Bottom-Up Construction of Bluetooth Topology under a Traffic-Aware Scheduling Scheme

Rajarshi Roy, Mukesh Kumar, Navin K. Sharma, and Shamik Sural, *Member*, *IEEE*

Abstract—We propose a Bluetooth topology construction protocol that works in conjunction with a priority-based polling scheme. A master assigns a priority to its slaves including bridges for each polling cycle and then polls them as many times as the assigned priority. The slaves can spend their idle time either in a power-saving mode or perform new node discovery. The topology construction algorithm works in a bottom-up manner in which isolated nodes join to form small piconets. These small piconets can combine to form larger piconets. Larger piconets can start sharing bridge nodes to form a scatternet. Individual piconets can also discover new nodes while participating in the master-driven polling process. The shutting down of master and slave nodes is detected for dynamic restructuring of the scatternet. The protocol can handle situations when all the Bluetooth nodes are not within radio range of each other.

Index Terms—Bluetooth, priority-based polling, bridge scheduling, scatternet formation, dynamic reconstruction.

1 Introduction

Bluetooth technology uses a frequency hopping spread spectrum technique for communication between active nodes [3], [9], [15]. A node, after being powered up, must first synchronize its frequency hopping sequence with an existing network before participating in packet exchange. A "piconet" is an ad hoc network of such frequency-synchronized Bluetooth devices. One of the nodes in a piconet takes the role of "master" while the rest become "slaves." A given piconet can contain one and only one master and up to seven active slaves at any point in time. A larger collection of Bluetooth devices forms a "scatternet," which is an inter-networking of piconets joined by "bridge" nodes.

A challenge in the successful operation of a network of Bluetooth devices is the setting up and maintenance of a scatternet topology. The task is complex due to a number of reasons. Bluetooth devices are symmetric in terms of their ability to take up any role; hence, there is no a priori identification of such devices as master, slave, or bridge. The symmetry is broken only after a piconet or a scatternet has been successfully set up with the master(s), the slave(s), and the bridge(s) identified. A scatternet is a truly ad hoc network with no infrastructure other than the nodes themselves. Hence, the nodes can be powered on, shut down, or moved around at any point of time. Any assumption about the initial state of the network in terms of the number and type of the devices is invalid. While it is always good to have a few desirable properties in a scatternet configuration, one cannot possibly run rigorous optimization algorithms to achieve such properties. The reason is twofold. First, Bluetooth devices are power and memory-constrained devices, so running computation-intensive algorithms is not feasible. Second, due to the ad hoc nature of the network, with possibly no information about the initial state itself, such algorithms cannot be implemented efficiently. The topology construction and reconstruction process must be carried out along with the usual packet exchange procedure governed by a master-driven scheduling scheme. Algorithms should have a localized nature so that small regional perturbations can be handled without disturbing the entire network.

In the next section, we provide a background on the various approaches to topology construction and scheduling in Bluetooth. We then give a brief overview of a priority-based scheduling scheme proposed earlier by us in Section 3. This is followed by a detailed discussion on the topology construction algorithm in Section 4. Simulation results are presented in Section 5 and we conclude in the last section of the paper.

2 Background

Attempts have been made during the last few years to develop new and innovative algorithms for Bluetooth topology construction. We review only those approaches that consider dynamic or distributed construction of topologies, since these are the two most important features of a Bluetooth scatternet formation protocol. Detailed surveys may be found in [1], [2], [11], [12], [14], [23]. One of the first such algorithms was proposed by Salonidis et al. [20], called the Bluetooth Topology Construction Protocol (BTCP). Tan et al. proposed a distributed Tree Scatternet

- *R. Roy is with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, 721302, India. E-mail: royr@ece.iitkgp.ernet.in.*
- *M. Kumar is with the Department of Computer Science and Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India. E-mail: sharmamukeshkumar@yahoo.co.in.*
- *N.K. Sharma is with the Computer Associates India Pvt. Ltd, Auriga Block (1st Floor, Left Wing), Vanenburg IT Park, Hi Tech City, Madhapur, Hyderabad, 500081, India. E-mail: sharmanavinkumar@yahoo.co.in.*
- *S Sural is with the School of Information Technology, Indian Institute of Technology, Kharagpur, 721302, India. E-mail: shamik@cse.iitkgp.ernet.in.*

Manuscript received 30 Mar. 2005; revised 7 Feb. 2006; accepted 17 May 2006; published online 15 Nov. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0084-0305.

Formation (TSF) protocol [22]. Their method works well in dynamic environments, in which nodes can arrive and leave at arbitrary time instants. Simulation results indicate relatively short latency during scatternet formation. However, TSF does not minimize the number of piconets. Each master usually has fewer than three slaves. Cuomo et al. have proposed a distributed algorithm that generates a treelike structure, which can dynamically adapt to the mobility of devices [7]. Their approach has a self-healing property since minor changes in the topology may be absorbed in the remaining network. QoS provisioning in scatternets has been considered in a separate work by Cuomo et al. [8]. After generating a tree-shaped scatternet, a second procedure is used to produce a meshed topology by applying a distributed scatternet optimization algorithm. A tree-structured scatternet was also proposed by Záruba et al. [24]. Chiasserini et al. suggested a distributed algorithm for topology formation that considers insertion and deletion of a node to/from a Bluetooth personal area network [6]. While this method can be implemented with moderate complexity, it may generate suboptimal topologies. Stojmenovic proposed a dominating set-based Bluetooth scatternet formation protocol [21]. It assumes that each node knows its position and that of all its neighbors.

Most of the above algorithms assume the Bluetooth nodes to be within radio range of each other. Recently, new scatternet formation algorithms have been proposed that work even without this restriction [13], [16]. Basagni et al. [1] compare a number of scatternet formation protocols for networks of Bluetooth devices. They show that Bluestar [16] is the fastest protocol for scatternet formation and also generates scatternets with lower numbers of piconets and shorter average route lengths. It may be noted that a number of the approaches discussed so far consider a fixed set of devices and do not handle the addition or deletion of nodes [14], [16], [24]. Many of them execute complex optimization algorithms as part of topology construction [12], [14], [21]. Also, most of the techniques use knowledge about the entire topology and the algorithms do not operate on localized information [14], [16], [20].

Research work in Bluetooth has also progressed in a different direction, in which efficient polling schemes have been proposed by researchers. The default polling scheme in Bluetooth is Round Robin (RR). Capone et al. [5] proposed a polling scheme that achieves high efficiency by reducing the rate of polling of slaves that return empty packets in the previous polling. Zhu et al. [25] suggested a policy in which a master serves the slave that has packet with smallest expected arrival time. Their policy predicts the packet length and the number of packets. Bruno et al. introduced an Efficient Double Cycle (EDC) to improve the RR polling [4]. Kalia et al. [10] presented a priority-based polling (PP) scheme and a K-fairness polling policy (KFP) for Bluetooth. We have earlier proposed a distributed Bluetooth topology construction protocol that can work in a dynamic environment where a node may be started or shut down at any point of time [18]. We have also suggested a polling scheme with priority for handling packet loss, power utilization, delay, and slot wastage problems faced in a Bluetooth piconet of heterogeneous devices [17]. Recently, we have extended this scheme to consider both intrapiconet as well as bridge scheduling in scatternets [19].

A careful analysis of existing research results in the field of Bluetooth, including our earlier work, reveals that the two problems, namely, topology construction and scheduling, have so far been considered independently. While setting up the topology, the fact that all message transfers can only be carried out through a master-driven polling process is overlooked. Similarly, at the time of suggesting efficient polling schemes, the amount of time a master or a slave must spend in new node/piconet discovery is not taken into consideration. To the best of our knowledge, there is no report on dynamic and distributed topology construction for Bluetooth that considers the finite amount of time available for node discovery along with master-slave polling for intrapiconet and interpiconet packet transfer. In this paper, we propose a simple and implementable Bluetooth scatternet formation protocol that uses only local information and considers the amount of time the devices need to spend in the master-slave polling process. The rest of their time is utilized for new node discovery, topology construction, and reconstruction. We consider both intrapiconet and interpiconet packets while designing the scatternet formation protocol. The Bluetooth nodes need not be within radio range of each other.

3 Priority-Based Polling Scheme with Bridge Scheduling

We have earlier proposed a polling technique in which the polling sequence is determined by a priority assigned by a master to each of its slaves, including bridges. The slaves are polled in decreasing order of their priority and a slave is polled as many times as its priority before the next slave is polled. While the details of the method are available in our earlier work [17], [19], we briefly describe the main features needed for building the context in which the proposed scatternet formation algorithm can be appreciated.

In each polling cycle, a master polls its slaves and also calculates their priority for the next cycle using a Master Communication Protocol (MCP). In order to initiate the polling process, slaves send their buffer sizes when they first communicate with the master. Thereafter, in each cycle, when a slave is polled for the last time, it lets the master know about the packets generated since the last polling cycle and the next k head-of-line packet length structure in its buffer using a Slave Communication Protocol (SCP). For scheduling the bridges, a bridge is assigned to each piconet for a quantum of time determined by the interpiconet traffic. We define three states of a bridge in a Bridge Communication Protocol (BCP). These are 1) to_stay: A bridge is part of a given piconet and will not leave the piconet during the next polling cycle, 2) to_join: A bridge is waiting to join a given piconet and its master will try to include it in the next polling cycle, and 3) to_release: A bridge is expected to leave its current piconet in the next polling cycle after receiving an *unhold* message from the new master. When any master gets control of a bridge, it first calculates the time for which the bridge can be retained using a bridge scheduling table. After the time is elapsed,

the current master releases the bridge if there is another master waiting for this bridge. Otherwise, the current master continues to use the bridge for the complete polling cycle and then tries to release it in the next polling cycle.

