our analysis setup implements the L2CAP Flow Control mode through a token bucket[13], with the data queue of size S and a token queue of size W wherein tokens arrive at a constant rate of tb. Furthermore, there is an outgoing queue for baseband data packets of size L, from which the data packets are serviced by the scheduler using the chosen intra-piconet scheduling policy. This implementation is shown in Fig. I. Note that the devices acting as slaves have one outgoing, or uplink, queue, whilst the device operating as the piconet master will maintain several downlink queues, one per each active slave, in parallel.

The parameters of this architecture, namely the sizes of the queues and the token rate in the token bucket, may be adjusted in order to achieve delay-throughput trade-off for each slave. This scheme can limit the throughput of the slave through the token rate, while simultaneously limiting the length of the burst of baseband packets submitted to the network. Depending on the token buffer size and traffic intensity, overflows of the token buffer can be detected through the TCP loss events such as three duplicate acknowledgments or time-outs. In the former case, the size of the congestion window will be halved and TCP will continue working in its Additive Increase-Multiplicative Decrease (AIMD) phase. In the latter case, the congestion window will shrink to one, and TCP will enter its slow-start routine.

B. Related Work

The performance of TCP traffic, in particular the steady-state send rate of the bulk-transfer TCP flows, has recently been assessed as the function of segment loss rate and round trip time (RTT)[14]. The system is modeled at the ends of rounds that are equal to the round trip times and during which a number of segments equal to the current size of the congestion window is sent. The model assumes that both the RTT and loss probability can be obtained by measurement, and derives average value of congestion window size.

The same basic model, but with improved accuracy with regard to the latency and steady state transmission rate for TCP Tahoe, Reno, and SACK variants, has been used in[15]. The authors have also studied the impact of correlated segment losses under three TCP versions.

In our approach, we will model the system at the moments of acknowledgement arrivals, instead of at the ends of successive rounds as in both papers mentioned above. In this manner, we are able to obtain more accurate information about performance, and to derive the TCP congestion window size and throughput in both non-saturated and saturated operating regimes. In addition, the blocking probabilities of all the queues along the path are known and each packet loss can be treated independently.

We note that both of the papers mentioned above consider RTT as a constant due to the large number of hops, whereas in our work there are two hops only, and RTT must be modeled as a random variable which is dependent on the current congestion window size.

Recently, the Adaptive Increase-Multiplicative Decrease (AIMD) technique was applied to control the flow over the combination of wireless and wireline path, by using the concept similar to the token bucket filter[16]. It is assumed that packets can be lost over the wireless link only, and that the all packet transmission times take exactly one slot. The system is modeled at the ends of the packet transmission times. This approach is orthogonal to ours, since we assume that the wireless channel errors will be handled through Forward Error Correction (FEC), and focus on finite queue losses at the data-link layer instead.

C. On the Choice of Intra-Piconet Scheduling Scheme

Of course, the performance of Bluetooth networks is also affected by the intra-piconet scheduling policy. As the master cannot know the status of all slaves' queues at any time, simpler policies, such as limited or exhaustive service, are to be preferred[3]. Both limited and exhaustive service are limiting cases of a family of schemes known as E-limited service[17]. In this polling scheme, the master and a slave exchange up to M packets, or less if both outgoing queues are emptied, before the master moves on to the next slave. (Limiting cases of $M = 1$ and ∞ correspond to limited and exhaustive service policies, respectively.) Our previous work indicates that the E-limited service offers better performance than either limited or exhaustive service[18]. E-limited service does not require that the master knows the status of slaves' uplink queues, and does not waste bandwidth when there are no data packets to exchange. Furthermore, it provides fairness by default, since the limit on the number of frames exchanged with any single slave prevents the slaves from monopolizing the network. Of course, the TCP protocol itself provides mechanisms to regulate fairness among TCP connections from one slave, but the bandwidth that can be achieved in Bluetooth networks is too small to warrant a detailed analysis in this direction.

III. ANALYSIS OF THE TCP WINDOW SIZE

A. TCP traffic in a Bluetooth piconet

Under the assumptions outlined above, the probability generating function (PGF) for the burst size of data packets at the baseband layer is $G_{bd}(z) = z^5$. The corresponding PGF for the acknowledgment packet burst size is $G_{ba}(z) = z$. The PGF for the size distribution of baseband data packets is $G_{pd}(z) = 0.8z^5 + 0.2z^3$, while the PGF for the size distribution of acknowledgment packets is $G_{pa}(z) = z^3$. Therefore, the PGF for the packet size as $G_p(z) = 0.5G_{bd}(G_{pd}(z)) + 0.5G_{ba}(G_{pa}(z))$, and the mean packet size as $\overline{L_{ad}} = 0.5G'_{bd}(G_{pd}(z))_{|z=1} + 0.5G'_{ba}(G_{pa}(z))_{|z=1} = 3.8$. (All time vari-

ables are expressed in units of time slots of the Bluetooth clock, $T = 625\mu s$.) We will also assume that the receiver advertised window is larger than the congestion window at all times.

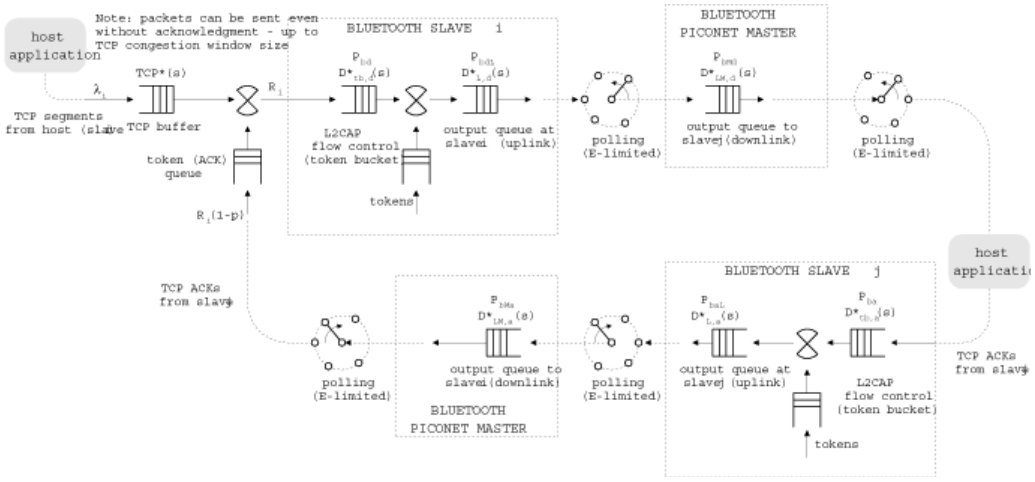

FIG. 2: The path of the TCP segment and its acknowledgment, together with the blocking probabilities and LSTs of the delays in respective queues.

The paths traversed by the TCP segment sent from slave i to slave j and the corresponding acknowledgment sent in the opposite direction, are shown schematically in Fig. 2. A TCP segment or acknowledgment can be lost if any of the buffers along the path is full (and, consequently, blocks the reception of the packet in question). In Appendices A and B we have calculated the probability distributions of token bucket queue lengths and outgoing buffer queue lengths at arbitrary times, as well as the corresponding blocking probabilities for TCP segments (P_{Bd}, P_{BdL}) and TCP acknowledgments (P_{Ba}, P_{BaL}). Segments/acknowledgments are also passing through the outgoing downlink queue at the master. However, we assume that this queue is much longer than the corresponding queues at the slaves, and the blocking probabilities P_{BMd}, P_{BMa} are much smaller and may safely be ignored in calculations. Then, the total probability p of losing a TCP segment or its acknowledgment is

$$p = 1 - (1 - P_{Bd})(1 - P_{BdL})(1 - P_{BMd})(1 - P_{Ba})(1 - P_{BaL})(1 - P_{BMa}) \tag{1}$$

