Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless Ad-hoc Network

Abhishek Das, Abhishek Ghose, Ashu Razdan, Huzur Saran† & Rajeev Shorey
IBM India Research Laboratory,
Block 1, Indian Institute of Technology,
Hauz Khas, New Delhi 110016, India
Email: srajeev@in.ibm.com
Phone: 91-11-6861100; Fax: 91-11-6861555

Abstract—**Emerging technologies such as Bluetooth are expected to become a ubiquitous solution for providing short range, low power, low cost, pico-cellular wireless connectivity. Bluetooth is a Master driven Time Division Duplex (TDD) system that supports an asynchronous channel for data traffic as well as synchronous channels for voice traffic. Data applications running over Bluetooth such as http, ftp and real audio will need transport layer protocols such as TCP and UDP to send packets over the wireless links. In this paper we study several schemes designed to improve the performance of asynchronous data traffic over a Bluetooth piconet that supports multiple active slaves. We propose and compare a number of SAR policies and MAC scheduling algorithms with a view towards enhancing the performance of transport layer sessions. We investigate the effect of different FEC and ARQ schemes at the baseband level, using a two-state Markov channel model for the Bluetooth RF link. We also study how the presence of circuit-switched voice impacts the performance of data traffic.**

Keywords—**Medium Access Control (MAC), Scheduling, Time Division Duplex (TDD), Segmentation and Reassembly (SAR), Forward Error Correction (FEC), Automatic Repeat Request (ARQ), TCP, UDP.**

I. Introduction

Bluetooth technology [1], [2] allows for the replacement of the numerous proprietary cables that connect one device to another with a universal short-range radio link. Beyond untethering devices by replacing cables, Bluetooth provides a universal bridge to existing data networks, a peripheral interface, and a mechanism to form small private ad-hoc groupings of connected devices away from fixed network infrastructures.

Bluetooth has a number of distinctive features compared to existing wireless LANs such as:

- Support for both data and voice traffic
- Frequency hopping to avoid interference
- A master driven Time Division Duplex (TDD) system at the Medium Access Control (MAC) layer to support full duplex transmission
- Segmentation and Reassembly (SAR) to handle large data packets
- Support for link level Automatic Repeat Request (ARQ) and Forward Error Correction (FEC) schemes

† Abhishek Das and Ashu Razdan are in the Computer Science and Engineering Department, Indian Institute of Technology at Kharagpur and Guwahati, respectively. Abhishek Ghose is in the Electrical Engineering Department, Indian Institute of Technology, Mumbai, India. Dr. Huzur Saran is a Professor in the Department of Computer Science and Engineering, Indian Institute of Technology, New Delhi, India. He is currently on a sabbatical at Stanford University, CA, USA. Email: saran@cse.iitd.ernet.in; Dr. Rajeev Shorey is a Research Staff Member in the IBM India Research Laboratory, New Delhi, India. This work was done while the first three authors were summer interns in the IBM India Research Laboratory, New Delhi from May to July'2000.

These key features of Bluetooth will significantly impact the performance of data traffic over Bluetooth. Fragmentation of large data packets by performing SAR, which allows them to be transmitted in small baseband packets, may increase their end-to-end delay. Master driven scheduling at the MAC level will affect throughput and queueing delay. The success rate of data transmissions will be affected by the presence of FEC and ARQ mechanisms at the link level. The bandwidth available for data traffic will be reduced in the presence of voice connections. The effect of these issues needs to be better understood in order to enhance the performance of asynchronous data traffic over Bluetooth.

Since TCP [3] is the most widely used transport protocol for reliable data services over the Internet, our primary focus in this paper is on the performance of TCP over Bluetooth. A key observation for wireless environments is that since TCP interprets packet loss as a sign of congestion and cuts back its window, its performance may deteriorate in the presence of random losses that cannot be attributed to congestion [4]. Considerable attention is being given to the design of a better TCP over the wireless link [5], [6]. Since Bluetooth is distinct from existing wireless LANs, these studies are not directly applicable to Bluetooth. We also study the performance of Constant Bit Rate (CBR) applications using UDP, a minimal non-guaranteed datagram service without any flow control or congestion avoidance.

In this paper, we propose two SAR policies with the aim of increasing link utilization and decreasing end-to-end delay of data packets. When multiple data transfers share the wireless link, as in a Bluetooth piconet, MAC scheduling algorithms are needed to achieve fair sharing of bandwidth, high link utilization and low queue occupancy. We demonstrate that Round-Robin scheduling is unable to meet these requirements and propose three new scheduling algorithms which meet these criteria adequately. We also incorporate Channel State Dependency [7] in these algorithms in order to improve the performance in the presence of bursty wireless errors. In accordance with observations reported in earlier studies of packet loss behavior in wireless LANs [5], [6], [18], a two state Markov model has been used to model the errors in the wireless channel. Bluetooth provides support for error correction at the link level through FEC and ARQ schemes. We investigate the performance improvement provided by FEC and ARQ schemes. We also compare the performance of different versions of TCP, namely, Tahoe, Reno, New Reno and Sack [3], [8] over Bluetooth.

Prior research closest to our work is that of Johansson *et al* [9]. They address the performance of TCP/IP over a Bluetooth wireless network but with very simplistic assumptions. They assume only two nodes (master and slave) in a Bluetooth piconet and study the behavior of TCP Vegas. Further, the authors model bit errors with a constant loss probability. They do not assume any FEC for data traffic arguing that doing so will yield largest ideal throughput. They do not specify any ARQ schemes at the baseband level to prevent packet loss. In [10], the authors have analysed and compared the behaviour of three different scheduling algorithms for Bluetooth: strict round robin polling, exhaustive polling and fair exhaustive polling. They study average delay versus bitrate for the three scheduling algorithms. The authors have demonstrated an increase in performance (high throughput and low delays) when allowing data packets to be sent in multi-slots (i.e., 3 or 5 slot baseband packets). The simulation study uses very simplistic assumptions: (i) the packet loss probability is constant for all packets, (ii) the master does not send any traffic to the slave but just forwards traffic from one slave to another, (iii) the buffers are assumed to be unbounded. Kalia *et al* [11] have proposed some simple SAR policies and MAC scheduling algorithms for Bluetooth. They propose two scheduling policies that utilize information about the size of the Head-of-the-Line packet at the master and slave queues to schedule the TDD slots effectively. These policies achieve high throughput and greater fairness compared to the Round-Robin based scheduling policies. The work in [11] is restricted only to the link layer and hence is not optimized for transport layer sessions.

The remainder of this paper is organized as follows. Section II gives a brief introduction to the Bluetooth technology. In Section III, we propose SAR and MAC scheduling policies and also discuss other important design issues in Bluetooth which affect the performance of asynchronous data traffic. The simulation model is presented in Section IV. In Section V, we present our simulation results and analyses. Finally, we conclude by presenting a summary of our results and some suggestions for future work in Section VI.

II. Bluetooth Technology

Bluetooth is a specification for the wireless communication of voice and data using a short-range radio. It is defined by a number of protocols residing in the physical and datalink layers in the OSI model, as shown in Figure 1. Bluetooth uses an adhoc, piconet structure with a single master and upto seven slaves. For a detailed description, see [1] and [2].