In each polling cycle, a slave with higher priority is polled a higher number of times than one with lower priority. The priority calculation step is based on 1) the packet arrival rate and the available buffer space in each slave, 2) the number of packets stored in the master buffer for each slave, and 3) the states of the bridges of the piconet. The slaves are made aware of the polling sequence so that they can go to the *hold* power-saving mode between polling in two successive cycles and wake up in between to listen to broadcast messages. This polling scheme has been shown to have better performance compared to other priority-based schemes in terms of throughput, packet loss, and delay [17], [19].

In static situations, the slaves spend their "idle" time in a power-saving mode. However, to take into account the dynamic growth of a scatternet, we modify our earlier protocol and let the nodes spend this idle time in new node/piconet discovery as well. When the traffic in a piconet is low, a master can also participate in the node discovery process.

4 Dynamic Topology Construction—A Bottom-Up Approach

To accommodate the requirements identified in the Introduction, we must consider the Bluetooth topology construction algorithm to be an inherent property of the devices just like their participation in the polling activity. The primary goal of the proposed protocol is twofold: 1) An isolated node should be able to connect to another isolated node or to an existing piconet within its radio proximity and 2) an existing piconet should be able to discover another existing piconet, thus forming or extending a scatternet. While building the topology, the protocol should also attempt to form a fully connected and balanced network. By fully connected, we mean a scatternet configuration in which *any* two piconets are directly connected by a bridge. If some of the nodes are outside the radio range of each other, such a fully connected scatternet may not be achieved. Further, if the node density is very high, although there is a higher probability of two piconets getting connected, they may only be connected through a third piconet instead of being connected directly through a bridge. It can be shown that a fully connected scatternet can be formed only when the number of Bluetooth nodes is less than or equal to 36 even when all the nodes are within radio range of each other. This situation is quite rare in practice and, even if it happens, our protocol attempts to minimize the number of hops to connect one piconet to another. A balanced network is one in which all the piconets have a similar number of nodes so that none of the masters is unduly loaded. In this paper, the term optimum denotes a configuration in which a scatternet is balanced and all the piconets are either fully connected or connected through the minimum number of hops.

4.1 Protocol Overview

In the topology construction protocol, any node can become ON at any instant in time. Here, ON means that either a new node has been powered up or an existing node has come within radio proximity of another node. There are four roles that an ON node can take up—Isolated (I), Master (M), Slave (S), and Bridge (B). A node can also become OFF at any instant in time. Here, OFF means that the node is unable to communicate with its master (or its slave(s) if the given node is a master) either because it is powered off or has moved out of the radio range. Due to mobility, a node that has become OFF in one piconet can become ON in another piconet. This reflects a real world situation in which battery-powered Bluetooth devices are powered up, join other nodes forming an ad hoc network, and then either shut down after exchanging voice/data packets for a limited period of time or moved to a new location.

In this paper, we do not mention the Page and the Page-scan states explicitly, but assume that they always follow Inquiry and Inquiry-scan. During the Page and Page-scan states, symmetric information exchange occurs between the nodes. Although Bluetooth baseband specifications do not put the restrictions, in the scatternet formed using the proposed algorithm, two piconets are connected through a single bridge. Also, only a slave is used as a bridge node, which is usually shared by at most two piconets. The reason is that, other than the masters, the bridge nodes are the ones that handle most of the scatternet traffic. If a bridge is shared by more than two piconets, throughput decreases and there is a possibility of bridge node failure due to high power consumption. This condition is relaxed only if there is no other way of forming a scatternet. The same arguments hold for our consideration of only slave nodes as bridges.

4.2 Protocol Details

In this section, we explain our protocol using all possible scenarios when two nodes discover each other. The implementation details are provided in Section 4.3. The different scenarios have been shown as different cases in Fig. 1. The figure refers to two routines, namely, Piconet Formation and Modification Routine (PFMR) and Scatternet Formation and Modification Routine (SFMR) of Figs. 2 and 3, respectively, depending on the roles of the nodes that discover each other and the configurations of the piconets (if any) to which they belong.

A newly started node takes the role I and executes an Isolated Node Protocol (INP), as shown in Fig. 4. It enters the Inquiry or the Inquiry-scan state with equal probability. The time it spends in either of the states is chosen randomly between 0.3125 and 1.35 sec. These two limits were determined through extensive simulation. Switching from one state to the other is repeated till it latches with another node. While running INP, the new node can be latched either to another isolated node or to a master/slave of an existing piconet performing node discovery. If the second node is also isolated, the two together form a piconet of just two nodes using the Piconet Formation and Modification Routine of Fig. 2, with one taking up the role of master and the other that of slave. The node with higher capability in terms of memory and available power is given the role of

```
CASE 1.  TWO ISOLATED NODES DISCOVER EACH OTHER
       Call PFMR
CASE 2.  A MASTER DISCOVERS AN ISOLATED NODE
       Case 2.a No. of active slaves in the existing piconet < 7
              Call PFMR
       Case 2.b No. of active slaves in the existing piconet = 7
              Determine if there is any existing non-bridge slave of the piconet that
              is within radio range of the isolated node
              If there is no such node
                   Send the least busy slave to the Park mode and include the new node
                   as an active slave
               Else
                   Call SFMR to make the isolated node as the master of a new piconet,
                   transfer up to 4 non-bridge slaves to this piconet that are within
                   its radio range, making the slave with the highest capacity as the
                   bridge node
CASE 3.  A SLAVE DISCOVERS AN ISOLATED NODE
       Case 3.a No. of active slaves in the existing piconet < 7
              If master of the existing piconet is within radio range of the isolated
              node
                   Call PFMR to include the isolated node in the existing piconet
              Else
                   Determine if there is any existing non-bridge slave of the piconet
                   that is within radio range of the isolated node
                     /* There is at least one such node, which is the slave that
                     discovered the isolated node */
                   Call SFMR to make the isolated node as the master of a new piconet
                   Transfer up to 4 non-bridge slaves that are within its radio range,
                   making one of them as the bridge node
       Case 3.b No. of active slaves in the existing piconet = 7
              Similar to the Else condition of Case 3.a above
CASE 4.  A SLAVE (S1 of piconet P1) DISCOVERS A MASTER (M2 of piconet P2) AND
              VICE VERSA
       Case 4.a Total No. of active slaves in P1 and P2 = 14
              If there is no bridge node between P1 and P2
                     Send the least busy slave of P2 to the Park mode
                     Make S1 as a bridge between P1 and P2
       Case 4.b Total No. of active nodes (master + slave) in P1 and P2 > 8 and < 16
              If No. of active slaves in P2 < 7
                   If there is no bridge between P1 and P2
                          Make S1 as a bridge between P1 and P2
                          Transfer up to 4 non-bridge slaves from the larger piconet
                          to the smaller piconet that are within radio range of the
                          master of the smaller piconet
                            /* The last step will hereinafter be referred to as
                            "Perform Load Balancing" */
                   Else
                          Perform load balancing
              Else
                   If there is a bridge between P1 and P2
                          Perform Load Balancing
                   Else
                          Determine if there is any non-bridge slave of P2 that is
                          within radio range of M1
                          If at least one such non-bridge slave exists
                                  Make that slave of P2 as a bridge between P1 and P2
                                  Perform load balancing
                          Else
                                  Send the least busy slave of P2 to the Park mode
                                  Make S1 as a bridge between P1 and P2
```

Fig. 1. Protocol for bottom-up construction of topology in Bluetooth.

master as shown in the figure. PFMR is also called if a master discovers an isolated node (Case 2 of Fig. 1) and the node can be included in the existing piconet (Case 2a).

However, if the piconet already has seven slaves and no nonbridge slave is in the radio range of the new node, the master sends the least busy slave into Park mode and includes the new node in its piconet. If there are more than seven slaves in a piconet, the master keeps on sending an active slave into Park mode and Unparking an already parked slave following either a least priority policy or any other policy as governed by the master, ensuring fairness among the slaves. Apart from listening to beacon train or broadcast messages sent by a master, a parked slave can also spend the rest of the time in device discovery, acting like an isolated node. If a master does not receive any response from a parked slave within the link supervision

```
      Case 4.c Total No. of active nodes (master + slave) in P1 and P2 <= 8
            If all the nodes of P1 are within radio range of M2 and all the nodes
            of P2 are within radio range of M1
                 Merge P1 and P2 by calling PFMR and make either M1 or M2 as the new
                 master of the combined piconet depending on whichever has higher
                 power
                 If there was a bridge between P1 and P2, make it a non-bridge slave
            If all the nodes of P1 are within radio range of M2
                 Merge P1 and P2 by calling PFMR and make M2 as the new master for
                 all the nodes of P1 and P2 combined
                 If there was a bridge between P1 and P2, make it a non-bridge slave
            If all the nodes of P2 are within radio range of M1
                 Same as above with roles of M1 and M2 reversed
            If some of the nodes of P1 are not within radio range of M2 and some of
            the nodes of P2 are not within radio range of M1
                 If there is no bridge between P1 and P2
                      Make S1 a bridge between P1 and P2
                      Perform Load Balancing
                 Else
                      Perform Load Balancing
CASE 5.  A SLAVE (S1 of a piconet P1) DISCOVERS A SLAVE (S2 of a piconet P2)
      Case 5.a Total No. of active slaves in P1 and P2 = 14
            Similar to Case 4a above
      Case 5.b Total No. of active nodes (master + slave) in P1 and P2 > 8 and < 16
            If there is a bridge between P1 and P2
                 Perform Load Balancing
            Else
                 If there is at least one non-bridge slave in P1 within radio range
                 of M2 or one non-bridge slave in P2 within radio range of M1
                         Make such a slave a bridge between P1 and P2
                         Perform Load Balancing
                 Else
                         Interchange roles of S1 and M1 or S2 and M2 depending on
                         whether P1 is smaller than P2 or not
                         Follow the fourth outer if condition of Case 4c above
      Case 5.c Total No. of active nodes (master + slave) in P1 and P2 <= 8
            The first three outer if conditions similar to Case 4c above
            If some of the nodes of P1 are not within radio range of M2 and some of
            the nodes of P2 are not within radio range of M1
                   Similar to Case 5b above
CASE 6.  A MASTER (M1 of a piconet P1) DISCOVERS A MASTER (M2 of a piconet P2)
      Case 6.a Total No. of active slaves in P1 and P2 = 14
            Similar to Case 5a above
      Case 6.b Total no. of active nodes (master + slave) in P1 and
            P2 > 8 and < 16
            Similar to Case 5b above
      Case 6.c Total No. of active nodes (master + slave) in P1 and P2 <= 8
            Similar to Case 5c above
```

Fig. 1. Continued.

timeout period set for that slave, it considers the slave to be disconnected or moved to another piconet. If there is at least one nonbridge slave of an existing piconet that is within radio range of the new node, a new piconet is created with the isolated node as its master. One such nonbridge slave within radio range of both the original master and the new master is considered the bridge node connecting the two piconets. The Scatternet Formation and Modification Routine of Fig. 3 is used to handle this condition. If there is more than one nonbridge slave of the original piconet that are within radio range of the new master, local load balancing is performed by transferring up to four such slaves for improving the scatternet configuration.