We will model the length of the TCP window at the moments of acknowledgments arrivals. Since TCP window of size w grows by $1/w$ after the successful acknowledgment, it is not possible to model the system using Probability Generating Functions – we have to find the probability distribution directly. This probability distribution is a hybrid function represented by the mass probability w_1 of window size being 1, and by the continuous probability density function $w(x)$ for window sizes from 2 to ∞. We also need to determine the probability distribution of the congestion window threshold $t(x)$ at the moments of acknowledgements arrivals. We will represent the probability of the time-out event as $P_{to} = p(1 - (1 - p)^3)$, and the probability of loss by three duplicate acknowledgements as

$P_{td} = p(1-p)^3$. These probability distributions can be described by following equations:

$$\begin{aligned} w_1 &= P_{to} + P_{td} \int_{x=2}^{3} w(x)dx, \text{ for } w = 1 \\ t(x) &= t(x)(1-p) + w(2x)p, \text{ for } w \geq 2 \\ w(x) &= \left(w(x) - w'(x)\frac{1}{x} \right)(1-p) \int_0^{x-1/x} t(y)dy \\ &\quad + w(x/2)(1-p) \int_{0.5x}^{\infty} t(y)dy + w(2x)P_{td} \end{aligned} \tag{2}$$

where $\int_0^{x-1/x} t(y)dy$ denotes the probability that the current congestion window size $x-1/x$ is above the threshold size (i.e., that TCP is working in the AIMD mode), and the probability that the current size $0.5x$ of the congestion window is lower than the threshold (i.e., that the system is in the slow start mode) is given by $\int_{0.5x}^{\infty} t(y)dy$.

The system (2) of integro-differential equations could be solved numerically, but it was found that sufficient accuracy may be obtained through the following approximation:

$$w(x) = C_1 e^{-0.25px^2(1+3p-3p^2+p^3)/(1-p)} \tag{3}$$

where the normalization constant C_1 is determined from the condition $1 - w_1 = \int_2^{\infty} w(x)dx$. The probability density function of window size for various values of TCP window size and segment loss probability p is shown in Fig. 3.

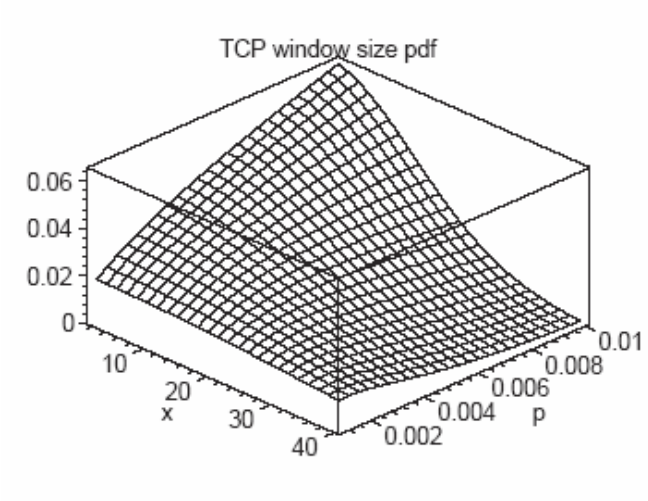

FIG. 3: Probability density function of TCP window size versus window size and segment loss probability.

The mean TCP window size $\overline{w}$ and mean threshold size $\bar{t}$ are calculated as

$$\begin{aligned} \overline{w} &= w_1 + \int_2^{\infty} xw(x)dx \\ \bar{t} &= \int_1^{\infty} xw(2x)dx \end{aligned}$$

since $t(x) = w(2x)$. The dependency of mean window size on TCP window size and segment loss probability p is shown in Fig. 4.

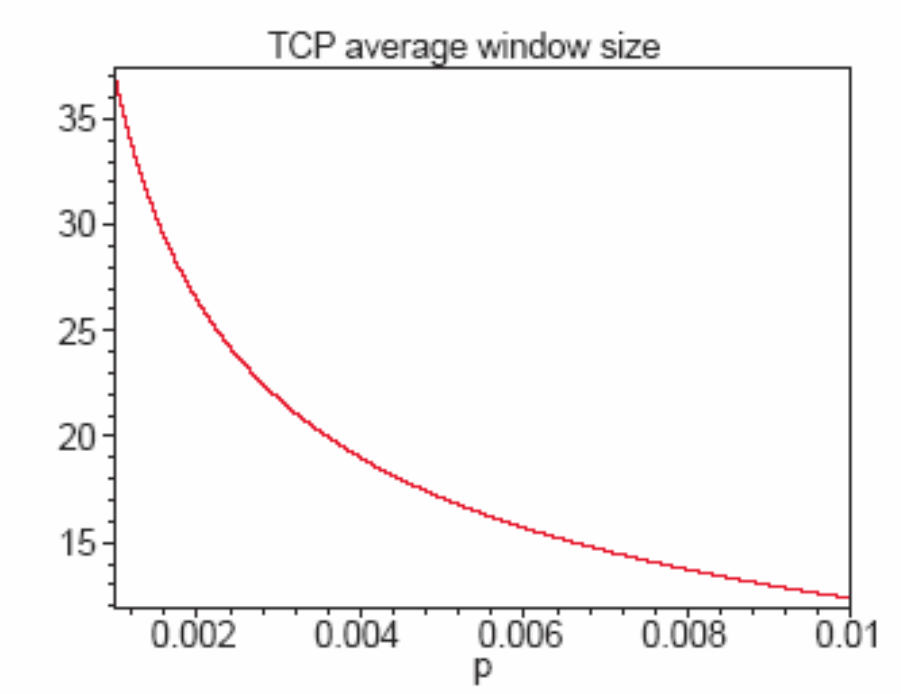

FIG. 4: Mean value of the TCP window size versus the segment loss probability.

B. TCP send rate and round-trip time estimation

As shown in Appendices A and B, we can find the probability distribution of delays through all the token bucket filters and the baseband buffers along the path of the TCP segment and its acknowledgment. The delay is calculated from the moment when the TCP segment enters the token bucket filter at the source device until the acknowledgment is received by that same device. This delay is equal to the sum of delays in all the buffers from Bluetooth protocol stack along the path. We will refer to this delay as the L2CAP round trip delay, and its probability distribution can be described through the corresponding Laplace-Stieltjes transform:

$$D^*_{L2CAP}(s) = D^*_{tb,d}(s)D^*_{L,d}(s)D^*_{LM,d}(s)D^*_{tb,a}(s)D^*_{L,a}(s)D^*_{LM,a}(s) \tag{4}$$

We can also calculate the probability distribution of the number of outstanding (unacknowledged) segments at the moments of segment acknowledgement arrivals. This number for an arbitrary TCP segment is equal to the number of segment arrivals during the L2CAP round trip time of that segment. The PGF for the number of segment arrivals during the L2CAP round trip time can be calculated, using the approach from[17], as

$$A(z) = D^*_{L2CAP}(\lambda_i - z\lambda_i).$$

The probability of k segment arrivals during the RTT time is equal to

$$a_k = \frac{d^k}{dz^k} D^*_{L2CAP}(\lambda_i - z\lambda_i)_{|z=0}.$$

Using the PASTA property (Poisson Arrivals See Time Averages), we conclude that arriving segment will see the same probability distribution of outstanding segments as the incoming acknowledgement. Therefore, the probability P_t that the arriving segment will find a free token and leave the TCP buffer immediately, is equal to:

$$P_t = \sum_{k=1}^{\infty} a_k \int_{k+1}^{\infty} w(x)dx \tag{5}$$

and the probability that the segments will be stored in the TCP buffer is $1 - P_t$. Then, the rate at which TCP sends the segments to the token bucket filter, which will be referred to as the TCP send rate R_i, has two components. One of these is contributed by the segments that find acknowledgments waiting in the token queue, and thus can leave the TCP buffer immediately; another one comes from the segments that have to wait in the TCP buffer until an acknowledgement (for an earlier segment) arrives. The expression for TCP send rate, then, becomes $R_i = P_t\lambda_i + R_i(1-p)(1-P_t)$, which may be simplified to

$$R_i = \lambda_i \frac{P_t}{P_t + p - pP_t} \tag{6}$$

As both components of the previous expression can be modeled as Poisson processes, we argue that the TCP sending process can still be modeled as Poisson process.