A. Bluetooth Baseband

The Baseband describes the specifications of the digital signal processing part of the hardware: the Bluetooth link controller, which carries out the baseband protocols and other low level link routines. Two link types are supported: (i) Synchronous Connection Oriented (SCO) (used primarily for voice) and (ii) Asynchronous Connectionless (ACL) (used primarily for packet data). Both link types use a Time Division Duplex (TDD) scheme for resolving contention over the wireless link, where each slot is 625 μs long. The SCO link is a point-to-point link between the master and a single slave, established by reservation of duplex slots at regular intervals. The ACL link is a point-to-multipoint link between the master and all the slaves in the piconet. A baseband packet may occupy one, three or five slots. The desired baseband packet length can be decided based on criterion such as the quantity of data to be transmitted or the number of contiguous slots available in the presence of voice traffic.

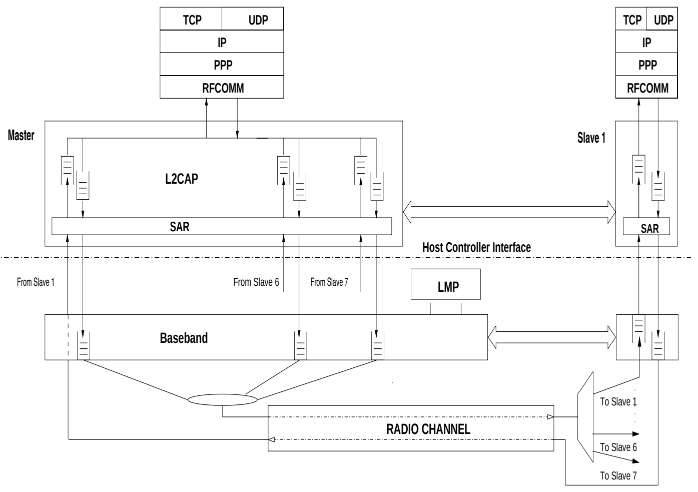

Fig. 1. The Bluetooth Protocol Stack

A fast unnumbered ARQ scheme is used to inform the source of the success or failure of transfer of payload. The baseband packets are retransmitted till a positive acknowledgement (ACK) is returned or until timeout is exceeded. An optional $2/3$ rate Forward Error Correction (FEC) can be used on the data payload to reduce the number of retransmissions. The packet header is always protected by a $1/3$ rate FEC since it contains valuable link information and should be able to sustain more errors.

B. Link Manager Protocol (LMP) and Logical Link Control and Adaptation Protocol (L2CAP)

LMP and L2CAP are layered above the Baseband Protocol and reside in the datalink layer. LMP assumes the responsibility of managing connection states, enforcing fairness among slaves, power management and other management tasks. L2CAP supports higher level protocol multiplexing since the Baseband does not support any *type* field identifying the higher layer protocol. L2CAP also supports packet segmentation and reassembly (SAR), and conveys quality of service information. It permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

C. IP over Bluetooth

TCP/IP/PPP is used for all Internet Bridge usage scenarios in Bluetooth [1]. UDP/IP/PPP is also available as transport for WAP (Wireless Application Protocol). In the Bluetooth technology, PPP (Point-to-Point Protocol) [13] is designed to run over RFCOMM [1] to accomplish point-to-point connections. RFCOMM provides serial cable emulation using a subset of the ETSI GSM 07.10 standard. However, the specification is open and it is also possible to configure IP directly over L2CAP. Thus two scenarios can be envisaged:

1. TCP/IP running over PPP over RFCOMM, which is layered over L2CAP. This enables smooth operation and interoperability with legacy applications. In this case, the framing information available through HDLC (Higher Level Data Link Control) in PPP should be passed to the L2CAP layer to make it aware of PPP packet boundaries, thus enabling efficient SAR [14].
2. TCP/IP layered directly over L2CAP.

The second approach has lesser overheads [14] and is used in our simulation model. However, our results also apply to the former case since the overheads due to RFCOMM and PPP headers (14 bytes) are negligible compared to the size of the TCP packet and the delay involved is a constant factor.

III. Design Issues in Bluetooth

In this section, we examine some of the design issues which have a significant impact on the performance of asynchronous data traffic over Bluetooth.

A. Segmentation and Reassembly schemes

Segmentation and reassembly (SAR) mechanisms are used to improve efficiency by supporting a maximum transmission unit (MTU) size larger than the largest baseband packet. This reduces overheads by spreading the packets used by higher layer protocols over several baseband packets, each covering 1, 3 or 5 slots. It is important to note that the payload size (without FEC) for a 5 slot packet (339 bytes in 5 slots or 67.8 bytes/slot) is significantly larger than that of 3 slot packets (183 bytes in 3 slots or 61 bytes/slot) and 1 slot packets (27 bytes/slot). We define *slot_limit* as the maximum number of slots across which a baseband packet can be sent. The *slot_limit* may be less than 5 due to presence of SCO connections or due to a very high bit error rate in the wireless channel. This parameter can be conveyed by the LMP to the L2CAP through a signalling packet. We now propose two SAR schemes.

A.1 SAR - Best Fit (BF)

This algorithm aims to reduce the wasted bandwidth in the baseband packets and uses a best-fit method to segment the higher layer packets. The algorithm can be summarized in the following steps:

1. If $slot\_limit = 5$, divide the L2CAP packet into an integral number $num5$ of 5 slot baseband packets.
2. If $slot\_limit \geq 3$, divide the remaining bytes into an integral number $num3$ of 3 slot baseband packets.
3. Divide the remaining bytes into an integral number $num1$ of 1 slot baseband packets.

For example, consider an L2CAP packet of size 556 bytes and $slot\_limit = 5$. The biggest data segment possible in this case is of 339 bytes, i.e. the best fit is a 5-slot packet. After this we are left with 217 bytes. Now the best fit is a 3 slot packet. After this iteration, we have 34 bytes left to be fragmented. The best fit here is a 1-slot packet and we will be left with 7 bytes, which is again sent in a 1 slot packet. Thus, we get one 5 slot packet, one 3-slot packet and two 1-slot packets after fragmentation.

TABLE I
Fragmentation of 556 byte L2CAP packet

	SAR-BF			SAR-OSU		
	$num1$	$num3$	$num5$	$num1$	$num3$	$num5$
Initial	0	0	0	0	0	0
Step 1	0	0	1	0	0	1
Step 2	0	1	1	0	0	2
Step 3	2	1	1	-	-	-
Step 4	-	-	-	-	-	-
Step 5	-	-	-	-	-	-

A.2 SAR - Optimum Slot Utilization (OSU)

This algorithm aims to decrease the transmission delay of L2CAP baseband packets by reducing the queueing delay of baseband packets. The lesser the number of baseband packets per L2CAP packet, the lesser is the end-to-end delay since only a single baseband packet is sent each time a slave is polled. Hence this algorithm maximizes the data sent each time a slave is polled by preferentially sending multi-slot packets. The algorithm can be summarized in the following steps:

1. If $slot\_limit = 5$, divide the L2CAP packet into an integral number $num5$ of 5 slot baseband packets.
2. If the size remaining to be fragmented is larger than that of a 3 slot packet, send it as a 5 slot packet.
3. If $slot\_limit \geq 3$, divide the remaining bytes into an integral number $num3$ of 3 slot baseband packets.
4. If the size remaining to be fragmented is larger than that of a 1 slot packet, send it as a 3 slot packet.
5. Divide the remaining bytes into an integral number $num1$ of 1 slot baseband packets.

