AD HOC ROUTING FOR BLUETOOTH USING EPIDEMIC ALGORITHM

A. SAI SUNEEL

Assistant Professor, Department of Electronics and, Communication Engineering, School of Engineering & Technology, Sri Padnavathi Mahila Visva Vidyalayam, Tirupati, India
saisuneel.adem@gmail.com

ABSTRACT

In an ad hoc network connection establishment is done to secure a path for routing packets from source to destination. This path is then maintained to secure further transmissions. In the case of Wireless Personal Area Networks (WPAN) which are more flexible, this is not usable. Some constraints in Bluetooth demand a more flexible on-demand ad hoc network compared to other types of ad hoc network protocols. Initially found that creating a scatternet is not practically possible using Bluetooth version 1.1 as the slave/slave bridge is not implemented into the hardware. In this paper, we present an Epidemic Algorithm to make it better suited for Bluetooth routing. Distributing information within networks can be complicated if hosts have only limited knowledge of the properties of the network. This leads to problems when it is highly important that specific information has to reach one particular host or all hosts within the entire network. Epidemic algorithms follow the paradigm of nature by applying simple rules to spread information by just having a local view of the environment. According to this fact, these algorithms are easy to implement and guarantee message propagation in heterogeneous and not all the time coherent environments.

I. INTRODUCTION

The number of devices making use of Bluetooth technology has rapidly increased in numbers thanks to the amount of implementations in cellular telephones, Personal Digital Assistants (PDAs), etc. Bluetooth is also very strong in secure communication as compared to other technologies used today. Instead of the point-to-point technique used today the wireless community demands more sophisticated solutions to transmit data between two devices, e.g. using a chat program within an ad hoc network.

By an on-demand ad hoc network, we refer to a network in which the path of a packet is not known a priori, but must rather be discovered first and in order to secure for subsequent transmissions. Each established connections are disconnected directly after a packet has been transmitted because of the potential of the nodes. This is a pertinent issue in nomadic environment such as sensor networks or Bluetooth Wireless Personal Area Networks (WPANs). The Bluetooth technology suffers from connections times sometimes being quite long (in the order of several seconds). In order to consider Bluetooth a viable option for use in a nomadic routing scenario, it is therefore imperative to reduce the number of connections made, even if this would mean the introduction of some overhead with regards to transmitted data.

A local Bluetooth ad hoc network is created by using from two up to eight devices in a piconet; one device is always a master. Using Bluetooth to create a local ad hoc network (piconet) is easily done. To create larger ad hoc networks (a so-called scatternet), several piconets are used. The formation of a scatternet is however not properly implemented in the hardware. The slave/slave bridge is supported in the specifications [1] and [2] but is not implemented by the hardware manufacturers. The master/slave bridge is also restricted to only one connection if the role is that to a slave. If a device becomes a master, it could handle a slave role too, but only in this order. As soon as a device becomes a slave, it is no longer reachable for any more connections. This constraint excludes the possibility to create different formations of a scatternet. To solve this problem each connection must be created when needed. After the packet is transmitted the connection is removed immediately. The ad hoc network connection is not maintained for other packets to be transmitted.

Routing algorithms and capabilities are not specified in Bluetooth (IEEE 802.15.1) version 1.1. This paper we suggest a routing algorithm is used epidemic algorithm. Even though it introduces some overhead, it offers significant gains in message transfer rates due to a smaller number of communications as compared to other algorithms as well as complexity of a network also reduced.

The paper is organized as follows: Section II describes related work and research within Bluetooth scatternet formation. Section III describes the issue of time-to-connect with related works. Section IV describes the routing and related protocols together with the Epidemic algorithm and compares the latter to other approaches. Section V presents the algorithms necessary for the routing protocol. In Section VI describes results. In section VII describes conclusions.

II. RELATED WORK

A three-stage scatternet formation scheme is described by Tadashi et al. [3]. First a neighbour discovery protocol is used. The reason is to create small piconets before one can create a scatternet. Second, a scatternet
formation protocol creates the scatternet itself. Finally, to maintain the scatternet just created a scatternet adjustment protocol is used to take care of new nodes or old ones leaving the area in range. All these stages are simulated and different results are presented.

A similar paper by JiHyuck et al. [4] describes a three-phase ad hoc network formation Bluestars and Bluestars*. The first phase is neighbour discovery, the second is neighbour grouping (scatternet formation). At last, a role is performed, which differs from the approach [3]. The last phase concentrates the behaviour on the role between the devices. The Bluestars and Bluestars* are compared to a tree topology proposed in [5]. A topology visualization tool was created to examine the behaviours. The proposed topology uses shorter average distance than the tree topology with a reasonable number of links.