In order to calculate the delay through the TCP buffer, we need to determine the probability distribution of the number of segments buffered by the TCP due to the insufficient size of the congestion window. (We assume that the TCP buffer has infinite capacity.) The probability that k segments are buffered is

$$q_k = \sum_{i=1}^{\infty} a_{i+k} \int_{i}^{i+1} w(x)dx + a_{k+1}w_1 \tag{7}$$

and the LST for the delay through the TCP buffer is

$$D^*_{TCP}(s) = \sum_{k=0}^{\infty} q_k e^{-sk} \tag{8}$$

The entire round trip time of the TCP segment, then, becomes

$$RTT^*(s) = D^*_{TCP}(s) D^*_{L2CAP}(s) \tag{9}$$

IV. SIMULATION RESULTS FOR TCP PERFORMANCE IN BLUETOOTH PICONET

We first consider a piconet with two slaves only, having two simultaneous (but independent) TCP connections: slave 1 to slave 2, and slave 2 to slave 1. In the first set of experiments we have varied token buffer queue size S and offered load per connection, while other parameters were fixed:

1. The value of the scheduling parameter was $M = 5$, so that the entire TCP segment can be transferred in one piconet cycle.

2. The token rate was fixed to $tb = 250kbps$.
3. The token buffer capacity was $W = 3$KB.
4. The output rate of the token bucket queue was set to $max\_rate = 1Mbps$.
5. The outgoing (uplink) queue size was $L = 20$.

The values of TCP goodput and RTT obtained through simulation, using the ns-2 network simulator[19] with Bluehoc extension[20], are shown in Fig. 5. The size of token bucket buffer varied from 5 to 25 baseband packets, and the offered load varied from 80 to 240kbps.

The topmost row of diagrams show the round-trip time RTT and goodput. We observe that only for $S = 25$ the goodput reaches the physical limit (for the given segment size) of approximately 180kbps per connection. At the same time, round trip delay reaches $400ms$.

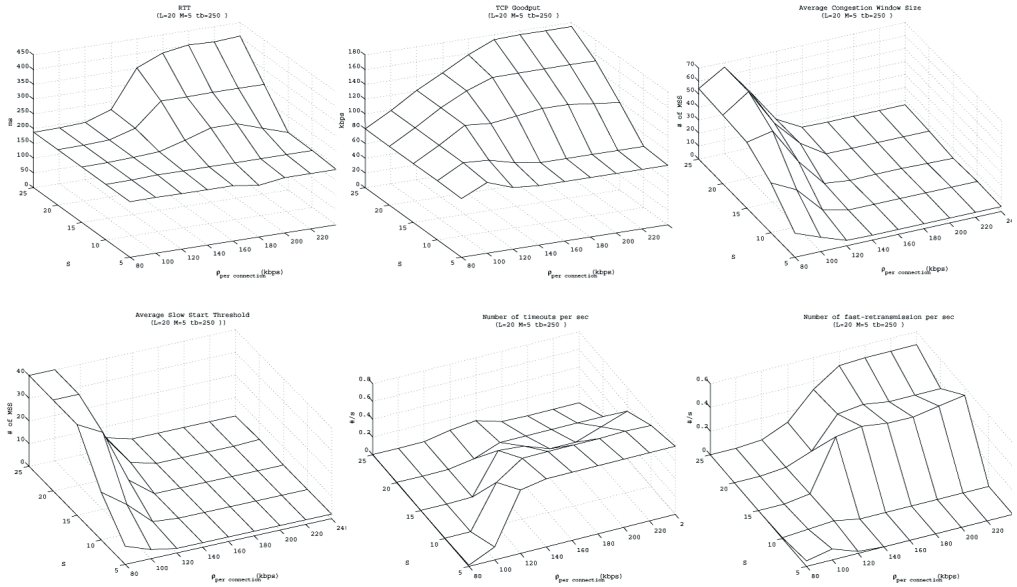

FIG. 5: TCP performance as the function of the buffer size S, in the piconet with two slaves. ((a) Round-trip time (RTT). (b) Goodput. (c) Mean size of the congestion window. (d) Mean value of the slow start threshold. (e) Time-out rate. (f) Fast retransmission rate.)

As can be seen from the two diagrams in the middle row, the mean congestion window size grows with S, which is expected since larger S means lower buffer loss. Furthermore, the congestion window grows with the segment arrival rate under very low loads, since there are not enough packets to expand the window size. For moderate and high offered loads, the average window size experiences a sharp decrease with the offered load. This is consistent with analytical results, since the packet loss probability is directly proportional to the offered load.

Finally, the bottommost row of Fig. 5 shows the rate of TCP timeouts and fast retransmissions as functions of offered load and buffer size S.

The dependencies shown in Fig. 5 hint that the value of $S = 25$ leads to maximum achievable throughput and negligible time-out rate for the offered load equal to the maximum achievable goodput (180kbps).

The next set of experiments considered TCP performance as a function of the scheduling parameter M, with constant values of $S = 25$ and $L = 20$. The resulting diagrams are shown in Fig. 6. We observe that the value of $M = 5$, which is sufficient to carry a TCP segment of five baseband packets, gives maximum goodput, minimum timeout rate and maximum fast-retransmission rate, when compared to larger values of M. Therefore, the minimal value of M which is sufficient to transfer a TCP segment in one piconet cycle appears also to be optimal with respect to goodput and other measures of performance.

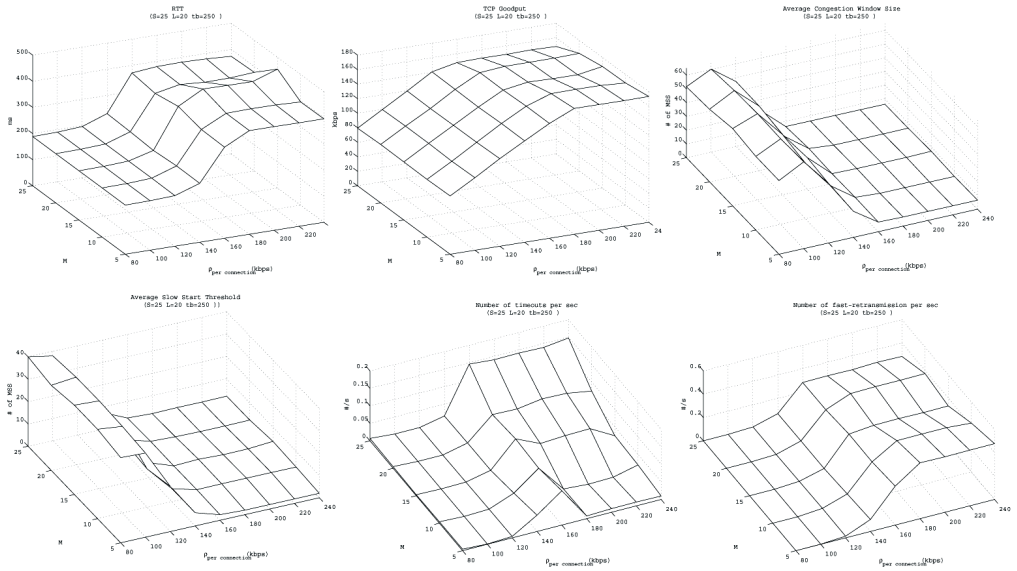

FIG. 6: TCP performance as the function of the scheduling parameter M, in the piconet with two slaves. ((a) Round-trip time (RTT). (b) Goodput. (c) Mean size of the congestion window. (d) Mean value of the slow start threshold. (e) Time-out rate. (f) Fast retransmission rate.)

We have also investigated TCP behavior with varying offered load and varying token rate; the corresponding diagrams are shown in Fig. 7. We observe that increasing the token rate over the maximal achievable goodput per connection can result only in marginal increase of mean congestion window size and goodput, despite the fact that the blocking probability at the token bucket filter is decreased. However, the blocking probability at the outgoing buffer at the baseband will still increase, and this increase leads to increased time-out rate and increased overall segment loss probability. Therefore, the token rate should not be set to the value much larger than the physical throughput limit per slave.