Revisiting the previous example, the biggest data segment possible in this case is again of 339 bytes. After this we are left with 217 bytes. Since this is larger than a 3-slot packet, these bytes are sent as a 5-slot packet. Thus, we get two 5-slot packets after fragmentation. The operation of the two algorithms is summarized in Table I.

In Section V-A, through simulations, we show that SAR-OSU outperforms SAR-BF in terms of throughput, link utilization and end-to-end delay.

B. Buffer-Size Optimization

As long as memory resources are abundant, one can keep the buffer sizes undetermined so as to provide the scheduling tool with the greatest flexibility and to prevent packet drops due to overflow. However, when memory resources are scarce, as is the case in small devices such as Personal Digital Assistants (PDAs) on which Bluetooth is likely to run, buffer sizes are preferably kept small. The primary data buffers in Bluetooth are at the L2CAP and at the Baseband. We keep the baseband buffer large enough to hold the baseband packets formed by fragmenting an L2CAP packet. We carry out simulations in Section V-B to find an appropriate size for the L2CAP buffer.

C. Scheduling Algorithms in Bluetooth

Multiple transport layer sessions share the wireless link in a piconet when multiple slaves are active or when a slave has multiple data connections. Master-driven Round-Robin scheduling achieves fair sharing of bandwidth and high link utilization when each such connection has equal data flow. In a typical situation, however, each slave in the piconet has varying data input rates. Consequently, numerous baseband slots are wasted by polling sources with low input rate, thereby decreasing link utilization, increasing queueing delay and leading to unfair sharing of bandwidth. To address these issues, we propose three scheduling algorithms, which incorporate the following methods:

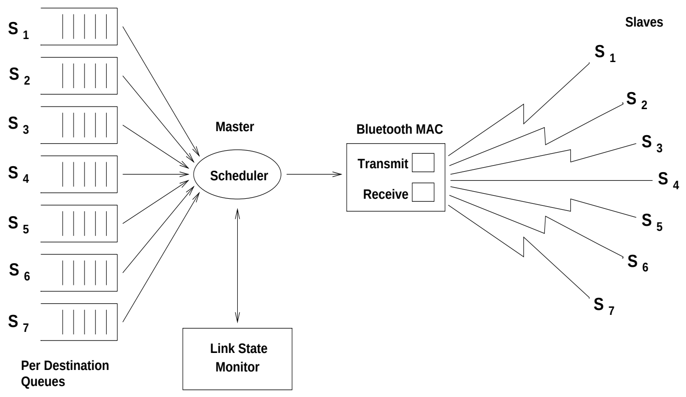

Fig. 2. MAC Scheduling in Bluetooth

Queue Priority based on Flow Bit: Per-slave baseband queues at the master and similar queues at the slave are maintained (as shown in Figure 2). We assign priority to these queues based on the pending data in the corresponding L2CAP buffers, and use the flow bit present in the payload header field of the baseband packet for this purpose. This flow bit is used to convey flow information at the L2CAP level as intended in the Bluetooth specification [1]. It is set when the number of packets in the L2CAP buffer for a particular slave is larger than a threshold *buf_thresh*. The LMP at the master monitors the flow bit of the baseband packets sent/received and conveys the traffic status to the Baseband. We define a variable *flow* to quantify the traffic rate on the wireless channel, which is set when the flow bits for packets traveling in either direction is turned on.

Queue Stickiness: In the case of Round-Robin scheduling, one packet is served at a time from each baseband queue. However the slaves with high data inflow may have their queues full while baseband slots are being wasted for slaves with low queue backlog. To reduce mean queue occupancy, we propose to transmit a number of baseband packets successively (quantified by a parameter num_sticky) for each queue having the *flow* parameter set.

Based on these ideas, we propose the following scheduling algorithms:

C.1 Adaptive Flow-based Polling (AFP)

Polling interval for a particular slave is defined as the maximum time limit, before which it must be served by the master and is decided based on QoS requirements. We define P_0 to be the polling interval negotiated during master-slave connection setup. We assume a homogeneous situation initially where all the slaves have the same value of P_0. AFP uses an adaptive polling interval P, whose value is changed based on the traffic rate in the wireless channel as indicated by the variable $flow$.

1. If $flow = 1$ and the Head-of-Line (HOL) packet is a data packet, transmit the data packet and set the polling interval P to P_0. In this case, there is a high flow rate for this slave, and hence its polling interval is reduced so that it can be served more frequently.
2. If $flow = 0$ and the HOL packet is a data packet, transmit the data packet and keep the polling interval unchanged.
3. If a poll packet is transmitted and a null packet is received, double the current polling interval P unless a threshold value P_{thresh} is reached. The polling interval is increased so as to reduce slots wasted when neither the master nor the slave have any data to transmit.

C.2 Sticky

Each slave is serviced in a cyclic fashion contingent on the state of $flow$:

1. If $flow = 1$, a maximum of *num_sticky* packets are transmitted for that queue. Here *num_sticky* is a variable parameter greater than one, whose optimal value is found through simulations in Section V-C.
2. If $flow = 0$, one packet is transmitted for that queue, as in Round-Robin scheduling.

C.3 Sticky Adaptive Flow-based Polling (StickyAFP)

This is similar to *AFP* except that when $flow = 1$ and the HOL packet is a data packet, a maximum of *num_sticky* packets are transmitted for that queue.

The hardware implementation of these algorithms is quite simple, the only additional hardware overheads being counters (for tracking number of packets transmitted in the Sticky algorithm), combinatorial logic gates (for *flow*) and registers (for keeping track of polling intervals). For a software implementation, a few additional instructions are needed.

The interaction between TCP's flow control and the link layer dispatching mechanism is quite complex. Since the behavior of TCP sources is difficult to capture by any closed form analytical expression, it is hard to analyze this system. We compare the performance of these algorithms through simulations in Section V-C.

D. Error Handling

For the ARQ scheme, we quantify *timeout* (described in Section II-A) by a maximum number of retransmissions tx_thresh. The purpose of the FEC scheme on the data payload is to reduce the number of retransmissions. However, in a reasonably error-free environment, FEC results in unnecessary overhead that reduces the throughput. Using a two state Markov channel model (described in Section IV-A) in our simulations, we study the effect of using FEC on the baseband payload and of varying the parameter tx_thresh in the ARQ scheme on the data throughput (Section V-D).

Since wireless channels are characterized by bursty errors, repeated transmission attempts of the head of line (HOL) packets may fail, blocking the transmission of packets to other receivers. Since the wireless links to various destinations are statistically independent [7], [16], packets for other slaves could be successfully transmitted during this interval. We propose Channel State Dependent Packet (CSDP) scheduling [7] versions of our algorithms to improve the data throughput over lossy wireless links. Upon encountering a packet loss (indicated by receipt of a negative ACK), CSDP policies defer the retransmissions to that slave till the next polling instant. If the deferred period length is more than TCP's timeout period, the source will timeout and retransmit a copy of the delayed packet, thereby unnecessarily increasing the load on the system. In practice, however TCP's timeout period is of the order of seconds, while the duration of burst periods is of the order of milliseconds. This time difference is sufficient for link layer mechanisms to attempt loss recovery by retransmission over the radio link. We compare the performance of *CSDP-AFP*, *CSDP-Sticky* and *CSDP-StickyAFP* through simulations in Section V-D.

