Journal of High Speed Networks 10 (2001) 59–76
IOS Press

Performance of network formation and scheduling algorithms in the Bluetooth wireless ad-hoc network

Apurva Kumar, Lakshmi Ramachandran * and Rajeev Shorey
IBM India Research Laboratory, Block 1, Indian Institute of Technology, Hauz Khas, New Delhi 110016, India
Tel.: +91 11 6861100; Fax: +91 11 6861555; E-mail: {kapurva,rlakshmi,srajeev}@in.ibm.com

Abstract. Bluetooth is a promising new technology for short range, low power, low cost, pico-cellular wireless connectivity between mobile devices. Efficient clustering or topology construction algorithms play a very important role in the fast connection establishment of such networks. The performance of these algorithms is mainly dependent on the device discovery time, i.e., the time taken by a node to discover and to connect to another node in its radio range, as well as the protocols used for organizing them into the required topology. We define metrics for device discovery performance and evaluate them by simulating the inquiry routine in Bluetooth for various ad-hoc scenarios. We indicate the scenario where device discovery in Bluetooth can become a bottleneck. We then describe some recent results on topology construction algorithms for Bluetooth. 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. We propose and compare a number of MAC scheduling algorithms with the aim of improving the performance of asynchronous data traffic over a Bluetooth piconet that supports multiple active slaves. We study how the presence of circuit-switched voice impacts the performance of data traffic.

Keywords: Bluetooth technology, scatternet, piconet, device discovery, inquiry, paging, scheduling, cluster, topology, Media Access Control (MAC), Time Division Duplex (TDD), TCP, IP, UDP, utilization, throughput, end-to-end delays

1. 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. These are:

- 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.

Bluetooth technology allows for the formation of network units called *piconets* and a connected set of piconets, called *scatternets* without the involvement of a central infrastructure, in an ad-hoc fashion. This is made possible using distributed algorithms for inquiry and paging. While device discovery in this paper refers to the process of obtaining Bluetooth addresses and clocks of neighboring devices using inquiry procedures, paging is used to setup

*Present address: Trillium Digital Systems India Pvt. Ltd., 2nd floor, Discoverer building, International Tech Park, Whitefield Road, Bangalore 560 066, India. E-mail: l_ramachandran@trillium.com.

a connection with a particular Bluetooth device. On top of these procedures, protocols are required for a set of Bluetooth devices to efficiently self-organize into a scatternet.

The total delay in a Bluetooth network includes delays in the (i) pre-connection, and (ii) post-connection phase. Pre-connection delays refer to the delays in the device discovery and scatternet formation process. Once a connection is established between a master and a slave (post-connection phase), packets can be exchanged between the two nodes. In this phase, the end-to-end packet delay is composed of the queueing delay at various buffers in the Bluetooth protocol stack, transmission delay and delay due to scheduling at the MAC layer.

There are many features of Bluetooth that may significantly impact the performance of data traffic. 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.

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.

TCP [3] is the most widely used transport protocol for reliable data services over the Internet and it is therefore important to study the performance of TCP over Bluetooth. One of the aims of this paper is to study how the various MAC scheduling algorithms in Bluetooth impact the performance of asynchronous data traffic (such as TCP and UDP). A detailed study of the performance of asynchronous data traffic over a Bluetooth piconet is to appear in [12].

There has been some work on the formation of clusters [25,27]. Some recent work [28] has been done on cluster formation such that a node is either a clusterhead or is at most d hops away from a clusterhead. Some work has also been done on randomized algorithms for initializing packet radio networks [26]. Each of these papers employ different models of communication and allow the passing of different types of messages during cluster formation which turns out to be expensive in the Bluetooth communication model. Some work on Bluetooth network formation has been done in [29].

Since Bluetooth is distinct from existing wireless LANs, the earlier studies on the performance of TCP over wireless links [4–6] are not directly applicable to 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 polling algorithms for Bluetooth: round robin, exhaustive and a modified exhaustive scheduling algorithm named Fair Exhaustive Polling. As in [9], the assumptions in the paper are simplistic. The authors stress the importance of multi-slot transmissions to increase throughput along with keeping the delays low. Kalia et al. [11] have proposed some simple SAR policies and MAC Scheduling algorithms for Bluetooth. They propose scheduling policies that utilize information about the Head-of-the-Line (HOL) packet at the master and slave queues to schedule the TDD slots effectively. However, their work 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 2 describes the device discovery procedures in Bluetooth. We formulate the scatternet formation problem and describe some recent results for the same. In Section 3, we discuss the Bluetooth protocol stack. In Section 4, we propose various 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 5. In Section 6, we present simulation results with different MAC scheduling algorithms, both in the absence and presence of voice traffic in a piconet. We conclude by presenting a summary of our results and some suggestions for future work in Section 7.

2. Scatternet formation

Bluetooth piconets have a star topology, with a master at the center of the star communicating directly with a number of slaves. A set of connected piconets is termed a scatternet (see Fig. 1), and inter-piconet communication is through bridge nodes. A bridge could be a common slave between one or more piconets (slave-slave bridge), or a master in one piconet and a slave in others (master-slave bridge).

In this section, we first give a brief overview of the device discovery procedure in Bluetooth before proceeding to describe our enhancements and simulation results. We then formulate the scatternet formation problem and describe some recent results.

2.1. Device discovery in Bluetooth

Bluetooth technology allows for the formation of networks called piconets and interconnection of such networks, called scatternets without the involvement of a central infrastructure, in an ad-hoc fashion. This is made possible using distributed algorithms for inquiry and paging. While device discovery in this paper refers to the process of obtaining Bluetooth addresses and clocks of neighboring devices using inquiry procedures, paging is used to setup a connection with a particular Bluetooth device [1].

