A Zone Routing Protocol for Bluetooth Scatternets
Rohit Kapoor, Mario Gerla
3803 H, Boelter Hall, UCLA

Abstract: Bluetooth is a low-cost, low-power technology initially intended as a replacement of cables between electronic devices. Bluetooth devices can form small network of up to 8 devices called piconets. The specification also defines networks of piconets called scatternets. Scatternets can have various uses such as for “monitoring” purposes in factories and warehouses or for ad hoc meetings. Scatternets require the use of a routing scheme to find paths in a dynamic network. Though considerable research has been done in the area of routing in ad hoc networks, the direct application of this may be inefficient to Bluetooth scatternets. Some previous work has also presented routing schemes for scatternets, but this does not present any results showing the performance of the routing scheme. In this paper, we present a routing scheme for Bluetooth scatternets which is based on the Zone Routing Protocol. We motivate the design of the routing scheme keeping in mind the specifics of the Bluetooth technology. We present simulation results for the scheme which show that the scheme gives very low overhead while keeping the route acquisition latencies low. The routing information at a node does not require a large amount of storage. In fact, a parameter in the scheme can be varied to trade-off storage information and routing overhead versus route acquisition latency.

I. INTRODUCTION

Bluetooth [1] is a low-power, low-cost, short-range technology that was developed to replace cables between electronic devices. Bluetooth devices can form small networks called “piconets” in which one node is a “master” and up to seven others are “slaves”. A piconet can be used to connect electronic devices such as mobile phones, PDAs, headsets; such a network is referred to as a PAN (Personal Area Network) [2].

Bluetooth piconets can also be interconnected to form larger networks called scatternets. Since Bluetooth devices are low power and are expected to cost as little as five dollars in the future, scatternets can find a number of uses. For example, a scatternet deployed in a warehouse can be used for inventory management or may be used to connect household electronic appliances into a network. Such environments will involve static scatternets where nodes are not mobile. Scatternets can also be used in dynamic environments, such as a conference or an ad-hoc meeting of friends, and will enable users to exchange various kinds of information, such as business cards or multimedia files. Such environments will involve frequent arrival and departure of nodes, causing changes to the scatternet structure.

Routing algorithms are needed to make scatternets functional. Though considerable research has been done in the area of routing in ad hoc networks, this mostly assumes 802.11 as the MAC technology. Since the Bluetooth MAC layer is different from 802.11, the direct application of current ad hoc routing algorithms may prove to be inefficient to Bluetooth scatternets. In fact, as we discuss later, a routing algorithm can be designed keeping in mind the specifics of the Bluetooth technology.

In this paper, we define a routing scheme for Bluetooth scatternets that is based on the Zone Routing Protocol (ZRP) [8]. We explain how the scheme takes into account the specifics of the Bluetooth MAC layer and also provide simulation results showing the performance of the scheme. Section 2 discusses the Bluetooth technology, while Section 3 discusses previously proposed routing schemes for ad hoc networks. In Section 4, we describe our routing scheme. Simulations results are presented in Section 5 and Section 6 concludes and discusses future work.

II. BLUETOOTH TECHNOLOGY

Bluetooth is based on a centralized connection-oriented approach. Bluetooth devices sharing a wireless channel form a piconet, in which one device has the role of the master and controls access to the channel, while the others are slaves. There may be up to 7 slaves in a piconet. Bluetooth uses a Time-Division Duplex (TDD) scheme to divide the channel into 625μs time slots. Each piconet is characterized by a particular fast frequency-hopping pattern; the frequency is uniquely determined by the master’s address and is followed by all the devices participating in the piconet.

III. RELATED WORK

Several ad-hoc routing protocols have been proposed in the literature [4] [5] [6] [7]. These generally fall under one of the two categories: (a) proactive or (b) reactive. In the former category of schemes, periodic advertisement of connectivity information takes place. Moreover, a change in connectivity can also lead to an immediate update. Two flavors of proactive schemes are distance vector schemes [4] and link-state based schemes [5]. Distance vector schemes such as DSDV [4] are simple and effective for small populations of mobile nodes, but suffer from the problem of slow convergence and a tendency to form routing loops. The slow convergence problem does not exist in link-state based periodic schemes [5], where a periodic flooding of link-state updates ensures that all nodes have global network topology information. In both kinds of periodic schemes, the control overhead is excessive due to the need for frequent system-wide broadcasts, particularly when mobility is high or when networks are large. In small networks, on the other hand, the control overhead is low and the route acquisition latency before transmission of the first packet is also low.

