Available online at www.sciencedirect.com
SCIENCE DIRECT®
Ad Hoc Networks 3 (2005) 119–140
Ad Hoc Networks
www.elsevier.com/locate/adhoc

An enhanced and energy efficient communication architecture for Bluetooth wireless PANs

Carlos de M. Cordeiro [a,*], Sachin Abhyankar [b], Dharma P. Agrawal [c]

[a] *Networking Technologies Lab., Nokia Research Center, Tampere, Finland*
[b] *Qualcomm Incorporated, San Diego, CA, United States*
[c] *OBR Center for Distributed and Mobile Computing, Department of ECECS, University of Cincinnati, Cincinnati, OH, United States*

Available online 20 August 2004

Abstract

Bluetooth is a radio technology for Wireless Personal Area Networking (WPAN) operating in the 2.4 GHz ISM frequency band, and allows devices to be connected into short-range ad hoc networks. The Bluetooth medium access control protocol is based on the Master/Slave paradigm wherein any communication between slave devices has to go through the Master. While this model provides for simplicity, it incurs a longer delay between any two slave devices due to far from optimal packet forwarding, the use of double the bandwidth, and also additional energy wastage at the Master. Moreover, if more than two devices want to communicate as a group, this can only be achieved by either multiple unicast transmissions or a piconet-wide broadcast, clearly resulting in inefficiency. In this paper, we propose a novel Dynamic Slot Assignment (DSA) scheme whereby the Master device dynamically assigns slots to Slaves so as to allow them to communicate directly with each other without any Master intervention. This proposed communication architecture also provides for Quality of Service (QoS) requests, admission control, and multi-device conversation by which a multicast-like communication is implemented within a piconet. Through extensive simulation, we observe that DSA drastically enhances Bluetooth performance in terms of delay and throughput, while significantly reducing power consumption at the master and the overall piconet.

Keywords: Bluetooth; Energy efficiency; Communication architecture; Performance evaluation

1. Introduction

Bluetooth [1,22,24] is a wireless communication technology that provides short-range, semi-autonomous radio network connections in the

* Corresponding author.
E-mail addresses: carlos.cordeiro@nokia.com (C. de M. Cordeiro), sachin.abhyankar@qualcomm.com (S. Abhyankar), dpa@ececs.uc.edu (D.P. Agrawal).

1570-8705/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2004.07.002

2.4 GHz Industrial-Scientific-Medical (ISM) band, and can establish ad hoc networks, called piconets. It has also been chosen to serve as the baseline of the IEEE 802.15.1 standard for Wireless Personal Area Networks (WPANs) [2], which can support both synchronous traffic such as voice, and asynchronous data communications. In Bluetooth, two or more devices using the same channel form a piconet, where one unit operates as a master and the others (at most seven active at a time) act as slaves. Bluetooth operates on a Master/Slave concept wherein the Master periodically polls the Slave devices and only after receiving such a poll is a Slave allowed to transmit. A Master device can directly control up to seven active Slave devices in what is defined as a piconet, with the Master transmitting in even numbered slots and the Slaves in odd numbered slots. Multiple piconets can be linked together through bridge devices to form what is called a scatternet.

So far, we can envision three waves of Bluetooth-based applications. Initially, Bluetooth was designed to enable a wide range of devices such as laptops, PDAs, mobile phones, and headsets, to form ad hoc networks in a semi-autonomous fashion [1]. The second wave of applications was the development of access points (with functionality similar to IEEE 802.11 access points) enabling hundreds of Bluetooth units to access the wired network in places such as theaters, stadiums, conferences, pavilions, and so on [28,29]. However, in the third wave of applications, the low cost, effortless and instant connection provided by Bluetooth technology has also become attractive for automatically forming an ad hoc network of a large number of low-power sensor nodes [30,31]. These sensor network applications are characterized by thousands of nodes embedded in the physical world, and a Bluetooth RF link to enable them to form a network by only bridging these sensor nodes within radio range [22,32,33]. These calls for solutions to be developed that are applicable to both small scale and large scale Bluetooth networks, while keeping interference at minimum levels [21,23] due to the large number of piconets. As we shall see, DSA satisfies such scalability requirements as opposed to existing solutions.

By virtue of the Master/Slave communication model, the Bluetooth medium access provides for simplicity, low power (as compared to other standards), and low-cost, these being the major forces driving the usefulness of the technology. However, this design choice also brings in major shortcomings, such as the inability for slaves to communicate directly with each other since their packets must be forwarded through the master device [1,22,24]. Moreover, there is no built-in support in Bluetooth for applications which require group communication [17,27], while this can only be achieved by either multiple unicast packets or by a piconet-wide broadcast. As a result, packet forwarding among slaves in Bluetooth is sub-optimal, bandwidth is wasted by forwarding through the master, end-to-end packet delay increases, and power consumption is significantly increased at the master unit due to its frequent medium access for both transmission and reception. Therefore, the adoption of the Master/Slave paradigm in its present form does not seem to be the most adequate solution.

To overcome these issues and address the shortcomings of the current Bluetooth Master/Slave communication model, we propose a novel Dynamic Slot Assignment (DSA) scheme to be coordinated by the master of a piconet [35,36]. Based on the piconet traffic patterns, the master device dynamically allocates slots for direct communication between slaves. This way, packets do not need to be forwarded by the master while it periodically re-evaluates slot assignments and changes it accordingly. This novel communication architecture enables for not only direct slave-to-slave communication but also serves as multi-slave communication, hence emulating a group (i.e., multicast-like) communication within the piconet. We have carried out extensive simulations of DSA and observe a drastic enhancement in current Bluetooth performance. In unicast scenarios, piconet throughput increases by up to 300%, delay is reduced to one-third, and overhead is halved, whereas in multicast-like communication throughput boosts up to 500%, delay decreases to approximately one-thirtieth, and overhead is merely one-seventh of existing Bluetooth implementations. Additionally, we have also shown

that power consumption at the master is dramatically reduced due to lesser number of transmissions/receptions, and in certain scenarios up to 80% reduction is achieved.

The rest of this paper is organized as follows. Section 2 gives an overview of the Bluetooth technology and also provides the motivation for our work, while Section 3 describes our proposed Dynamic Slot Assignment scheme. Next, Sections 4 and 5 present the simulation methodology and results of extensive runs, along with comparisons of our scheme with existing Bluetooth. Related work is then given in Section 6. Finally, the paper is concluded in Section 7.

2. Bluetooth overview and motivation

The details of the Bluetooth system, architecture and protocols are defined in [1]. A brief overview is provided here for completeness. Bluetooth is a short-range (up to 10 m) wireless link technology aimed at replacing cables that connect phones, laptops, PDAs, and other portable devices. Bluetooth operates in the ISM frequency band starting at 2.402 GHz and ending at 2.483 GHz in the USA and most European countries. A total of 79 RF channels of 1 MHz width are defined, where the raw data rate is 1 Mbit/s. A Time Division Duplex (TDD) technique divides the channel into 625 μs slots and, with a 1 Mbit/s symbol rate; a slot can carry up to 625 bits. Transmission occurs in packets, where DHx (without forward error correction) and DMx (with forward error correction) packets are usually employed for data traffic, while HVx packets are used for voice applications. Here, x represents the number of slots and can be equal to 1, 3 or 5.

The Bluetooth specification [1] defines two different types of links for data and voice applications. The first link type Synchronous Connection Oriented (SCO) is treated as a circuit-switched, point-to-point traffic, whereas the second link type Asynchronous Connectionless Link (ACL) acts as a packet-switched, point-to-point data traffic. Usually, SCO links are used for audio applications with strict quality of service requirements where packets are transmitted at predefined regular intervals, while ACL links are often used in data applications where there is no such strict requirement on end-to-end delay. At most, three SCO connections can be supported within a piconet and the polling cycle varies when connections of such a type are present, ranging from 6 for HV3 packets to 2 for HV1 packets. Stating that SCO packets from a given connection have a polling cycle of 6 slots means that one SCO packet has to be sent in every 6 slots, so as to achieve the 64 Kbps bandwidth required for voice applications. On the other hand, ACL connections do not have such requirements and the polling cycle can be expanded or shrunk according to the number of slaves in a piconet and their traffic demands. For simplicity, we assume that every SCO connection uses HV3 packets which have a polling cycle of 6 slots, even though our scheme is applicable to any SCO packet type. Fig. 1 illustrates a full polling cycle highlighting the use of 1, 3 and 5-slot packets.

Fig. 2 depicts a Bluetooth piconet comprised of five devices. Fig. 2(a) illustrates the case where the slave device S_1 communicates with another slave S_3, and all packets have to be forwarded through the master device M. Here, the packet forwarding between S_1 and S_3 is clearly sub-optimal, bandwidth is wasted by forwarding through the master

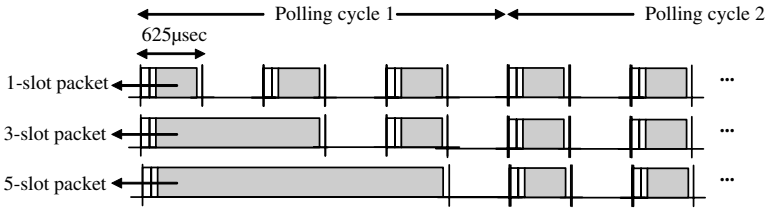

Fig. 1. Packet transmission in Bluetooth.

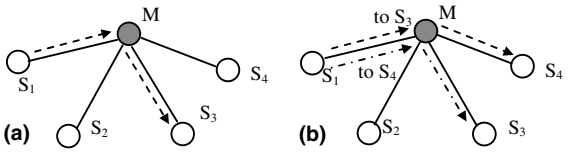

Fig. 2. Master/Slave communication in Bluetooth: (a) forwarding is done through the master; (b) communication with more than one device.

M, end-to-end packet delay increases, and, as shown later, power consumption is significantly increased at the master unit due to its frequent medium access for both transmission and reception. Therefore, we believe that for existing wireless and mobile devices which possess limited battery power and for application efficiency, the Master/Slave paradigm of Bluetooth should not be adopted in its current form.