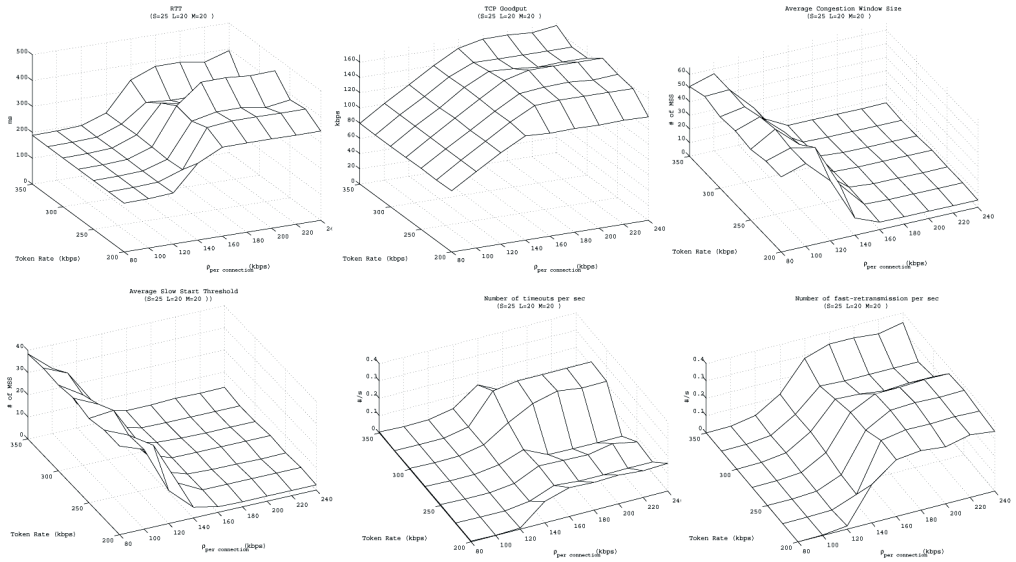

FIG. 7: TCP performance as the function of token rate, in the piconet with two slaves. ((a) Round-trip time (RTT). (b) Goodput. (c) mean size of the congestion window. (d) Mean value of the slow start threshold. (e) Time-out rate. (f) Fast retransmission rate.)

V. TCP PERFORMANCE IN THE PICONET WITH SEVEN SLAVES

We have conducted a similar set of experiments in a piconet with seven active slaves. In this case, each slave i, $i = 1..7$ creates a TCP connection with another slave $j = (i+1) \bmod 7$, giving rise to a total of seven identical TCP connections. Consequently, the maximum goodput per slave is limited to $\frac{360}{7} \approx 50$ kbps.

The performance of TCP traffic, when the token buffer size varies from 5 to 25 baseband packets and the offered load varies 50% around the physical goodput limit, is shown in Fig. 8. Again, the value of $S = 25$ gives the goodput which is close to the limit under high load, as well as a low time-out rate. The shape of the time-out rate surface can be explained by the fact that packets do not arrive too frequently under low offered loads, and time-outs occur before three new packets are generated to provoke three duplicated acknowledgments.

Finally, Fig. 9 shows TCP performance with seven slaves under varying value of the scheduling parameter M and the token rate. We observe that this behavior is similar to that in the case of two slaves, although the optimality of the value $M = 5$ is much less pronounced. We again note that good performance is obtained if the token rate does not exceed about 50% of the maximum physical goodput.

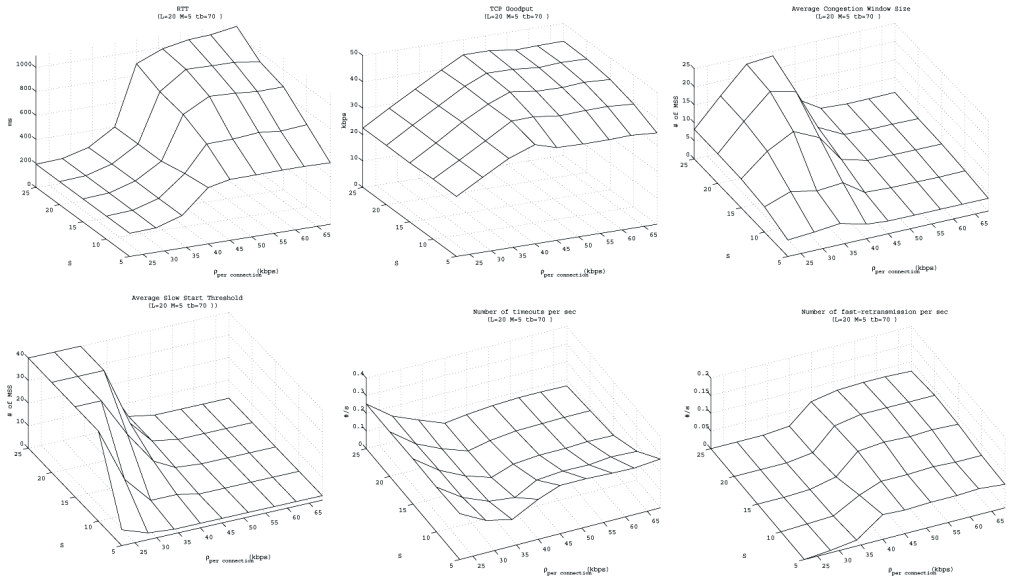

FIG. 8: TCP performance as the function of the buffer size S, in the piconet with seven slaves. ((a) Round-trip time (RTT). (b) Goodput. (c) Mean size of the congestion window. (d) Mean value of the slow start threshold. (e) Time-out rate. (f) Fast retransmission rate.)

VI. CONCLUSION

We have analyzed the performance of the Bluetooth piconet carrying TCP traffic, assuming TCP Reno is used. We have analytically modeled the segment loss probability, TCP send rate, and the probability density functions of the congestion window size and the round trip time. We have investigated the impact of token bucket buffer size, token rate, outgoing baseband buffer size, and scheduling parameter on the goodput, round-trip time, congestion window size, time-out rate and fast-retransmission rate. Our results show that buffer sizes around 25 baseband packets are sufficient for achieving maximal goodput. Scheduling should be confined to the E-limited policy with scheduling parameter being equal to the number of baseband packets in the TCP segment. Token rate should be set to value only around 50% larger than the portion of the total piconet throughput dedicated to particular slave.

APPENDIX A: QUEUEING ANALYSIS OF TOKEN BUCKET FILTER UNDER TCP TRAFFIC

We consider the token bucket (TB) filter, the queueing model of which is shown in Fig. 10. We assume that the TCP segments arrive at the TCP sending rate R_i calculated in (6), while the TCP acknowledgements arrive at the rate of $R_i(1-p)$. The token arrival rate will be denoted as tb, which means that the token arrival period will be $T_b = 1/tb$. The queue which holds tokens has length of W baseband packet tokens. Packets leave the queue at *max_rate*, which is much larger than the

