Special Issue on Self-organisation in Mobile Networking

A self-organising protocol for Bluetooth scatternet formation

Rajarshi Roy¹, Mukesh Kumar², Navin K. Sharma³ and Shamik Sural⁴*

¹Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721302, India
²Department of Computer Science and Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India
³Computer Associates India Pvt. Ltd., Auriga Block, 1st Floor, Left Wing, Vanenburg IT Park, Hi Tech City, Madhapur, Hyderabad 500081, India
⁴School of Information Technology, Indian Institute of Technology, Kharagpur 721302, India

SUMMARY

We propose a self-organising protocol for Bluetooth topology construction in which scatternet structures are maintained even in the presence of dynamic perturbation of network configuration. The protocol does not require the presence of a central topology construction or re-construction process. Instead, the nodes can self-organise to create a stable configuration both after start-up as well as after intermediate changes due to the arrival of new nodes or shutting down of existing nodes. The proposed algorithm uses a bottom-up approach, which is built into the regular operations of the Bluetooth devices. Isolated nodes as well as masters and slaves of existing piconets can discover other nodes, thus extending the scatternet. Copyright © 2005 AEIT.

1. INTRODUCTION

For wireless communication in the unlicensed ISM band (2.4 GHz), Bluetooth (BT) is one of the most promising technologies for ad hoc networks [1, 2]. A newly powered-up Bluetooth node must discover other nodes in an existing network by synchronising the frequency hopping patterns before it is able to exchange packets. The synchronised Bluetooth devices form an ad hoc network called a ‘piconet’ in which one of the devices assumes the role of ‘master’ and the rest become ‘slaves’. There can be one master and up to seven active slaves in any piconet. If there are more than eight active Bluetooth devices, they form a ‘scatternet’, which is a collection of piconets connected by ‘bridge’ nodes. All the devices in a piconet share a common radio frequency channel using a slotted time division duplex (TDD) protocol. Packet transfer always takes place through the master. The first step in the successful operation of a Bluetooth network is, thus, the formation of a network topology—a configuration in which each node in a piconet is synchronised with its master and multiple piconets are connected by bridge nodes.

Bluetooth nodes are symmetric devices in which any node can act either as a master or as a slave. However, these roles are assigned only after two nodes are able to synchronise their frequency hopping patterns through an asymmetric process as mentioned in the baseband specifications. These standards specify that one of the nodes is designated as a ‘sender’ and the other node is specified as a ‘receiver’. While the baseband specifications talk about the link establishment procedure between a sender and a receiver resulting in the roles of master and slave, it is not explicit in describing how the symmetric nodes can take up such asymmetric roles after power-up without manual intervention. In the absence of a meta-process or central management to direct a particular node to act as a sender or as a receiver, we need to construct a self-organising protocol that can not only connect two isolated nodes, but also make the topology dynamic.

In the next section, we review related work on scatternet formation in Bluetooth and then present our protocol in Section 3. This is followed by discussions on the algorithm in Section 4 and simulation results in Section 5. We conclude the paper in Section 6.

*Correspondence to: Shamik Sural, School of Information Technology, Indian Institute of Technology, Kharagpur 721302, India.
E-mail: shamik@cse.iitkgp.ernet.in

Copyright © 2005 AEIT Accepted 13 December 2004
2. RELATED WORK ON BLUETOOTH SCATTERNET FORMATION