In case slave S_1 wants to send packets (e.g., a business card) to both S_3 and S_4, it has to send two unicast packets as illustrated in Fig. 2(b). On top of that, these multi-slave packets may have to be kept in a device's buffer for as many polling cycles as is the number of recipients, since a device can only address one other unit in a given polling cycle. Note that a piconet-wide broadcast may be a feasible option, however as the master is the only device capable of sending a broadcast, packets will still have to be forwarded by it. Additionally, the trade-off between when to employ multiple unicasts or a broadcast is still unclear since slaves within a piconet are stateless. Hence, given the importance and wide applicability of multicasting, a basic support for group communication should also be provided.

3. The Dynamic Slot Assignment (DSA) scheme

To address these major shortcomings in the Bluetooth design while keeping the simplicity of the Master/Slave paradigm, we propose a novel Dynamic Slot Assignment (DSA) scheme [35,36]. The basic idea behind DSA is to appropriately manage the polling cycle conducted by the master of the piconet. As devices initiate or terminate communication with each other within the piconet, we restructure the piconet polling cycle (expanding it with a new connection or shrinking it upon termination), build the transmission schedule for each unit, and then propagate the resulting schedule to the members of the piconet. This way, slaves know exactly in which slot to transmit and/or to listen. Therefore, not only direct communication between slave devices is supported, but also a multicast-like communication is achieved by having destination slaves listen to the same slot. If we assume a uniform distribution of connection requests within a piconet, approximately 75% of all connections would constitute slave-to-slave communication [35], thereby stressing the need for supporting and optimizing such cases.

3.1. Connection request, release and admission control

In order for the piconet master to optimally assign and reserve slots for piconet devices, it should know the Quality of Service (QoS) requirements for each connection. In DSA, whenever a slave device wishes to establish a connection with another device (whether another slave or the master itself), it sends a CONNECTION_REQ message to the piconet master, specifying in its payload: (i) the destination (if unicast) or destinations (if multicast) address to which the device wants to establish a connection; (ii) the baseband packet type to be used in its transmission; (iii) desired transmission rate; and (iv) acceptable transmission rate. For instance, when slave S_1 in Fig. 2(a) wants to establish an FTP connection with slave S_3, it could send a connection request to its master M, as CONNECTION_REQ(S_3, DH5, 30, 50), while it would send a CONNECTION_REQ(S_3, S_4, DH5, 30, 50) for a multicast FTP connection to slaves S_3 and S_4. The transmission rate defines the frequency (in number of slots) a device desires to transmit. In our last example, slave S_1 notifies the master that it desires to transmit one DH5 packet every 30 slots, but it can accept to transmit in every 50 slots if the desired transmission rate cannot be supported.

When a connection request arrives at the master, it takes the requesting slave address (contained in the packet header), the packet type—which identifies the type of connection (voice or data)

requested, the destination address contained in the payload, and the transmission rates. If the master can grant this request, it allocates a unique identifier to the connection, recalculates the appropriate schedule (detailed in the following subsection), and broadcasts the scheduling information to all active slaves. If the connection cannot be supported with this QoS level, the master returns a CONNECTION_REJ message back to the source. In our earlier example, having requested a DH5 packet does not guarantee that the slave's request will be granted. As explained later, based on the current traffic and schedule, the master decides which packet is the best and propagates this information to all slaves with a broadcast. Also, we shall see that the traffic type influences the slot assignment, as SCO packets have to be scheduled periodically due to their time constraints, while no such restriction is present for ACL connections.

An important issue arises in the case of bi-directional flows (e.g., TCP traffic) as slots have to be reserved for the reverse traffic. In this case, it is up to the destination (here, destination is a broadly defined term and can be a Bluetooth device, a TCP entity, or an application) to similarly make a slot reservation in the reverse direction through a CONNECTION_REQ message. However, as this message is sent in response to a bi-directional flow, it is handled differently by the master who immediately allocates the required slots specified within the request (e.g., DH3 for TCP ACKs) to be used in the reverse direction. If the master cannot satisfy this reverse connection request due to absence of enough resources, it returns a CONNECTION_REJ message back to the source and also drops the associated forward connection, along with its reserved slots, which generated this bi-directional flow. We note that this special TCP support is *optional* in DSA and has to be configured/requested by an external agent in order to become effective.

Similar to connection requests, slaves also send connection termination messages to the master device. A slave transmits a CONNECTION_REL message, specifying the connection identifier as originally allocated by the master. Returning to the previous example of Fig. 2(a) where we assume $C_{1,3}$ as being the connection identifier, upon termination of the communication between S_1 and S_3, the slave S_1 would simply send a CONNECTION_REL($C_{1,3}$) to the master device M. The master M would free the slot allocated previously to this connection, calculate the new schedule for the remaining slaves and redistribute the new schedule within the piconet. Also, the master associates an idle timer (T_{IDLE}) with each connection to handle the case when a device moves out of range or the connection is idle for too long. Upon expiration of T_{IDLE}, the connection is terminated and its resources are freed up. Note that the master keeps track of all connections within the piconet in order to assign slots to the devices; however, it is not generally believed [21] that piconets, which are capable of having at most eight devices [1], will have a large number of connections. Given that, in our design, we assume that at most CON-$N_{THRES} = 16$ connections can be simultaneously present in a piconet, which can be said to be a very reasonable number for most piconet applications.

In the following section, we give details on how the slot scheduling and assignment is performed in DSA.

3.2. Slot scheduling and assignment

Every time the master of a piconet receives either a connection request or termination, it computes a new schedule for devices. This schedule contains information about which slot(s) belong to which device, and in which slot(s) a given slave needs to listen. With this mechanism, devices are able to directly talk to each other in either a unicast or multicast-like communication.

The schedule has to be transmitted to all the slave devices so that each one of them can determine when to transmit and when to listen. For that, we have defined a new broadcast message called SLOT_SCHED, which has a format depicted in Fig. 3 and is described in detail in Section 3.3. For simplicity, we assume that broadcast messages are reliable. Several approaches could be used to achieve reliability such as broadcasting the same message more than once or employing some sort of reliable broadcast. In our implementation, reliability is achieved by retransmission of the same broadcast message three times.

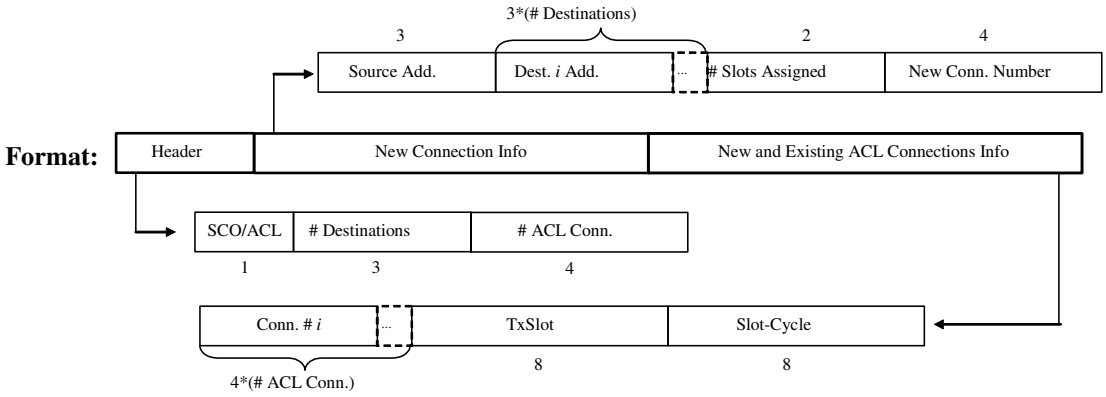

Fig. 3. Scheduling message format (numbers expressed in bits).

3.3. Slot scheduling message format

Fig. 3 illustrates the proposed format for the scheduling message employed by DSA, with the size of each field in bits as indicated. With this single message format, we can provide both unicast and multicast-like communication within a piconet. As we can see from Fig. 3, the scheduling message consists of three parts: the header, the information about the new connection request which caused the scheduling of this new message, and information about the slot assignment in the new and existing ACL connections. As we shall see, SCO connections are treated differently since they have a fixed scheduling.

The header part of the message is composed of three fields. The first field indicates whether the connection request, that caused the transmission of this new scheduling broadcast, corresponds to either ACL or SCO connection. Alternatively, this field determines whether the information about the new connection contained in the second part of the message is related to an SCO or ACL connection. This makes a difference in the slot scheduling as ACL connections can be scheduled at any time, while SCO connections have to appear at every polling cycle, where a polling cycle is equal to 6 in Bluetooth (see Section 2). The second field of the header indicates the number of destinations addressed by this new connection request. In case of a slave-to-slave communication, this field would be equal to one, whereas it would be equal to the group size in a multi-device communication. If this broadcast message is the result of a connection termination, this field contains zero. The third and final field in the header indicates the number of ACL connections contained in the third part of the scheduling message (to be soon described). In other words, this field is used to determine the boundaries of the message.

The second part of the scheduling message provides information about the connection request that has caused the generation of this scheduling broadcast, and absence of this part implies a connection termination. As mentioned earlier, both unicast and multicast-like communications are supported by this scheduling message. With this in mind, the first field indicates the source of data for the connection (i.e., the device which generated the connection request), followed by as many destination devices as specified in the number of destinations field contained in the header. Since each Bluetooth active member address is 3 bits [1], the size of this field is a multiple of 3. Next is the field which indicates how many slots have been allocated by the master for transmission by the source of the connection. Recall from Section 3.1 that together with the connection request message, the requesting device also specifies the type of packet it wishes to use in its transmissions. Depending upon the traffic pattern and the presence or absence of SCO connections, the master may or may not meet the device's request. For example, assume that a device S_1 is currently engaged in a

SCO connection with device S_2. In the mean time, another slave S_3 requests to establish a connection with slave S_4 for a file transfer by employing DH5 packets. It is well known that SCO connections are periodically scheduled in every polling cycle due to their strict QoS requirements. Since the Bluetooth slot cycle is equal to 6 and two out of these 6 slots are already being used for a SCO connection between S_1 and S_2, the master cannot satisfy slave S_3's request to use DH5 packets, as only 4 slots out of 6 are available. Therefore, in this case, the master would respond with a value 3 in this field, thus indicating that slave S_3 can use at most 3-slot packets. Finally, the last field of this message part is devoted to assigning a unique connection identifier to each connection within the piconet. The allocation of this identifier is managed by the master and communicated to the slaves in this broadcast message. This is the number that is used by the source slave when sending a connection release message to the master, and is also employed in the third part of the message scheduling.