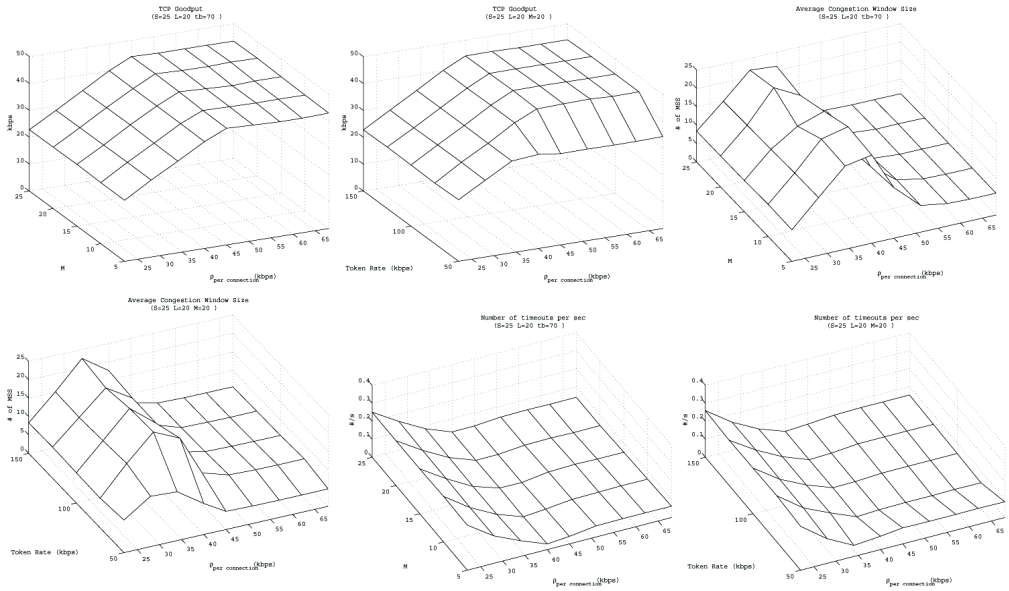

FIG. 9: TCP performance in the piconet with seven slaves. ((a) Goodput as the function of the scheduling parameter M. (b) Goodput as the function of token rate. (c) Mean size of the congestion window as the function of the scheduling parameter M. (d) Mean size of the congestion window as the function of token rate. (e) Time-out rate as the function of the scheduling parameter M.(f) Time-out rate as the function of token-rate.)

token arrival rate. Using this model, we first derive the probability distribution function (PDF) of the number of the packets in the token bucket queue at the moments of token arrival, and then proceed to calculate that same PDF at arbitrary time.

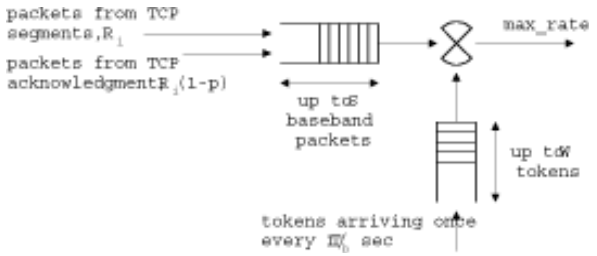

FIG. 10: Queueing model of the token bucket with finite capacity, accepting two types of packets.

The probability of a_k baseband packet arrivals in the TB queue is equal to the sum of probabilities

of data and acknowledgment packet arrivals:

$$\begin{aligned} a_k &= \sum_{i=1}^{k} \left[\frac{1}{i!} \frac{d^i}{dz^i} \left(e^{-R_i T_b (1-G_{bd}(z))} \right)_{|z=0} \right. \\ &+ \left. \frac{1}{(k-i)!} \frac{d^{k-i}}{dz^{k-i}} \left(e^{-R_i T_b (1-G_{ba}(z))} \right)_{|z=0} \right] \end{aligned} \tag{A1}$$

The TB can be modeled as a discrete-time Markov chain, in which the state i (when $0 \leq i \leq W$) corresponds to the situation when $W-i$ tokens are available in the token buffer, but no data packets are present in the data queue. The remaining states from $W+1$ to $W+S$ correspond to the situation where there are data packets in the data queue, but the token queue is empty. Since both queues have finite lengths, the Markov chain, which is shown in Fig. 11, is finite as well. The balance equations for this Markov chain are

$$\begin{aligned} \pi_0 &= a_0 \pi_1 + (a_0 + a_1) \pi_0 \\ \pi_i &= \sum_{j=0}^{i+1} a_{i-j+1} \pi_j, \text{, for } 0 < i < W+S-1 \\ \pi_{W+S-1} &= \textstyle\sum_{j=0}^{S-1} \pi_j \sum_{k=S-j}^{\infty} a_k \end{aligned} \tag{A2}$$

The PDF for the queue lengths at the moments of token arrivals can be found by solving this system with the condition $\sum_{k=0}^{S-1} \pi_k = 1$.

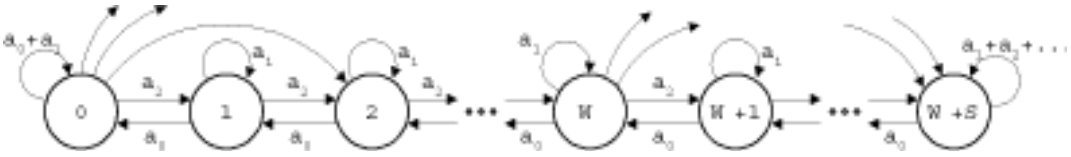

FIG. 11: Token bucket may be represented as a discrete Markov chain.

1. Analysis of the TB queue length at arbitrary times and the derivation of blocking probability

By using the probability distribution of TB queue length at moments of token arrivals, we can derive the joint probability distribution of TB queue length and the remaining time before the token arrival. This will help us to obtain the blocking probability which is important since it determines the segment loss probability at the TCP level. We will introduce the following variables:

- The total queue length (including both token queue and data queue), L_q.
- The elapsed token time – from a given token arrival to the arbitrary time before the arrival of the next token, $T_{b\_}$.
- The remaining token time – from the arbitrary time between two successive token arrivals till the next token arrival, T_{b+}.
- The number of packet arrivals (results of burst arrivals) during the elapsed token time, $A(T_{b\_})$.

- The blocking probability at arbitrary time, P_B. Since the burst represents a TCP segment, we will adopt the total rejection policy for calculating the blocking probability. In other words, if there is not enough room for all baseband packets which belong to the burst (TCP segment), the entire burst will be rejected.
- Finally, the probability distribution function $T_b(x)$ of the token inter-arrival time, and the corresponding probability density function $t_b(x)$.

For the time between two successive token arrivals, the joint probability distribution of the TB queue length and remaining token time is

$$\Pi_k^* = \int_0^\infty e^{-sy}\text{Prob}[L_q = k, y < T_{b+} < y+dy], \quad 1 \le k \le W+S \tag{A3}$$

By using the TB queue length distribution at the time of arrival of the previous token, we obtain

$$\begin{aligned}\Pi_k^*(s) &= R_i(\overline{G_{bd}}+\overline{G_{ba}})T_b(1-P_B) \\ &\quad \cdot \sum_{j=0}^{k} \pi_j E[e^{-sT_{b+}}|_{A(T_{b\_})=k-j}]\text{Prob}[A(T_{b\_})=k-j], \text{ for } 1 \le k < W+S\end{aligned} \tag{A4}$$

$$\Pi_{W+S}^*(s) = \sum_{j=0}^{W+S-1} \pi_j \sum_{k=S-j}^{\infty} E[e^{-sT_{b+}}|_{A(T_{b\_})=k-j}]\text{Prob}[A(T_{b\_})=k-j] \tag{A5}$$

The system (A4) and (A5) can be simplified using the following expression:

$$\begin{aligned}\psi_k^*(s) &= \sum_{l=0}^{\infty}\sum_{i=1}^{k}\left(\frac{1}{i!}\frac{d^i}{dz^i}G_{bd}(z)^l + \frac{1}{(k-i)!}\frac{d^{k-i}}{dz^{k-i}}G_{ba}(z)^l\right)_{|z=0} \\ &\quad \cdot \int_0^\infty \frac{(R_i x)^l}{l!}e^{-R_i x}\frac{1-T_b(x)}{T_b}dx \int_0^\infty e^{-sy}\frac{t_b(x+y)}{1-T_b(x)}dy\end{aligned} \tag{A6}$$

Expression (A6) can be further simplified to

$$\begin{aligned}\psi_k^*(s) &= \frac{1}{T_b}\sum_{i=1}^{k}\left(\frac{1}{i!}\frac{d^i}{dz^i}_{|z=0}\frac{e^{(R_iG_{bd}(z)-R_i)T_b}-e^{-sTb}}{R_iG_{bd}(z)+s-R_i}\right. \\ &\quad \left. + \frac{1}{(k-i)!}\frac{d^{k-i}}{dz^{k-i}}_{|z=0}\frac{e^{(R_iG_{pa}(z)-R_i)T_b}-e^{-sT_b}}{R_iG_{pa}(z)+s-R_i}\right)\end{aligned} \tag{A7}$$