During the last few years, a number of scatternet formation algorithms have been suggested. One of the first such algorithm was proposed by Salonidis et al. [3]. They introduced a scatternet formation algorithm called Bluetooth topology construction protocol (BTCP). It has three phases: (i) leader election—in which one node acquires complete knowledge of all the nodes, (ii) determination of masters of individual piconets by the leader and (iii) formation of scatternet through bridge nodes. Miklos et al. [4] suggested the use of several heuristics to generate scatternets with a few desirable properties. They evaluated the performance of these scatternets with different characteristics through simulations. Záruba et al. [5] suggested a scatternet configuration called Bluetrees. In this configuration, they assume that node mobility is very low and that each node is aware of the number and identities of each of its neighbours. In situations with higher mobility, this assumption may not always be valid. Petrioli et al. [6] proposed a three-phase scatternet formation protocol for the construction of multi-hop scatternets called Bluestars. In this approach, the first phase is topology discovery, the second phase is Bluestar piconet and the third phase is selection of gateway devices forming BlueConstellations. Tan et al. [7] proposed a distributed tree scatternet formation (TSF) protocol. Their method is designed to work in dynamic environments in which nodes can arrive and leave at arbitrary time instants. The simulation results indicate relatively short latency during scatternet formation. However, TSF does not minimise the number of piconets. Each master usually has fewer than three slaves.

Law et al. [8] presented a Bluetooth scatternet formation protocol in which they try to optimise some of the quality measures, namely, the number of piconets, maximum degree of the devices and the network diameter. Their protocol has \( O(\log n) \) time complexity and \( O(n) \) message complexity. Yun et al. [9] also proposed a three-phase protocol for the construction of their version of Bluestar. The stages in their protocol are neighbour discovery, neighbour grouping and role assignment. Basagni et al. [10] compare the scatternet formation protocols for networks of Bluetooth devices—Bluetree, Bluestar, Bluenet [11] and the protocol proposed in Reference [12]. They have analysed the performance of these protocols in terms of the overhead, quality of the generated scatternet, time needed for scatternet formations, number of piconets formed, number of slaves per piconet and the scatternet route lengths. They have shown that Bluestar is the fastest protocol for scatternet formation and also generates scatternets with lower number of piconets and average route length.

Cuomo et al. [13] have proposed a distributed algorithm that generates a tree-like structure, which can dynamically adapt to the mobility of devices. Their approach has self-healing property since minor changes in the topology may be absorbed in the remaining network. QoS provisioning in scatternets has been considered by Cuomo et al. [14]. This approach builds on the idea of SHAPER. After generating a tree-shaped scatternet, a procedure is defined that produces a meshed topology by applying a distributed scatternet optimisation algorithm. Chiasserini et al. [15] suggested a distributed algorithm for topology formation that considers insertion and deletion of a node to/from a Bluetooth personal area network. While this can be implemented with moderate complexity, they may generate sub-optimal topologies. Guerin et al. [16] investigated the problem from an algorithmic perspective. They conclude that the minimum weight spanning tree algorithm is the only one that is guaranteed to always satisfy the usual Bluetooth degree constraints in a two-dimensional scenario. Stojmenovic [17] proposed a dominating set-based Bluetooth scatternet formation protocol.

It may be noted that none of the protocols except those proposed in References [7, 13, 14] is self-organising. Some of these have a leader election process for initial topology construction [3]. They assume that all the nodes are started up at the same time [3, 10, 15]. The situation in which new nodes are powered up from time to time or existing nodes are shut down remains unaddressed. The problem of topology ‘reconstruction’ is often deferred as a future extension [5, 6, 8, 9, 11, 12,]. Most importantly, the topology construction problem is treated as a stand-alone problem and not as an outcome of specified properties of Bluetooth nodes [3, 5–9, 11, 12, 15–17].

While forming a dynamic scatternet, one should ideally take into account various parameters like time and message complexity, number of piconets, network diameter, number of bridges etc., and try to optimise the overall performance of the scatternet. At the same time, it must be noted that the Bluetooth nodes are usually portable, power-constrained devices. As a result, rigorous optimisation algorithms like Reference [18] may not possibly be run on such nodes in practical applications whenever a topology is disturbed due to shutting down of nodes or joining of new nodes. Instead, we propose a simple and implementable self-organising protocol in which topology formation specific properties are integrated with the normal operations of Bluetooth nodes and the topology attains the ideal configuration over a period of time.
3. SELF-ORGANISING TOPOLOGY CONSTRUCTION PROTOCOL