The third and last part of the scheduling message contains the information about the schedule itself, that is, when each device is supposed to transmit and/or receive. The first field(s) contains lists of all the connection identifiers currently present in the piconet, and the order in which a connection identifier appears in this field determines the order in which devices associated with this identifier (either as source or destinations) have their slots assigned. To indicate to the slaves the starting slot for transmission (in case of the source of a connection) or reception (in case of the destination(s)), we use the field called *TxSlot* (transmit slot). The values permitted for *TxSlot* are 1, 3, or 5, depending upon the type of packet a connection uses. Lastly, as we have already mentioned, in DSA we employ a scheme where we expand or shrink the slot-cycle according to the number of ACL connections in the piconet. Therefore, conveying the slot cycle information to all slaves is the objective of the last field, called *Slot-Cycle*, as shown in Fig. 3. This field is always a multiple of 6, as this value is the required periodicity of SCO connections. Based on that, a device determines the slot—$slots = polling\_cycle \times index\_in\_list) + Txslot$—in which it is supposed to transmit and/or receive. Here, *polling_cycle* depends upon the SCO packet in use and is equal to 6 in case of HV3 packets, *index_in_list* is the index where the device's associated connection identifier appears in the first field of the third message part, and *TxSlot* is as described earlier.

3.4. Example scenario

To better illustrate the dynamics of DSA, consider again Fig. 2(a) where we have a piconet composed of 5 devices, one master M and four slaves labeled S_i, for $0 < i < 5$. For simplicity purposes, we do not include transmission rate in our discussion here, but note that QoS admission control is currently incorporated into DSA implementation, and consider unidirectional traffic only while bidirectional traffic is also presently supported by DSA. Assume that no traffic currently exists in the piconet, at which time slave S_1 sends a connection request to the master M in order to establish an ACL connection with slave S_2, specifying DH5 as the desired packet for its transmissions. Upon receipt of the connection request, the master M allocates a connection identifier, say $C_{1,2}$, to the connection between S_1 and S_2 and calculates the scheduling message to be broadcast as follows.

Initially, when there is no traffic, the slot cycle in the piconet is equal to 6 as polling is the only activity and is carried out by the master M. Fig. 4(a) illustrates this scenario, where slots are numbered from 1 to 6. Upon receipt of the aforementioned ACL connection request, the master M increases slot cycle by 6—making it 12—and assigns the first slot cycle for direct communication between the requesting device S_1 and destination S_2, while it always keeps the last slot cycle for polling (see Fig. 4(b)). Therefore, we see that the slot cycle is directly proportional to the number of ACL connections existing within a piconet, being always a multiple of 6. During the last slot cycle used by the master for standard polling, priority is given to those devices currently without open connections in order to achieve fairness, while those devices engaged in traffic connections are polled last.

The master also has to indicate the source and destination slaves (S_1 and S_2 in our example) the index within the assigned slot cycle where source

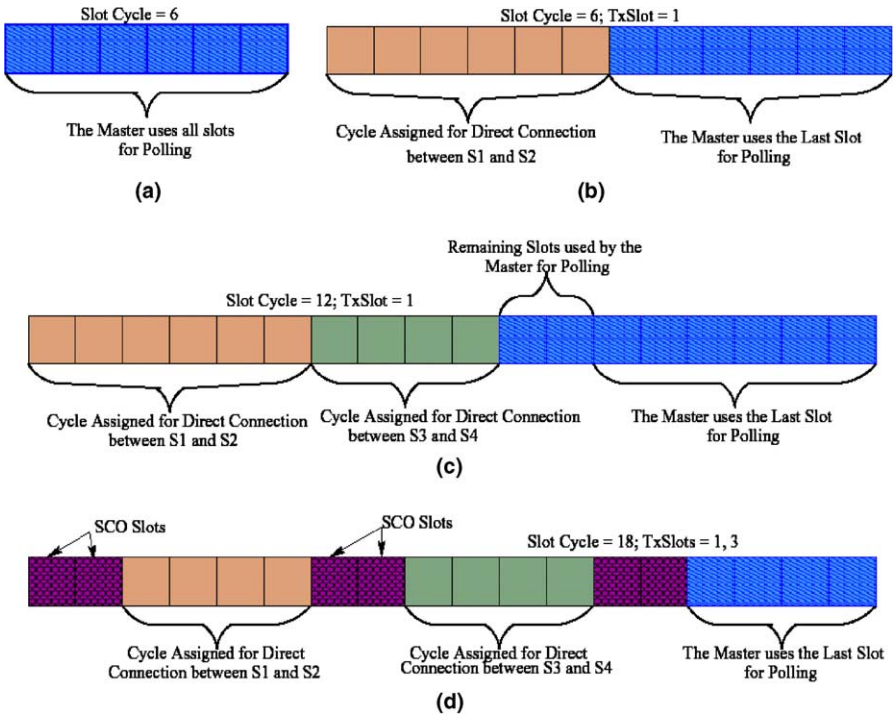

Fig. 4. Dynamic assignment of slots and expanding slot cycle: (a) no traffic (base slot cycle); (b) one ACL connection (slot cycle expanded); (c) two ACL connections (slot cycle expanded); (d) two ACL connections and one SCO connection (SCO traffic is scheduled periodically).

(S_1) is supposed to transmit and the destination (S_2) is supposed to listen. This is accomplished by the field *TxSlot* where, in our example, the master M would set *TxSlot* to 1 as to allow slave S_1 to use the entire slot cycle for the transmission of its DH5 packets. The scheduling message is then broadcast to all slaves. If, for instance, another ACL connection request to employ DH3 packets from slave S_3 to S_4, arrives at the master M, it proceeds in a similar manner, by assigning a connection identifier, say $C_{3,4}$, increasing the slot cycle from 12 to 18 to accommodate the new connection, building the new schedule, and broadcasting it throughout the piconet. This new schedule is illustrated in Fig. 4(c).

Assume now that a SCO connection request from device S_1 to S_3 arrives at the master M. Given the periodicity of SCO traffic which has to be present in each and every slot cycle, the master now has to reorganize the slot assignments. In our example, the connection $C_{1,2}$ will no longer be allowed to use DH5 packets as this would prevent SCO packets from being sent during this slot cycle, while the connection $C_{3,4}$ could continue employing the same packet type. To incorporate the SCO traffic within the schedule, the master changes the value of *TxSlot* from 1 to 3 indicating to all ACL connections that they will, from now on, transmit starting at slot 3, while the SCO connection is assigned the first two slots of every slot cycle. As a matter of fact, in DSA, SCO connections always use the first slots of a cycle while ACL connections follow next. Furthermore, note that contrary to ACL connections which have their polling cycle expanded according to the number of existing connections, the slot cycle of SCO connections is always 6, which means that SCO packets have to be transmitted in every slot cycle and hence need not be specified in the scheduling message. This way, we adhere to the requirements of the Bluetooth specification [1]. This new schedule is illustrated in Fig. 4(d).

3.5. Comments and observations

In this section we elaborate on some very important issues regarding DSA. We begin with the multicast address allocation, followed by multicast and scatternet support in DSA, overhead analysis, impact on existing Bluetooth architecture and, finally, the relation of DSA with the IEEE 802.15.3 standard.

A very important issue to be discussed regards the multicast addressing in DSA given that every Bluetooth packet contains a single three bit active member address (AM_ADDR), where seven of these addresses are used by the slaves and the remaining one is employed for broadcast. In the design and implementation of DSA, we considered two options to tackle this problem: we could increase the AM_ADDR field in the Bluetooth packet from three to four bits and use one out of these four bit addresses for multicasting, or we could simply temporarily allocate one of the three bit addresses for this purpose. The first option would make DSA incompatible with current Bluetooth specifications. Therefore, we have selected the second option, where we temporarily allocate the broadcast address for the purpose of multicasting. However, note that by allocating the broadcast address for multicasting does not imply that all devices will receive such a packet since the packet is discarded once the device realizes it is not supposed to listen at that particular slot as specified in the slot scheduling message currently in effect. Hence, only the actual multicast members (specified in the slot scheduling message) will accept the corresponding packet. On the other hand if the packet is actually to be broadcast throughout the piconet, then the given slot carrying the broadcast packet will not be present in the slot scheduling message, resulting in all piconet devices accepting such packet.

As explained in Section 3.1, connection request messages always arrive at the master and are not propagated to the destination. Implicitly, we assume that the destination device will always accept new connection requests. This can be extended to include the destination in the connection request exchange. However since the master has information about all connections in the piconet, we opted to omit this additional transmission at the expense of the destination's willingness to accept the connection.

Also, note that when a device requests to employ DH1 or DH3 packets for a connection and no SCO traffic is present in the piconet, they will always be granted an entire slot cycle even if they do not make full use of it. In these cases, the master uses the free slots for polling, hence preventing wastage (see Fig. 4(c)).

In the previous section, we have only provided a scenario where unicast communication takes place. However, note that multicast-like transmission does not require any more changes than to specify the entire list of destinations in the second part of the scheduling message format. In this case, all devices belonging to the list of destinations would record the *slot cycle* and *TxSlot*, resulting in all of them tuning to the same slots and hence receiving the same message. It is important to note that we have chosen a sender-initiated approach to support multicasting over Bluetooth, as it is optimal for small groups [18] where the sender has information about all the destinations. It contrasts with receiver-oriented approaches [17,27] where receivers initiate group membership, and are best applicable for medium-to-large groups. Also, while we have restricted our discussion to piconet multicasting, note that DSA could be used to support scatternet multicasting with some extensions. For that, we basically need to include the bridge node in the piconet multicast and have it convey the packets to its other associated piconets. In addition, we could synchronize slot assignments of neighboring piconets for enhanced performance.

It is worth commenting on the overhead of our DSA scheme. In most cases, the scheduling message of Fig. 3 can be transmitted using a DH1 packet, that is, by using only one slot we can implement DSA over current Bluetooth and virtually eliminate its major shortcomings. We discuss this again in the performance evaluation section.

We would also like to analyze the impact of the implementation of DSA within the current Bluetooth architecture. Basically, the changes would be confined to the Link Manager Protocol (LMP) and Baseband layers. At the LMP level,

messages such as connection request, release, the slot scheduling, and timers would be incorporated. However, the functionally of DSA as to when to transmit/listen would be better placed at the Baseband layer.