If a slave discovers a node, it forms a temporary piconet with the discovered node for information exchange. We use a special Link Manager Protocol PDU (Protocol Data Unit) similar to [16] in order to facilitate such communication and information exchange in the temporary piconet. Details of such a PDU are given in the next section. While choosing a slave for device discovery, a master prefers the one with the least traffic load. In order to cover a large geographical area surrounding its piconet, the master can also choose a slave at random. Such a node is polled at the start of a polling cycle if its calculated priority is greater than the average priority of all the slaves. Otherwise, the master polls it at the end of the polling cycle. Thus, the master ensures that the designated slave has the maximum time for device discovery. When a slave discovers an isolated node (Case 3 of Fig. 1) and the number of existing slaves in the piconet is less than seven (Case 3a), the isolated node can be included in the existing piconet by calling PFMR if it is within radio range of the master. However, if the isolated node is not within radio range of the master or the number of existing slaves is seven (Case 3b), we create a new piconet having the isolated node as master and one of the nonbridge slaves of the existing piconet as the bridge. Since the slave under consideration

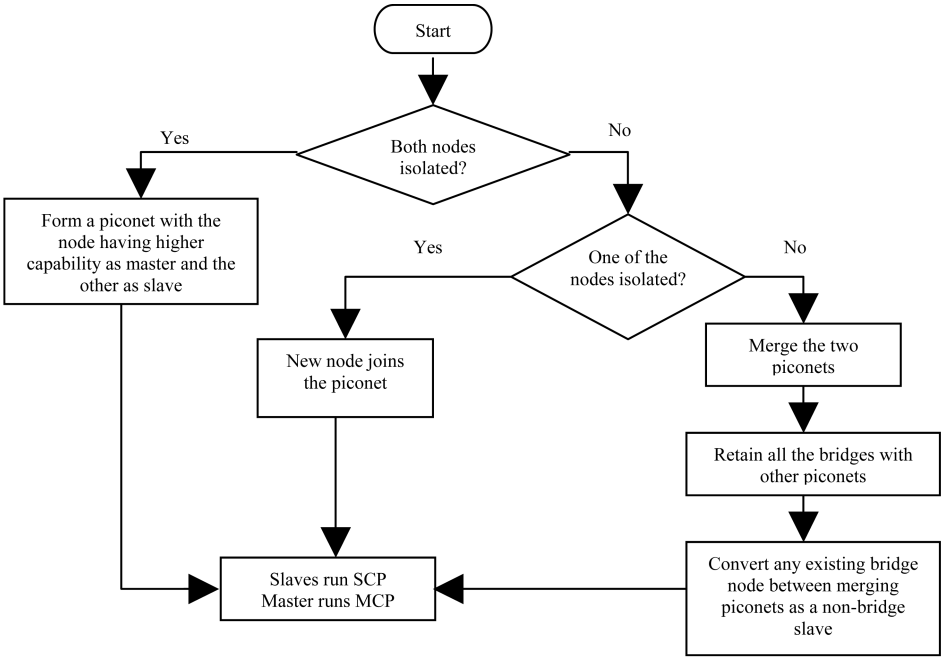


Fig. 2. Piconet Formation and Modification Routine (PFMR).

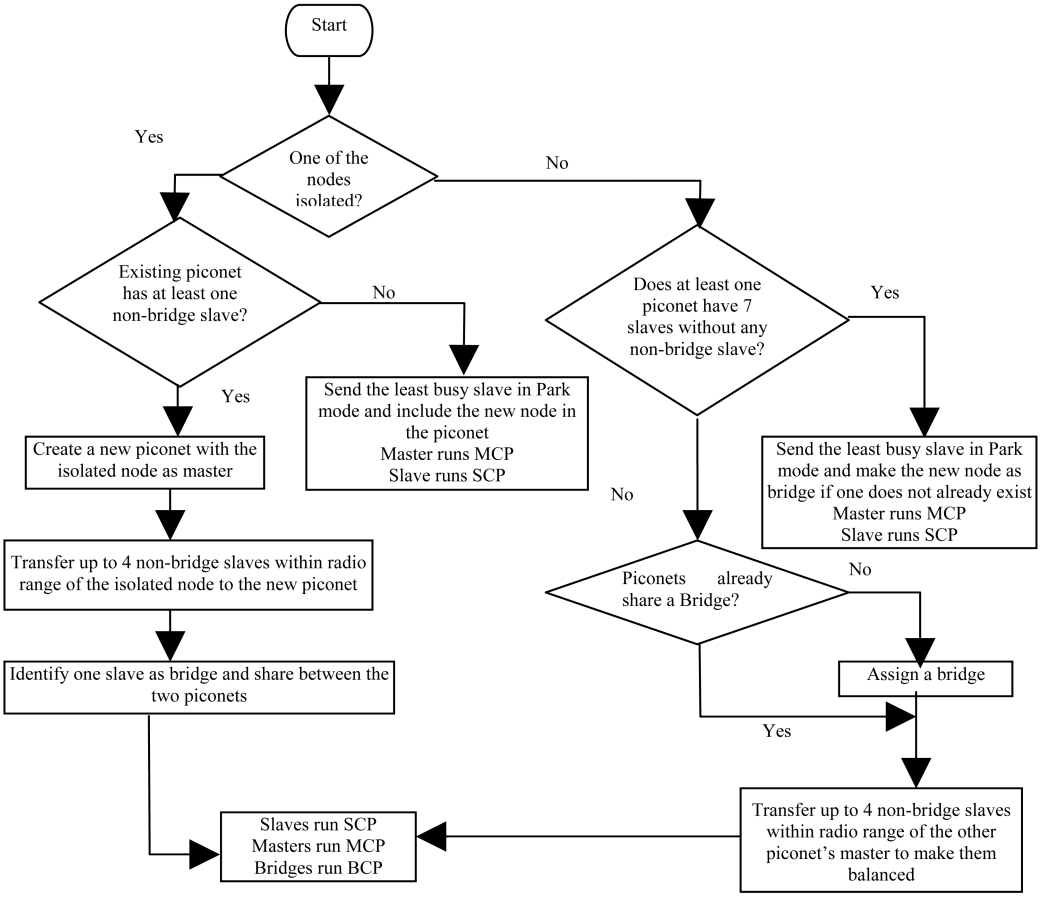


Fig. 3. Scatternet Formation and Modification Routine (SFMR).

had discovered the isolated node, existence of at least one slave within radio range of the isolated node (master of the new piconet) is guaranteed. SFMR is used to handle this situation. Local load balancing is also applied, if possible.

Instead of discovering an isolated node, it is possible that, during node discovery, the slave discovers or is discovered by an existing master or a slave of another piconet (Cases 4 and 5, respectively). In these situations, if

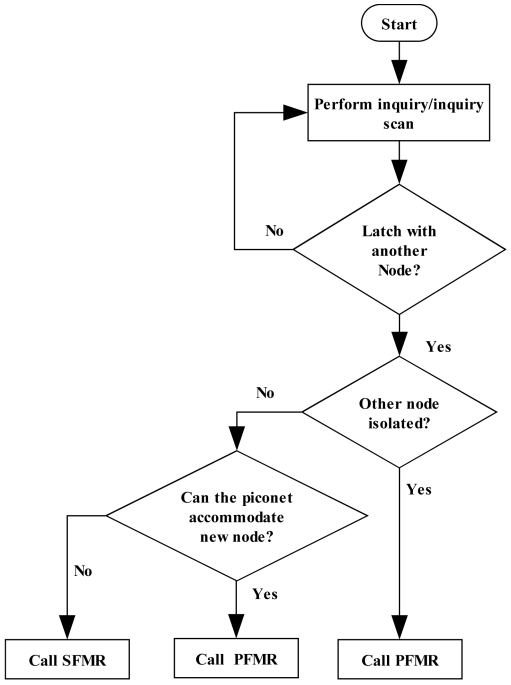


Fig. 4. Isolated Node Protocol (INP).

the total number of slaves in the two piconets discovering each other is 14 and no bridge exists between these two piconets, the master sends its least busy slave into Park mode and makes the discovering or discovered slave a bridge node between these two piconets. It may be argued that one could create three piconets using the two masters and the 14 slaves if a sufficient number of bridges were identified. However, this leads to high message complexity in scatternet reconfiguration and, hence, we avoid such restructuring. If the total number of slaves in the two piconets is less than 14, restructuring is done as required by calling either PFMR or SFMR.