The primary goal of the self-organising protocol is two-fold: (i) an isolated node should be able to connect to another isolated node or to an existing piconet within its radio proximity and (ii) an existing piconet should be able to discover another existing piconet, thus forming or extending a scatternet. All the steps required to achieve these two goals should be ingrained in the basic communication protocol of the Bluetooth nodes. 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. Evidently, if the number of nodes is high, such a fully connected scatternet cannot be achieved. However, this situation is quite rare in practice and even if it happens, our protocol attempts to minimise the number of hops to connect one piconet to another. A balanced network is one in which all the piconets have similar number of nodes so that none of the master nodes is unduly loaded. For a balanced network, average inter-slave data transmission delay is less since the slave nodes are polled by their respective masters at equal intervals of time. In this paper, the term optimum denotes a specific scatternet configuration in which all the piconets are either fully connected or connected through minimum number of hops. It also has the property that the scatternet is balanced. However, every time an existing network is changed, we do not try to reduce the number of piconets immediately in order to avoid high message complexity and device power wastage.

3.1. Protocol overview

In the topology construction protocol, we assume that any node becomes ON at any instant of time. Here ON means that either a new node has been powered up or an existing node has come within radio proximity of another piconet. There are four roles that an ON node can take up, namely, Isolated (I), Master (M), Slave (S) or Bridge (B). Any node can also go to the OFF state at any instant of time where OFF means that a node is not able to communicate with its master either because it is powered-off or it has moved out of the radio range of the master. Due to mobility, a node that becomes OFF in one piconet can now become ON with respect to another piconet. This reflects the real world situation where a small battery powered Bluetooth node is powered-up, joins other nodes forming an ad hoc network and then shuts down after exchanging voice/data packets for a limited period of time or moves to a new location.

The Bluetooth specifications mention that if a node performs Inquiry and latches with another node performing Inquiry-scan, the former goes into the Page state and the latter goes into the Page-scan state for exchanging information. In this paper, we do not mention the Page and Page-scan states explicitly but assume that they always follow Inquiry and Inquiry-scan states as per the specifications. During the Page and Page-scan states, symmetric information exchange occurs between the nodes. In the scatternet formed using the proposed algorithm, two piconets are connected through a single bridge. Only a slave is used as a bridge node, which can be shared by at most two piconets.

3.2. Protocol details

We propose a protocol suite that consists of four basic protocols—one for each role of the Bluetooth node. These are named as isolated node protocol (INP), master communication protocol (MCP), slave communication protocol (SCP) and bridge communication protocol (BCP). The INP is used to connect an isolated node to another isolated node or to an existing piconet as shown in Figure 1. If two isolated nodes discover each other, they first exchange device information and then form a piconet by calling the piconet formation and modification routine (PFMR) of Figure 2. In this situation, Case 1 of PFMR is executed. In PFMR, the node with higher memory and power assumes the role of master and the other node assumes the role of slave. However, if the isolated node latches with a master, there are two possibilities. If the existing piconet has less than seven slaves, it can accommodate the new node by calling PFMR. In this situation, Case 2 of PFMR is executed. Else, two piconets are formed connected by a bridge node using the scatternet formation and modification routine (SFMR) of Figure 3. In this situation, Case 1 of SFMR is executed. The new node is assigned the role of Master or Slave depending on its memory and available power and that of the master of the piconet in which the node joins. It should be noted that if the isolated node latches with a slave of an existing piconet, the latter provides node discovery information to its own master. The master then completes the topology re-construction process.

After a node joins a piconet, it participates in voice/data packet exchange. If it has been assigned the role of Master, the node executes MCP of Figure 4. In the MCP, a master may swap its role with a slave if its power falls below a
certain threshold. If a master detects that one of its slaves is not responding to polling packets for a long time, the master assumes that the slave has become OFF. The master then removes this slave from its piconet and notifies all the remaining slaves. After some regular intervals, a master signals its most idle non-bridge slave to participate in node/piconet discovery. When a slave receives this signal, it goes to Inquiry-scan state. If the slave then latches with another node, it notifies the master. The master then calls PFMR or SFMR depending on the existing number of nodes to extend the scatternet. The master may also participate in node/piconet discovery by going to the Inquiry or the Inquiry-scan state if traffic is low in its piconet. During discovery, if the master detects an isolated node, it calls PFMR or SFMR depending on the existing number of nodes to extend the scatternet. Either Case 2 of PFMR or