Now the system (A4), (A5) can be written as:

$$\begin{aligned}\Pi_k^*(s) &= R_i(\overline{G_{bd}}+\overline{G_{ba}})T_b(1-P_B)\sum_{j=0}^{k}\pi_j\psi_{k-j}^*, \text{ for } 1 \le k < W+S \\ \Pi_{W+S}^*(s) &= R_i(\overline{G_{bd}}+\overline{G_{ba}})T_b(1-P_B)\sum_{j=0}^{W+S-1}\pi_j\sum_{k=S-j}^{\infty}\psi_{k-j}^*\end{aligned} \tag{A8}$$

The probabilities that the TB filter has exactly k packets can be simply expressed as $P_k = \text{Prob}[L_q = k] = \Pi_k^*(0)$. Also, we note that the probability of the empty queue is equal to $P_0 = 1-\lambda(\overline{G_{bd}}+\overline{G_{ba}})T_b(1-P_B)$. It should be observed that all queue state probabilities at arbitrary time

are functions of P_B, therefore we need to express the blocking probability as the function of the queue state probabilities, and then solve this equation for P_B. The blocking probability under two types of packet bursts in the TB filter is

$$P_B = \sum_{k=0}^{W+S-1} P_k \sum_{j=W+S-k+1}^{\infty} 0.5(g_{bd,j} + g_{ba,j}) + P_{W+S} \tag{A9}$$

where $g_{bd,j} = \frac{1}{j!}\frac{d^j}{dz^j} G_{bd}(z)_{|z=0}$ and $g_{ba,j} = \frac{1}{j!}\frac{d^j}{dz^j} G_{ba}(z)_{|z=0}$ are mass probabilities of the burst size probability distribution for the TCP data segment and the TCP acknowledgment segment, respectively. Expression (A9) can be rearranged to find the blocking probability P_B, which leads to the queue length distribution. After that, the individual blocking probabilities for each traffic type can be found as

$$\begin{aligned} P_{Bd} &= \sum_{k=0}^{W+S-1} P_k \sum_{j=W+S-k+1}^{\infty} g_{bd,j} + P_{W+S} \\ P_{Ba} &= \sum_{k=0}^{W+S-1} P_k \sum_{j=W+S-k+1}^{\infty} g_{ba,j} + P_{W+S} \end{aligned} \tag{A10}$$

The delay through the token bucket filter should be calculated separately for the data segments and for the acknowledgments. The queueing delay of the entire TCP segment is equal to the queueing delay of the first baseband packet from the burst representing that TCP segment. The LST of the delay for the data segment is given with

$$D^*_{tb,d}(s) = \frac{\left(P_0 + \sum_{l=1}^{W} \Pi^*_k(0)\right) \sum_{k=1}^{W+S-l} g_{bd,k} + \sum_{k=W}^{W+S-1} \Pi^*_k(s) [G^*_{pd}]^{k-1} \sum_{j=1}^{W+S-k} g_{bd,j}}{1 - P_{Bd}} \tag{A11}$$

The LST for the delay of the acknowledgment, $D^*_{tb,d}(s)$, can be determined in an analogous manner.

APPENDIX B: ANALYSIS OF THE OUTGOING QUEUE AT THE BASEBAND LEVEL

We will now derive the expressions for the delay and blocking probability for the other queue (buffer): the outgoing buffer at the baseband level. This buffer has finite length of L baseband packets, and it is fed by packets that pass through the token buffer filter. We assume that the TCP segment packets arrive at the rate of $\lambda_{i,d} = R_i(1 - P_{Bd})$, and that the TCP acknowledgment packets arrive at the rate of $\lambda_{i,a} = R_i(1-p)(1-P_{Ba})$. We again assume that each buffer has the total rejection policy, i.e., the entire burst is rejected if it cannot fit into the buffer.

The baseband queue is serviced by the baseband scheduler using the E-limited policy. The master sends a downlink packet to the slave, and receives an uplink packet from it. Empty (POLL or NULL) packets are sent if there are no data packets in the corresponding outgoing queue. Under the E-limited policy, the master stays with the slave (and services its outgoing queue) for at most M packet are transmitted, or less it if both outgoing queues are empty. Therefore, the operation of the outgoing queue has to be analyzed using the theory of $M^{[x]}/G/1$ queues with vacations, where vacation corresponds to the time when the master is serving other slaves. We will first determine the probability distribution of slave queue lengths in imbedded Markov points that correspond to vacation termination times and uplink transmission completion times[17]. Then, we will determine the PDF for the slave queue length at arbitrary point of time, and use it to derive the access delay and the blocking probability of the burst.

1. Analysis of outgoing queue length distribution in Markov points

Let $q_{k,i,u}$ denote the joint probability that a Markov point in the uplink queue of slave i is a vacation termination time and that there are $k = 0,1,2\ldots$ packets at the outgoing queue of the slave i at that time. Also, let $\pi^{(m)}_{k,i,u}$ denote the joint probability that a Markov point in the uplink queue i is the m-th uplink transmission completion time and that there are k packets in the queue, where $m = 1\ldots M$ and $k = 0,1,2,\ldots$. The analogous probabilities for the corresponding downlink queue are denoted with $q_{k,i,d}$ and $\pi^{(m)}_{k,i,d}$.

Let $g_p(x)$ and $v_i(x)$ denote the probability density functions of the packet transmission time and vacation time, respectively, at the uplink queue of slave i; their LST transforms will be $G^*_p(s)$ and $V^*_i(s)$. The downlink packet transmission, immediately followed by the uplink transmission, will be denoted as a frame; the corresponding LST transform is $G^*_p(s)^2$. Let us also denote the probability of k packet arrivals at the uplink queue of slave i during the frame time as $a_{k,i,u}$, and the probability of k packet arrivals during the vacation time (i.e., while master is serving other slaves) as $f_{k,i,u}$. These probabilities may be calculated as

$$\begin{aligned}
a_{k,i,u} &= \sum_{i=1}^{k} \frac{1}{i!}\frac{d^i}{dz^i}\left(G^*_p(\lambda_{i,d}-\lambda_{i,d}G_{bd}(z))\right)^2\Big|_{z=0} \\
&\quad + \frac{1}{(k-i)!}\frac{d^{k-i}}{dz^{k-i}}\left(G^*_p(\lambda_{i,a}-\lambda_{i,a}G_{ba}(z))\right)^2\Big|_{z=0} \\
f_{k,i,u} &= \sum_{i=1}^{k} \frac{1}{i!}\frac{d^i}{dz^i}\left(V^*(\lambda_{i,d}-\lambda_{i,d}G_{bd}(z))\right)^2\Big|_{z=0} \\
&\quad + \frac{1}{(k-i)!}\frac{d^{k-i}}{dz^{k-i}}\left(V^*(\lambda_{i,a}-\lambda_{i,a}G_{ba}(z))\right)^2\Big|_{z=0}
\end{aligned} \tag{B1}$$

where $g_p * g_p(x)$ denotes the convolution of $g_p(x)$ with itself. Note that the expressions $(G^*_p(\lambda_{i,d} - \lambda_{i,d}G_{bd}(z)))^2$ and $G^*_p(\lambda_{i,a}-\lambda_{i,a}G_{ba}(z))$ denote the PGFs for the number of packet arrivals in the uplink queue from TCP data segments and acknowledgments, respectively, during the frame time. Also $(V^*(\lambda_{i,d}-\lambda_{i,d}G_{bd}(z)))^2$ and $V^*(\lambda_{i,a}-\lambda_{i,a}G_{ba}(z))$ denote the corresponding PGFs for the number of packet arrivals in the uplink queue, but during the vacation time.