0-7803-7700-1/03/$17.00 (C) 2003 IEEE
Reactive schemes, on the other hand, invoke a route determination procedure only on demand. When a route is needed, some sort of global search procedure such as flooding is employed. Various reactive routing schemes have been proposed [6] [7]. Schemes such as DSR [6] and AODV [7] unicast the route reply packet back to the source along a path of node addresses accumulated during the route reply phase. In the case of DSR, the node addresses are stored in the header of the packet. In AODV, these paths are stored as next-hop information at each node.

The advantage of the proactive schemes is that there is little delay in determining a route. In reactive protocols, there can be significant delay in determining a route. On the other hand, the periodic dissemination of global information makes proactive schemes unsuitable for large networks. Reactive schemes scale well as networks become larger.

Work has also been done in the area of routing in Bluetooth scatternets. In [9], the authors present a Routing Vector Scheme (RVM) in which the complete path is carried in the header (i.e., source routing) and Bluetooth addresses are expressed very efficiently. Such a scheme can lead to a large packet overhead as Bluetooth scatternets get large. In [10], a self-routing scheme is presented in which Blue-tree structures are embedded into scatternets. In [11], a routing protocol that employs flooding to obtain battery levels of nodes is described. None of these papers presents results on the performance of the routing scheme (some results are presented in [11], but these deal more with power-saving issues than with routing).

IV. ROUTING SCHEME

Before describing our routing scheme, we observe the following three characteristics of Bluetooth. Firstly, Bluetooth nodes are expected to be small and resource-constrained, and thus, any routing scheme that requires large tables of information to be stored or complicated processing at the nodes cannot be viable. Thus, link-state or distance-vector approaches that keep global information are not good contenders. Secondly, sending routing queries over multiple hops could be time-consuming in Bluetooth networks due to the nature of the gateway, which divides its time between its piconets [12]. This could give very large route acquisition latencies. Thirdly, Bluetooth networks are structured such that one node (master) controls a cluster of nodes (piconet). The master node is, in a sense, the representative of its piconet. In such a structured network, a node would only need to maintain information regarding master nodes and this would provide it with information about all the other nodes (slaves) of the piconet too (if master nodes have information about all slave nodes in their piconet).

These three observations lead us to consider the use of a routing scheme based on the Zone Routing Protocol (ZRP) for Bluetooth scatternets. The ZRP is a hybrid reactive-proactive scheme in which a node limits the scope of the proactive procedure to a certain number of neighboring hops, called the routing zone. The routing zone contains a subset of all nodes of the network. A node in a Bluetooth scatternet can obtain information regarding all the nodes in a piconet by receiving routing information only from the master of the piconet. Thus, a large subset of nodes can be included in the routing zone at the overhead associated with receiving routing packets only from masters. This also means that the route acquisition latency will be small for finding routes to a larger number of nodes.

When a node falls outside the routing zone, the ZRP behaves in a reactive manner. Of the reactive schemes, source routing (DSR) can lead to a large overhead for large scatternets. We, thus, choose AODV, which is a table-based scheme, to base the reactive part of our routing scheme on.

A routing scheme based on the ZRP is thus, appropriate for Bluetooth scatternets since it involves low routing overhead and ensures that route acquisition latencies are not too large. Though our routing scheme, which we define in the next section, is closely modeled on ZRP and AODV, we make some changes to these and customize the scheme for use in Bluetooth scatternets.

Our routing scheme consists of two parts - the proactive part and the reactive part. The source node first checks whether it has a path to the destination (the destination node can be either in the routing zone of the source node, as determined by the proactive part, or in the routing tables set up in the reactive part). If the source does not have a route to the destination, the node employs the reactive part of the scheme. Routing information obtained from the proactive and reactive parts of the scheme is stored separately.