Finally, the IEEE 802.15.3 standard for WPANs also supports direct communication between devices. However, the design of 802.15.3 is considerably different from DSA. First, 802.15.3 devices possess clear channel assessment capability which enables a contention-based access period. Also, 802.15.3 is based in a superframe structure with the addition of a beacon period. In addition, contentionless access in 802.15.3 is provided by guaranteed time slots which can be of variable size. Clearly, all these and other features found in 802.15.3 are not present in Bluetooth. Therefore, we have developed DSA specifically to the existing Bluetooth technology, taking its design and slot structure into consideration.

3.6. Analysis of DSA and new piconet per connection

An alternative scheme to DSA is to create an individual piconet for every new connection [14,28], and hence seek to optimize aggregate throughput at the expense of increased interference. Here, we refer to this technique as New Piconet Per Connection (NPPC). To better understand the trade-offs of NPPC and DSA, it becomes inevitable to study the nature of inter-piconet interference [21,28], also referred to as intermittent interference [23], and compare the schemes with respect to this aspect. Numerical results, proof of correctness, and details of the analytical model presented here briefly can be found in [34].

In Bluetooth, packet transmissions in different piconets are asynchronous and are transmitted with period T_p, which depends upon the Bluetooth packet type p. For instance, if p is equal to DH1 or DM1 we have that $T_p = 2* \ slotsize$, where $slotsize$ is the size of a Bluetooth slot, and is equal to 625 μs. The probability of packet collision $p_c(i,j)$ between any two piconets i and j is the probability of packet overlap both in time and frequency. Therefore, if we assume that any packet collision incurs packet loss (strong interference) [21], the packet collision probability between packets p and z of piconets i and j with sizes $S_{p,i}$ and $S_{z,j}$, respectively, is given by

$$p_c(i,j) = \frac{S_{p,i} + S_{z,j}}{(\max(slotsperpacket(p), slotsperpacket(z)) + 1) \times slotsize} \cdot \frac{1}{C}, \quad (1)$$

where C is the number of available frequency channels and is equal to 79 in most countries [1]. Moreover, the function $slotsperpacket(X)$ gives the number of slots occupied by a Bluetooth packet X, and $\max(p, q)$ returns the largest value of two numbers p and q, or any of them if $p = q$. We can extend this analysis by deriving the packet collision probability in a network comprised of N piconets. The packet collision probability with a packet originated at the ith piconet is given by

$$\bar{p}_c(i) = 1 - (1 - p_c(i,j))^{N-1}. \quad (2)$$

Aggregate throughput is another important performance measure when you consider a collection of piconets operating in proximity. In Bluetooth, data packets are followed by an acknowledgement in the opposite direction from the recipient. Thus, we can obtain the aggregate throughput of N piconets relative to a Bluetooth packet X as given by

$$S(X) = \mu(X)N\left[1 - \frac{2(ACKinbits(X) + sizeinbits(X))}{(slotsperpacket(X) + 1) \times slotsize} \cdot \frac{1}{C}\right]^{N-1}, \quad (3)$$

where $ACKinbits(X)$ and $sizeinbits(X)$ are the size in bits of the Baseband acknowledgement and data packet type X, with $\mu(X)$ being the nominal throughput of X [1]. The acknowledgement information is included in the header of the return packet, and hence called piggy-backing. In our analysis, we consider the common case where Baseband acknowledgements do not carry payload information, thereby making $ACKinbits(X)$ in Eq. (3) always equal to 126 bits [1,34].

Fig. 5(a) depicts the packet collision probabilities for all the three slot sizes of Bluetooth packets as a function of the number of piconets. As we can see, the packet collision probability always

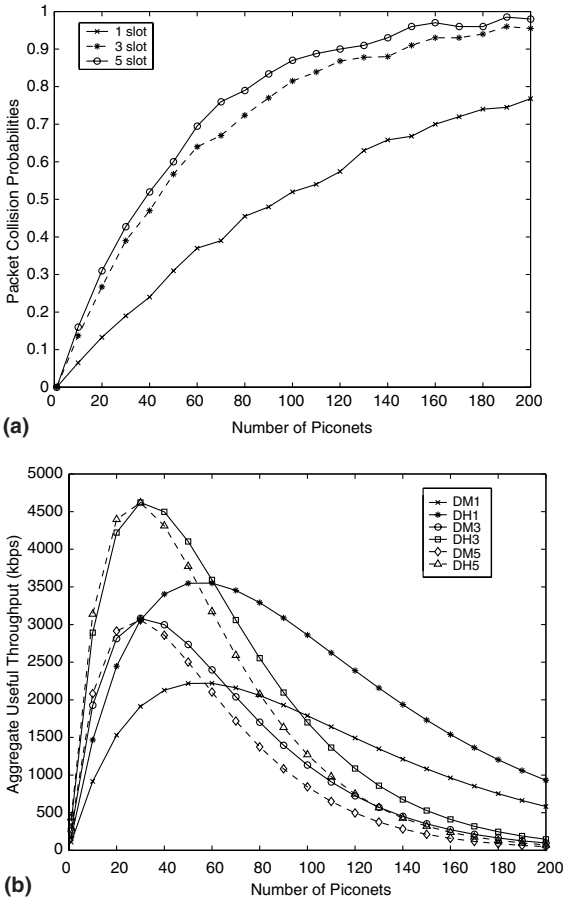

Fig. 5. Packet collision probability and aggregate througput in Bluetooth: (a) packet collision probability; (b) aggregate throughput.

increases with the number of piconets, while the aggregate throughput increases only up to a certain number of piconets as shown in Fig. 5(b). Thus, an obvious enhancement to the basic NPPC scheme could be to keep on partitioning the piconet for every new connection until the point where an additional piconet will deteriorate the aggregate throughput. However, there are some problems with this solution:

- As the number of piconets are increased so as to improve the aggregate throughput (Fig. 5(b)), the packet collision probability rises rapidly as depicted in Fig. 5(a). Thus a larger number of packets will have to be re-transmitted, incurring a higher delay and increased energy consumption due to frequent transmissions/receptions.
  - This is not true for audio connections, where lost packets are not re-transmitted. Hence, the increase in the interference level will drastically affect the performance and quality of the connection.
  - This can also impact TCP performance by continuously adjusting the congestion window due to packet drops for FTP-type of applications. Moreover, TCP may interpret packet loss due to interference as congestion and inadverdently apply its congestion control algorithms, hence negatively affecting throughput.
- Usually, it is really difficult to know how many piconets exist in a scatternet, unless a rough estimate is extracted by measuring the interference. We plan to investigate this as a future research topic.
- Creating a new piconet for every connection arrival might result in a condition where inter-piconet communication has to be addressed and could possibly lead to dynamic and continuous joining/leaving of the nodes in the piconets.
- Previous results [23] show that a considerable amount of time is wasted in switching from one piconet to another if a single node is involved in more than one traffic connection either as a source, a destination or both. That is, NPPC will eventually suffer from inter-piconet communication issues.

Therefore, in the design of DSA, we have decided not to create a new piconet for each connection, even when the number of piconets are within acceptable interference level. Our goal is not to always achieve the maximum possible aggregate throughput, but to keep interference at acceptable levels, effectively and efficiently support the three waves of Bluetooth-based applications (see Section 1) with their increased scalability requirements, and re-organize piconet scheduling so that application needs can be satisfied. Moreover, as we show in Section 5, the greedy throughput approach taken by NPPC is observed to

drastically increase interference levels in medium to large network sizes, hence not being scalable. On the other hand, DSA is shown to effectively handle networks of varying sizes.

4. Simulation environment and methodology

To evaluate our proposed DSA scheme, we have implemented it in the Network Simulator (ns-2) [19] and BlueHoc [20], an open-source Bluetooth simulator provided by IBM. Since BlueHoc only provides the basic functionality of Bluetooth, we have made considerable extensions to this simulator in order to conduct our experiments including the support for direct slave-to-slave and multicast-like communication.

The simulation experiments share the same network topology. It uses a single piconet comprised of eight devices placed randomly within a $17\,m \times 17\,m$ region. Radio propagation range is taken as 10 m, and nominal channel capacity is set to 1 Mbps.

We have thoroughly compared our DSA scheme with the existing Bluetooth under the following metrics:

- Aggregate throughput—This is the average number of data bytes correctly received by all piconet devices per unit time.
- Delay—This measures the delay per bit in transferring data from a given source to a given destination.
- Overhead—This measures the efficiency of the scheme. It relates the total number of data and control bytes sent to the total number of data bytes received by all devices.
- Power consumption—With this metric, expressed in Megabytes/Joule (Mbytes/J), we evaluate the power efficiency of DSA as compared to Bluetooth. Bluetooth radios operate at 2.7 V and 30 mA, resulting in a 115 nJ/bit for transmission [1,25]. Moreover, measurements have shown [26] that receive:send ratio is usually 1:1.4, while power consumption in idle times is negligible. In our simulations, we have used these relations to compute power consumption for the Bluetooth devices.

To evaluate the flexibility of DSA, both slave-to-slave and multi-slave communications have been studied where we vary the number, packet type, and bandwidth requirement of each connection. In order to cover a broad range of scenarios, we have run our simulations with FTP, Telnet, and SCO (voice) traffic, where FTP and Telnet connections use DHx and DMx packets, respectively, while SCO traffic employs HV2 and HV3 packets. Each traffic connection is pair-wise distinct nodes, and we consider up to three connections of each type. To achieve reliability in the broadcast of the DSA scheduling message (Section 3.2) and at the same time determine an upper bound in the overhead of the DSA scheme, we perform the broadcast of a slot-scheduling message three times in all simulation runs. This number has been reached after extensive simulations under various traffic scenarios, where we have noted that three broadcasts suffice to cover up all piconet devices. Thus, we believe it to be a reasonable number of broadcasts to compare our DSA with existing Bluetooth, and to serve as a worst-case analysis.

5. Simulation results

Results for unicast communication and multicast-like communications are given separately.