A master can also go in device discovery mode if the traffic load is low in its piconet or the number of slaves is low (two or three). The slaves stay in power-saving mode during this phase. Also, a master can send a slave for more than one polling cycle in device discovery phase if the number of slaves is low (two or three). Situations in which a master discovers an isolated node or a slave of another piconet have already been discussed above. However, a master may also discover or be discovered by another master (Case 6). Such situations are handled in a manner similar to the case where a slave discovers a slave.

4.3 Implementation Details

In this section, we give a detailed description about how the above-mentioned protocols can be implemented in Bluetooth. We propose a special Link Manager Protocol PDU (named LMSFP–Link Manager Scatternet Formation Protocol) for control and information exchange during scatternet formation. It is a special type of DM3 packet [3] whose payload body is shown in Fig. 5. The first bit is the transaction ID, which is 0 if the PDU forms part of a transaction that was initiated by a master and 1 if initiated by a slave. A 15-bit operation code represents the type of information being exchanged by a particular LMSFP. The 3-bit No_Device (Number of Devices) field indicates the number of devices whose information is present in the payload body. This field is followed by a 1-bit OTR (Operation Time Range) that takes a value of 1 if the payload body contains the operation start time (T_{Start}) and operation end time (T_{End}) and 0 otherwise.

At the start of any polling cycle, a master polls the slave that was sent for device discovery in the previous cycle. In response to the polling, the slave sends back an LMSFP packet. The LMSFP packet includes the device ID, clock value, and role of each member of the other piconet if the slave had discovered a slave or the master of another piconet. If an isolated node was discovered, the LMSFP packet contains the device ID and clock value of the isolated node. For any discovered node, LMSFP also contains (in the T_{start}, T_{end} values) the time negotiated between the slave and slave/master of the other piconet to start the piconet merging/reshuffling procedure. If no node was discovered, the slave responds with a NULL packet.

After receiving this packet from the slave, the master broadcasts a new LMSFP packet to all its slaves and then starts piconet merging/reshuffling at the negotiated time. At the end of this stage, the master broadcasts the updated piconet information to all its slaves. This update contains the device ID, clock value, and role of each of the members of the existing piconet in the form of an LMSFP packet with OTR bit set to zero. The master then calculates the priority and timing sequence of each slave using the above-mentioned method and sends the new slave to device discovery. It broadcasts the new configuration within its piconet.

Whenever a slave in the device discovery phase detects another slave or a master, it makes a temporary piconet with the peer and exchanges piconet information (ID, clock value, and role of each member of its piconet) using an LMSFP. The LMSFP also contains the time after which that slave will be polled for the next cycle. Since the slave has all this information about its own piconet, it can determine

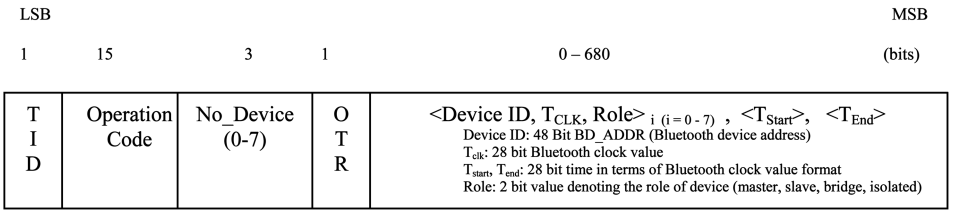


Fig. 5. Payload body of LMSFP PDU.

whether piconet merging or reshuffling is possible. Based on this, the slave chooses the appropriate protocol (SFMR or PFMR) and also the state (page or page scan) in which the slave/master should go. The slave considers the larger of the two time values left for polling in the two piconets as the reference time (suppose that, for one piconet, the next polling starts after time T1 and, for the other piconet, it starts after time T2, where $T2 > T1$; then, T2 is taken as the reference time) and they exchange necessary information with each other. Then, the temporary piconet is broken and both the nodes go back to their previous states.

As discussed in case 2b of Fig. 1, when a master having seven active slaves discovers an isolated node, it first determines if there is any nonbridge slave of its piconet in the radio range of the isolated node. The procedure for range or neighbor determination works as follows:

Step 1. The master sends the device ID and clock value of all the members of its piconet, paging start time (the time after which the isolated node should start paging each nonbridge slave of the piconet) and paging end time (the time within which the isolated node should complete paging of all nonbridge slaves) using an LMSFP packet. The paging start time is generally the time when the master comes to normal communication in the piconet and broadcasts the control information (directing each nonbridge slave to go into page scan mode) in its piconet. Since the isolated node has the device ID and clock value of each node, it can directly page each device. This is done in order to check the availability of nodes in its communication range.

Step 2. At the beginning of the next polling cycle, the master sends a control packet to all its members, directing each nonbridge slave to go into page scan mode.

Step 3. The isolated node pages all the nonbridge slaves one after another and decides which is in its communication range depending on the response from the slave. It then sends the list of observed slaves to the master using an LMSFP packet.

Step 4. If there is at least one slave in the observed list mentioned above, the master calls SFMR. It uses load balancing as mentioned in Fig. 1. Each master broadcasts its piconet information to all its slaves. The master also updates the information of its own piconet as well as the neighboring piconet.

Step 5. If there is no such slave as mentioned in Step 4 above, the master sends its least busy slave into Park Mode and includes the isolated node as a new active slave. The slave in Park mode can also go in device discovery and behaves like an isolated node while discovering other devices. If it gets connected to some other piconet, it leaves the existing piconet.

In a manner similar to above, when a slave discovers an isolated node, it sends a control packet and directs the isolated node to go into page scan mode for a time estimated by the slave. The estimated time is the time after which the paging of isolated node would be completed. At the start of next polling cycle, the master polls this slave for any discovery information. The slave responds with an LMSFP packet containing information only about the isolated node. Then, the master goes in page mode and starts paging the isolated node. If the master can discover the isolated node during paging, it includes the isolated node into the existing piconet following the same procedure as discussed above. Otherwise, it assumes that the isolated node is not in its radio range. In the latter case, the previous slave again pages the isolated node and sends an LMSFP packet containing the device ID, role, and clock value of all the devices in the existing piconet. It also directs the isolated node to go into page mode after a quantum of time. This is the time when the master sends all its nonbridge slaves to the page scan mode. The process of neighbor node detection and load balancing is similar to the case described before. The only difference is that the information exchange between the isolated node and master of the existing piconet through LMSFP packet is done via the slave that discovered the isolated node. It may be noted that all the information exchanged between a paging and a paged device is done after successful paging and setting up of a temporary piconet. As soon as the LMSFP packet exchange is over, the temporary piconet is broken. Each slave in a piconet has all the required information (namely, device ID, clock value, and role of each member of its piconet), so that it can take the decision, exchange the piconet information with a discovered device, and direct the isolated node to go into page or page scan mode accordingly.

When a master discovers a slave or vice versa, the procedure of finding nodes in radio range and load balancing is similar to that discussed above. The only difference is in the steps involved in information exchange between the discovering device (master or slave) and the discovered device (master or slave).

4.4 Discussions on the Protocol

It is observed from the protocol details that we avoid the reassigning of bridges since it has high communication overhead and also disrupts services of those parts of the scatternet that were not involved in the current node discovery. When two piconets discover each other and get merged, any existing bridge node is converted to a nonbridge slave. If both the piconets had bridges with a third piconet, one of the two bridges is not used anymore. Since a typical routing strategy in Bluetooth is on-demand routing, such conversion of one bridge node does not greatly affect the overall routing in the scatternet.

It may be noted that we do a relative priority calculation. If traffic is high for every node, the one having maximum traffic load gets maximum priority and the others are assigned priority based on their ratio of traffic load. Similarly, if traffic load is low, the one having maximum traffic load gets maximum priority and the others get in the same ratio. It means that even if the traffic is low, the cumulative priority calculation of all slaves, and thus, the time left for inquiring, does not have a large variation unless the number of slaves varies. If the number of slaves is low, the master can send one of them in node discovery mode for more than one cycle. The master itself may also go for device discovery when traffic is light or the number of slaves is low. If traffic is light, the average packet transmission delay will not be high. On the other hand, if the number of slaves is low, the master polls the slaves more frequently after coming back from the node discovery phase. Once again, the impact on average packet transmission delay is low. Our algorithm has two main advantages over other existing algorithms. First, the