The Bluetooth physical layer uses frequency hopping which conforms to FCC regulations for the 2.4 GHz ISM band. It uses a 79 hop system with each hop occupying 1 MHz. These restrictions have implications on the device discovery process. One implication is: when a device wakes up to scan for inquiry messages, it can not do so at a fixed dedicated frequency. Instead the wake up frequency could be any of the 32 frequencies in the inquiry hopping sequence. The actual frequency used depends upon the clock of the scanning device, which is not known to the inquirer. Moreover since the onus of finding a new device is deliberately kept with the inquirer, the scanning device does not do continuous scanning. Hence, in the inquiry procedure there is uncertainty in both time and frequency at which the scanning device(s) might be listening. The overall connection establishment delays are dominated by inquiry delays. In the paging procedure, the hopping sequence and approximate value of the clock of the paged device are known through inquiry. Further, paging, unlike inquiry involves interaction of only two devices. These factors result in relatively small and predictable paging delays. It is with this in mind that we neglect the delays due to the paging procedure.

Due to the interaction of several devices during inquiry, the problem of finding the delay distribution or even average delays is not easily tractable. These delays depend upon the number of devices participating in inquiry procedure, the number of inquiring devices, bandwidth reserved for inquiry and the number of responses required from an inquiry. In this section we give a brief overview of the inquiry procedure used in Bluetooth and define various delays which are important in ad-hoc networks. We obtain these delays for various scenarios by means of simulation.

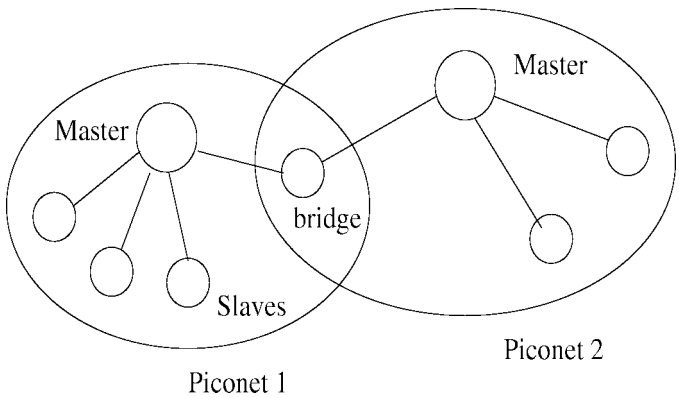

Fig. 1. Bluetooth scatternet with two piconets.

2.2. Definitions

In this section we explain some terms defined in the Bluetooth Specification [1].

- *Inquiry procedure*: The procedure carried out by a unit to discover other units.
- *Inquiry scan procedure*: The procedure carried out by devices which allow themselves to be discovered.
- *Device discovery procedure*: The combination of the above two procedures carried out in a distributed fashion.
- *GIAC*: General Inquiry Access Code is an inquiry access code common to all devices. All devices which wish to reply to inquiry will reply to an ID packet containing the GIAC.
- *DIAC*: Dedicated Inquiry Access Code is common for a dedicated group of Bluetooth units that share a common characteristic.
- *Inquiry hopping sequence*: 32 unique wakeup frequencies distributed over 79 MHz with a period length of 32. GIAC is used to derive this sequence.
- *Inquiry response hopping sequence*: 32 unique response frequencies which are in one-to-one correspondence to the current inquiry hopping sequence. GIAC is used to derive this sequence.

We define the following delays which are used to evaluate performance of the device discovery algorithm:

- *Inquiry delay*: The time elapsed between a unit starting an inquiry procedure and the earlier of the following two events: *MaxResponses* number of responses being received or inquiry timeout.
- *Device discovery delay*: The average delay in an ad-hoc environment for a device to discover any other device. The devices in the environment may or may not already be part of a network. The device discovery delay in an environment with *n* devices is given by:

$$D_{dd} = \frac{1}{n(n-1)} \sum_{a=1}^{n} \sum_{b=1, a \neq b}^{b=n} d_{ab}, \tag{1}$$

where d_{ab} is the average delay for device *a* to discover device *b*.

2.3. Overview of inquiry procedure in Bluetooth

The inquiry procedure used for device discovery in Bluetooth can be described with the help of Fig. 2. The first timeline shows the inquiring device. The 32 inquiry hop frequencies are divided into 2 trains of 16 frequencies each. The frequencies in a train are determined by the inquirer's clock. The 16 frequencies in the first train are repeated 256 times before the second train is started. Every even slot, 2 frequencies are sent and every odd slot, responses are awaited from devices scanning at these frequencies.

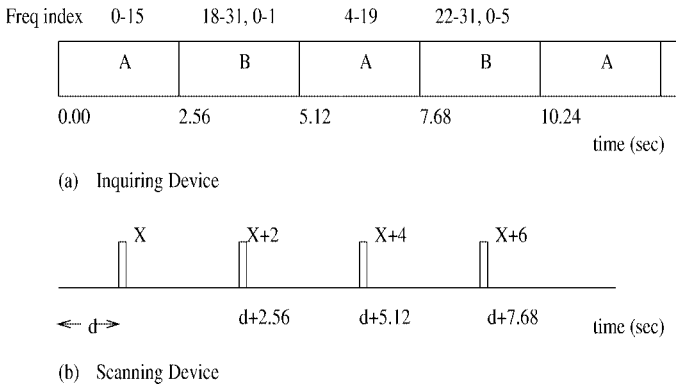

Fig. 2. Inquiry procedure in Bluetooth.

As seen from Fig. 2, train B is not exactly complementary to train A, but there is an overlap of two frequencies. So if we enumerate the frequencies in the A train as 0-15, frequencies in the B train are 18-31, 0-1. Thus frequencies 0 and 1 overlap. The overlapping is done because the scanning frequency changes by two between two consecutive inquiry scans as is explained below. The second timeline shows the scanning device. There is an uncertainty in time, d as well as frequency X which are not known to the inquirer. The scanning device scans for a window of 18 slots every 2.56 sec. The phase in the inquiry hopping sequence during scanning is determined by CLKN16-12, i.e., bits 12-16 of the native clock of the scanning device. This changes every 1.28 sec, and since scanning takes place every 2.56 sec, the scanning device scans at alternate frequencies in the hopping sequence as shown in the figure by X, X + 2, X + 4 and so on. In the absence of any errors, the scanning device will receive an inquiry packet by the time the second inquiry scan window is over. However, it does not send a response in the very next slot, but returns to the previous state (standby or connection), waits for a random period of maximum 1023 slots and returns to inquiry scan and listens at the same frequency. When it receives an inquiry packet again, it sends an FHS packet containing its address and clock information. The FHS packet contains the clock and device address of the sending device. The address is used to generate the frequency hopping sequence for paging the device and the clock is used to determine the phase in the hopping sequence. In this procedure even if the environment is error-free and collisions of FHS packets are neglected, device discovery delays could still be 10.24 sec. However when there are more than one inquiring devices, these delays would be still higher and difficult to predict.