5.1. Unicast communication

In what follows, we study the behavior of our four selected metrics under three different traffic scenarios which could reflect various applications such as multi-party gaming where connections are not biased towards particular devices. We note, however, that even in scenarios where certain connections are more likely than others, we expect to see a considerable improvement with the use of DSA [35]. These scenarios considered here are

- Scenario A—In this scenario, a total of three FTP connections are considered where each connection is initiated consecutively in different points in time. Table 1 depicts the connection endpoints employed in this scenario, as well as shows the various connection requests issues

Table 1
Scenario A connection pattern

Connection	CONNECTION_REQ
$S_1 \leftrightarrow S_3$	S_1: (S_3, DH5, 12, 18)
$S_0 \leftrightarrow S_4$	S_0: (S_4, DH3, 6, 18)
$S_5 \leftrightarrow S_6$	S_5: (S_6, DM1, 12, 18)

by the slave devices. As we can see, each FTP connection has a different packet type and demand for bandwidth allocation, thus exercising the flexibility of DSA with respect to ACL traffic;

- Scenario B—This scenario is devoted to analyze voice traffic performance. It employs a total of two SCO connections also initiating consecutively at different points in time. Various connection requests and their demands for this scenario are illustrated in Table 2. With this scenario we evaluate the flexibility of DSA with respect to SCO traffic;
- Scenario C—This scenario considers a mixed case of connections. Total of three connections are initiated consecutively where the first one is a SCO connection, the second one a Telnet, and the third one an FTP. Table 3 describes the characteristics of the various connections employed in this scenario. Most importantly, we note that this scenario is of major importance as it contains a mix of SCO and ACL traffic, hence requiring DSA to handle heterogeneous traffic connections.

Table 2
Scenario B connection pattern

Connection	CONNECTION_REQ
$S_1 \leftrightarrow S_3$	S_1: (S_3, HV3, 4, 6)
$S_5 \leftrightarrow S_6$	S_5: (S_6, HV2, 4, 6)

Table 3
Scenario C connection pattern

Connection	CONNECTION_REQ
$S_1 \leftrightarrow S_4$	S_1: (S_4, DH3, 18, 24)
$S_3 \leftrightarrow S_2$	S_3: (S_2, DM3, 12, 18)
$S_5 \leftrightarrow S_6$	S_5: (S_6, HV3, 4, 6)

5.1.1. Aggregate throughput

Fig. 6(a)–(c) depict the aggregate throughput for DSA and existing Bluetooth for the scenarios under consideration. In general, we can observe that DSA always achieves higher throughput than existing Bluetooth. On an average, DSA experiences up to 300% improvement. The reason is simple: while in Bluetooth all traffic has to go through the master, path becomes non-optimal and bandwidth is wasted, hence compromising the number of data bytes received per unit time. On the other hand, DSA employs direct slave-to-slave communication, thus avoiding additional delay due to packet transmission, propagation time, and queuing, and hence boosting the number of data packets received per unit time.

In Fig. 6(a), we see that with one FTP connection Bluetooth can still provide a performance comparable with that of DSA as the delay effects are minimized. However, when the number of connections increases to two and three, Bluetooth can no longer sustain the same performance whereas DSA scales approximately linearly.

5.1.2. Delay

The delay experienced in each of the three scenarios are illustrated in Fig. 7(a)–(c) respectively. Due to the direct communication between slaves, and hence optimal communication path, DSA is shown to provide reduced delay in all scenarios, with delays being as low as one-third of the current delay in Bluetooth. Note that in scenarios A and C where we have traffic connections involving TCP, the delay increases as we increase the number of connections. This is due to the burstiness of TCP traffic which increases the average queue length in every burst.

On the other hand, this is not the case in scenario B where only SCO connections are considered and the traffic pattern follows a constant bit rate (CBR) stream. Here, the delay for DSA is constant due to our direct slave-to-slave communication, while in Bluetooth the delay initially increases rapidly and then tends to stabilize. Note that in all scenarios, the delay experienced by the DSA scheme is dramatically lower than that of Bluetooth.

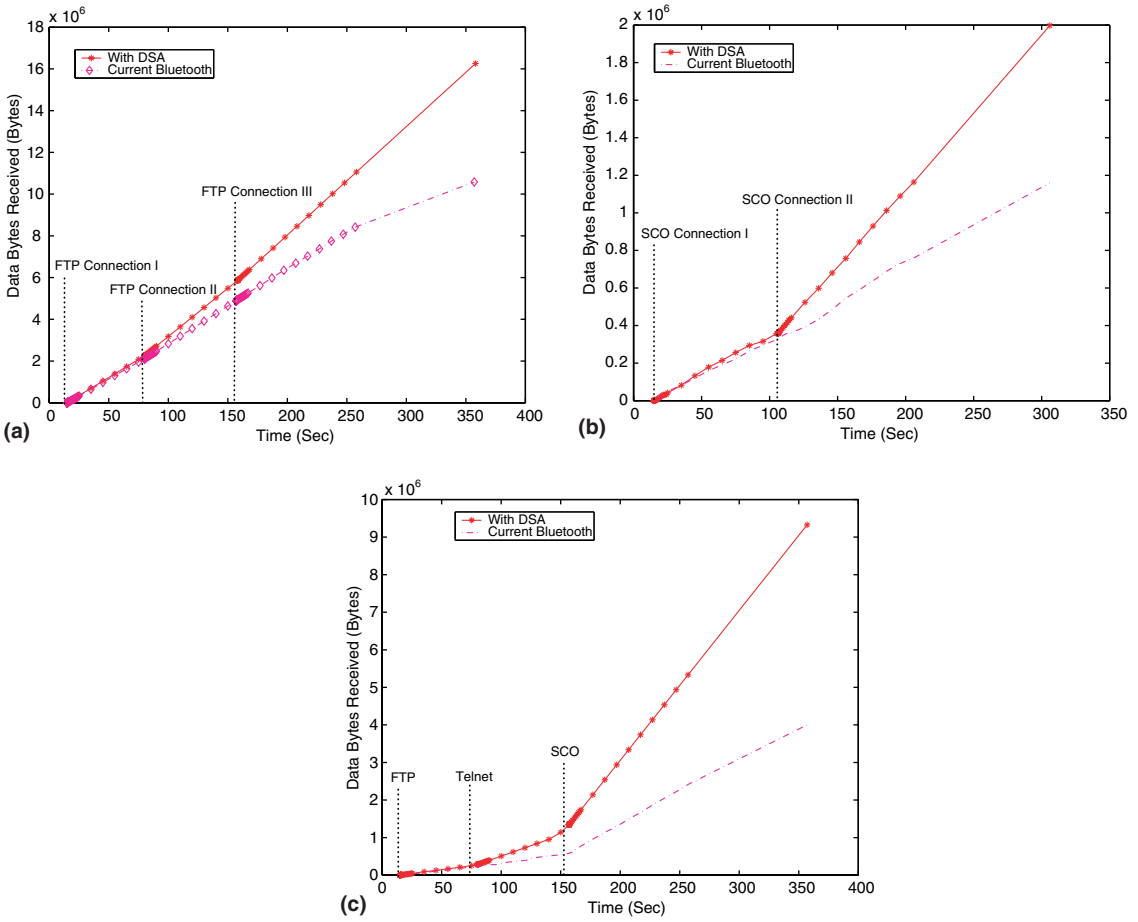

Fig. 6. Aggregate throughput for the three scenarios considered: (a) scenario A; (b) scenario B; (c) scenario C.

5.1.3. Overhead

Fig. 8(a)–(c) compare the overhead of Bluetooth and DSA for scenarios A, B, and C respectively. Here, we see that although DSA transmits three broadcasts for each scheduling message, the associated overhead is always approximately half that of Bluetooth for the three scenarios considered. The reason for this is that Bluetooth's overhead is present for every packet since all of them have to be forwarded by and routed through the master hence incurring an additional transmission. In DSA, the overhead is only tied to the broadcast message while all data packets are directly sent to the intended destination.

In Fig. 8(b), we can see that the overhead of Bluetooth for scenario B remains nearly the same as in DSA while only one SCO connection is present in the piconet, whereas it increases rapidly when the second SCO connection arrives while DSA maintains a proportional grade of service. The reason being that the current Bluetooth can support at most three simplex SCO connections, that is, having the master as either a source or a destination of SCO traffic. Therefore, when it comes to duplex connections, that is, from slave-to-slave, Bluetooth can support only one SCO connection. With our DSA scheme employing direct slave-to-slave communication, we overcome

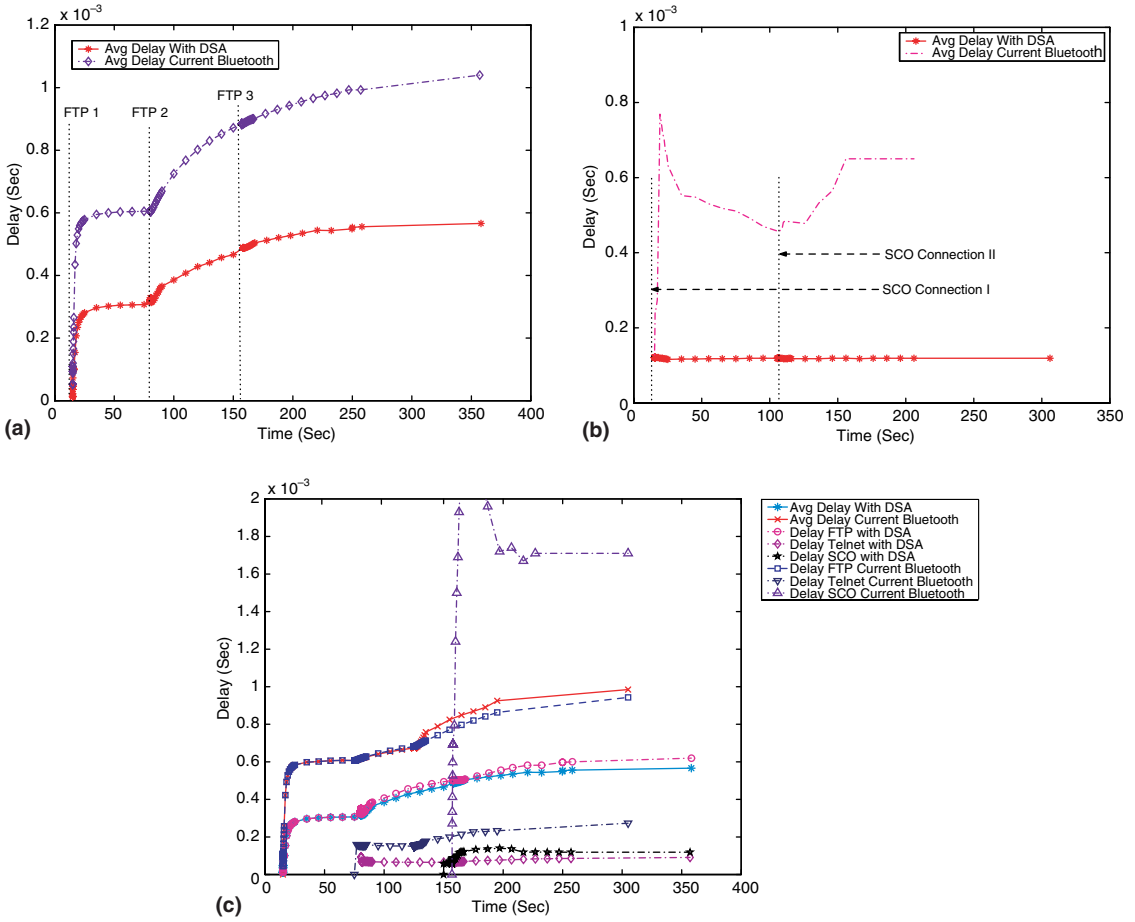

Fig. 7. Delay for the three scenarios considered: (a) scenario A; (b) scenario B; (c) scenario C.