![Diagram of isolated node protocol (INP).](image-url)
Case 1 of SFMR is executed. However, if the master detects another master or the slave of another piconet, PFMR is called if the two piconets can be merged together. In this situation, Case 3 of PFMR is executed. On the other hand, if the total number of slaves in the two piconets exceeds eight, SFMR is called to connect the two piconets by bridge nodes and the piconets are balanced. In this situation, Case 2 of SFMR is executed.

In Slave role, a node executes the SCP of Figure 5 and takes part in the usual data/voice exchange. As mentioned above, it can go to Inquiry-scan for node discovery. During node discovery, the slave can discover either an isolated node or a master node. It should be noted that a slave cannot discover another slave as both of them perform node discovery in the Inquiry-scan state. If a node is discovered, the master is informed and the slave goes back to its usual communication mode. The master completes the scatternet extension process as mentioned in the MCP. If the master does not poll a slave for sometime, the slave assumes that the master has become OFF. The slave then considers itself isolated and starts Inquiry/Inquiry-scan using the INP.

Case 2 - An isolated node joins an existing piconet with less than 7 slaves

If (getRole()==1 and getRole()==M)
    if (F(m,p)=F(m,p))
        if i becomes the new master
            i runs MCP
            Send P(i) to all slaves
            else
                Same as above with roles of i and j reversed.

Case 3 - Two piconets with a total of less than 7 slaves are merged

If (getRole()==M and getRole()==M)
    if (F(m,p)=F(m,p))
        Merge P(i) and P(j)
        Send message to bridge between P(i) and P(j)
        else
            Same as above with roles of i and j reversed.

In the Bridge role, a node executes the BCP of Figure 6 and takes part in usual data/voice exchange across the two piconets joined by it. If a Bridge node does not receive a polling packet from one of its masters while executing the BCP, it assumes that particular master to be OFF. The Bridge then becomes a non-bridge slave of its other master and starts SCP.

4. DISCUSSIONS ON THE PROTOCOL

In the proposed protocol suite, there is no leader election process so that we eliminate the fault situation which may result due to leader failure. Also, master and slave failure detection is built in the protocol itself. There are a few important considerations for distributing nodes while running the SFMR. Since we try to balance the scatternet and at the same time avoid high message complexity, only

Copyright Ⓒ 2005 AEIT

non-bridge slaves are transferred during re-arrangement. Out of the non-bridge slaves, at most half of the available slaves are assigned to the new piconet. While assigning a bridge node, the slave with highest available memory and power is considered. If the master of a piconet with seven slaves discovers the master of another piconet also with seven slaves and they are still disconnected, then the protocol does not make an attempt to connect all the nodes.

Figure 4. Master communication protocol (MCP).

Figure 5. Slave communication protocol (SCP).
by dividing into three piconets. The reason is, this process has high message complexity since all the piconets that share a bridge with these two piconets, will have to be updated about the changed configuration. However, this is quite an uncommon situation. Due to ephemeral nature of the Bluetooth nodes, one of them may free a slave in future at which time the two piconets can be joined.

The node discovery process in the proposed approach is handled both by the master and the slaves. If a master detects a new node, it exchanges information for learning the details about the node like power and memory capacity. However, when a slave of an existing piconet detects a new node, it informs the master about node discovery. The master then communicates to the new node for extending the piconet. This step assumes that the new node is not only in radio proximity of the slave but also of the master. Another important property of the protocol is that, we introduce not only the concept of isolated node discovery but also of piconet discovery. This enables dynamic growth of the scatternet.

The communication overhead of the nodes is low in the protocol since the piconet/scatternet formation and modification goes along with data/voice exchange among other nodes. At any instant of time, communication of only that piconet is blocked whose master is involved in scatternet formation and modification. The rest of the scatternet keeps the normal communication process running.