E. Number of SCO connections

Upto three simultaneous SCO links for supporting real-time traffic, such as voice, can be supported by the master. The master will send SCO packets at regular intervals, the so-called SCO intervals T_{SCO} (counted in slots) in the reserved master-to-slave slots. The SCO slave is always allowed to respond with an SCO packet in the following slave-to-master slot.

Since SCO links reserve slots, no ACL packets can be supported in the presence of three SCO links. In the presence of two SCO links, only 1 slot ACL packets can be sent. In the presence of one SCO link, 1 and 3 slot ACL packets can be supported. The performance of data (ACL) traffic in the presence of varying number of SCO links is described in Section V-E.

F. TCP variants

Current implementations of TCP, namely TCP Tahoe and TCP Reno [8], use an acknowledgement number field that contains a cumulative acknowledgement, indicating that the TCP receiver has received all of the data upto the indicated byte. A selective acknowledgement option allows receivers to additionally report non-sequential data they have received (implemented in Sack TCP [8]). New-Reno [8] avoids many of the retransmit timeouts of Reno without requiring SACK. Through simulations in Section V-F, we compare the performance of TCP Tahoe and Reno with two modified versions of TCP Reno: TCP New Reno and TCP Sack.

IV. Simulation Model

We have developed an extensive simulation model for Bluetooth, using the Network Simulator (ns) [15] and the MATLAB package, containing the core Bluetooth protocol layers (shown in Figure 1) as well as TCP/IP. The network is modeled as a Bluetooth piconet with one master and seven slaves.

The traffic sources shown in Figure 3 generate TCP/UDP traffic which is transmitted to the transport layer. After the TCP/UDP header is added, the packets are sent to the network layer. The L2CAP layer receives data segments from the upper layer, adds an L2CAP header and enqueues them in the L2CAP buffer. Large L2CAP packets are then segmented into multiple smaller baseband packets by a SAR module at the L2CAP and are enqueued in a baseband buffer. Using a MAC scheduling algorithm, the packets are then sent at appropriate intervals through the physical RF link. This is modeled in Section IV-A.

When the slave's baseband layer receives a packet, it is enqueued in the baseband buffer and then sent to the SAR module in L2CAP for reassembly. The reassembled packets are transmitted upwards through the protocol stack to the transport layer. TCP then sends an ACK for correctly received packets (UDP does not send any ACKs). In our simulations, we use a TCP/UDP Packet size of 512 bytes and a TCP ACK size of 40 bytes.

The actual traffic sources may be located at any point in the Internet. However, as the bandwidth associated with wireline networks is much higher than is available in the Bluetooth wireless link, the latter becomes a bottleneck. Thus, for the purpose of our simulation, we may equivalently place our sources at the master and account for the wired part of the network by a constant delay. In the simulated network (see Figure 3), slaves 1 and 2 have persistent TCP (*ftp*) connections which are active from 0 to 60 and 10 to 20 seconds respectively. Slaves 3 to 7 receive CBR traffic running over UDP with rates ranging from 5520 bps to 17664 bps, as shown in Figure 3. Note that we have tried to capture a variety of traffic sources in the traffic model, which is evident from the choice of flows. While we study this traffic model, we look into the effect of all sources rather than a single one and hence we use the performance metrics of link utilization, average throughput and average end-to-end delay which reflect the overall data performance rather than the performance of a single slave in a piconet.

We have used TCP-Reno [8] in our simulations since it is one of the most common reference implementations for TCP.

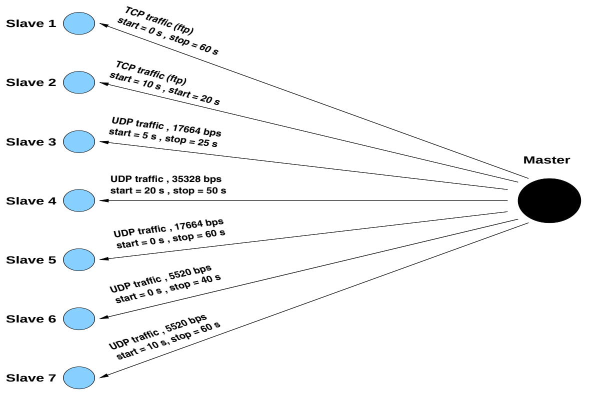
Slave 1
TCP traffic (ftp) start = 0 s , stop = 60 s
Slave 2
TCP traffic (ftp) start = 10 s , start = 20 s
Slave 3
UDP traffic , 17664 bps start = 5 s , stop = 25 s
Master
Slave 4
UDP traffic , 35328 bps start = 20 s , stop = 50 s
Slave 5
UDP traffic , 17664 bps start = 0 s , stop = 60 s
Slave 6
UDP traffic , 5520 bps start = 0 s , stop = 40 s
Slave 7
UDP traffic , 5520 bps start = 10 s, stop = 60 s

Fig. 3. Traffic Sources used in Simulation

Performance Metrics

We study three performance metrics: throughput, end-to-end delay and link utilization. *Throughput* is an indication of how much data the user can receive per second. In our simulation results, the throughput is averaged every 0.5 seconds. We define *End-to-end delay* of packets to be the delay incurred from the time it is enqueued in the transport layer buffer to when it is received. Note that this includes the queueing delay at the buffers. *Link Utilization* quantifies how much of the available bandwidth is actually being used by baseband packets.

A. *Correlated Fading Channel Model*

Since packet-error rates critically depend on the distance between the transmitter and receiver [17], the wireless channel varies with each user, depending on their location. For high speed radio transmissions at a high carrier frequency, as in Bluetooth (1 Mb/s at 2.4 GHz), the fade durations are comparable to transmission times of TCP packets [5] and hence, packet losses cannot be modeled as being independent of each other. However, the frequency-hopping scheme, incorporated in the Bluetooth radio, leads to considerably smaller bursts of errors compared to other wireless links. In accordance with common practice, we model the Bluetooth RF link as a discrete two-state Markov Chain [5], [6], [18] as illustrated in Figure 4. Since wireless channels have been shown to be distinct and time-varying for each user [16], we associate independent Markov channel models with each master-slave connection. At any point

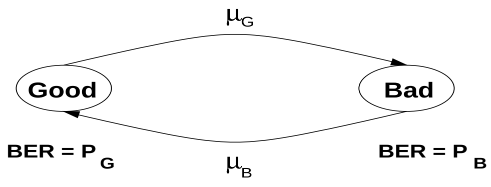

Fig. 4. Transition Structure of the Markov Loss Model

of time, we model the channel as being in one of the two possible states, G (Good) with BER P_G or B (Bad) with BER P_B where $P_B \gg P_G$. These BER values are dependent on the characteristics of the propagation environment and the transmission modulation scheme. The mean residency time in states with high Bit Error Rate (BER) are longer than a single packet transmission time, resulting in correlated packet losses. We assume that the time spent in each state is exponentially distributed, with different mean values, that is, different rates of state transitions μ_G and μ_B. According to the properties of exponentially distributed random variables, the average time between state transitions can be expressed by $\rho_G = 1/\mu_G$ and $\rho_B = 1/\mu_B$.