Another approach used is a distributed scatternet formation protocol. salonidis et al. [6] describes a scatternet formation protocol. The Bluetooth Topology Construction Protocol (BTCP) uses a two-stage formation scheme. The first phase is used to select a single global leader within the range of the scatternet. This leader stores knowledge about other devices. In the next phase the leader tells other devices to form the scatternet. Law et al. [7] reduce the steps to only one phase. A vote between two or several devices is used to elect a leader. This is recursively done until the entire scatternet is created.

To test all these protocols and algorithms on Bluetooth hardware, these papers assume that the slave/slave bridge is implemented in the Bluetooth hardware. According to the findings the slave/slave bridge is not implemented, which makes a work-around necessary.

III. TIME-TO-CONNECT

1. Neighbour Discovery

The main idea for each client and access point (router) is that they should maintain their own list of discovered neighbors. This is done by using the Host Controller Interface (HCI) periodic inquiry command. The list is established before any message is being sent through the ad hoc network. If a node enters the area in range of an ad hoc network this could take some time because of the inquiry procedure [8].

For Bluetooth the selection box and the state machine play an important role in the process of creating an ad hoc network. The state machine together with the selection box derives the next frequency used by the device. According to the specification [1] a maximum value of 10.24 s for the inquiry procedure may be required. For on-demand ad hoc networks this is not efficient enough. The relatively high time-to-connect is a huge disadvantage when establishing connection between devices for version 1.1 of Bluetooth. In [9], the role of using the random back off parameter is discussed and improved settings of this single parameter are reported using an optimized random back off parameter in Bluetooth’s Frequency Hopping Spread Spectrum (FHSS). The modified state machine allows us to finalize the inquiry process within about 6 s with a mean of 2.0 s.

2. Connection

Using version 1.1 of Bluetooth the mean time for page procedure is 1.6 s [8]. If version 1.2 of Bluetooth should be used, which contains a faster page procedure, the mean time of the page procedure is most likely to be reduced.

IV. ROUTING AND PROTOCOL SWITCHING

1. Store and Forward Packet Switching

This broadcasting routing type includes Reverse Path Forwarding (RPF). This is a straightforward algorithm which forwards a packet to all other links except the one from on which the packet was received. Otherwise, the packet is discarded if the incoming packet is known to the router and accordingly not being forwarded. There are types of broadcast such as Hot Potato Forwarding or Spanning Tree Forwarding [10] that are left for further study.

Let us consider a simple example presented in Fig. 1 and take into account that as the connections are extremely short lived, i.e. a sending node is in general not able to receive data simultaneously. Let us assume that node A sends a message first to node B (1.) and then to node C (2.). The RPF algorithm will then make B send the message to C and vice versa. In case (a), node B is first to send the message to C (3.) while C has to wait for B to become idle. However, there is a chance that B starts the transmission to D (4.) before C is able to transmit to B (5.). In case (b), node C is first to send the message to B (3.).

![Fig. 1: RPF](image)

Now, it may happen that B sends the message first to C (4.) and then to D (5.), which implies an unnecessary delay due to one unnecessary message transfer from C to B. In large-scale scenarios, such superfluous messages may cause network flooding, which is also known as the broadcast storm problem [11].
2. MODIFIED REVERSE PATH FORWARDING (MRPF) FOR BLUETOOTH

The modification to the RPF algorithm consists in avoiding unnecessary message transfers by supplying a message header with routing information. As the Enhanced RPF algorithm reference involves sending notification messages before sending the very message, the basic problem of establishing unnecessary connections is basically the same as with RPF. The difference between RPF and MRPF is seen from the composition of Figures 1 and 2. Again, we assume that node A sends a message first to node B (1.) and then to node C (2.). In case (a), B is first to send the message to C (3.). MRPF prevents C from sending the message back to B again. In case (b), C is first, and B is able to transmit the message to D (4.) earlier than in the corresponding RPF case.

![Fig 2: MRPF](image)

Thus, the Modified RPF for Bluetooth reduces the number of connections. Computing & battery power is saved by applying less no. of connections but more memory is needed. Because the information about the packets that already passed the node has to be stored. Complexity of the simulation environment in larger networks.

3. EPIDEMIC ALGORITHM

The proposed Epidemic algorithm [12] is reduced the complexity of the simulation in larger networks. Epidemic algorithms describes whenever two hosts come into communication range if host has the lower ID start anti-entropy session and exchange all messages, that one of the hosts has not seen yet.

For example when a host S wants to send a message to another host D, S passes the message to its neighbor hosts and they do the same until the message reaches D. If there is a partition of the network on the way to D, it is likely that one (or more) host(s) of the current section come into contact with another fraction at a later time due to node movement. So, the message is passed throughout the network and will eventual received by D with high probability (fig.3).

![Fig 3:Message delivery from node S to D](image)

The exchange of information between two nodes, when they come close to each other, uses the technique of anti-entropy in the context of updates in replicated databases. Originally, anti-entropy means that a host chooses randomly another host for data exchange. For the use in Epidemic Routing, happenstance is “simulated” by the event of two hosts coming into communication range.