The effect of mobility of the Bluetooth nodes is handled in the Master, the Slave and the Bridge communication protocols. If a particular slave goes out of the radio proximity of its master, the loss of communication is detected both by the master (MCP) as well as by the slave itself (SCP). The master then removes the slave from its piconet by releasing its Bluetooth node address. The slave, on the other hand, detects that it has lost contact with its master and goes in the Inquiry or the Inquiry-scan state. The effect of mobility of the master is handled in a similar manner. All its slaves and bridges can detect the OFF state of the master caused by mobility or by a power down.

The main features of the proposed protocol are explained through a number of stages in Figure 7. In the figure, master nodes are labelled as ‘m’ and bridge nodes as ‘b’. All the remaining unmarked nodes are non-bridge slaves. First, the master of an existing piconet P-1 discovers an isolated node. The node joins the piconet P-1 through PFMR. Then the master of P-1 latches with the master of another piconet P-2. The total number of slaves is greater than seven. Hence they form a scatternet using SFMR and start sharing a Bridge node. It should be noted that during the scatternet formation, one non-bridge slave is transferred from P-1 to P-2 for load balancing. Also, it is the piconet P-1 that contributes one of its slaves as bridge node for sharing with P-2. In the next stage, another piconet P-3 joins the scatternet by sharing a bridge, again through SFMR. At this stage the three piconets P-1, P-2 and P-3 are connected forming a scatternet. However, they are not fully connected. While piconets P-1 and P-2 share a
bridge and piconets P-1 and P-3 share another, there is no direct bridge sharing between piconets P-2 and P-3. This temporarily results in higher number of hops for packet transfer between piconets P-2 and P-3. In the next stage, as master of piconet P-2 discovers master of piconet P-3, they start sharing a bridge so that the scatternet is fully connected with only one hop required for a packet to reach from any slave to another in the scatternet. Finally, a piconet P-4 with one master and two slaves discover piconet P-2 and gets merged with it through PFMR since the total number of slaves is seven.

5. SIMULATION RESULTS

We have developed a simulator for testing Bluetooth topology formation and scheduling protocols. The test bed emulates the Bluetooth devices and the communication process between master and slaves using TDD channel and frequency hopping sequence [19]. The simulation tool allows creation and destruction of Bluetooth nodes at any point of time by the user through command line interface. The standard features of Bluetooth networks like Inquiry/Inquiry-scan, Page/Page-scan and Power saving modes have been implemented following the Bluetooth specifications [1, 2].

We first verified the correctness of the proposed protocol by simulating all possible combinations of node and piconet discovery as explained in the last section. For each combination, different initial conditions were used during simulation. A connected scatternet was formed under every circumstance. In the simulations, the function $F(m,p)$ is considered simply as the power of a device since many of the Bluetooth devices have similar memory capacity.
We next measure the performance of the protocol through extensive simulation. The protocol performance is observed under three broad categories: (i) How closely a scatternet formed by the protocol can track an ideal scatternet configuration in terms of connectivity and node distribution, (ii) routing delay in a scatternet formed by the protocol and (iii) isolated node connection delay in the protocol. In Figure 8(a), we show how the average number of slaves per piconet varies with the number of nodes. Both the ideal average and the actual average are plotted in the graph. The ideal average number is obtained when the number of piconets is minimum and they are fully connected. From the figure it is seen that the average number of slaves per piconet is the same as the ideal average number when the number of nodes is small. The reason is that, up to seven slaves, there is only one piconet both in our protocol and in the ideal case. When the number of nodes exceeds 12, our protocol tends to form higher number of piconets as shown in Figure 8(b). However, the number of piconets is either equal to the ideal number or is only one greater than the ideal. The reason is that we do not modify an existing scatternet every time new nodes are added as explained through the example of Figure 7. In the cases where number of piconets formed is same as the ideal number in Figure 8(b), the average number of slaves per piconet of Figure 8(a) also matches well with the ideal average number. Overall, it is observed that the performance of our protocol is quite close to the ideal performance.