A. Proactive Part

In the proactive part, each node maintains routing information for nodes within MAXHOPS hops from it. Each master node sends out the addresses of its immediate neighbors (slaves) to nodes that are up to MAXHOPS hops away. This information is sent in a Link Manager PDU, which we call the LMP_neighbor PDU. We assume that the master is aware of the addresses of the slaves in its piconet. The master can obtain this information when the connection is established with the slave. A time-to-live field in the PDU header of the LMP_neighbor PDU is used to restrict the PDU to MAXHOPS hops. The LMP_neighbor PDU may be sent in a 1-slot or 3-slot packet, depending upon whether the amount of routing information regarding the neighbors’ addresses fits into a 1-slot packet or not.

When to Send

A master node sends the LMP_neighbor PDU if:

a) there is a change in the immediate one-hop neighboring topology (if one of its connections has been broken or a new one has been established)

1 Route acquisition latency refers to the delay in discovering the route to a destination
b) there is a change in the \textit{routing zone} of the node caused due to an LMP\_neighbor PDU received from a new node within MAXHOPS-hops from the node or

c) it has not sent an LMP\_neighbor PDU in the previous LMP\_neighbor\_timeout sec.

A master can determine condition (a) if it has acquired a new slave or if one of its slaves has not responded to its polling messages for some time. A master can determine condition (b) if it receives an LMP\_neighbor PDU from a master node that was previously not present in its \textit{routing zone}. The master keeps a timer to determine condition (c). In condition (a) or (c), a node is trying to inform all its neighbors within MAXHOPS-hops that its neighboring topology has changed, whereas in condition (b), a node is trying to send its neighboring information to one particular node. Thus, the PDU sent in condition (a) or (c) is broadcast, whereas the PDU sent in condition (b) is unicast to the new node.

\textbf{How to Send}

The LMP\_neighbor PDU sent in condition (b) is sent in a unicast manner to the destination node.

In condition (a) or (c), the LMP\_neighbor PDU has to be broadcast and restricted to within MAXHOPS-hops of the sending node; a time-to-live field in the PDU is used to achieve this. The node that sends out the PDU sets this field equal to MAXHOPS. A gateway or a master node receiving the PDU decrements the value of this field by one and broadcasts it again if the decremented value is greater than zero.

The notion of broadcast in Bluetooth is different from that in 802.11, since neighbors of a node are not determined by physical proximity, but by the logical structure imposed by the Bluetooth scatternet. Thus, we define the notion of broadcast in Bluetooth as:

- If a master needs to broadcast, it sets the AM\_ADDR field to 0 and transmits the message. All slaves in the piconet (except gateways which may be visiting some other piconet) get this message. If a gateway was visiting some other piconet at the time of the broadcast, the master sends the message to it in a unicast manner when it is present in the piconet.
- If a non-gateway slave needs to broadcast, it sends the message in a unicast manner to its master. If the slave is a gateway, it sends the message to each of its masters when it visits the piconet.

Each master/gateway node maintains a Routing Map, which contains an entry for each master node from which it has received an LMP\_neighbor PDU and a list of the addresses of that master node’s immediate neighbors. When a master/gateway node receives an LMP\_neighbor PDU, it updates its Routing Map in the following manner:

- If the LMP\_neighbor PDU is received from a master node that is not an element in its Routing Map, it adds an entry corresponding to this master node in the Routing Map (along with a list of the master node’s slave neighbors).
- If the LMP\_neighbor PDU is received from a master node that is an element in its Routing Map, it updates the list of neighbors (slaves) for the master node in its Routing Map according to the list received in the PDU.
- If it has not received an LMP\_neighbor PDU from a master node that has an entry in its Routing Map for the last (2 * LMP\_neighbor\_timeout) sec, it deletes the master node’s entry from the Routing Map. The value of LMP\_neighbor\_timeout seconds is typically set to 6 seconds as a balance between keeping the overhead low and exchanging information frequently.

A gateway or a master node creates a routing table using the information received in the LMP\_neighbor PDUs. A slave node does not store any information and only forwards each packet to the master, where routing lookup takes place. This structure also suits a piconet quite well, since a master is typically expected to be a bigger node than the slaves in its piconet.

\textbf{Storing Routing Information}

The routing information is stored such that retrieval (which is expected to be more frequent) is more efficient than updating (which is less frequent, typically once in LMP\_neighbor\_timeout sec). We explain how the routing information is stored at the master. The information is stored in a similar manner at the gateway.