2.4. Device discovery performance

Having described the device discovery procedure in Bluetooth, we now describe our simulation model. Our model makes the following assumptions regarding the system and the input parameters:

(i) The environment is assumed to be error-free. Our aim is to study the effect of interaction of several devices during inquiry and not the degradation caused by wireless errors.
(ii) All devices are considered to be in communication range of each other.
(iii) All devices taking part in device discovery reserve 10 percent of their resources for this purpose.
(iv) Inquiry timeout period is assumed to be 10.24 sec.

The performance metrics are:

– *number of inquiry responses per unit time*;
– *inquiry delay* ;
– *inquiry timeouts*.

Figure 3 shows the device discovery performance on the basis of these assumptions and the inquiry procedure described above for simulation time varying between 10–40 sec and number of device varying between 5 and 14. We observe that for simulation time less than 30 sec number of useful responses increases almost linearly with number of devices. Since number of responses in infinite time is an environment with N devices is $N(N-1)$, we see that less than 25 percent of the possible responses are obtained in 30 sec.

Figure 4 shows the average inquiry delays and inquiry timeouts in a simulation time of 50 sec in an environment consisting of 14 devices for several values of *MaxResponses*. The maximum value of inquiry delay can be inquiry timeout which is assumed to be 10.24 sec. Hence increasing *MaxResponses* results in more timeouts.

From the results we conclude that the inquiry procedure in Bluetooth does not perform very well in case of multiple simultaneous inquirers. The reason for this being the unavailability of a device doing inquiry for scanning. This causes an increase in inquiry delay and timeouts for other devices which results in higher average device discovery delays. Hence the device discovery procedure in Bluetooth imposes restrictions on ad-hoc applications which depend upon fast synchronization of a large number of devices.

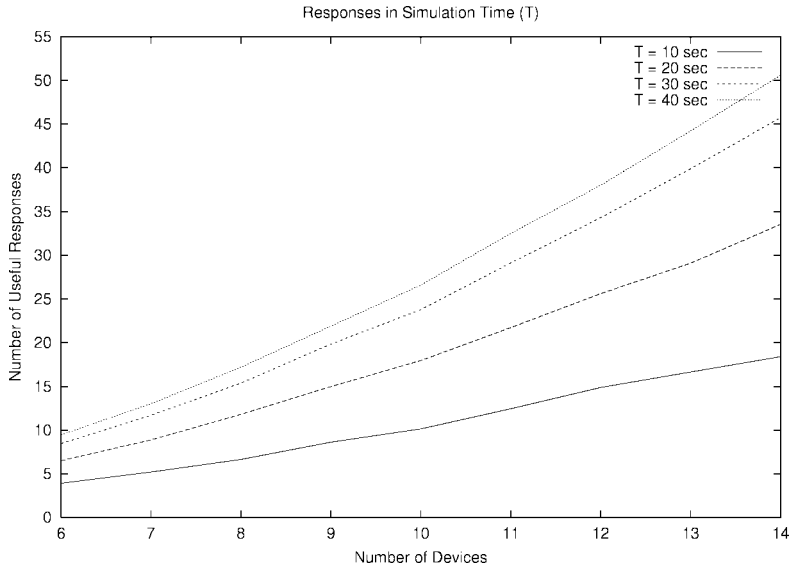

Fig. 3. Number of responses v/s number of devices.

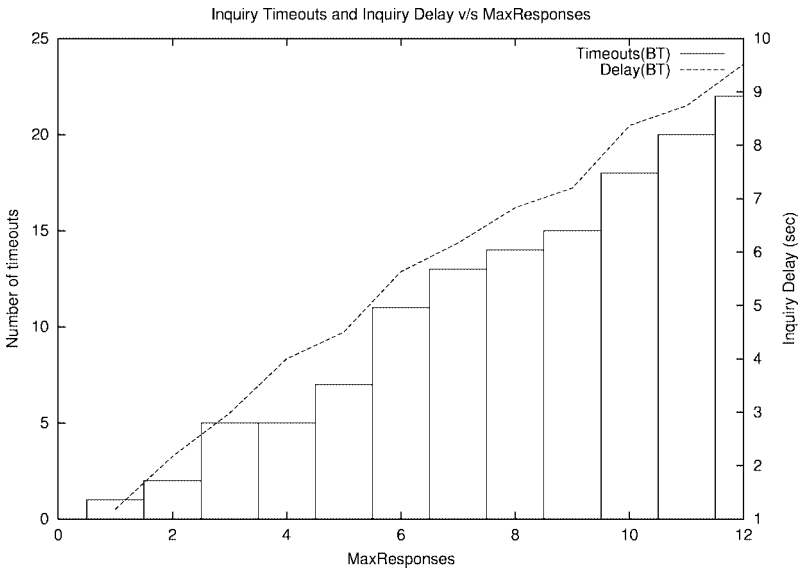

Fig. 4. Inquiry delay and timeouts.

2.5. Star-clustering algorithms

In [17], the authors have investigated the problem of distributed cluster formation for the Bluetooth communication model described above. The Bluetooth wireless ad hoc network is modeled as an undirected graph, where the set of nodes represents the Bluetooth devices, and there is an edge between two nodes if they are within radio range of each other. Each node has a unique Id known to itself, but not to other nodes. It is assumed that the underlying topology is a complete graph, and the total number of nodes, N, and the maximum number of nodes per piconet, S, are also known to all the nodes. The network is asynchronous, and there is no notion of global time, with each node keeping its own local clock.