this restriction by supporting up to three duplex SCO connections within a piconet. This is one major improvement due to our proposed DSA scheme. As a result, an additional SCO connection initiated in the piconet disturbs current traffic which incurs increased overhead in existing Bluetooth, while it achieves a proportional overhead in DSA.

5.1.4. Power consumption

Fig. 9 shows the power consumption comparison of DSA and Bluetooth for the three scenarios under evaluation. We can clearly notice a drastic reduction in power consumption at the master device as a result of the proposed DSA scheme. As we can see from Fig. 9, the master does not get involved in the communication between slaves in DSA, resulting in a significant increase in the energy efficiency (Mbytes/J). Table 4 illustrates the average power consumption improvement of DSA both at the master as well as in the slaves for the scenarios analyzed.

As can be seen from Table 4, the average energy saved at the slaves is mainly due to reduction in the number of polls by the master device, as these slaves are involved in data communication. Hence, power consumption at the slaves is much less as compared to the master device where energy conservation is substantial as a result of DSA (see Table 4). Finally, it is important to note the energy-per-byte improvement ratio of DSA over Bluetooth in Fig. 9. In other words, with the

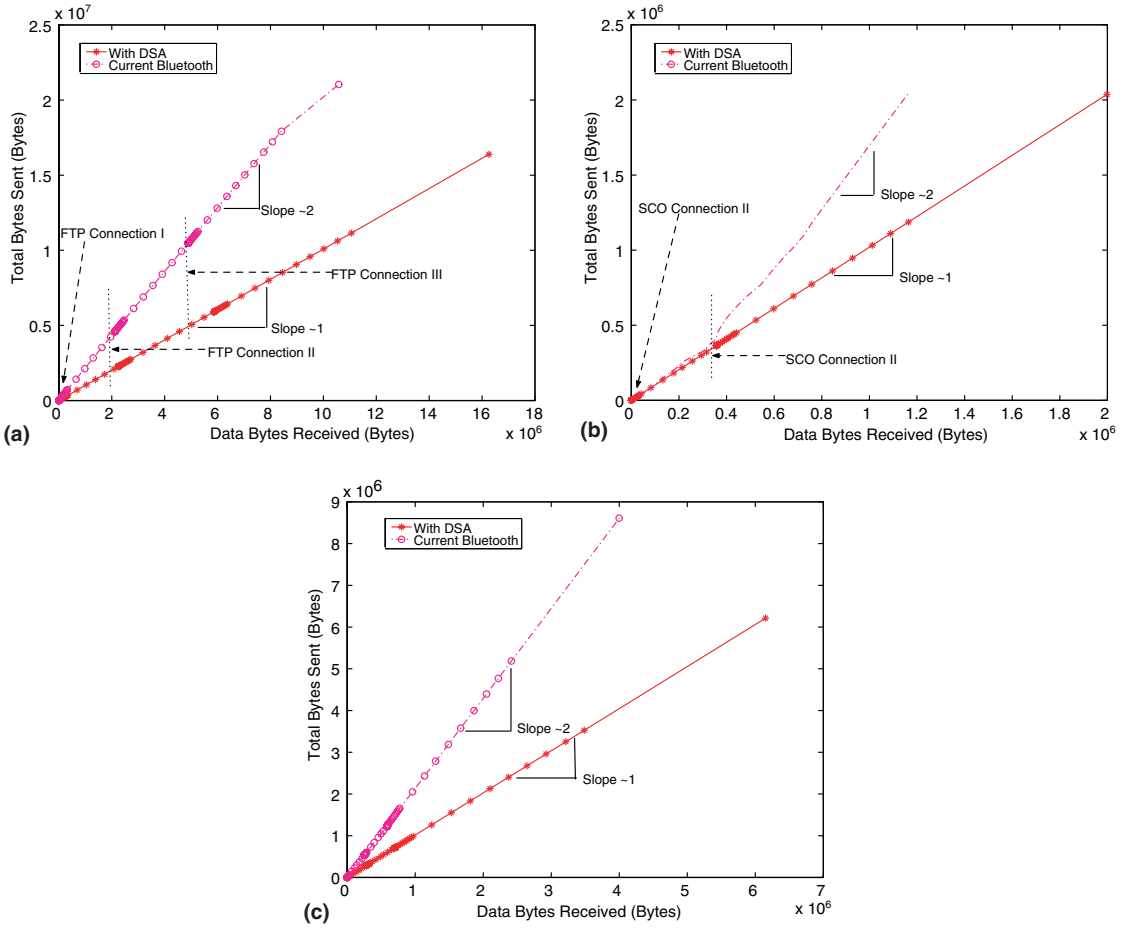

Fig. 8. Overhead for the three scenarios considered: (a) scenario A; (b) scenario B; (c) scenario C.

adoption of DSA, a significantly larger amount of data can be transmitted with the same energy. A direct consequence of this fact is that the piconet has its lifetime drastically increased.

5.2. Multicast communication

In this section, we study the multicast support in DSA and compare it with the native way of implementing group communication in Bluetooth by multiple unicasts. For this experiment, we consider two multicast sessions wherein each of the two multicast sources has exactly three other devices as group members. Here, the first multicast session is initiated at simulation startup, while the second is initiated 100 seconds later. Exponential traffic sources are considered.

5.2.1. Aggregate throughput

Fig. 10 depicts the aggregate throughput of both DSA and Bluetooth. As expected, DSA greatly outperforms Bluetooth by employing a single transmission for all group members, while Bluetooth transmits a packet per destination. As a matter of fact, most of the well-known benefits of multicasting [17,27] over multiple unicasts are embodied in the DSA, making it more efficient than Bluetooth. DSA improves the average throughput by up to 500% as shown in Fig. 10.

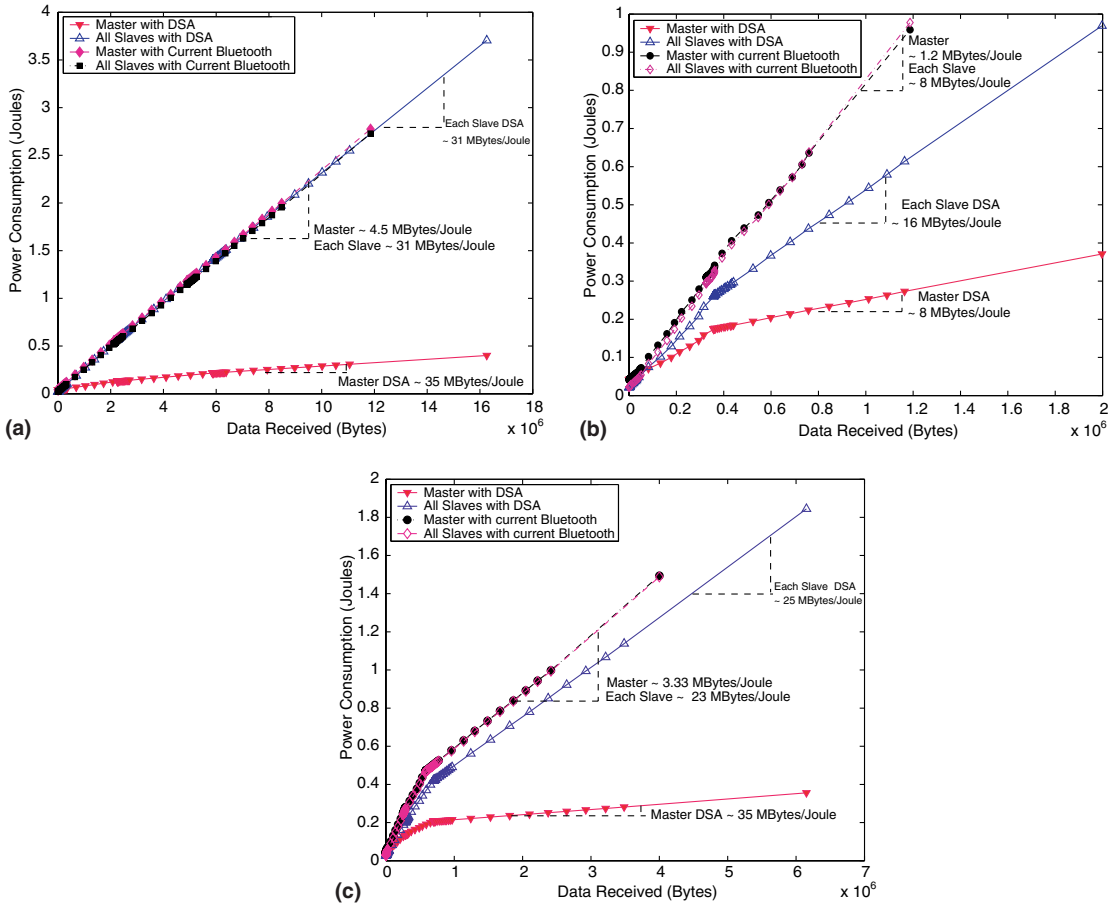

Fig. 9. Power consumption comparison of DSA and Bluetooth: (a) scenario A; (b) scenario B; (c) scenario C.