The information is stored in two tables, the \textit{node\_linked\_list} (which is shown in Fig 1) and the \textit{node\_hash\_table}. Each node of the \textit{node\_linked\_list} contains the address of a master and the slaves (gateway and non-gateway) of its piconet. Each gateway slave in this node has a link to nodes that have the addresses of each of its masters and their slaves. For example, from Fig 1, the first node of the \textit{node\_linked\_list} contains the address of the master whose routing information is being stored, and the addresses of its slaves. The gateways in this node have a link to nodes that contain addresses of their other masters and so on. This structure clearly directly corresponds to the node’s \textit{routing zone}, and it is easy to see the nodes at 1-hop, those at 2-hop etc. The information in the \textit{node\_linked\_list} is used to fill up the \textit{node\_hash\_table}.

The \textit{node\_hash\_table} is an array of a fixed number of elements (we used the value of 15 for this in our simulations), where each element is a linked list. For each node whose address is received in the LMP\_neighbor PDUs, a hash function is applied to its address to obtain a Hash Index:

\[\text{Hash Index} = \text{Address} \mod \text{Size}\]

An entry for this node is added at the Hash Index. If there are already other entries at the element indexed by the Hash Index, the new entry is added at the end of the linked list corresponding to the index. Along with each such entry, the node to which a packet that has this node as destination is
to be forwarded is also stored. This can be obtained from the node_linked_list.

Note that retrieval of the node to which a packet is to be forwarded is very quick. The hash function gives a direct reference to the element in the node_hash_table, from which the address of the node to which the packet is to be forwarded is obtained. Update of the node_linked_list and the node_hash_table requires more operations, but since this is an infrequent procedure, it does not burden the system much. Even so, only those portions of the node_linked_list and the node_hash_table that are changed need to be updated (and not the complete tables) each time an LMP_neighbor PDU is received. Also, note that all this storage and processing of routing information is taking place only at the master and gateway nodes. This saves the (typically smaller) slave nodes from being involved in routing procedures.

![Diagram of Node Linked List for a Master Node](image)

Fig 1: Node_linked_list for a master node

**B. Reactive Part**

When a node does not have a path to the destination, it does path discovery by broadcasting (note that broadcasting here is the Bluetooth broadcasting we defined earlier) a route request (RREQ) packet, as in AODV [7]. The first node that has a path to the requested node sends a route reply (RREP) packet back to the source. The node may have obtained the path either from the reactive or the proactive part of the routing scheme. As the RREQ packet is flooded, nodes set up reverse paths by recording the address of the neighbor from which the first RREQ was received. Each node maintains a counter, called the broadcast_id, which is incremented each time a RREQ packet is sent. Each RREQ contains the source address and the broadcast_id, which uniquely identify it. Using this, a node can discard multiple copies of the same RREQ packet and not broadcast these. A RREQ also contains a field called the hop_count, which is incremented each time the RREQ is broadcast.

The RREP packet finds its path back to the source using the reverse paths that were set up. As the RREP traverses back to the source, each node along the path sets up a forward pointer to the node from which the RREP was received. This creates forward paths to the destination.

The value of the hop_count obtained from the RREQ packet received at the node replying to the RREQ is copied in the hop_count field of the RREP packet. If a node receives more than one RREPs in response to the same RREQ, it updates its routing table and propagates the RREP only if the RREP has a smaller hop_count than the previous RREP. A forward or reverse path is removed if no packet for the destination is received within path_timeout (= 3 sec).

**V. SIMULATION RESULTS**

To evaluate the performance of our routing scheme, we used the Bluehoc [13] simulator (which is built on NS-2 [14]) and added our scatternet routing scheme to it. The routing scheme resides just above the LM (Link Manager) layer of the Bluetooth stack.

The two important parameters we study are the routing overhead and the route acquisition latency. We consider scatternets of different sizes consisting of 50, 100, 150 and 200 nodes. The value of MAXHOPS is varied between 0 (completely reactive), 1, 2, 3 and 4. The routing table to store the paths in the reactive part of the routing scheme has a size of 15 entries.