The algorithms described in [17] apply to the case where a set of Bluetooth nodes are powered on at the same time, and need to efficiently organize themselves into a scatternet. These algorithms ensure that the nodes identify themselves as either master or slave nodes, and each slave also knows the master it belongs to. The resulting scatternet has a minimum number of connected piconets, given a bound on the number of nodes per piconet. One of the goals is also to avoid the exchange of roles between a master and a slave, which is an expensive operation. On termination, a single node (super-master) has complete information on all the clusters that have been formed. This node can then run any centralized algorithm and determine the bridges. Although there has been considerable research on the star clustering/leader election problems [19–24], they become infeasible to implement in this context, as they use models where the discovering devices broadcast their Ids, and exchange substantial information in the initial stages of the algorithm. In this section, we give a brief over view of the two-stage distributed randomized algorithm and a more detailed description of the deterministic algorithm.

The two-stage distributed $O(N)$ randomized algorithm for an N-node complete network proposed uses continuous inquiry and scan. This is an important idea and makes a device listen continuously in order to increase the probability of the message reaching another device. This corresponds to continuous inquiry and scan. The first stage of the algorithm is randomized, at the end of which each node either becomes a master-designate or a slave-designate. Each of the nodes conducts several rounds of Bernoulli trials (i.e., independent coin tosses) in order to determine this.

The second stage of the algorithm is used to correct the effect of the randomness by using a deterministic algorithm to decide on the final set of masters and slaves, and to efficiently assign slaves to masters. In this stage, all master-designates carry out inquiry, while all slave-desigates carry out inquiry scan. A master-designate becomes a master once it collects the maximum number of slave-designates. Timeouts are defined both for piconet formation as well as super-master election. The algorithm also uses the concept of Proxy-slaves, which are used in the super-master election. The first slave-designate that responds to an inquiring master-designate is made a Proxy-slave. This node continues scanning and responds to inquiry messages from other master-designates. The inquiry response message is made to carry the extra information that it is a response from a Proxy-slave. The election of the super-master is interleaved with the cluster formation in order to speed up the scatternet formation. Since the total number of nodes in the network is known to all the nodes, the node which has received information about all other nodes is declared the super-master. For a detailed description of the algorithm and the cases where the number of actual master-designates is greater than, less than or equal to the ideal number of masters, the reader is referred to [17].

An $O(N)$ deterministic algorithm is also proposed in [17]. The basic idea of this algorithm is that nodes discovering each other form a tree of responses, the root of each tree becoming a master.

The next problem of interest is when all nodes are not within radio range of each other, i.e., the underlying topology is an arbitrary graph. The feasibility problem of scatternet formation, i.e., the problem of organization of a set of nodes into a connected set of star-shaped clusters of bounded size, the connection between clusters being made through non-center nodes, has been proved to be NP-Complete [18]. The optimization problem of scatternet formation, i.e., the above problem with the additional requirement that the number of masters be minimized, is therefore NP-hard [18].

The greedy centralized algorithm proposed in [18] assumes that a central entity has knowledge of the underlying topology. The algorithm tries to minimize the number of piconets by marking nodes which have higher degree as masters. The algorithm proceeds as follows. Initially, all nodes are unmarked. The node with the highest degree in the entire graph is marked as a master. If the degree of a node marked as master is less than the cluster size, then all its neighbours are marked as its slaves. Otherwise, the neighbour set of this node is sorted in the increasing order of degrees and the first k1 nodes are marked as its slaves. The neighbourhood of each slave is searched for $k2 - 1$ nodes of highest effective degree, i.e., the degree after removing edges to already marked nodes. These nodes are marked as the next masters and the corresponding slave becomes a bridge. The algorithm proceeds in a breadth-first fashion until all nodes are marked. In cases where a master has degree greater than k1, and the remaining nodes in the neighbourhood do not get marked, they are marked as masters. However, there could be cases where the graph is disconnected, in which case, we make master-master edges, which correspond to master-slave bridges.

Approximation bounds for the number of piconets have also been derived for a restricted class of graphs, namely, the clique-coverable graphs. This algorithm is a $1 + 7k/N$ approximation, where the input graph with N nodes has a clique covering with k cliques. The reader is referred to [18] for the proofs.

Any algorithm intended for practical application to an ad hoc environment like that of Bluetooth should be completely distributed. There is no centralized entity which has complete information on the whole network. In order to minimize the number of message exchanges, while providing useful information to the nodes about their neighbourhood, each node is assumed to exchange its immediate neighbourhood information with each of its neighbours. This means that every node has a two-hop neighbourhood information. A distributed heuristic is also described in [18] and the performance of the two schemes have been compared. Simulation experiments show that both these algorithms perform well for dense graphs.

3. Bluetooth protocol stack

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 Fig. 5. Bluetooth uses an ad-hoc, piconet structure with a single master and upto seven slaves. For a detailed description, see [1] and [2].

3.1. 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:

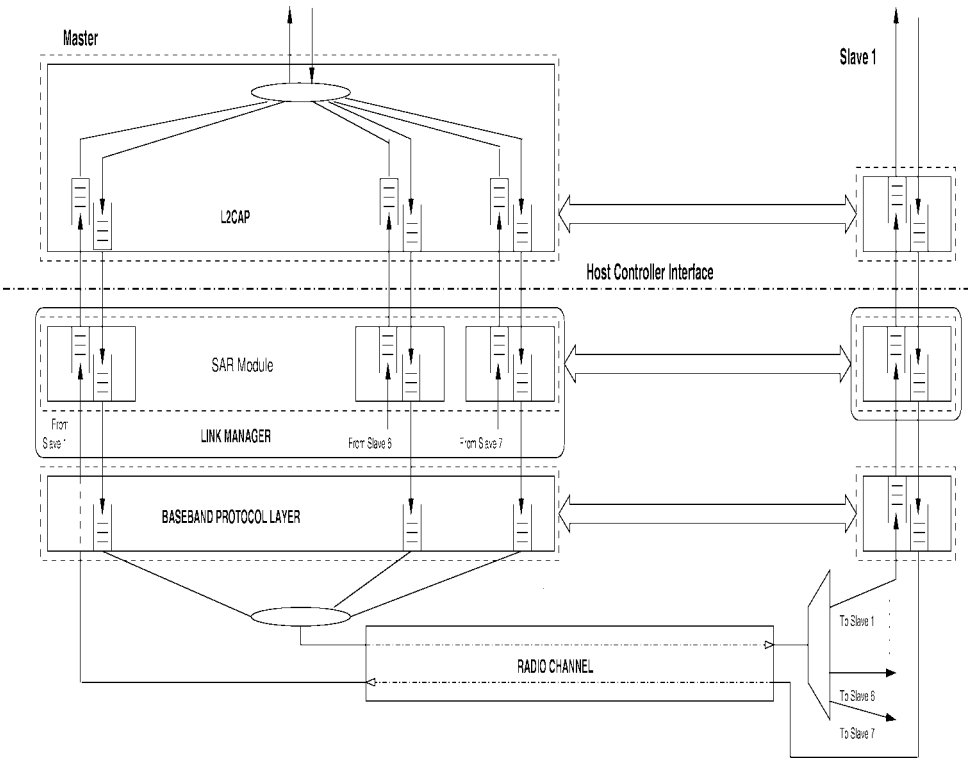

Fig. 5. The Bluetooth protocol stack.

(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 $625\ \mu$s 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.

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.

3.2. 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.

3.3. 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.

4. 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.

4.1. 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:

Queue priority based on flow bit: Per-slave baseband queues at the master and similar queues at the slave are maintained (as shown in Fig. 6). 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:

4.1.1. 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 the 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 it's 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.

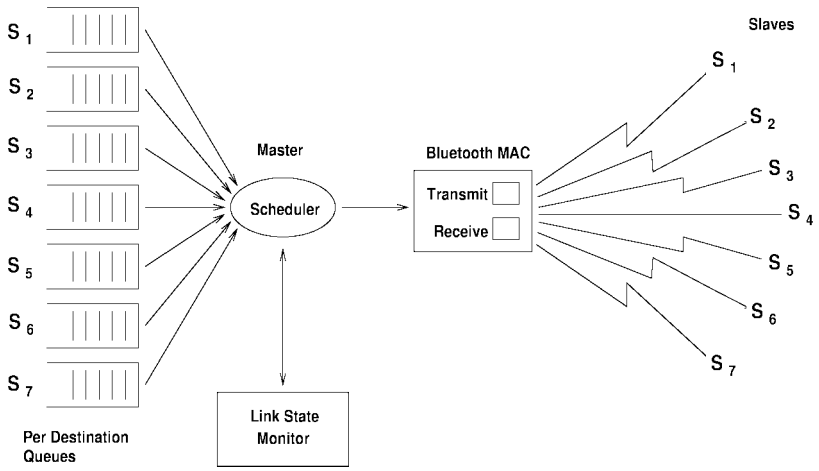

Fig. 6. MAC scheduling in Bluetooth.

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.

4.1.2. 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 6.1.
2. If $flow = 0$, one packet is transmitted for that queue, as in Round-Robin scheduling.

4.1.3. 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 the 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 mathematically. We compare the performance of these algorithms through simulations in Section 6.1.

4.2. Channel state dependent scheduling

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 6.1.2.

4.3. 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. We present the performance of data traffic in the presence of varying number of SCO links in Section 6.2.

5. 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 Fig. 5) as well as TCP/IP. The network is modeled as a Bluetooth piconet with one master and seven slaves.

The traffic sources shown in Fig. 7 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 in 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.

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, slaves 1 and 2 have persistent TCP (*ftp*) connections which are active from 0 to 60 and 10 to 20 sec respectively. Slaves 3 to 7 receive CBR traffic running over UDP with rates ranging from 5520 bps to 17664 bps, as shown in Fig. 7. We use TCP-Reno [8] in our simulations since it is one of the most common reference implementations for TCP.

5.1. Performance metrics

We have used 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 sec. 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. Notice 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.

6. Simulation results and performance evaluation

6.1. Scheduling algorithms

In this section, we assume that the channel is error free and simulate the algorithms described in Section 4.1.

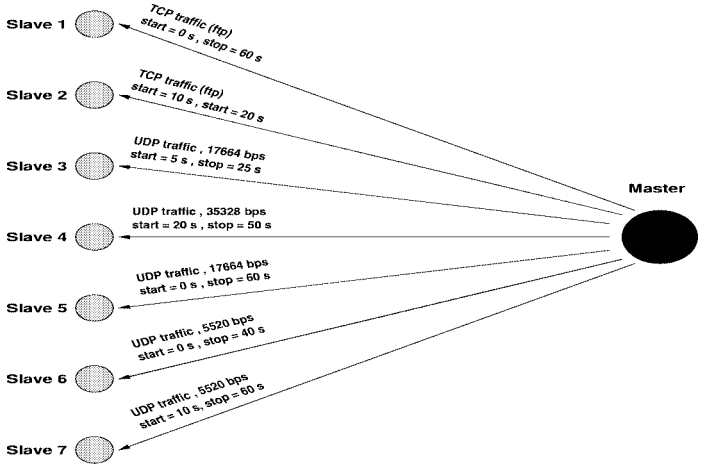

Fig. 7. Traffic sources used in simulation.

In Fig. 8, 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-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 Fig. 9, 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 Fig. 10, the link utilization under different scheduling algorithms is compared. From the graph, we see that high link utilizations are obtained for StickyAFP ($num\_sticky = 16$), Sticky ($num\_sticky = 16$) and AFP, as compared to Round-Robin. In Fig. 11, 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

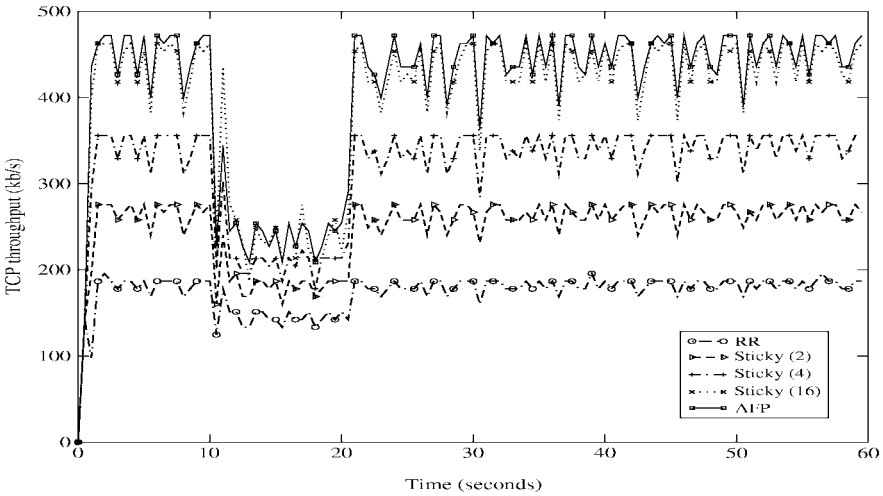

Fig. 8. TCP throughput vs time for AFP and Sticky.