To obtain average parameter values of the indoor radio environment, time-varying channel impulse responses were simulated using a wide band channel model. As GFSK is used in the Bluetooth Radio, GFSK BER vs E_b/N_0 results are taken from [20]. We take a log-normal distribution for the received signal amplitudes as it provides the closest fit for indoor wireless radio environments [21]. The two states of the Markov model are constructed by partitioning the envelope of the received signal into two intervals, and thus determine the mean residency times and BERs of the two states. The parameter values obtained are $P_G = 6.879 \times 10^{-5}$, $P_B = 1.263 \times 10^{-3}$, $\rho_G = 437.5$ ms and $\rho_B = 55.8$ ms.

Note that the objective in this paper is to illustrate the behavior of a transport session when packets are subject to bursty losses. An approximate characterization of the wireless channel is sufficient to illustrate these effects.

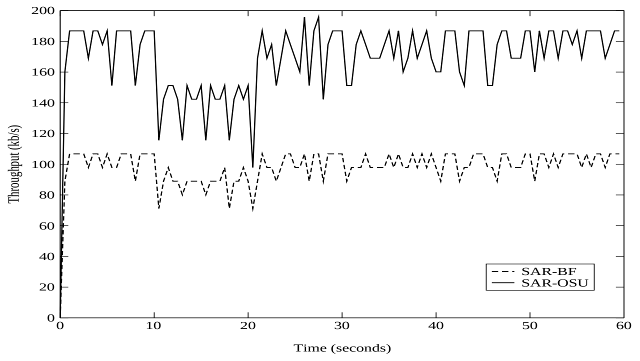

Fig. 5. TCP throughput vs time [slot_limit = 5]

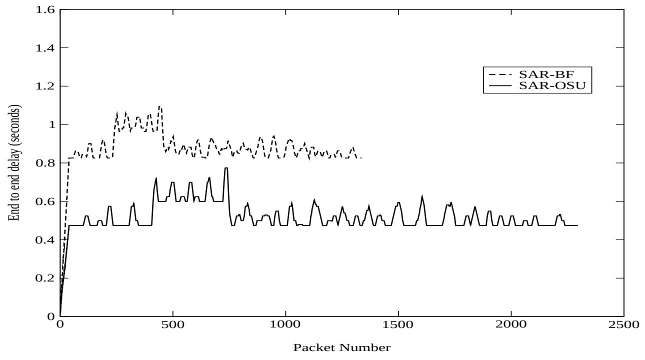

Fig. 6. End-to-end delay vs Packet Number [slot_limit = 5]

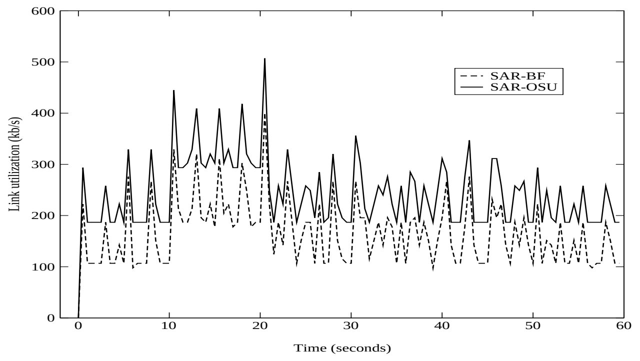

Fig. 7. Link utilization vs time [slot_limit = 5]

V. Simulation Results and Performance Evaluation

A. *Segmentation and Reassembly (SAR)*

To compare the performance of SAR-BF and SAR-OSU, we use an L2CAP buffer of size equal to 50 TCP/UDP packets. The large buffer size allows us to neglect the effect of packet drops and focus on the interaction between the SAR policy and data performance. We use Round-Robin scheduling at the MAC level and assume an error-free channel so as to isolate the effect of the SAR policies on data performance. In Figures 5 and 6, the throughput and end-to-end delay of a persistent TCP source (to slave 1) is compared for SAR-BF and SAR-OSU

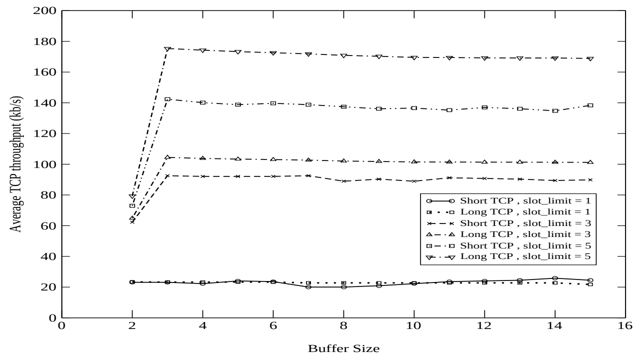

Fig. 8. Average TCP throughput vs L2CAP Buffer Size

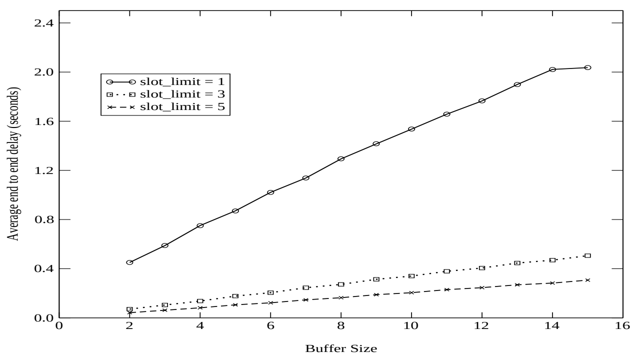

Fig. 9. Average end-to-end delay vs L2CAP Buffer Size

using $slot\_limit = 5$. Figures 5 and 6 clearly illustrate that higher throughput and lower end-to-end delays can be obtained by using SAR-OSU over SAR-BF. The overall link utilization of SAR-OSU is also shown to be higher than that of SAR-BF (Figure 7).

Two interesting trends are observed in Figures 5, 6 and 7. Firstly, the frequent fluctuations seen in these figures are due to the bursty sources (simulated by intermittent CBR traffic) in our model. Secondly, since two TCP connections (to slaves 1 and 2) are active in the period between 10 to 20 seconds, fair sharing of bandwidth leads to a drop in the individual throughput of the TCP connection to slave 1, while the overall link utilization is seen to increase.

Table I in Section III-A illustrates that SAR-OSU uses 12 slots (including slots for ACKs) and wastes 122 bytes in the last 5 slot packet whereas SAR-BF uses 14 slots and wastes 20 bytes. Since SAR-OSU uses a lesser number of slots and consequently uses lesser time to transmit the same amount of data, it results in a higher throughput and lower end-to-end delays, and hence higher link utilization. If $slot\_limit = 3$ or 1, both SAR-BF and SAR-OSU will fragment the 556 byte L2CAP packet into the same number of baseband packets, and hence there will not be any appreciable difference in their performance. Due to the superior performance of SAR-OSU, it is used for the remaining simulations in this paper.

B. L2CAP Buffer size

In this section, we use Round-Robin scheduling and assume an error-free channel to isolate the effect of buffer size on TCP performance. In Figure 8, we plot the average TCP throughput as a function of buffer size for a persistent TCP connection (to Slave 1) as well as for a short TCP transfer (to Slave 2) for $slot\_limit$ equal to 1, 3 and 5. The buffer size is expressed in multiples of transport layer packets.