We generate CBR connections where the source and destination points of the connections are chosen randomly. The connections start at a random time between 0 and 10 sec and run for a random time varying between 1 sec and till the end of the simulation. 150 connections are generated with the end time of the simulation being 200, 500 or 900 seconds. This causes the rate at which the connections are generated to have various values (e.g., 150 connections in 200 seconds is a higher rate of generation of connections than 150 connections in 500 seconds). Varying the rate of generation of connections shows the difference in performance due to the MAXHOPS parameter. A more proactive scheme (higher value of MAXHOPS) would be less efficient when rate of generation of connections is lower and vice versa.

Each scatternet is generated in the following manner: for each node, we randomly decide whether it is a master with a probability of 25%. Slaves are then randomly distributed around the masters. Some slaves are chosen as gateways with the restriction that a gateway may belong to a maximum of 2 piconets and a piconet may have a maximum of 3 gateways.

We do not model mobility in the scatternet. Mobility effects in Bluetooth scatternets will be dependent on the connection establishment procedure of Bluetooth (and hence, on the scatternet formation and reorganization algorithm) since Bluetooth is a connection-oriented technology. Nevertheless, our routing scheme deletes old entries from the routing table (as a routing scheme that supports mobility would, as is explained in the Reactive Part of the scheme). The work in this paper focuses only on the routing scheme. In future, we plan to extend this work to combine routing and formation in scatternets.
Fig 2(a): Proactive routing overhead when time is 200sec

Fig 2(b): Reactive routing overhead when time is 200sec

Fig 2(c): Total routing overhead when time is 200sec

Fig 3: Route acquisition latency when time is 200sec

Fig 4: Total routing overhead when time is 500sec

The proactive part of our routing scheme transfers routing information for all nodes of a piconet in a single packet. Moreover, for each route request query, each piconet may have to expend a few packets to forward it. Since this constitutes a very small fraction of the bandwidth, only a small amount of the piconet bandwidth is used to propagate routing information in the reactive part of the routing scheme.

Also, since nodes are not mobile, only periodic updates of routing information take place in the periodic part. When nodes are mobile, there will be added overhead due to immediate updates of routing information. Nevertheless, our routing scheme has very low overhead. Another point to observe is that the amount of storage for routing information will be roughly proportional to the PRO shown in Fig 2 (a). This is highest for MAXHOPS = 8.

The route acquisition latency (Fig 3) is lowest for MAXHOPS = 8 since routes are already present for a large number of connections.

Fig 4 shows the TRO (we do not show the PRO and RRO for lack of space) when the connections span over 500 seconds and the value of LMP_neighbor_timeout is the same as earlier; Fig 5 shows the same when connections span over 900 seconds. From Fig 4 and 5, it can be seen that as the rate of generation of connections is decreased, higher values of MAXHOPS give a higher overhead since bandwidth is wasted in exchanging routing messages even when no connections need to be routed. Middle values of MAXHOPS such as 2 and 4 are able to achieve low overhead.
VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a routing scheme for Bluetooth scatternets based on the Zone Routing Protocol. We motivated the design of the scheme keeping in mind considerations of Bluetooth scatternets. We evaluated the performance of the scheme using simulations. The parameter MAXHOPS determines the performance of the algorithm. The simulations showed that the proactive part of the scheme is able to communicate a large amount of routing information at very low overhead. A higher value of MAXHOPS leads to very small route acquisition latencies, but leads to higher routing overhead and higher cost of storing information. Also, in networks that are largely idle, the proactive part may cause unnecessary overhead, particularly for large values of MAXHOPS. In fact, a node may be able to determine a “good” value of MAXHOPS for itself by considering its resource (memory etc.) constraints.

In future, we would like to enhance the routing scheme such that a node can request a change in the size of the MAXHOPS parameter according to its resource constraints. The node may achieve this by broadcasting control packets that request nodes to start or discontinue sending LMP_neighbor PDUs to it. Such control over the size of the routing zone may be very useful in networks that are constantly changing. Thus, a node which realizes that due to a bigger and denser network, its routing tables have become very large, can request a decrease in the value of MAXHOPS. This strategy comes with some overhead and it remains to be seen if this overhead justifies its use.

Another area we are working on is combining routing and formation in Bluetooth scatternets. To evaluate our routing scheme in the face of mobility, it is necessary to have a scatternet formation (and reorganization) algorithm. We are developing a formation algorithm that works closely with our routing scheme. We will then be able to provide a complete solution for handling mobility in Bluetooth scatternets.

REFERENCES