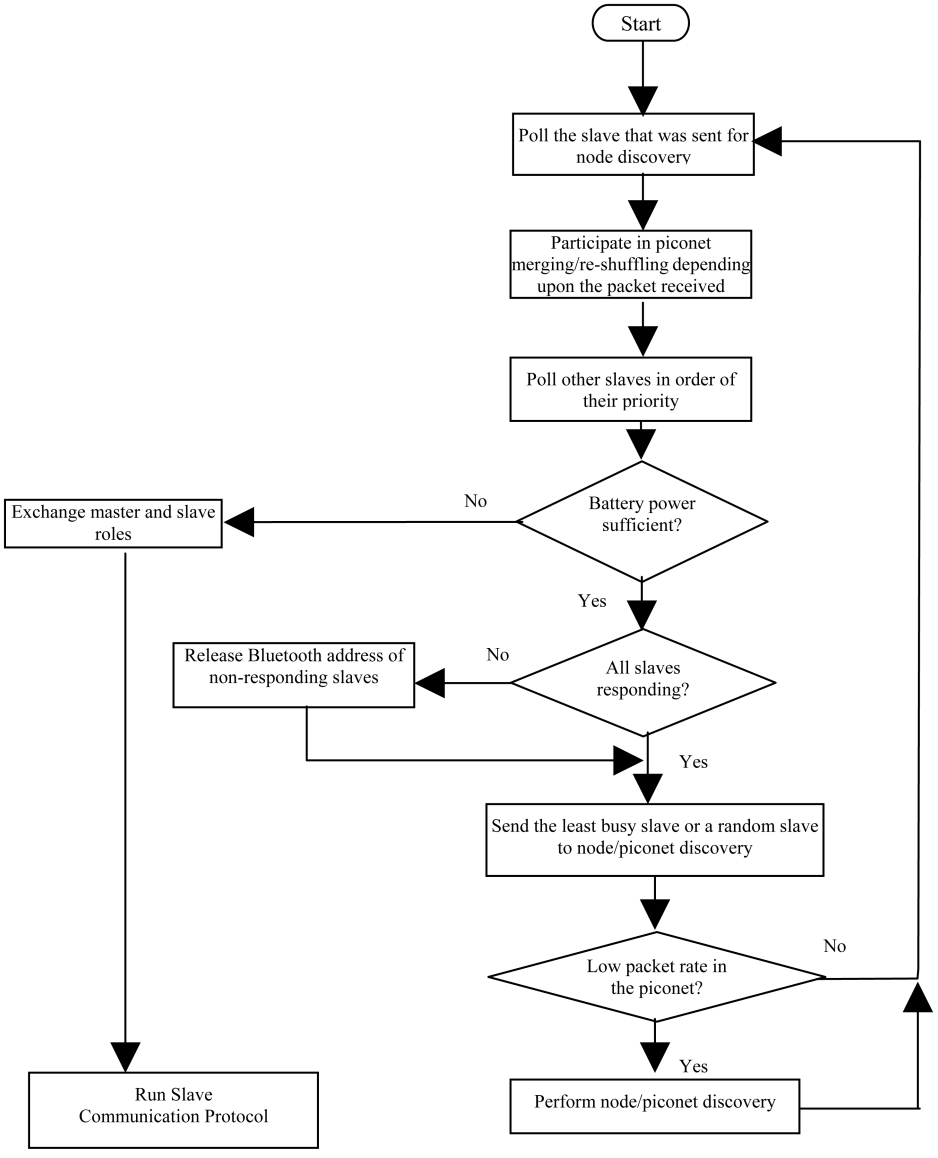


Fig. 6. Enhanced master communication protocol.

topology construction protocol is ingrained in the basic communication infrastructure itself rather than being a separate standalone process. Second, it does not disturb the whole scatternet for small local perturbations, thus reducing the message complexity and time delay in scatternet reconstruction and preventing the network resources from being wasted in topology construction itself because of the frequent mobility of Bluetooth devices.

In our discussions above, we have not mentioned the effect of shutting down or moving away nodes. If a slave is shut down, the master can detect it since the slave will not respond to polling packets in successive polling cycles. The master then releases the Bluetooth address of this slave for reuse. Also, if the available battery power of a master falls below that of a slave in its piconet, the master and the slave swap their roles for maximizing the lifetime of the network. This enhanced Master Protocol is shown in Fig. 6.

Similarly, a slave detects master failure if it is not polled at the expected time and the broadcast packet is also not received. The slave then considers itself to be isolated and runs Isolated Node Protocol. The enhanced Slave Protocol is shown in Fig. 7. The enhanced Bridge Protocol of Fig. 8 also shows detection of master failure. If a bridge detects that one of its masters is not responding, it becomes a nonbridge slave of the other master.

4.5 Analysis of the Protocol

In this section, we show that the proposed protocol can indeed form scatternets if the nodes participate in the routines described above with a lower bound on the maximum priority that can be assigned to each slave. We first formulate the problem of maximum time required for latching two Bluetooth nodes while one is in Inquiry and the other is in the Inquiry-scan state. This analysis is done considering a situation in which all the nodes are within radio range of each other.

The node doing Inquiry runs its clock—at double the rate of the node running Inquiry-scan. So, the worst-case

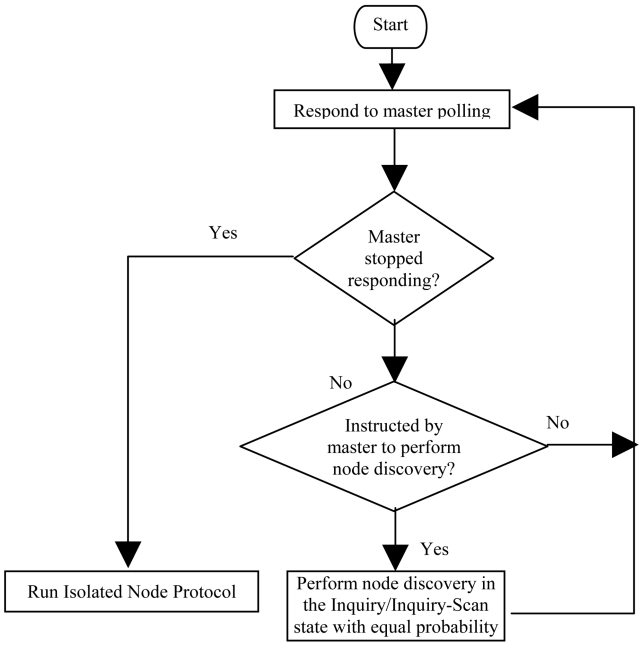


Fig. 7. Enhanced slave communication protocol.

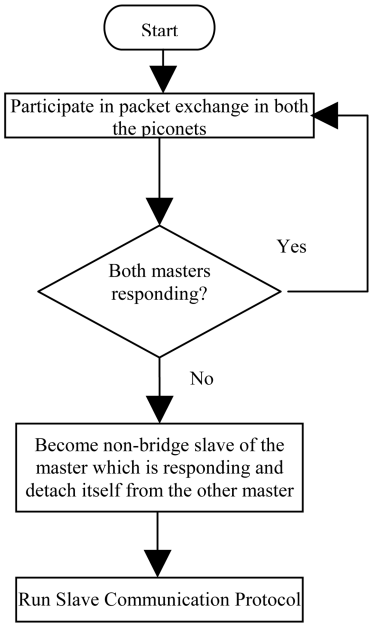


Fig. 8. Enhanced bridge communication protocol.

latching time is 32 normal Bluetooth clock pulses since there are 32 frequencies in the hopping sequence at the time of Inquiry. In the best case, the latching time is one clock cycle and the average case latching time is 16 clock cycles. The amount of time a slave spends in Inquiry-scan depends on the current traffic condition in the piconet. Let us assume that each node is being polled by the maximum priority number "P" in the polling scheme of our algorithm. Let us also assume that each slave generates one, three, or five slot-length packets with equal probability and, hence, the average packet size is three slots. So, each slave gets $(3+1)\mathrm{P} = 4\mathrm{P}$ slots for its use. If there are seven slaves, then the total number of slots used is 28P. The first slave to be polled gets the broadcast packet, uses 4P slots for packet transfer, and then gets 24P slots before the next broadcast from the master. So, for each slave, the 28P slots are divided in three parts: 1) Time from broadcast to the time it gets polled first, 2) slots used for its own packet transfer, and 3) remaining time before the next broadcast. Table 1 shows these three components of time for all the seven slaves of a piconet.

It is seen from Table 1 that the worst-case time for node discovery is 12P. It should be noted that, if all the slaves reach the maximum priority level, then none of these can be released for node discovery. If one node goes for node discovery from a piconet, it is most likely a node with a least priority of 1. Hence, this slave will be polled last in a polling cycle according to our priority-based polling algorithm shown in Fig. 6. Thus, the time available for this slave for new node discovery will be 24, 28, 32, ..., up to 24P slots, depending on the priority structure assigned to the other slaves by the master in the current polling cycle. In the case when the other slaves have highest priority P, the time available for node discovery by the least-priority slave will be 24P slots. This time will be even less if there are less than seven slaves in the piconet. A slave goes in node discovery and stays in the Inquiry-scan state for 24P slots before it gets polled again. If we consider the discovery of an isolated node, then the isolated node toggles between Inquiry and Inquiry-scan states, and let us assume that, in these two states, it spends TI and TIS number of time slots, respectively. Therefore, the least time available for latching is $(24\mathrm{P} - \mathrm{TIS})/2$ and the maximum time available is $\min(24\mathrm{P} - \mathrm{TIS}, \mathrm{TI})$. For latching, this minimum time available must be greater than the maximum time required for two nodes to align their frequency cycles and the same is already calculated to be 32 slot times. Therefore, $(24\mathrm{P} - \mathrm{TIS})/2 > 32$.

TABLE 1
Time Available for Different Slaves in a Polling Cycle

Slave No.	Time from Broadcast to Poll	Time for Polling	Time from Poll to Next Broadcast	Total time between two Broadcasts
1	0	4P	24P	28P
2	4P	4P	20P	28P
3	8P	4P	16P	28P
4	12P	4P	12P	28P
5	16P	4P	8P	28P
6	20P	4P	4P	28P
7	24P	4P	0	28P

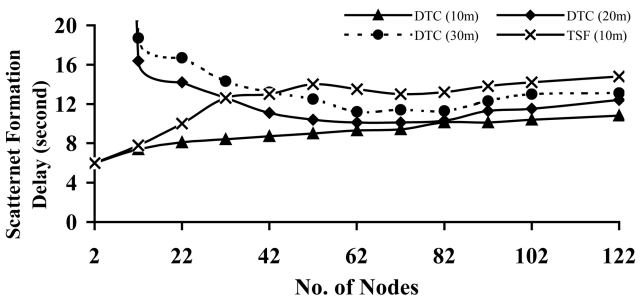


Fig. 9. Scatternet formation delay.

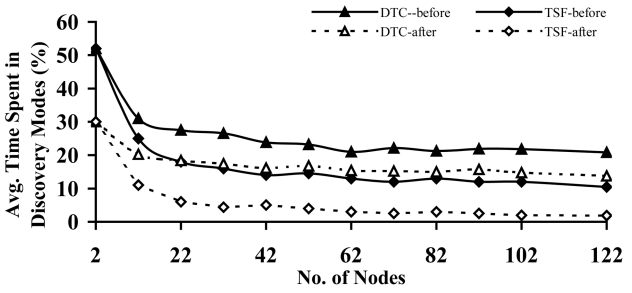


Fig. 10. Average time spent in discovery mode.