For the packet departure times when the master polls the slaves, we note that the buffer occupancy can be between 0 and $L-1$. The probabilities that the uplink queue contains k packets in Markov points are given by

$$\begin{aligned}
\pi^{(1)}_{k,i,u} &= \sum_{j=1}^{k+1} q_{j,i,u}a_{k-j+1,i,u},\ 0 \le k \le L-2 \\
\pi^{(1)}_{L-1,i,u} &= \sum_{j=1}^{L} q_{j,i,u}\sum_{k=L-j}^{\infty} a_{k,i,u} \\
\pi^{(m)}_{k,i,u} &= \sum_{j=1}^{k+1} \pi^{(m-1)}_{j,i,u}a_{k-j+1,i,u},,\ 0 \le k \le L-2,\ m = 2\ldots M \\
\pi^{(m)}_{L-1,i,u} &= \sum_{j=1}^{L-1} \pi^{(m-1)}_{j,i,u}\sum_{k=L-j}^{\infty} a_{k,i,u},,\ m = 2\ldots M \\
q_{k,i,u} &= \left(\sum_{m=1}^{M-1}\pi^{(m)}_{0,i,u} + q_{0,i,u}\right) f_{k,i,u} + \sum_{j=0}^{k}\pi^{(M)}_{j,i,u}f_{k-j,i,u},\ 0 \le k < L \\
q_{K,i,u} &= \left(\sum_{m=1}^{M-1}\pi^{(m)}_{0,i,u} + q_{0,i,u}\right) \sum_{k=L}^{\infty} f_{k,i,u} + \sum_{j=0}^{L-1}\pi^{(M)}_{j,i,u}\sum_{k=L-j}^{\infty} f_{k,i,u}
\end{aligned} \tag{B2}$$

Also, $\sum_{k=0}^{L} q_{k,i,u} + \sum_{m=1}^{M} \sum_{k=0}^{L-1} \pi_{k,i,u} = 1$. The distribution of the queue length in Markov points may be obtained.

Let us denote the probability that the vacation starts after the uplink transmission as $h_{i,u} = \sum_{m=1}^{M-1} \pi_{0,i,u}^{m} + \sum_{k=0}^{L-1} \pi_{k,i,u}^{M}$. Then, the probability that the vacation will start after an arbitrary Markov point is $q_{0,i,u} + h_{i,u}$. The average distance in time between two consecutive Markov points at slave i is

$$\eta_{i,u} = (q_{0,i,u} + h_{i,u})\overline{V_{i,u}} + (1 - q_{0,i,u} + h_{i,u})2\overline{L_{ad}} \tag{B3}$$

2. Analysis of the outgoing queue length distribution at arbitrary time

By using the probability distribution of the uplink queue length in Markov points, we can derive the probability distribution of this queue length at arbitrary time between two Markov points, together with the PDF of the remaining vacation time (if the previous Markov point was the start of a vacation) or the PDF of the remaining frame service time (if the previous Markov point was the start of a packet service). We will introduce the following variables:

- The probability density function of the vacation time, $v_i(x)$, and its PDF, $V_i(x)$.
- The queue length at an arbitrary time, $L_{q,i,u}$.
- The elapsed vacation time, $V_{\_,i,u}$.
- The remaining vacation time, $V_{+,i,u}$.
- The number of packet arrivals resulting from packet burst arrivals in the elapsed vacation time, $A(V_{\_})$.
- The probability density function of the frame service time, $g_p * g_p(x)$, and its PDF, $F_s(x)$.
- The elapsed frame service time, $X_{\_,i,u}$.
- The remaining frame service time, $X_{+,i,u}$.
- The number of packet arrivals resulting from packet burst arrivals in the elapsed frame service time, $A(X_{\_})$.

For the time between the start and end of vacation, we define the joint probability of the queue length and the remaining vacation time as

$$\Omega_{k,i,u}^{*}(s) = \int_0^{\infty} e^{-sy} \text{Prob}[L_{q,i,u} = k, y < V_{+,i,u} < y + dy],\ 0 \le k \le L \tag{B4}$$

For the time between the start and end of the frame service, for the frame $1 \le m \le M$, we define the joint probability of the queue length and remaining frame service time as

$$\Pi_{k,m,i,u}^{*}(s) = \int_0^{\infty} e^{-sy} \text{Prob}[L_{q,i,u} = k, y < X_{+,i,u} < y + dy],\ 1 \le k \le K,\ 1 \le m \le M \tag{B5}$$

Then, by using the probabilities of the uplink queue state in the previous Markov point, we obtain

$$\begin{aligned}
\Omega^*_{k,i,u}(s) &= \frac{\overline{V_{i,u}}}{\eta_{i,u}}\big(q_{0,i,u}+\sum_{m=1}^{M-1}\pi^m_{0,i,u}\big)E[e^{-sV_{+,i,u}}|_{A(V_{_,i,u})=k}]\text{Prob}[A(V_{_,i,u})=k] \\
&\quad +\frac{\overline{V_{i,u}}}{\eta_{i,u}}\sum_{j=0}^{k}\pi^M_{j,i,u}E[e^{-sV_{+,i,u}}|_{A(V_{_,i,u})=k-j}]\text{Prob}[A(V_{_,i,u})=k-j], \\
&\qquad\qquad \text{for } 0\le k\le K-1 \\
\Omega^*_{L,i,u}(s) &= \frac{\overline{V_{i,u}}}{\eta_{i,u}}\big(q_{0,i,u}+\sum_{m=1}^{M-1}\pi^m_{0,i,u}\big)\sum_{k=L}^{\infty}E[e^{-sV_{+,i,u}}|_{A(V_{_,i,u})=k}]\text{Prob}[A(V_{_,i,u})=k] \\
&\quad +\frac{\overline{V_{i,u}}}{\eta_{i,u}}\sum_{j=0}^{k}\pi^M_{j,i,u}\sum_{k=L-j}^{\infty}E[e^{-sV_{+,i,u}}|_{A(V_{_,i,u})=k}]\text{Prob}[A(V_{_,i,u})=k]
\end{aligned}$$

$$\begin{aligned}
\Pi^*_{k,1,i,u}(s) &= \frac{2\overline{L}}{\eta_{i,u}}\sum_{j=1}^{k}q_{j,i,u}E[e^{-sX_{+,i,u}}|_{A(X_{_,i,u})=k-j}]\text{Prob}[A(X_{_,i,u})=k-j], \\
&\qquad\qquad \text{for } 1\le k\le L-1 \\
\Pi^*_{L,1,i,u}(s) &= \frac{2\overline{L}}{\eta_{i,u}}\sum_{j=1}^{L}q_{j,i,u}\sum_{k=K-j}^{\infty}E[e^{-sX_{+,i,u}}|_{A(X_{_,i,u})=k}]\text{Prob}[A(X_{_,i,u})=k], \\
&\qquad\qquad \text{for } 1\le k\le L-1 \\
\Pi^*_{k,m,i,u}(s) &= \frac{2\overline{L}}{\eta_{i,u}}\sum_{j=1}^{k}\pi_{j,i,u}E[e^{-sX_{+,i,u}}|_{A(X_{_,i,u})=k-j}]\text{Prob}[A(X_{_,i,u})=k-j], \\
&\qquad\qquad \text{for } 1\le k\le L-1,\ 2\le m\le M \\
\Pi^*_{L,m,i,u}(s) &= \frac{2\overline{L}}{\eta_{i,u}}\sum_{j=1}^{L}\pi^m_{j,i,u}\sum_{k=K-j}^{\infty}E[e^{-sX_{+,i,u}}|_{A(X_{_,i,u})=k}]\text{Prob}[A(X_{_,i,u})=k], \\
&\quad \text{for } 2\le m\le M
\end{aligned} \tag{B6}$$