We observe that the average TCP throughput becomes almost constant for a buffer size greater than four for the persistent TCP connection. Though the throughput varies for L2CAP buffer size larger than four for the short transfer, the variance is small. In Figure 9, the end-to-end delay of TCP is plotted as a function of buffer size. This graph shows an increasing trend which is justified by the fact that a larger buffer size leads to an increased queueing delay, which in turn increases the average end-

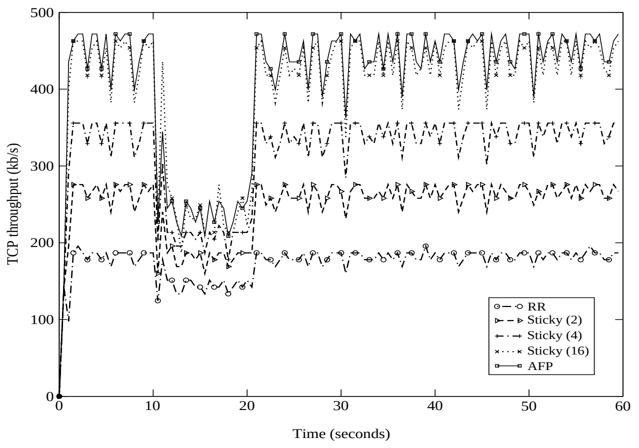

Fig. 10. TCP throughput vs time for AFP and Sticky

to-end delay. Since Bluetooth will typically be implemented in small devices with limited memory resources, we conclude that a buffer size of four to six will optimally satisfy the memory requirements of a generic Bluetooth device. For the rest of our simulations, we use an L2CAP buffer size of five.

This buffer size can be justified by calculating the bandwidth-delay product for our model. From Figure 8, we get an average throughput of 39.8 packets/second and from Figure 9, we get an average delay of 0.11 seconds for a buffer size of 5 and $slot\_limit = 5$. This gives a bandwidth-delay product equivalent to 4.4 packets, which is quite close to 5. This reasoning, however, is approximate since the entire 1 Mb/s bandwidth of the radio link is not available to a single connection.

From Figure 8, it is also observed that the throughput performance with $slot\_limit = 5$ is better than that for $slot\_limit = 3$ which is in turn better than that for $slot\_limit = 1$. This is expected since 5 slot packets have higher payload content than 3 slot and 1 slot packets, as described in Section III-C.

C. Scheduling Algorithms

In this section, we assume that the channel is error free and simulate the algorithms described in Section III-C.

In Figure 10, the TCP throughput for different values of the parameter num_sticky in the Sticky algorithm is compared with that of the Adaptive Flow-based Polling (AFP) and Round-

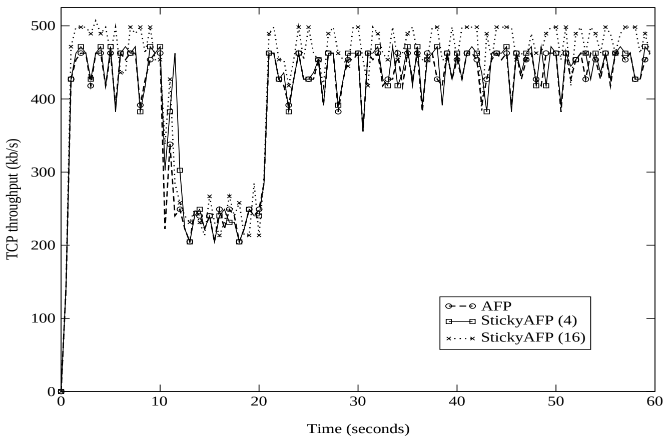

Fig. 11. TCP throughput vs time for AFP and StickyAFP

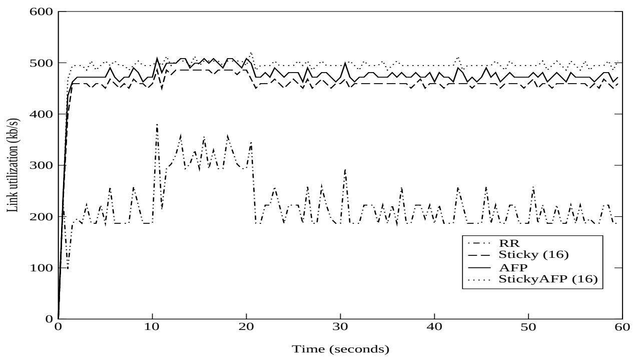

Fig. 12. Link utilization for scheduling algorithms

Robin (RR) algorithms. From this graph, we clearly observe that the AFP and Sticky algorithms give significantly improved performance compared to RR. The throughput of Sticky increases with increase in the value of num_sticky and is approximately the same as AFP for a num_sticky value of 16.

In Figure 11, the TCP throughput of AFP is compared to that of StickyAFP for num_sticky values 4 and 16. From this graph, we observe that the throughput performance of StickyAFP for a num_sticky value of 16 is better than that of AFP and StickyAFP for a num_sticky value of 4, the latter two having almost the same throughput. However, the performance improvement obtained for a num_sticky value of 16 is not very significant.

In Figure 12, the link utilization under different scheduling algorithms is compared. From the graph, we see that high link utilization is obtained for StickyAFP ($num\_sticky = 16$), Sticky ($num\_sticky = 16$) and AFP, as compared to Round-Robin. In Figure 13, the average end-to-end delay is plotted for each of the slaves under different scheduling algorithms. The Sticky algorithm is found to have the lowest end-to-end delay while StickyAFP has the highest. By increasing the polling interval for those queues that have less data, AFP decreases the number of poll packets which otherwise cause underutilization of available bandwidth, and hence increases link utilization. Sticky reduces queue occupancy by transmitting multiple packets consecutively from queues with a high backlog, hence preventing

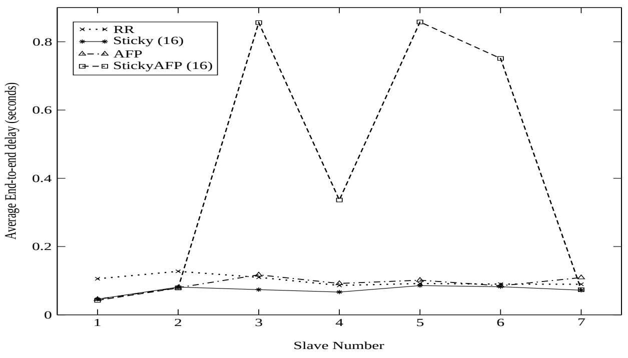

Fig. 13. End-to-end delay for scheduling algorithms

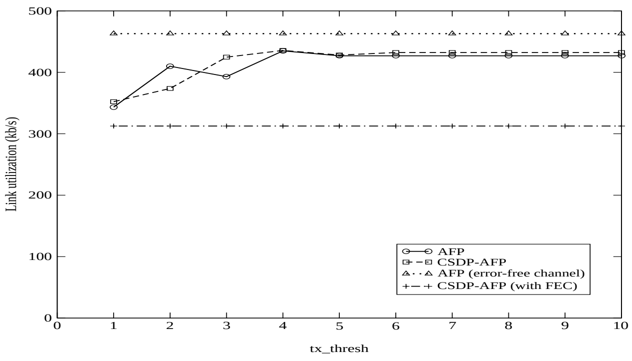

Fig. 14. Link utilization vs tx_thresh for versions of AFP

queue overflow and reducing end-to-end delay. StickyAFP, on the other hand, causes a marked increase in the end-to-end delay of intermittent CBR traffic because *flow* is set infrequently for such bursty sources. Additionally, each cycle has a larger duration due to other slaves being served num_sticky times.