When two isolated nodes try to find each other, both of them toggle between Inquiry and Inquiry-scan states for TI and TIS time slots, respectively. But, their sequence of changing states could be out of phase. So, eventually, we get a situation when, for the min(TI,TIS) slot time, one is in Inquiry and the other is in the Inquiry-scan state. This is the maximum amount of time we can get for latching these isolated nodes. By the aforementioned logic, $\min(\mathrm{TI}, \mathrm{TIS}) > 32$. Combining the two relations, we get $32 <= \mathrm{TIS} <= (24\mathrm{P} - 64)$, so that $\mathrm{P} >= 4$. We typically set a value of 5 for P in our algorithm so that the probability of latching becomes higher.

For situations in which all the nodes are not within the radio range of each other, there is no such straightforward way of determining the value of P. We therefore ran extensive simulations to find the most suitable value of P for node discovery under various types of traffic load parameters. It was found that a value of P between 4 and 6 is most favorable. Hence, we have taken $\mathrm{P} = 5$ as the optimal choice.

5 Simulation Results

We measure the performance of the protocol through extensive simulation studies. The TSF approach proposed by Tan et al. [22] is the one closest to our work when all the nodes are within radio range of each other. To make an effective comparison, we show the relative performance of the proposed Dynamic Topology Construction (DTC) algorithm with TSF under situations when the geographical area of random node deployment is in a circle of diameter 10m. As per the specifications, 10m is the radio range of Bluetooth devices. We also consider two other situations in which the nodes are distributed over circular areas of diameters 20m and 30m, respectively. For studying multihop results, we compare our protocol with the multihop algorithm proposed by Petrioli et al. [16]. For these comparisons, we consider the nodes to have uniform distribution within a square area of size (30m $\times$ 30m). Performance of the protocols is observed under three broad categories: 1) static analysis, 2) dynamic analysis, and 3) quality of network formed. In static analysis, we consider that new nodes are not joining and nodes are not leaving a scatternet. The joining and leaving of nodes are considered in dynamic analysis.

For static analysis, we consider two metrics, namely, 1) scatternet formation delay and 2) time spent in node discovery as shown in Figs. 9 and 10, respectively. From Fig. 9, it is observed that when the number of nodes is low, the scatternet formation delay for DTC (20m) and DTC (30m) is high. The reason is that the probability of finding a node in the opposite state in the communication range of any given node is low. However, as the node density increases, the scatternet formation delay decreases. In case of TSF, a larger number of connected components finally merge into a single one. Here, a component refers to a tree scatternet. When there are a higher number of components, latching becomes easier and components merge quickly. But, finally, when only a few components (about two or three) remain, frequency synchronization becomes difficult, resulting in a substantial delay in TSF.

The average time spent in discovery mode both before as well as after establishment of networks is plotted in Fig. 10. It is seen that the proposed algorithm has a higher amount of time spent in discovery mode. This is due to the fact that we give more importance to the dynamic maintenance and expansion of the scatternet. The priority-based polling scheme enables us to perform this step, which is not available in TSF. A coordinator node in TSF is a tree node that is being elected by its root node for discovering other tree scatternets or components. In our protocol, all the nonbridge slaves can potentially go for node discovery while, in case of TSF, only the coordinator can go in the discovery phase, which is only one for each component. When the initial network formation is over, only one component is left with one coordinator. In a static environment, our algorithm spends more time in searching. Since it is not really known whether the network formation is complete, we choose to keep the option of node discovery running.

To study the dynamic performance of the protocol, we consider free node connection delay and recovery delay caused by master failure. Fig. 11 shows free node connection delay. When the number of nodes is low, the isolated node

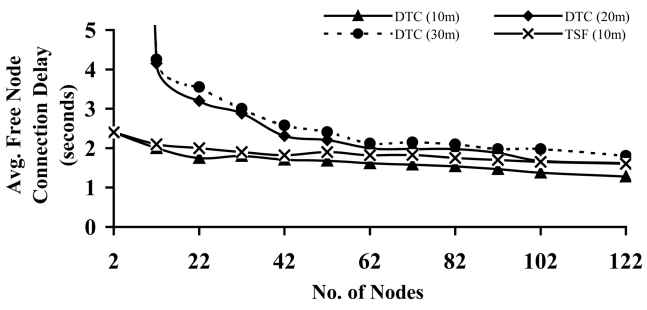


Fig. 11. Average free node connection delay.

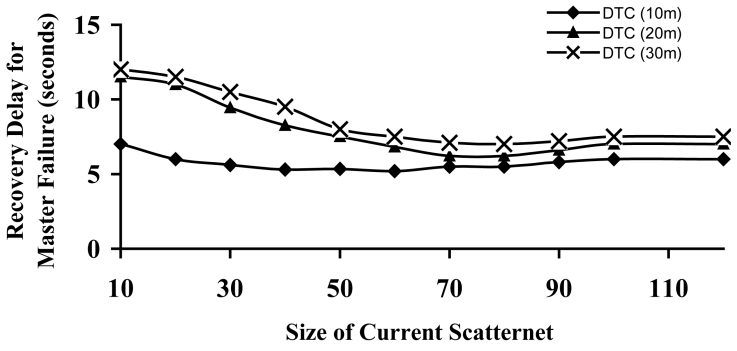


Fig. 12. Recovery delay for master failure.

connection delay for DTC (20m) and DTC (30m) is high. However, as the node density increases, the node connection delay decreases. It is also observed that the performance of DTC and TSF are comparable when the nodes are within radio range of each other. In DTC, only the nonbridge slaves and the master go for discovery. Initially, with an increase in the number of nodes, the number of piconets increases. This, in turn, increases the number of slaves performing discovery, resulting in lower isolated node discovery time. However, when the number of nodes is very high, then many of the slaves are utilized in forming bridges and, thus, the number of slaves left for discovery does not increase any further. As a result, the free node connection delay becomes almost constant.

The recovery delay caused by a single master failure is plotted in Fig. 12 for DTC only. Here, recovery delay means the average time taken by all nodes of that piconet in again becoming part of a full scatternet. Recovery delay depends on the existing number of piconets. If there are a larger number of piconets, then the dangling nodes can quickly attach to other piconets. The average number of nodes in each piconet depends on the particular configuration achieved. It is seen that, for a node size of 10, the delay is high. For DTC operating on larger areas, initial performance is low. However, as the number of nodes goes up, they become comparable with DTC (10m). While a larger number of piconets increases the chances of latching with more slaves participating in discovery, there is a balancing effect caused by the unavailability of nonbridge slaves. This observation is similar to the "healing partition delay" of the TSF algorithm. However, we cannot compare these two metrics since TSF forms a tree topology with a single node failure partitioning the whole network. On the other hand, in our approach, we obtain a mesh topology. Here, a single node failure does not necessarily partition the network. If a slave fails, the master simply removes that slave from its piconet without affecting the rest of the network, as explained in Fig. 6. If a bridge node is shut down, then there is link failure between only the piconets sharing it. It also does not necessarily partition the network since there are other indirect links connecting the two, resulting in increased hop counts for interpiconet packets. In case of a master failure, once again, there is no partitioning as in TSF. This only causes some of the slaves to become free nodes as explained in Fig. 7. Bridges of that master become nonbridge slaves of the other master as provisioned in Fig. 8.

We next compare the characteristics of the scatternets formed by the proposed algorithm and other approaches. The metrics studied are:

1. Number of piconets formed with size of network.
2. Average piconet size with size of network.
3. Standard deviation of load of the piconets representing how balanced the piconets are. Here, balanced means all the piconets have almost the same number of slaves.
4. Average path length between any two nodes of the network. Path length is a measure of routing delay too.
5. Maximum path length between any two nodes of the network.

While comparing the performance, we consider the optimum parameter values for TSF as suggested by its authors. Figs. 13a and 13b show the variation of the number of piconets formed with the number of Bluetooth nodes. In

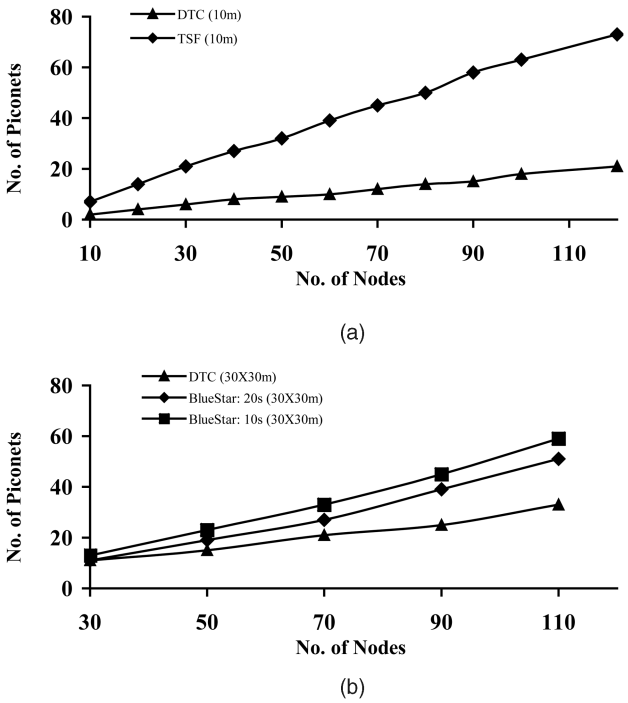
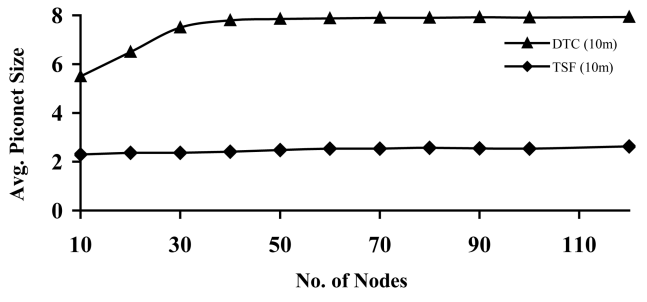
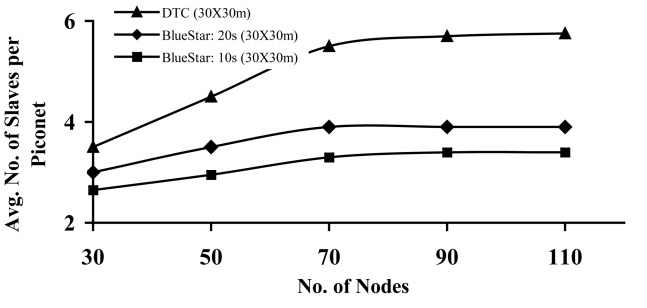


Fig. 13. Number of piconets versus number of nodes. (a) DTC versus TSF. (b) DTC versus Bluestar.