Table 4
Energy improvement of DSA over Bluetooth

	Master	Slaves
Scenario A	83.06%	2.12%
Scenario B	43.62%	8.95%
Scenario C	54.36%	6.60%

5.2.2. Delay

Delay is another aspect in which the multicasting mechanism built in DSA, brings it to a minimum level. As we can see from Fig. 11, the delay in DSA is practically constant due to its direct one-to-many communication, whereas it is approximately 30 times higher in Bluetooth.

5.2.3. Overhead

As Bluetooth employs multiple unicasts in order to provide a group-like communication mechanism, it is expected to have a much higher overhead than DSA. Fig. 12 confirms this assertion by revealing that DSA experiences around one-seventh the overhead of Bluetooth, despite the control messages broadcasted in DSA.

5.2.4. Power consumption

Finally, Fig. 13 presents the power consumption comparison of Bluetooth and DSA. Similar to the unicast case, multicasting with DSA also significantly reduces the power consumed at the master device as well as at all slaves. On average, the master experiences a 54.17% reduction in

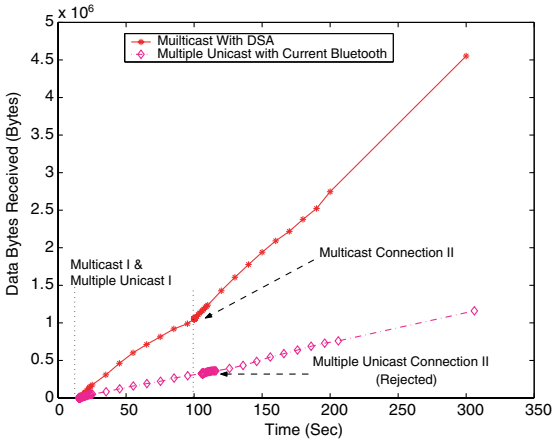

Fig. 10. Aggregate throughput comparison of DSA and Bluetooth.

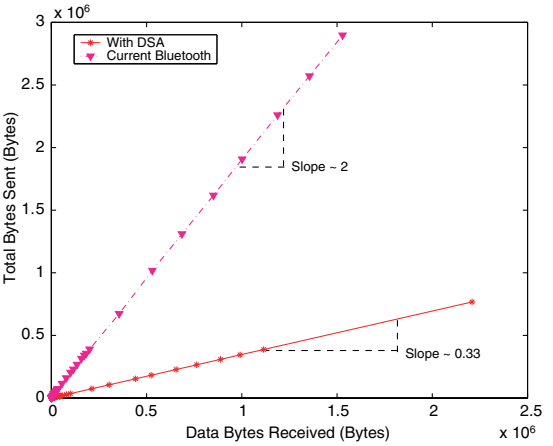

Fig. 12. Overhead comparison of DSA and Bluetooth.

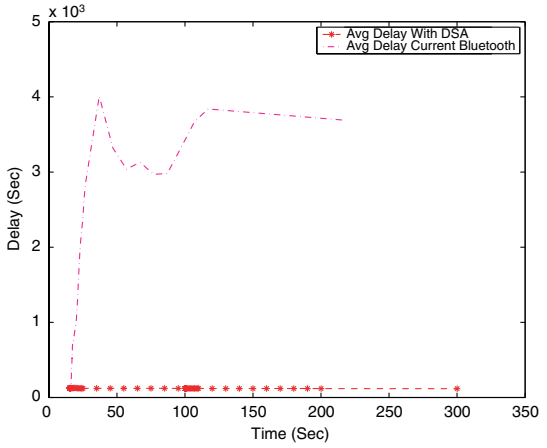

Fig. 11. Delay comparison of DSA and Bluetooth.

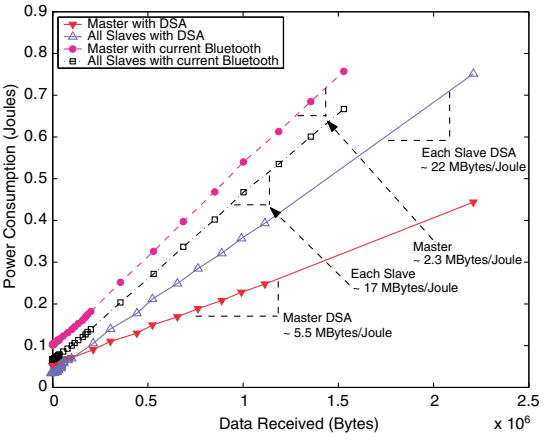

Fig. 13. Power consumption of DSA and Bluetooth.

energy consumption, while the slaves have a 33.05% reduction. As a consequence of this fact, we can see from Fig. 13 that the energy efficiency (Mbyte/J) ratio of DSA overwhelms that of current Bluetooth, hence prolonging the lifetime of the overall network.

5.3. Performance comparison of DSA and NPPC

In this section, we return to the comparison of DSA and NPPC. In order to obtain a deeper insight between their pros and cons, Fig. 14(a) and (b) compare the performance of the two schemes for different network sizes (represented by the different number of nodes) and increasing number of connections. For this performance evaluation, we have considered a uniform node distribution of 6 per piconet and Bluetooth DH5 packets only. Also, given the fact that the interfering range of Bluetooth devices can go up to 24 m and the density of nodes can be as high as 20 nodes/m^3 [38], we have considered that any

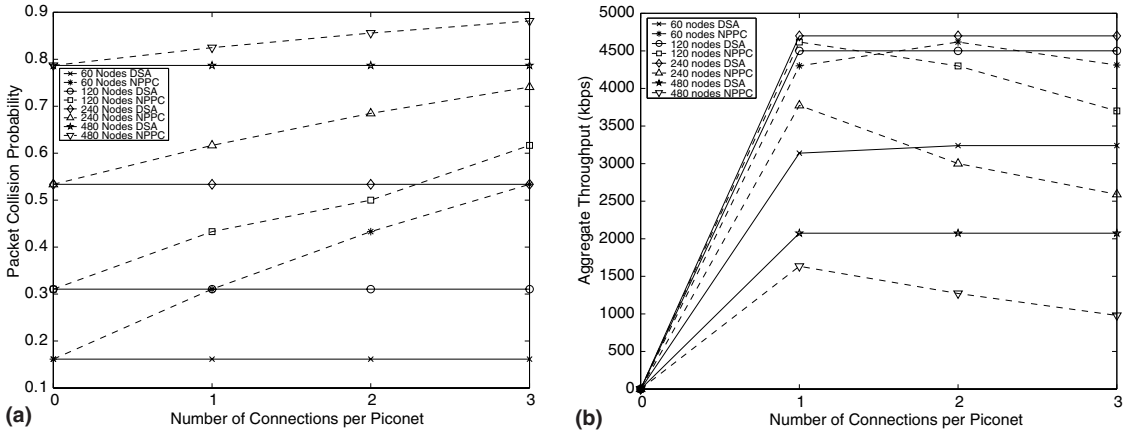

Fig. 14. Performance comparison of DSA and NPPC: (a) packet collision probability; (b) throughput.

transmission can interfere with any other hence reflecting some high-density application scenarios.

As we can see from Fig. 14(a), DSA is much more resilient to interference than NPPC given its more conservative approach of rearranging piconet schedules instead of partitioning them. On the other hand, NPPC is noted to increase network interference levels for every new connection, and this behavior becomes increasingly unacceptable as network size grows larger. Analyzing the throughput comparison of both DSA and NPPC in Fig. 14(b) reinforces this fact. Here, NPPC is only observed to surpass DSA on network sizes of 60 nodes or less, while DSA outperforms NPPC in all other scenarios as NPPC starts experiencing high levels of interference. In summary, this is to say that the approach taken by NPPC does not scale to medium (meaning 60 nodes or more) or large network sizes, whereas DSA is observed to give overall superior performance by keeping interference at minimum possible levels and making better use of piconets. We also note that the greedy throughput approach taken by NPPC does not reflect practical applications where traffic is usually bursty. Therefore, continued creation of piconets may render the piconets being idle most of the time, whereas DSA seeks to better utilize piconet resources by supporting a larger number of connections through appropriate scheduling of channels.

However, this is not to say that DSA cannot be employed in conjunction with some modified version of NPPC where partitions would occur not for every new connection, but according to aspects such as QoS parameters, number and type of traffic connections, and so on. We have investigated this issue and the results can be found in [35].

6. Related work

Recent research studies have tackled the issue of Bluetooth piconet performance improvement from different angles. From the scheduling perspective, [3–6] pointed out the drawbacks of existing scheduling techniques and several modifications are suggested to enhance the polling mechanism from the master device. However, the issue of packet forwarding through the master device is not addressed, leading to the problems already mentioned in this paper.

[7–13] proposes algorithms to generate the scatternet topology with properties such as limiting the maximum roles and degrees of any node, evenly distributing topology generation, limiting the number of piconets and the maximum number of hops between any pair of devices, and so on. However, the major shortcoming of these approaches is the disregard of traffic characteristics when building the topology. That is, even if any two slaves

need to communicate frequently, there is no guarantee that they will be within the same piconet in the resulting scatternet.

As we have discussed, another solution would be piconet partitioning (e.g., NPPC), wherein a pair of slaves form a piconet by themselves if frequent communication is taking place between them [14]. However, as we have seen, this approach incurs additional problems such as scatternet scheduling, and, more importantly, this cannot be done indefinitely as interference levels may become unacceptable [16,21,23]. Additionally, this solution makes a very unlikely built-in assumption that all connections are pair-wise distinct nodes in order to piconet partitioning to be successful, and also it cannot support multicast communication.

An enhanced pseudo master–slave switch and pseudo piconet partitioning algorithms are presented in [37]. We note, however, that the algorithms in [37] complement the ones we present here and could, therefore, operate in conjunction.

Finally, [15] proposes a time-slot leasing approach where the master device allocates slots to slaves for direct communication. While this scheme has a few points in common with our proposal, there are fundamental differences. Firstly, SCO connections are not evaluated. While this greatly simplifies design, as periodic scheduling of SCO packets has not to be taken into account, it is unrealistic to assume that SCO traffic will not be present [2]. Second, the mechanism in [15] does not provide for multicast-like communication, while this has been a major concern of our DSA scheme. Third, this scheme does not provide any mechanism to adjust bandwidth allocation as traffic demand increases or decreases, while our proposed DSA mechanism dynamically adjusts the slot cycle so as to meet traffic needs.