Although AFP, StickyAFP ($num\_sticky = 16$) and Sticky ($num\_sticky = 16$) give the best results in terms of link utilization as well as throughput (to Slave 1), we observe that StickyAFP leads to significantly higher end-to-end delay. Thus we infer that AFP and Sticky (with a high value of num_sticky) result in the best overall performance.

D. Effect of Error Correction Schemes

Figures 14 and 15 show the link utilization and average end-to-end delay for different values of tx_thresh (maximum number of retransmissions of baseband packets) under AFP and CSDP-AFP in the presence and in the absence of a 2/3 rate FEC. The corresponding values for AFP in an error free channel are also plotted for comparison. These figures clearly indicate the performance degradation in the presence of errors, as well as the additional reduction in link utilization and increase in end-to-end delay due to the use of FEC. A remarkable observation is that when FEC is added, the performance is independent of tx_thresh, and hence of the ARQ scheme. We also observe that ARQ leads to efficient error recovery and for values of $tx\_thresh > 4$, the performance does not vary significantly. This motivates us to choose the optimum value of tx_thresh to be 5.

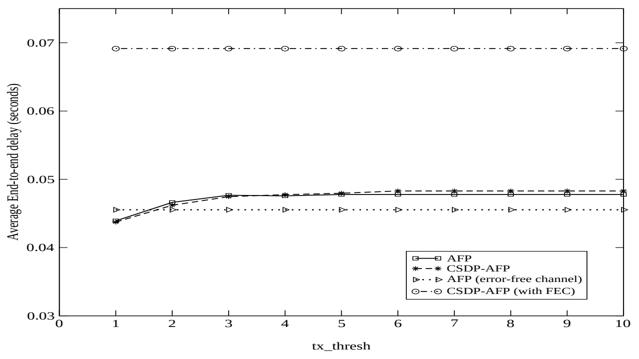

Fig. 15. End-to-end delay vs tx_thresh for versions of AFP

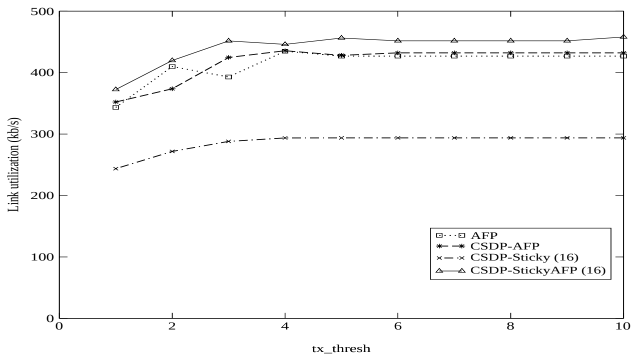

Fig. 16. Link utilization vs tx_thresh for CSDP algorithms

Figure 16 clearly illustrates that the CSDP versions of the proposed scheduling algorithms do not give a significant performance improvement and their relative performance is the same as that in the error-free channel condition. Since the burst error periods in the wireless channel are short enough to allow the packets to be successfully retransmitted before tx_thresh retransmissions, CSDP versions do not improve the performance in the presence of a link level ARQ scheme.

E. Varying voice connections

In Figures 17, the throughput for the persistent TCP transfer (to slave 1) is shown using AFP and with varying number of SCO connections (N_{SCO}). The average end-to-end delays for each of the slaves under the same conditions is shown in Figure 18. From these graphs we see that the throughput decreases and end-to-end delay increases as the number of SCO connections increase. As is expected, the lowest throughput and highest end-to-end delay is obtained when $T_{SCO} = 6$, with 2 SCO connections. For a T_{SCO} value of 6 and one SCO connection, two different values of $slot\_limit$ (1 and 3) are possible. The graphs show that a higher throughput and lower end-to-end delay is obtained for $slot\_limit = 3$ than for $slot\_limit = 1$. Further, for one SCO connection, a lower throughput and higher end-to-end delay is obtained for $T_{SCO} = 6$ than for $T_{SCO} = 4$. This is expected since for a larger T_{SCO}, more slots are available for ACL packets. Further, as shown in the graphs, the highest throughput and lowest end-to-end delay is obtained for AFP with no voice connections.

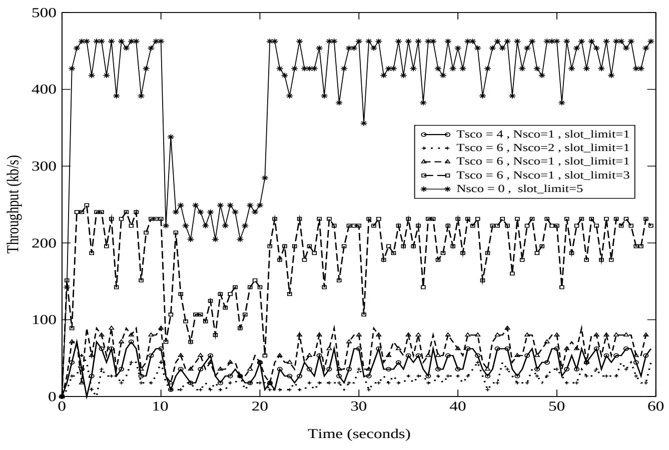

Fig. 17. Throughput degradation in the presence of voice

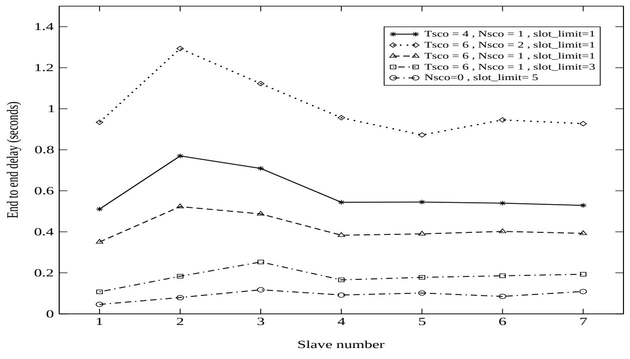

Fig. 18. End-to-end delay in the presence of voice

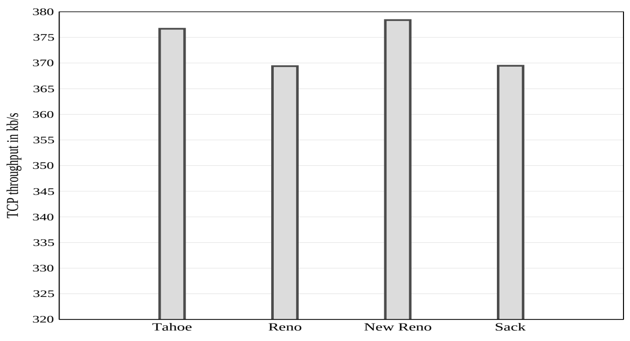

Fig. 19. Performance of TCP Versions

F. TCP variants

The performance of the different TCP versions in the presence of errors, for the TCP connection to slave 1, is shown in Figure 19. NewReno performs marginally better than the other versions. However, the difference in throughput is insignificant which clearly illustrates that the efficient link layer ARQ scheme of Bluetooth eliminates the need for modifications at the transport layer for error recovery.