(a)



(b)

Fig. 14. (a) Average piconet size versus number of nodes. (b) Average number of slaves per piconet versus number of nodes.

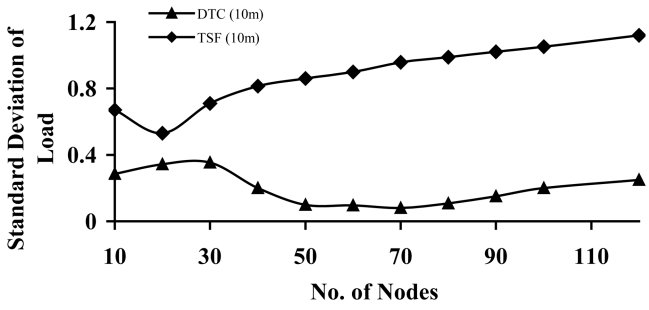


Fig. 15. Standard deviation of load versus number of nodes.

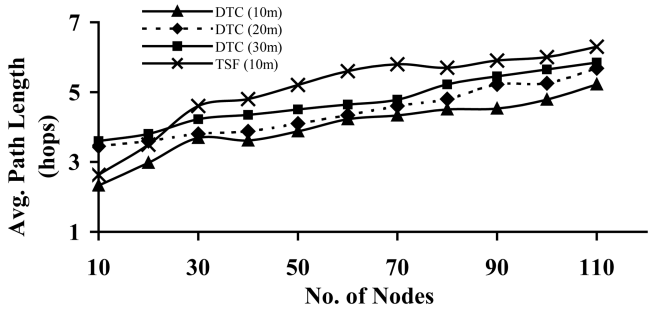


Fig. 16. Average path length versus number of nodes.

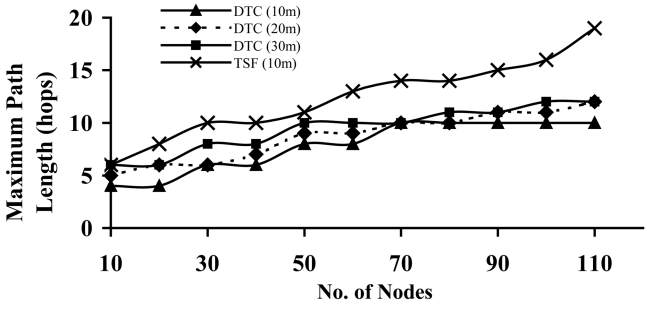


Fig. 17. Maximum path length versus number of nodes.

Fig. 13a, we compare the performance of DTC and TSF both operating in a scenario where the Bluetooth nodes are distributed over an area of diameter 10m so that the nodes are within radio range of each other.

From the figure, it is seen that, in DTC, the number of piconets varies from two to 24 as the number of nodes varies from 10 to 120. On the other hand, in TSF, the number of piconets varies, starting from seven to finally 73. Thus, DTC outperforms TSF by a large margin. A large number of piconets leads to a higher interpiconet packet transfer delay. In Fig. 13b, we compare the performance of DTC with Bluestar, proposed in [16]. For Bluestar, we consider phase II and III of their algorithm with two different values of the parameter t_{disc}, namely, 10s and 20s. It is seen that the number of piconets in DTC is fewer, compared with Bluestar.

Figs. 14a and 14b show the variation in size of the piconet with the number of nodes in the two algorithms. Fig. 14a shows the average piconet size versus the number of nodes. Here, average piconet size means the average number of nodes in a piconet including the master and all its slaves. In DTC(10m), the average piconet size increases with the number of nodes till 30 and then remains almost constant at 7.5 to 8. This implies that most of the piconets formed by our approach attain the full size of 8 (one master + seven slaves). Only a few piconets are formed with a fewer number of slaves. On the other hand, in case of TSF, the average piconet size remains between 2.5 to 3. This means that most of piconets have just two devices—one master and one slave. This structure is inefficient compared to ours since a lot of time is wasted in interpiconet routing, which also involves bridge switching. Any inefficient bridge-scheduling algorithm will make a node wait too long, thus causing delay and possible buffer overflow. In Fig. 14b, we compare the variation of the average number of slaves per piconet with the number of nodes for DTC and Bluestar. It is seen that the average number of slaves per piconet is higher in DTC.

In Fig. 15, we show the effect of the topology construction algorithm on load balancing. The standard deviation of the number of nodes in each piconet is plotted against the number of nodes in the scatternet. After an initial rise, the standard deviation falls sharply in our approach. This implies that almost all the piconets have a similar number of nodes.

The average path length in the number of hops between any two nodes is shown in Fig. 16. The path length between any two slaves in the same piconet is considered to be 2 since a packet has to move from the originating slave to the master (first hop) and then from the master to the destination slave (second hop). Similarly, the path length between two slaves in neighboring piconets (piconets sharing a bridge) is 4 (slave-master-bridge-master-slave). It is seen that DTC consistently outperforms TSF with a difference of approximately 1.5 hops ($\sim$ 25%).

In Fig. 17, we show the maximum path length in the scatternets formed by the two algorithms. In our approach, the maximum path length initially increases from 4 to 10 for about 70 nodes and then remains almost constant even for 120 nodes. On the other hand, the maximum path length for

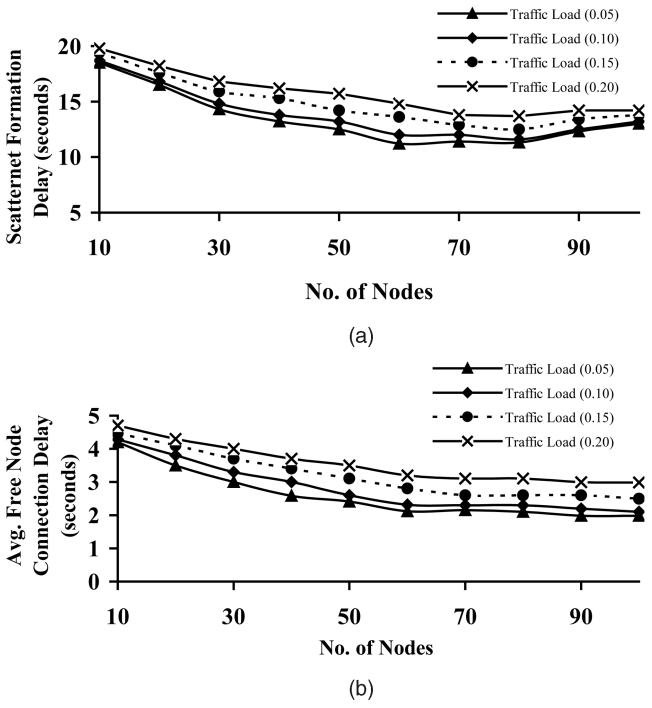


Fig. 18. (a) Scatternet formation delay. (b) Average free node connection delay versus number of nodes under different traffic conditions.

TSF keeps on increasing, reaching a value of 18 for the same 120 nodes. It can be shown that the worst-case routing delay is proportional to the maximum path length. Hence, in terms of routing delay also, the proposed algorithm performs better than TSF.

Finally, we study the effect of traffic load on scatternet formation delay and average free node connection delay in Figs. 18a and 18b. For this study, we consider the nodes to be uniformly distributed within a square area of size (30m $\times$ 30m). The traffic load shown in the figures is the average packet arrival rate at each node in terms of the length of packets (in units of time slots) generated for each time slot unit. It is seen in Fig. 18a that there is about a 3-4 second difference in scatternet formation delay between traffic loads of 0.05 and 0.2. On the other hand, the difference in the free node connection delay between the two extreme load levels is about 2 seconds. Both the delays decrease with initial increase in the number of nodes and then become almost constant similar to Figs. 9 and 11, respectively.

6 Conclusions

We have suggested a novel protocol for topology construction in Bluetooth that is simple and uses only localized information. The protocol works in an unsupervised manner in the context of a priority-based polling scheme for both intrapiconet and interpiconet packets. Bluetooth nodes are treated as symmetric devices and are capable of running identical routines. The complete protocol consists of a set of simple routines which handle connection between isolated nodes, between an existing piconet and an isolated node, as well as between two existing piconets. If the number of nodes to be connected is less than eight and the nodes are within radio range of the master, a single piconet is formed with one of the nodes acting as master and the rest acting as slaves. If, however, the total number of nodes exceeds eight, a scatternet is formed or an existing scatternet is extended by forming bridge nodes. In situations where all the nodes are not within radio range of each other, multihop paths are formed to extend the piconet. The routines detect situations in which a master, slave, or a bridge node is shut down so that local topology reconstruction can be done without affecting the entire network.