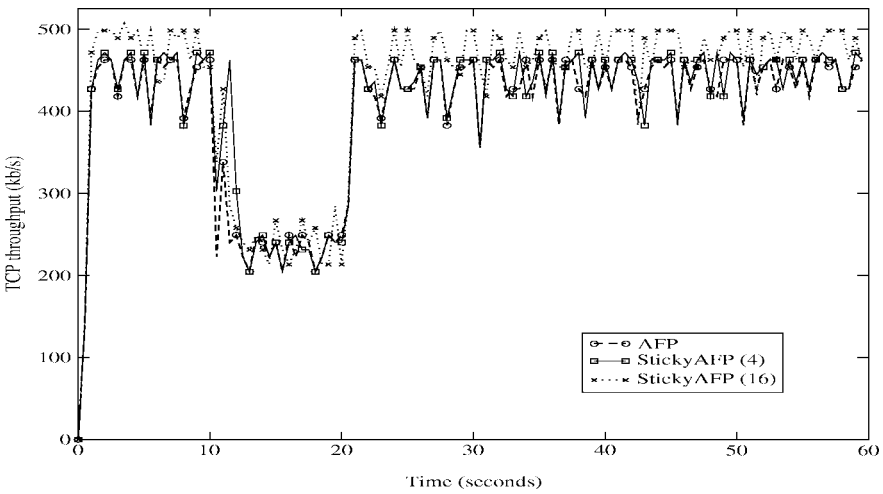

Fig. 9. TCP throughput vs time for AFP and StickyAFP.

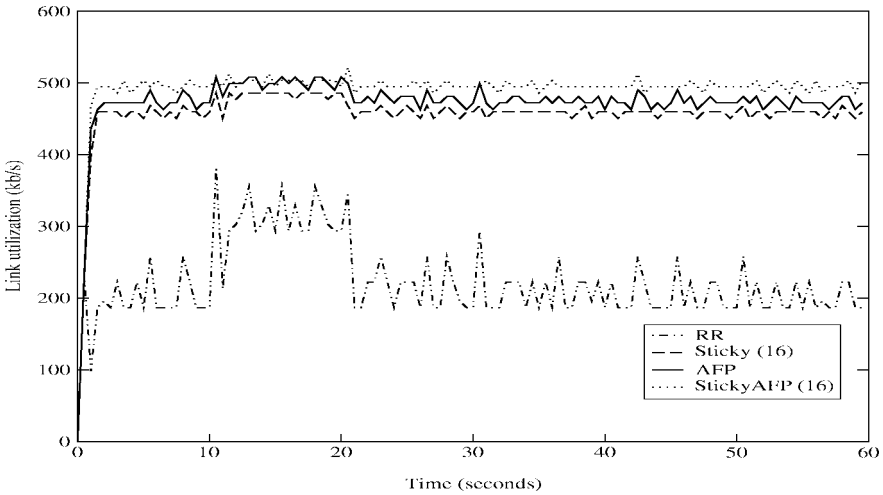

Fig. 10. Link utilization for scheduling algorithms.

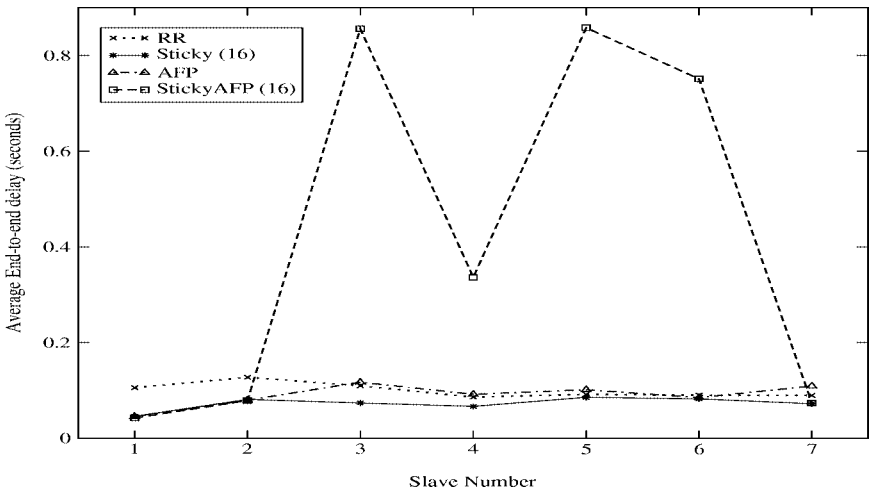

Fig. 11. End-to-end delay for scheduling algorithms.

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 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.

6.1.1. Static FEC/ARQ

Figures 12 and 13 show the link utilization and average end-to-end delay for different values of tx_thresh (maximum number of retransmissions) under AFP and CSDP-AFP in the presence and in the absence of a $2/3$ rate

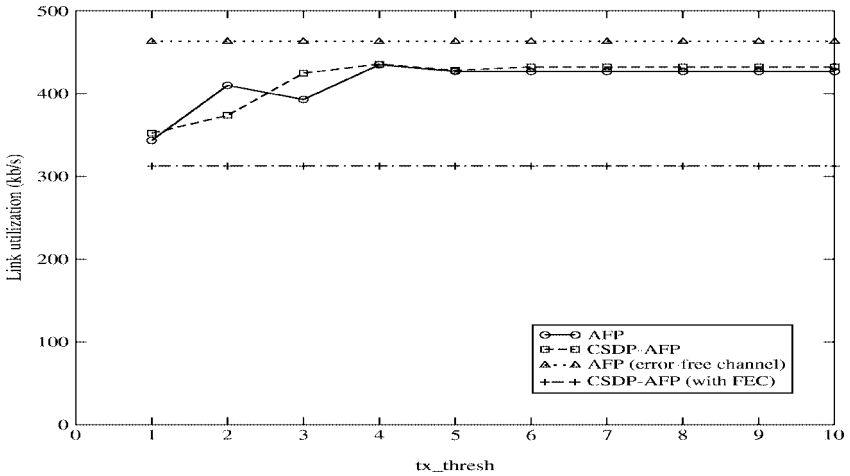

Fig. 12. Link utilization vs tx_thresh for versions of AFP.