VI. Conclusion and Future Work

We have presented methods to enhance the performance of asynchronous data traffic over a Bluetooth piconet with a master and seven active slaves. We have used both TCP and UDP traffic sources in our study. To begin with, we proposed two new SAR mechanisms: SAR-BF and SAR-OSU, that increase the link utilization and throughput, as well as reduce the end-to-end packet transmission delays. SAR-OSU outperforms SAR-BF in all respects, i.e., throughput, link utilization and packet transmission delays. Further, we demonstrated that an L2CAP buffer, capable of holding five transport layer packets is well suited for Bluetooth devices (which typically have low memory resources) and also reduces the queueing delay. Assuming a transport layer packet of size 512 bytes, we concluded that even real time UDP applications running at 32 kb/s will not experience any packet drops.

We observed that a simple MAC scheduling algorithm such as Round-Robin is not suitable for Bluetooth as it is unable to minimize delay for interactive sessions. Further, it does not distribute bandwidth fairly amongst all active sessions. With these issues in mind, we proposed and compared three new scheduling algorithms: AFP, Sticky and StickyAFP, which have a simple implementation. Our results highlight the significant increase in performance obtained by their use. AFP and Sticky (with num_sticky set to 16) have been shown to have the best performance.

Using an appropriate two state Markov model to characterize the indoor wireless channel, we studied the performance of the Bluetooth system with link level ARQ and FEC schemes of Bluetooth. We emphasized the fact that the use of link level ARQ is adequate for efficient error recovery, with FEC only adding to the overheads. An optimum value for the maximum number of retransmissions, *tx_thresh*, in the ARQ scheme was found to be five. We demonstrated that the channel state dependent (CSDP) versions of the proposed scheduling algorithms do not lead to significant gains in performance. Further, we showed that the presence of voice connections degrades the performance of data traffic. We observed that the performance gains across TCP variants are marginal, from which we concluded that efficient link-layer error recovery mechanisms in Bluetooth obviate the need for transport layer enhancements.

It is important to point out that the error model can be extended, at the expense of an increased computational complexity, to a transmission channel modeled as a finite state Markov chain with more than two states. Further study is required to incorporate low power modes (*sniff, hold, park*) into MAC scheduling and to explore the effect of varying number of slaves in a piconet. Since performance is expected to vary significantly with varying channel status, a clever design of adaptive FEC and ARQ schemes may be very effective in Bluetooth. An important area of future research is to study the performance of TCP traffic over a Bluetooth scatternet with multiple overlapping piconets. Many earlier papers have focussed on TCP performance over mobile ad-hoc networks (see [12] and the references therein), it however remains to be seen how TCP will perform over a Bluetooth scatternet with a unique physical and TDD MAC layer.

Acknowledgment

The authors would like to thank Apurva Kumar (Research Staff Member, IBM India Research Laboratory, New Delhi, India) and Vijay Gupta (Department of Electrical Engineering, Indian Institute of Technology, New Delhi, India) for their help in designing the Bluetooth channel model. The parameters of the two state Markov model for Bluetooth have been obtained by them.

References

[1] Bluetooth Special Interest Group, “Specification of the Bluetooth System 1.0b, Volume 1: Core,” *http://www.bluetooth.com*, Dec. 1999.

[2] J. Haartsen, “The Bluetooth Radio System,” *IEEE Personal Communications*, Vol. 7, No. 1, pp. 28-36, Feb. 2000.

[3] W.R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison Wesley, 1994.

[4] H. Chaskar and U. Madhow, “TCP over wireless with link level error control: Analysis and design methodology,” *Proc. MILCOM*, 1996.

[5] A. Kumar, “Comparative performance analysis of versions of TCP in local network with a lossy link,” *IEEE/ACM Trans. on Networking*, Vol. 6, No. 4, pp. 485-498, Aug. 1998.

[6] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz, “A comparison of mechanisms for improving TCP performance over wireless links,” *IEEE/ACM Trans. on Networking*, Vol. 5, No. 6, pp. 756-769, Dec. 1997.

[7] P. Bhagwat, P. Bhattacharya, A. Krishna and K. Tripathi,“Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs,” *Wireless Networks*, pp. 91-102, 1997.

[8] K. Fall and S. Floyd, “Simulation-based Comparisons of Tahoe, Reno, and Sack TCP,” *ftp://ftp.ee.lbl.gov*, Mar. 1996.

[9] N. Johansson, M. Kihl and U. Korner, “TCP/IP over the Bluetooth wireless ad-hoc network,” *Networking 2000*, Paris, France, May 2000.

[10] N. Johansson, U. Korner and P. Johansson, “Performance evaluation of scheduling algorithms for Bluetooth”, In *Broadband Communications: Convergence of Network Technologies*, Edited by Danny H. K. Tsang and Paul J. Kuhn, Kluwer Academic Publishers, pp 139-150, 2000.

[11] M. Kalia, D. Bansal and R. Shorey, “MAC scheduling and SAR policies for Bluetooth: A master driven TDD pico-cellular wireless system”, *IEEE International Workshop on Mobile Multimedia Communications (MoMuc’99)*, San Diego, CA, USA, pp. 384-388, Nov. 1999.

[12] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks”, *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom’99)*, Seattle, USA, August, 1999.

[13] W. Simpson, “The Point-to-Point Protocol (PPP),” RFC 1661, July 1994.

[14] P. Bhagwat, I. Korpeoglu, C. Bisdikian, M. Naghshineh and S. K. Tripathi, “Bluesky: A cordless networking solution for palmtop computers,” *Mobicom’99*, Seattle, Washington, Aug. 1999.

[15] S. McCanne and S. Floyd, “NS-Network Simulator,” 1995, http://www-nrg.ee.lbl.gov/ns.

[16] D. Moldkar, “Review on radio propagation into and within buildings,” IEE Proc.-H, Vol. 138, No. 1, Feb. 1991.

[17] D. Duchamp and N. Reynolds, “Measured performance of a wireless LAN,” *17th Conference on Local Computer Networks*, Minneapolis, MN, pp. 494-499, Sept. 1992

[18] H. S. Wang and N. Moayeri,“Finite-state Markov channel - a useful model for radio communication channels,” *IEEE Trans. Veh. Technol.*, Vol. 44, pp. 163-171, Feb. 1995.

[19] G. T. Nguyen, B. D. Noble, R. H. Katz and M. Satyanarayanan, “A trace-based approach for modeling wireless channel behavior,” in *Proc. Winter Simulation Conf.*, Dec. 1996.

[20] M. Shimizu *et al.*, “New method of analyzing BER performance of GFSK with postdetection filtering,” *IEEE Trans. Commun.*, Vol. 45, No. 4, pp. 429-436, April 1997.

[21] R. Ganesh and V. Pahlavan, “Effects of local traffic and local movements on the multipath characteristics of the indoor radio channel,” *IEE Elect. Let.*, 26(12), pp.810-812, 1990.