Detailed simulation studies reveal that the proposed protocol has superior performance compared to protocols with similar features. Another important advantage of the proposed work is that it considers the finite amount of time that is available for the masters and nonbridge slaves to perform node discovery. We would like to perform a queuing theoretic analysis of our work to show that the queues are indeed stable under the conditions in which polling is done and scatternet protocol is executed.

References

[1] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli, "Comparative Performance Evaluation of Scatternet Formation Protocols for Networks of Bluetooth Devices," *Wireless Networks,* vol. 10, no. 2, pp. 197-213, 2004.

[2] S. Basagni, R. Bruno, and C. Petrioli, "Device Discovery in Bluetooth Networks: A Scatternet Perspective," *Proc. IFIP-TC6 Networking Conf.,* pp. 1087-1092, 2002.

[3] *Bluetooth Core Specifications,* http://www.bluetooth.org/, 26 May 2006.

[4] R. Bruno, M. Conti, and E. Gregori, "Wireless Access to Internet via Bluetooth: Performance Evaluation of the EDC Scheduling Algorithm," *Proc. First Workshop Mobile Internet,* 2001

[5] M. Capone, S. Gefia, and R. Kapoor, "Efficient Polling Schemes for Bluetooth Picocells," *Proc. IEEE Int'l Conf. Comm.,* pp. 1990-1994, 2001.

[6] C.F. Chiasserini, M.A. Marsan, E. Baralis, and P. Garza, "Towards Feasible Topology Formation Algorithms for Bluetooth-Based WPANs," *Proc. 36th Ann. Hawaii Int'l Conf. System Sciences,* pp. 313-318, 2003.

[7] F. Cuomo, G. Di Bacco, and T. Melodia, "SHAPER: A Self-Healing Algorithm Producing Multi-Hop Bluetooth Scatternets," *Proc. IEEE Global Telecomm. Conf.,* pp. 236-240, 2003.

[8] F. Cuomo, T. Melodia, and I.F. Akyildiz, "Distributed Self-Healing and Variable Topology Optimization Algorithms for QoS Provisioning in Scatternets," *IEEE J. Selected Areas in Comm.,* vol. 22, no. 7, pp. 1220-1226, 2004.

[9] J. Haartsen, "Bluetooth—The Universal Radio Interface for Ad-Hoc Wireless Connectivity," *Ericsson Rev.,* vol. 3, pp. 110-117, 1998.

[10] M. Kalia, D. Bansal, and R. Shorey, "Data Scheduling and SAR for Bluetooth MAC," *Proc. IEEE Vehicular Technology Conf.,* pp. 716-720, 2000.

[11] C. Law, A.K. Mehta, and K.-Y. Siu, "A New Bluetooth Scatternet Formation Protocol," *ACM/Kluwer J. Mobile Networks and Applications,* vol. 8, no. 5, pp. 485-498, 2003.

[12] X. Li, I. Stojmenovic, and Y. Wang, "Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Scatternet Formation," *IEEE Trans. Parallel and Distributed Systems,* vol. 15, no. 4, pp. 350-361, 2004.

[13] Y. Liu, M.J. Lee, and T.N. Saadawi, "A Bluetooth Scatternet-Route Structure for Multihop Ad Hoc Networks," *IEEE J. Selected Areas in Comm.,* vol. 21, no. 2, pp. 229-239, 2003.

[14] M.A. Marsan, C.F. Chiasserini, A. Nucci, G. Carello, and L. De Giovanni, "Optimizing the Topology of Bluetooth Wireless Personal Area Networks," *Proc. IEEE Int'l Conf. Computer Comm. (INFOCOM),* 2002

[15] B.A. Miller and C. Bisdikian, *Bluetooth Revealed: The Insider's Guide to an Open Specification for Global Wireless Communications.* Prentice Hall, 2000.

[16] C. Petrioli, S. Basagni, and I. Chlamtac, "Configuring BlueStars: Multihop Scatternet Formation for Bluetooth Networks," *IEEE Trans. Computers,* vol. 52, no. 6, pp. 779-790, June 2003.

[17] R. Roy, M. Kumar, N.K. Sharma, and S. Sural, "P[3] - A Power-Aware Polling Scheme with Priority for Bluetooth," *Proc. Int'l Conf. Parallel Processing (ICPP) Workshops,* pp. 480-487, 2004.
[18] R. Roy, M. Kumar, N.K. Sharma, and S. Sural, "A Self-Organizing Protocol for Bluetooth Scatternet Formation," *European Trans. Tele-Comm.,* vol. 16, no. 5, pp. 483-493, 2005.
[19] R. Roy, M. Kumar, N.K. Sharma, and S. Sural, "A Power-Aware Master and Bridge Scheduling Scheme for Bluetooth Scatternets," *Int'l J. Ad-Hoc and Ubiquitous Computing,* vol. 2, nos. 1/2, pp. 119-132, 2007.
[20] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks," *Proc. IEEE Int'l Conf. Computer Comm.,* pp. 1577-1586, 2001.
[21] I. Stojmenovic, "Dominating Set Based Bluetooth Scatternet Formation with Localized Maintenance," *Proc. IEEE Int'l Parallel and Distributed Processing Symp.,* pp. 148-155, 2002.
[22] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "An Efficient Scatternet Formation Algorithm for Dynamic Environment," *Proc. Int'l Assoc. of Science and Technology for Development Conf. Comm. and Computer Networks,* 2002.
[23] R.M. Whitaker, L. Hodge, and I. Chlamtac, "Bluetooth Scatternet Formation: A Survey," *Ad Hoc Networks,* vol. 3, no. 4, pp. 403-450, 2004.
[24] G. Záruba, S. Basagni, and I. Chlamtac, "BlueTrees - Scatternet Formation to Enable Bluetooth-Based Personal Area Networks," *Proc. IEEE Int'l Conf. Comm.,* pp. 273-277, 2001.
[25] H. Zhu, G. Cao, G. Kesidis, and C. Das, "An Adaptive Power-Conserving Service Discipline for Bluetooth," *Proc. IEEE Int'l Conf. Comm.,* pp. 303-307, 2002.

Rajarshi Roy received the bachelor's degree in electronics and telecommunication from Jadavpur University, Kolkata, India, with first class honors in 1992 and the MSc (Engg) degree in electrical communication engineering from the Indian institute of Science, Bangalore, Karnataka, India, under the supervision of Dr. Utpal Mukherji in 1995. He received the PhD degree in electrical engineering with a specialization in networking from Polytechnic University, Brooklyn, New York, in 2001 under the supervision of Dr. Shivendra Singh Panwar. Since July 2002, he has been employed as an assistant professor in the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, West Midnapur, West Bengal, India. Prior to this, he worked for the Performance Analysis Group of the Advanced Development Department of Comverse, Wakefield, Massachusetts, as a performance analyst software engineer and as a systems engineer for Lucent Technologies' India Development Center in Bangalore, India. He also worked in the Applied Statistics Unit of the Indian Statistical Institute, Kolkata, India, as a visiting scientist, was an academic visitor with the Helsinki University of Technology, Espoo, Helsinki, Finland during Fall 2004, and has worked for the High Speed Networking Research Department of Bell Laboratories, Lucent Technologies, Holmdel, New Jersey, as an intern during Summer 1997. He has obtained corporate training from Opnet.com in Washington, D.C. and Empirix.com in Wilmington, Massachusetts. He also attended summer school in astronomy and astrophysics at the Inter University Center for Astronomy and Astrophysics, Pune University Campus and at the National Center for Radio Astronomy, Pune, Maharashtra, India. In his present position as an assistant professor, he has served as a supervisor for three doctoral, 12 master's, and nine bachelor's theses, written various journal and conference papers, served as reviewer of various IEEE transactions and conferences, and supervised project works funded by various agencies of the Human Resource Development Department (MHRD) of the Government of India. Previously, he worked on DQDB, ATM, and TCP/IP networks, performance analysis, and control of queues. His current research interest includes sensor networks, underwater acoustic sensor networks, wireless networks and wireless ad hoc networks, MIMO systems, optical networks, and complex networks. He is especially interested in control, optimization, and algorithmic issues involved in the optimal design of engineering systems in the above-mentioned domain, performance analysis, and distributed computing.

Mukesh Kumar received the BTech degree in computer science and engineering from the Institute of Technology, Banaras Hindu University, India, in 2005. Since then, he has been working at Sapient Corporation Pvt. Limited, Bangalore, India. His research interests include ad hoc networks, Bluetooth and sensor networks, wireless communication, mobile computing, network security, and distributed systems.

Navin K. Sharma received the BTech degree from the Indian Institute of Technology, Kharagpur, India, in 2004. Since then, he has been working as a software engineer at Computer Associates, India Technology Center, Hyderabad, India. His research interests include ad hoc networks, Bluetooth and sensor networks, wireless communication, mobile computing, and distributed systems.

Shamik Sural (M '93) received the BE degree in electronics and telecommunication engineering from Jadavpur University, Calcutta, India, in 1990, the ME degree in electrical communication engineering from the Indian Institute of Science, Bangalore, in 1992, and the PhD degree from Jadavpur University in 2000. He has worked in a number of companies belonging to the IT industry, both in India as well as the United States, in various capacities. Since 2002, he has been an assistant professor at the School of Information Technology, Indian Institute of Technology, Kharagpur, India. Dr. Sural has served on the program committee and executive committee of a number of international conferences, including the International Database Engineering and Applications Symposium, the IEEE Conference on Fuzzy Systems, the Annual Computer Security Applications Conference, the Asian Mobile Computing Conference, the International Workshop on Distributed Computing, and others. His research work has been funded by the Ministry of Communication and Information Technology, Department of Science and Technology, Government of India, and the National Semiconductors Corporations. He has been elected as the chairman of the IEEE Kharagpur Section for 2006. He has published more than 50 papers in reputed journals and conferences. His research interests include Bluetooth and ad hoc networks, multimedia database systems, database security, and data mining. He is a member of the IEEE and the IEEE Computer Society.