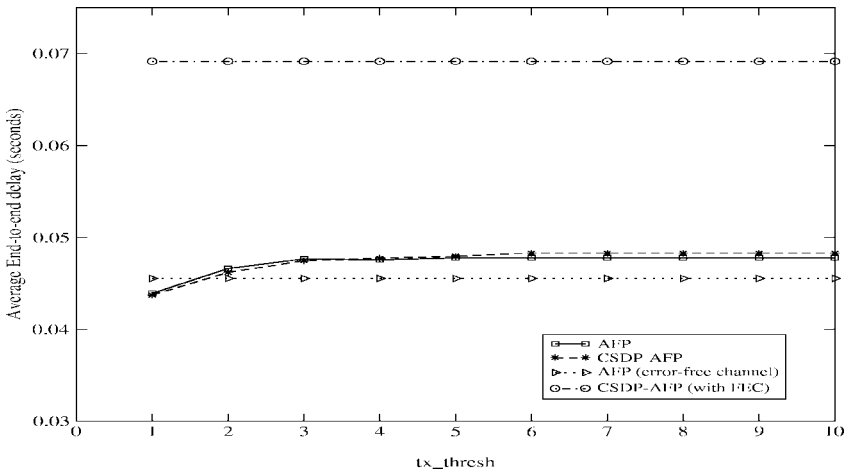

Fig. 13. End-to-end delay vs tx_thresh for versions of AFP.

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.

6.1.2. Effects of CSDP scheduling

Figure 14 clearly illustrates that the CSDP versions of the proposed scheduling algorithms do not give a significant performance improvement and that 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.

6.2. Varying voice connections

In Fig. 15, the throughput for the persistent TCP transfer (to slave 1) is shown using AFP and with varying number of SCO connections. The average end-to-end delays for each of the slaves under the same conditions

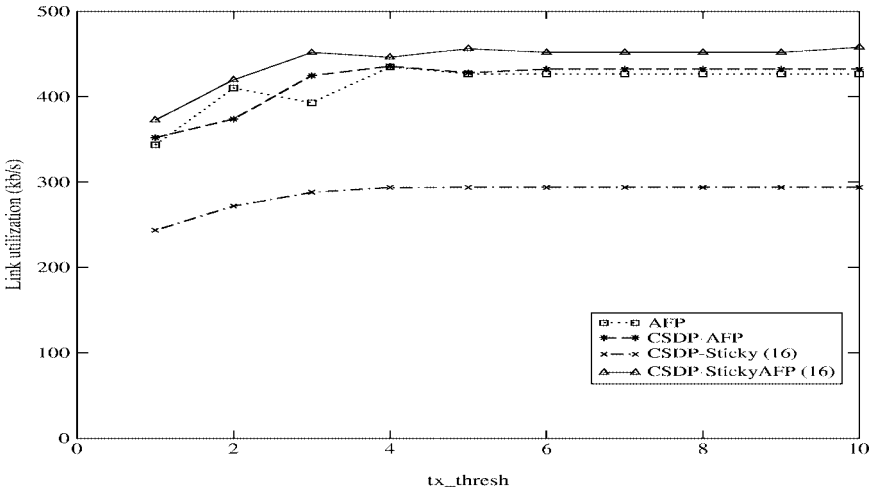

Fig. 14. Link utilization vs tx_thresh for CSDP algorithms.

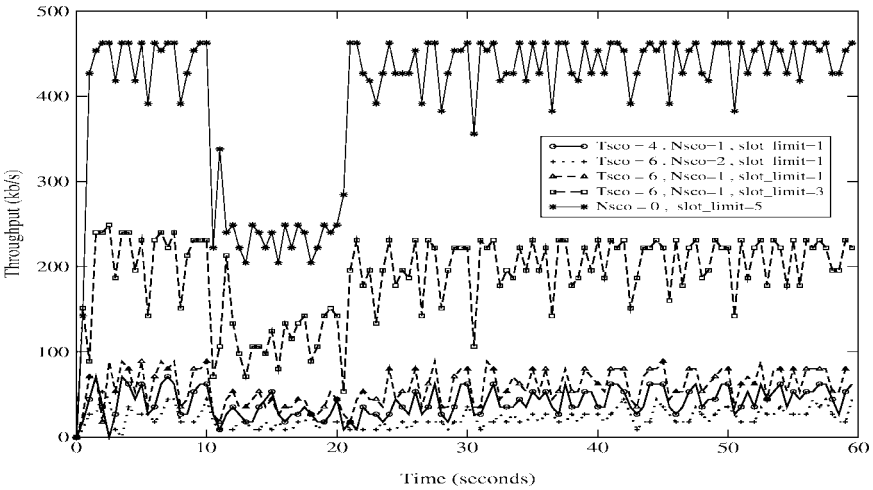

Fig. 15. Throughput degradation in presence of voice.

is shown in Fig. 16. 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. As expected, the highest throughput and lowest end-to-end delay shown in the graphs is for AFP with no voice connections.

7. Conclusion and future work

Motivated by the tremendous interest in the Bluetooth technology, in this paper, we have proposed and studied algorithms for (i) device discovery, (ii) scatternet formation and (iii) MAC scheduling.