The system of equations (B6) can be simplified using the following expressions:

$$\begin{aligned}
\phi_k(s) &= E[e^{-sV_{+,i,u}}|A(V_{_,i,u})=k]\text{Prob}[A(V_{_,i,u})=k] \\
&= \sum_{i=1}^{k}\Bigg[\frac{1}{\overline{V_i}i!}\frac{d^i}{dz^i}|_{z=0}\Big(\frac{V^*(-\lambda_i G_{bd}(z)+\lambda_{i,d})-V^*(s)}{\lambda G_{bd}(z)+s+\lambda_{i,d}}\Big) \\
&\quad +\frac{1}{\overline{V_i}(k-i)!}\frac{d^{k-i}}{dz^{k-i}}|_{z=0}\Big(\frac{V^*(-\lambda_i G_{ba}(z)+\lambda_{i,a})-V^*(s)}{\lambda G_{ba}(z)+s+\lambda_{i,a}}\Big)\Bigg]
\end{aligned}$$

$$\begin{aligned}
\psi^*_k(s) &= E[e^{-sX_{+,i,u}}|A(X_{_,i,u})=k]\text{Prob}[A(X_{_,i,u})=k] \\
&= \sum_{i=1}^{k}\Bigg[\frac{1}{2\overline{L}i!}\frac{d^i}{dz^i}|_{z=0}\Big(\frac{G^*_p(-\lambda_{i,d}G_{bd}(z)+\lambda_i)^2-G^*_p(s)^2}{\lambda G_{bd}(z)+s+\lambda_{i,d}}\Big) \\
&\quad +\frac{1}{2\overline{L}(k-i)!}\frac{d^{k-i}}{dz^{k-i}}|_{z=0}\Big(\frac{G^*_p(-\lambda_{i,a}G_{ba}(z)+\lambda_{i,a})^2-G^*_p(s)^2}{\lambda G_{ba}(z)+s+\lambda_{i,a}}\Big)\Bigg]
\end{aligned}$$

Then, (B6) and (B6) can be transformed to

$$\begin{aligned}
\Omega^*_k(s) &= \frac{\overline{V}}{\eta_{i,u}}\big(q_{0,i,u}+\sum_{m=1}^{M-1}\pi^m_{0,i,u}\big)\phi^*_k(s)+\frac{\overline{V_{i,u}}}{\eta_{i,u}}\sum_{j=0}^{k}\pi^M_{j,i,u}\phi^*_{k-j} \\
\Omega^*_{L,i,u}(s) &= \frac{\overline{V_{i,u}}}{\eta_{i,u}}\big(q_{0,i,u}+\sum_{m=1}^{M-1}\pi^m_{0,i,u}\big)\sum_{k=K}^{\infty}\phi^*_k(s) \\
&\quad +\frac{\overline{V_{i,u}}}{\eta_{i,u}}\sum_{j=0}^{k}\pi^M_{j,i,u}\sum_{k=K-j}^{\infty}\phi^*_k(s)
\end{aligned} \tag{B7}$$

$$\begin{aligned}
\Pi^*_{k,1,i,u}(s) &= \frac{2\overline{L_{ad}}}{\eta_{i,u}}\sum_{j=1}^{k}q_{j,i,u}\psi^*_k(s),\ 1\le k\le L-1 \\
\Pi^*_{L,1,i,u}(s) &= \frac{2\overline{L_{ad}}}{\eta_{i,u}}\sum_{j=1}^{L}q_{j,i,u}\sum_{k=K-j}^{\infty}\psi^*_k(s),\ 1\le k\le L-1 \\
\Pi^*_{k,m,i,u}(s) &= \frac{2\overline{L_{ad}}}{\eta_{i,u}}\sum_{j=1}^{k}\pi_{j,i,u}\psi^*_{k-j}(s),\ 1\le k\le L-1,\ 2\le m\le M \\
\Pi^*_{L,m,i,u}(s) &= \frac{2\overline{L_{ad}}}{\eta_{i,u}}\sum_{j=1}^{L}\pi^m_{j,i,u}\sum_{k=L-j}^{\infty}\psi^*_k(s),\ 1\le k\le L-1,\ 2\le m\le M
\end{aligned} \tag{B8}$$

The distribution of the size of the uplink queue at the slave, at arbitrary time, is given by

$$\begin{aligned}
\text{Prob}[L_{q,i,u}=0] &= \Omega^*_0=\frac{1}{\lambda_i\eta_{i,u}}\sum_{m=1}^{M}\pi^m_{0,i,u} \\
\text{Prob}[L_{q,i,u}=k] &= \Omega^*_k(0)+\sum_{m=1}^{M}\Pi^*_{k,m,i,u}(0) \\
&= \frac{1}{\lambda_i\eta_{i,u}}\sum_{m=1}^{M}\pi^m_{k,i,u},\ 1\le k\le K-1 \\
\text{Prob}[L_{q,i,u}=K] &= \Omega^*_L(0)+\sum_{m=1}^{M}\Pi^*_{L,m,i,u}(0)
\end{aligned} \tag{B9}$$

With this distribution, we are able to calculate the burst blocking probability in the uplink queue at an arbitrary time. To that end, let us denote the mass probability of the burst size being exactly l packets as $g_l=\frac{1}{l!}\frac{d^l}{dz^l}G_b(z)_{|z=0}$. Then,

$$P_{B,i,L} = \sum_{k=0}^{L}\text{Prob}[L_{q,i,u}=k]\text{Prob}[\text{burst}>L-k] \tag{B10}$$

and the blocking probabilities for TCP segments and acknowledgments become

$$P_{BdL} = \sum_{k=0}^{L}\text{Prob}[L_{q,i,u}=k]\sum_{l=L-k}^{\infty}g_{bd,l} \tag{B11}$$

$$P_{BaL} = \sum_{k=0}^{L}\text{Prob}[L_{q,i,u}=k]\sum_{l=L-k}^{\infty}g_{ba,l} \tag{B12}$$

The delay of the entire TCP segment is equal to the delay of the first baseband packet from that segment:

$$D^*_{L,d}(s) = \frac{1}{1-P_{BdL}} \left(\sum_{k=0}^{K-1} \Omega^*_{k,i,u}(s) G^*_{pd}(s)^{2k} V^*_i(s)^{\lfloor k/M \rfloor} + \sum_{k=1}^{K-1} \sum_{m=1}^{M} \Pi^*_{k,m,i,u}(s) G^*_{pd}(s)^{2(k-1)} V^*_i(s)^{\lfloor (k+m-1)/M \rfloor} \right)$$

It should be noted that tis delay is not equal to the delay for acknowledgment segment $D^*_{L,a}(s)$, even though the expressions used to derive them are similar.

1

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

3 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, June 2001, pp. 1990–1994.

4 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, Apr. 2001, pp. 591–600.

5 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, Nov. 1999, pp. 139–150.

6 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, Nov. 1999, pp. 384–388.

7 J.-B. Lapeyrie and T. 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*, vol. 2, New York, NY, Apr. 2003, pp. 1322–1332.

8 Bluetooth SIG, *Specification of the Bluetooth System*, Version 1.1, Feb. 2001.

9 ——, *Specification of the Bluetooth System – Core System Package [Host volume]*, Version 1.2, Nov. 2003, vol. 3.

10 ——, *Specification of the Bluetooth System – Architecture & Terminology Overview*, Version 1.2, Nov. 2003, vol. 1.

11 ——, *Specification of the Bluetooth System – Core System Package [Controller volume]*, Version 1.2, Nov. 2003, vol. 2.

12 M. Kalia, D. Bansal, and R. Shorey, "Data scheduling and SAR for Bluetooth MAC," in *Proceedings VTC2000-Spring IEEE 51st Vehicular Technology Conference*, vol. 2, Tokyo, Japan, May 2000, pp. 716–720.

13 D. P. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.

14 J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *ACM/IEEE Transactions on Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

15 B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytical models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK," *ACM/IEEE Transactions on Networking*, vol. 11, no. 6, pp. 959–971, Nov. 2003.

16 L. Cai, X. Shen, and J. W. Mark, "Delay analysis for AIMD flows in wireless/IP networks," in *Proceedings Globecom'03*, San Francisco, CA, Dec. 2003.

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

[18] J. Mišić, K. L. Chan, and V. B. Mišić, "Performance of Bluetooth piconets under E-limited scheduling," Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, Tech. report TR 03/03, May 2003.

[19] *The Network Simulator ns-2*. Software and documentation available from `http://www.isi.edu/nsnam/ns/`, 2003.

[20] Bluehoc, *The Bluehoc Open-Source Bluetooth Simulator, version 3.0*. Software and documentation available from `http://www-124.ibm.com/developerworks/opensource/bluehoc/`, 2003.