7. Conclusions

By virtue of the Master/Slave communication model, Bluetooth medium access provides for simplicity, low-power, and low-cost, these being the major forces driving the technology. However, this medium access scheme brings in several problems such as inefficient packet forwarding and wastage of bandwidth by forwarding through the master, increased end-to-end packet delay, and additional power consumption at the master unit due to frequent medium access. Moreover, multicast-like communication is not supported which could be employed by many applications envisioned for WPAN environments.

With that in mind, we proposed a dynamic scheduling scheme in this paper with two novel approaches, namely, the implementation of direct communication between any two slaves within a piconet, and support for multicast-like transmissions for group communication. Through extensive simulation, we have shown DSA to greatly outperform existing Bluetooth implementation by providing optimal bandwidth usage, lower delay, and low overhead. More importantly, we have shown DSA to be an extremely energy efficient scheme for Bluetooth as this is decisive for the success of envisioned WPAN applications. We believe this work will lead to new research directions. It is worth mentioning that the DSA scheme is simple and can be easily incorporated into current Bluetooth architecture.

Acknowledgment

This work has been supported by the Ohio Board of Regents Doctoral Enhancement Funds and the National Science Foundation under grant CCR-113361.

References

[1] Bluetooth SIG, Bluetooth Specification, <http://www.bluetooth.com>.
[2] C. Bisdikian, An overview of the Bluetooth wireless technology, IEEE Communications Magazine 39 (12) (2001) 86–94.

[3] M. Kalia, S. Garg, R. Shorey, Efficient policies for increasing capacity in Bluetooth: an indoor pico-cellular wireless system, in: Proceedings of IEEE VTC, May 2000.
[4] A. Capone, M. Gerla, R. Kapoor, Efficient polling schemes for Bluetooth piconets, in: Proceedings of IEEE ICC, June 2001.
[5] M. Kalia, D. Bansal, R. Shorey, Data scheduling and SAR for Bluetooth MAC, in: Proceedings of IEEE VTC, May 2000.
[6] M. Kalia, D. Bansal, R. Shorey, MAC scheduling and SAR policies for Bluetooth: a master driven TDD Pico-cellular wireless system, in: MoMuC, May 1999, pp. 384–388.
[7] L. Ramachandran, M. Kapoor, A. Sarkar, A. Aggarwal, Clustering algorithms for wireless ad hoc networks, in: Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 2000.
[8] T. Salonidis, P. Bhagwat, L. Tassiulas, Proximity awareness and fast connection establishment in Bluetooth, in: Proceedings of MobiHoc, 2000.
[9] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, Distributed topology construction of Bluetooth personal area networks, in: Proceedings of IEEE Infocom, 2001.
[10] C. Law, A. Mehta, K. Siu, Performance of a new Bluetooth scatternet formation protocol, in: Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing, October 2001.
[11] C. Law, K. Siu, A Bluetooth scatternet formation algorithm, in: Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks, November 2001.
[12] G. Zaruba, I. Chlamtac, S. Basagni, Bluetrees—scatternet formation to enable Bluetooth-based ad hoc networks, in: Proceedings of IEEE ICC, June 2001.
[13] Z. Wang, R. Thomas, Z. Haas, Bluenet—a New Scatternet Formation Scheme, in: Proceedings of the 35th HICSS, Big Island, Hawaii, January 2002.
[14] W. Zhang, H. Zhu, G. Cao, On improving the performance of Bluetooth networks through dynamic role management, Technical Report, CSE-01-018. <http://www.cse.psu.edu/~gcao/paper/bluetooth.ps>, May 2001.
[15] W. Zhang, H. Zhu, G. Cao, Improving Bluetooth network performance through a time-slot leasing approach, in: Proceedings of WCNC, 2002.
[16] C. Cordeiro, D. Agrawal, Mitigating the effects of intermittent interference on Bluetooth ad hoc networks, in: 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Lisbon, Portugal, September 2002.
[17] H. Gossain, C. Cordeiro, D. Agrawal, Multicast: wired to wireless, IEEE Communications Magazine 40 (6) (2002) 116–123.
[18] L. Ji, M.S. Corson, Differential destination multicast—a MANET multicast routing protocol for small groups, in: Infocom, 2001, pp. 1192–1202.
[19] The Network Sumulator (ns-2). <http://www.isi.edu/nsnam/ns/>.
[20] BlueHoc, IBM Bluetooth Simulator. <http://oss.software.ibm.com/developerworks/opensource/bluehoc/>.
[21] C. Cordeiro, D. Agrawal, D. Sadok, Piconet interference modeling and performance evaluation of Bluetooth MAC protocol, IEEE Transactions on Wireless Communications 2 (6) (2003) 1240–1246.
[22] D. Agrawal, Q.-A. Zeng, Introduction to wireless and mobile systems, Brooks/Cole, Florence, KY, 2003.
[23] C. Cordeiro, D. Agrawal, Employing dynamic segmentation for effective co-located coexistence between Bluetooth and IEEE 802.11 WLANs, in: IEEE GLOBECOM, Taiwan, November 2002.
[24] Y. Bin Lin, I. Chlamtac, Wireless and Mobile Network Architectures, Wiley, New York, 2000.
[25] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of HICSS, January 2000.
[26] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: Proceedings of IEEE Infocom, June 2002.
[27] C. Cordeiro, H. Gossain, D. Agrawal, Multicast over wireless mobile ad hoc networks: present and future directions, in: IEEE Network, Special Issue on Multicasting: An Enabling Technology 17 (1) (2003) 52–59.
[28] Y. Lim, S. Min, J. Ma, Performance evaluation of the Bluetooth-based public internet access point, in: Proceedings of the 15th ICOIN, 2001, pp. 643–648.
[29] N. Rouhana, E. Horlait, BWIG: Bluetooth web internet gateway, in: Proceedings of IEEE Symposium on Computer and Communications, July 2002.
[30] O. Kasten, M. Langheinrich, First experience with Bluetooth in the smart-its distributed sensor network, in: Proceedings of the Workshop in Ubiquitous Computing and Communications (PACT), October 2001.
[31] F. Siegemund, M. Rohs, Rendezvous layer protocols for Bluetooth-enabled smart devices, in: Proceedings of International Conference on Architecture of Computing Systems, April 2002.
[32] D. Estrin, R. Govindan, J. Heidmanm, New century challenges: scalable ordination in sensor networks, in: ACM Mobicom, 1999, pp. 263–270.
[33] J. Kahn, R. Katz, K. Pister, New century challenges: mobile networking for smart dust, in: ACM Mobicom, 1999, pp. 271–278.
[34] C. Cordeiro, S. Abhyankar, R. Toshiwal, D. Agrawal, BlueStar: enabling efficient integration between Bluetooth WPANs and IEEE 802.11 WLANs, Mobile

Networks and Applications, Special Issue on Integration of Heterogeneous Wireless Technologies 9 (4) (2004) 409–422.
[35] C. Cordeiro, S. Abhyankar, D.P. Agrawal, Design and Implementation of QoS-driven dynamic slot assignment and piconet partitioning algorithms over Bluetooth WPANs, in: IEEE INFOCOM, March 2004.
[36] C. Cordeiro, S. Abhyankar, D.P. Agrawal, A novel energy efficient communication architecture for Bluetooth ad hoc networks, in: Proceedings of Personal Wireless Communication (PWC), Venice, Italy, September 2003.
[37] S. Abhyankar, R. Toshiwal, C. Cordeiro, D. Agrawal, On the application of traffic engineering over Bluetooth ad hoc networks, in: ACM MSWiM, in conjunction with ACM Mobicom, September 2003.
[38] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102–114.

Carlos de M. Cordeiro is a Senior Research Engineer at Nokia Research Center where he is involved in the home networking arena and also with standardization activities within the IEEE 802.11 working group. He received his PhD in computer science and engineering in 2004 from the University of Cincinnati, OH, USA, where he won the honorable Outstanding Doctoral Dissertation Award and the prestigious 2003/2004 The National Dean's List Award. Earlier, he obtained a M.S. and B.Sc. in computer science in 1998 and 2000, respectively, from the Federal University of Pernambuco, Brazil. His research interests are in the broad area of wireless and mobile communication including MAC protocol analysis and design, MIMO systems, IEEE 802.11, IEEE 802.15, IEEE 802.16, cognitive radios, power control, spectrum management, ad hoc and sensor networks, routing, multicast, TCP over wireless, and cellular networks. He has published numerous papers in the wireless area and holds 5 pending patents involving MAC protocols for WLANs and WPANs. He has delivered tutorials in areas such as directional antenna systems, wireless broadband and mobile ad hoc and sensor networks, and in the past was the recipient of best paper awards from refereed networking conferences. He has worked for IBM in San Jose, CA, and is a member of the IEEE.

Sachin Abhyankar has concluded BS and MS in Computer Science. He worked as a software engineer for couple of years before joining the MS program at the University of Cincinnati, where he worked as a research assistant in the Center for Distributed and Mobile Computing in the area of wireless ad hoc networks. He has published various papers related with wireless networks in magazines and conferences. Presently, he is working as a software developer for Qualcomm.

Dharma P. Agrawal is the Ohio Board of Regents Distinguished Professor of Computer Science and Computer Engineering and the founding director of the Center for Distributed and Mobile Computing at the University of Cincinnati, OH. He has been a faculty member at the NC State University, Raleigh, NC (1982–1998) and the Wayne State University, Detroit (1977–1982). His current research interests include wireless and mobile networks, distributed processing, and scheduling techniques. He has authored a textbook on Introduction to Wireless and Mobile Systems published by Brooks/Cole in August 2002. He is an editor for the Journal of Parallel and Distributed Systems and the International Journal of High Speed Computing. He has served as an editor of the IEEE Computer magazine, and the IEEE Transactions on Computers. He has been the Program Chair and General Chair for numerous international conferences and meetings. He has received numerous certificates and awards from the IEEE Computer Society. He was selected for the "Third Millennium Medal" by the IEEE for his outstanding contributions. He is a fellow of the IEEE, ACM, and AAAS.