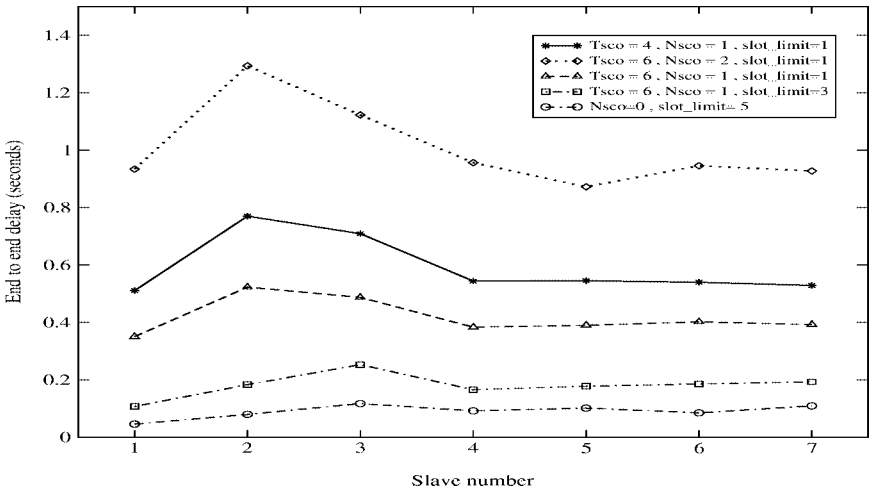

Fig. 16. End-to-end delay in presence of voice.

Device discovery in Bluetooth is a complex procedure involving inquiry and inquiry scan phases. We have studied the performance of device discovery in detail and have shown that the inquiry procedure in Bluetooth does not perform very well in case of multiple simultaneous inquirers.

We have reviewed recent results in scatternet formation: the problem of organization of a set of nodes into a connected set of star-shaped clusters of bounded size, the connection between clusters being made through non-center nodes, and the number of clusters being minimized.

The total delay in a Bluetooth network is a sum of delays in the (i) pre-connection, and (ii) post-connection phase. Pre-connection delays refer to the delays in the device discovery and scatternet formation process. Once a connection is established between a master and a slave (post-connection phase), packets can be exchanged between the two nodes. In this phase, the end-to-end packet delay is composed of the queueing delay at various buffers in the Bluetooth protocol stack, transmission delay at the wireless link and delay due to scheduling at the MAC layer.

Scheduling in a Bluetooth piconet is complex due to (i) TDD MAC, (ii) variable sized data (ACL) packets, (iii) reserved slots for voice (SCO) traffic. We have presented efficient MAC scheduling methods that 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.

We have 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 have 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.

We then 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 degrade the performance of data traffic.

Future work in scatternet formation could include mobility models and could take into account the service classes of the nodes for cluster formation. 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. An interesting area of future research is routing over a Bluetooth scatternet: how existing routing algorithms for ad hoc networks perform over a Bluetooth scatternet.

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* **7**(1) (2000), 28–36.

[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, in: *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* **6**(4) (1998), 485–498.

[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* **5**(6) (1997), 756–769.

[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*, 1997, 91–102.

[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, in: *Networking 2000*, Paris, France, 2000.

[10] N. Johansson, U. Korner and P. Johansson, Performance evaluation of scheduling algorithms for Bluetooth, in: *Broadband Communications: Convergence of Network Technologies*, D.H.K. Tsang and P.J. Kuhn, eds, Kluwer Academic Publishers, 2000, pp. 139–150.

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

[12] A. Das, A. Ghose, A. Razdan, H. Saran and R. Shorey, Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network, to appear in: *IEEE INFOCOM'2001*, Anchorage, Alaska, USA, 2001.

[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, in: *Mobicom '99*, Seattle, WA, 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* **138**(1) (1991).

[17] L. Ramachandran, M. Kapoor, A. Sarkar and A. Aggarwal, Clustering algorithms for wireless ad hoc networks, in: *Dial M for Mobility 2000, 4th International Workshop on Discrete Algorithms for Mobile Communication*, in conjunction with *ACM MobiCOM*, 2000.

[18] K. Balaji, S. Kapoor, A.A. Nanavati and L. Ramachandran, Star-clustering algorithms for wireless ad hoc networks, IBM Research Report, IBM India Research Laboratory, 2000.

[19] C.V. Ramamoorthy, J. Srivatsava and W.T. Tsai, Clustering techniques for large distributed systems, in: *Proceedings of IEEE INFOCOM*, 1986.

[20] H.H. Abu Amara, Fault-tolerant distributed algorithm for election in complete networks, *IEEE Trans. Computers* **37**(4) (1988), 449–553.

[21] H.H. Abu Amara and J. Lokre, Election in asynchronous complete networks with intermittant link failures, *IEEE Trans. Computers* **43**(7) (1994), 778–788.

[22] M. Gerla and J.T.C. Tsai, Multicluster, mobile, multimedia radio network, *ACM Baltzer Journal of Wireless Networks* **1**(3) (1995), 255–265.

[23] H.H. Abu Amara and A. Kanevsky, On the complexities of leader election algorithms, in: *5th Int. Conf. on Computing and Information*, 1993.

[24] G. Singh, Efficient distributed algorithms for leader election in complete networks, in: *11th Int. Conf. on Distributed Computing Systems*, 1991.

[25] B. Das, E. Sivkumar and V. Bhargavan, Routing in ad hoc networks using a spine, in: *IEEE Int. Conf. On Computers, Communication, and Networks*, 1997, pp. 1–20.

[26] T. Hayashi, K. Nakano and S. Olariu, Randomized initialization protocols for packet radio networks, in: *13th Int. Parallel Processing Symp* and *10th Symposium on Parallel and Distributed Processing*, 1999.

[27] R. Dechter and L. Kleinrock, Broadcast communication and distributed algorithms, *IEEE Trans. Computers*, 1986, 210–219.

[28] A.D. Amis, R. Prakash, T.H.P. Vuong and D.T. Huynh, Max-min D-cluster formation in wireless ad hoc networks, in: *Proceedings of IEEE INFOCOM*, 2000.

[29] A. Mizutani, T. Aihara, S. Shimotono and H. Ishikawa, Bluetooth scatternet, IBM Research report RT5176, IBM Tokyo Research Lab, 1999.