Average inter-piconet routing delay in the number of hops is plotted in Figure 9 against the number of nodes. Here, by number of hops between two piconets we mean hop counts between the masters of those piconets. Thus, for packet transfer between two slaves of the same piconet, hop count is zero. Packet transfer between two slaves of two different piconets sharing a bridge node is two since the packet has to hop from the master of the first to the bridge node and then again hop to the second master. It is observed in Figure 9 that when the number of nodes is less than or equal to eight, routing delay is zero. Between 8 and 28, the routing delay is 2. Beyond 28 nodes, it is not possible to achieve full one-hop connectivity of piconets. Hence the average piconet routing delay slowly increases.

In Figures 10(a), (b), we plot isolated node connection delay in terms of clock time and number of transmitted packets respectively. It is seen that when the number of existing nodes is less, there is a comparatively higher value of set-up delay. This can be explained by the fact that at this stage, only a few masters alternate in Inquiry and Inquiry-scan states. Since the number of slaves is also less, not many of them can go in the Inquiry-scan state for node discovery. On the other hand, as the number of nodes increase, more masters and slaves become available which can participate in the node discovery process. Due to this, both the clock time as well as the number of packets for link set-up initially go down and then remain almost constant with increase in the number of nodes. Again, as we try to create fully connected piconets, for higher number of nodes, a comparatively larger percentage of slaves
perform the role of bridge nodes. Such nodes are not allowed to join in the node discovery process. This results in a steady state for the set-up time as well the number of set-up packet exchanges.

6. CONCLUSIONS

A self-organising Bluetooth scatternet formation protocol has been presented in this paper. This protocol addresses the need for handling dynamic situations in which Bluetooth nodes may join or leave a scatternet at any point of time. One of the goals of the protocol is to maintain full minimum hop connectivity between piconets. Further, we attempt to form a scatternet where the piconets are balanced in terms of the number of slaves. It has been shown that our proposed protocol generates balanced scatternets for small to moderate number of Bluetooth nodes. Even for larger number of nodes, this approach generates a scatternet, which is close to an optimal configuration. The scatternet self-organises and creates a network of balanced piconets by merging small piconets with larger ones if necessary when new nodes join the network.

Extensive simulation was done to measure the performance of the protocol. It is seen that the nodes can achieve the desired connectivity within a reasonable amount of time and the number of message exchanges. It is further observed that there is a slow increase in the average inter-piconet routing delay with the number of Bluetooth nodes. Thus, we achieve a scatternet configuration with a balanced number and size of piconets. At the same time, we can restrict the set-up time, message complexity and inter-piconet routing delay to acceptable limits. The protocol operates in a fully distributed bottom-up manner with no leader election process. This renders the scatternet self-organising. Computational complexity of the protocol is low since it involves evaluation of only a few branch conditions and simple arithmetic. Another strength of this approach is that the protocol operates within the defined specifications of Bluetooth devices. We would like to compare the quality of scatternet formed by our protocol with other approaches as an extension of the current work.

REFERENCES


AUTHORS’ BIOGRAPHIES

Rajarshi Roy is an assistant professor at the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, India. He has worked in the Tele-Communication industry companies like Lucent Technologies both in India and the U.S.A. His research interests are Bluetooth and ad hoc networks, performance studies and QOS.

Mukesh Kumar is a final year under-graduate student in the Department of Computer Science and Engineering at the Institute of Technology, Banaras Hindu University, India. His research interests are Bluetooth and ad hoc networks.

Navin K. Sharma is as associate software engineer at the Computer Associates India. He received his B.Tech. degree from the Indian Institute of Technology, Kharagpur, India. His research interests are software engineering, Bluetooth and ad hoc networks.

Shamik Sural is an assistant professor at the School of information Technology, Indian Institute of Technology, Kharagpur, India. He has worked in the industry on a large number of projects both in India and the U.S.A. His research interests are Bluetooth and ad hoc networks, multimedia database systems and data mining.