Each host stores messages in a buffer (hash table), indexed by the unique ID of each message. If an anti-entropy session is established, the two nodes exchange messages they do not know. For the gain of performance, only a summary vector (bit vector) is exchanged, indicating which entries in the buffer are set.

If the message buffer overflows, old messages are flushed. This has to be taken in to account especially if limitations are made to the buffer size or the maximum hop count. In this case, there has to be a fair queuing strategy to achieve fairness and quality of service.

3.1. Classification of epidemic algorithms

In each epidemic algorithm there exists a so-called population, a set of interactive, communicating units. These units use a ruleset that defines how to spread specific information that might be of interest to other units. This ruleset if absolutely affected by the design of the algorithm and can be freely chosen. The only requirement is, that at a specific time a unit must have one of the following states regarding to specific information:

- **Susceptible**: the unit does not know anything about the specific information but it can get this specific information.
- **Infective**: the unit knows the specific information and uses the ruleset to spread the information.
- **Removed (Recovered)**: the unit knows the specific information but does not spread it.

Based on these definitions, we can define different classes of epidemic algorithms as described. These classes define how units can handle incoming information in general.

3.1.1. Susceptible-Infective (SI)

This class defines epidemic algorithms where nearly each unit is initially susceptible. By receiving updated information a unit becomes infective and remains it, until the whole population is infective. Therefore we need some additional methods to decide whether to stop spreading information or not.
3.1.2. Susceptible-Infective-Susceptible (SIS)
In difference to the model of SI algorithms, SIS algorithms are able to decide to stop spreading information before the whole population is infective. For example, if a unit realizes that all of his last five communication partners are already infective, it might decide that the specific information, which has to be spread, is old, and stops spreading it. But removed units can still become infective again. If a unit gets specific information, that it has stopped spreading before, it will spread it again until it loses interest again.

3.1.3. Susceptible-Infective-Removed (SIR)
SIR algorithms are nearly the same as SIS algorithms with the only difference that removed units remain removed for specific information. This means that a unit would never get infective again, after it once stopped spreading specific information. For instance, a unit, which did not spread specific information in the last x rounds, stops spreading that specific information for all the time, assumes that the whole population is already infective. Mathematical modeling of this class is not an easy issue. Up to today, there is no closed formula available for describing the infective rate. There are only numerical analyses available, which show, as expected, that at the end a constant number of units keep susceptible. This is obvious, as the units stop spreading, because they mainly choose infective units as communication partner and not one of the last remaining susceptible units that are very few.

3.2. Anti Entropy
Anti-entropy is a kind of SI class epidemic algorithm. According to that, the three states of an epidemic environment can be applied in terms of our databases:
- **Susceptible**: the database does not know a certain update
- **Infective**: the database knows a certain update and spreads it
- **Removed**: the database knows a certain update but does not spread it

Regularly each of our n replicas D1 . . . Dn chooses a small number m of the other n − m replicas uniformly at random as synchronization partner. After choosing a site, the whole data is synchronized between these two databases. Synchronizing the whole database is expensive because of the need to transfer somehow the whole database through the network, in order to be able to compare the data. Thus this technique can not be used too frequently. It is suggested to be an update strategy besides other techniques like the direct mail strategy.

We now present three synchronization strategies. For these, we consider S to be the database that chooses T for the anti

3.3. Epidemic Routing Protocols
These protocols depend heavily on which method the algorithm uses to spread the information. Especially the direction of the flow of information affects the number of transmission for specific information to reach the whole population. We will now introduce some communication techniques and show, how long they will take, to infect the whole population.

3.3.1. Push algorithms
In this algorithm describes each node \( u \) chooses a communication peer \( v \) and sends it any new information it has. In this mode, the infective nodes are the initiators of the communication.

![Push algorithm](image)

At the beginning, only a few nodes are infective and the chance to choose a susceptible unit is high. So the chance to infect at least one more units in the first round is more or less reliable. For the first third of the population, the number of infections \( #_{\text{infective}} \) in the round \( t \) is at least

\[
#_{\text{infective}}(t) = 2 \cdot #_{\text{infective}}(t-1) \cdot \left( 1 - \frac{e \cdot (\ln n)^2}{n} \right)
\]

In the following rounds, when \( #_{\text{infective}}(t) \) is in the interval of \([n / 3 , n]\), the chance for an infective unit to choose a susceptible unit as communication partner gets even smaller. There is not much space left to spread the information. The expected number \( S(t) \) of susceptible units in round \( t \) is less than:

\[
S(t) = e^{-\frac{1}{3} \cdot #_{\text{infective}}(t-1)}
\]

As expected these result are very close to the properties of the SI class. In [13] a push-based SI algorithm needs \( \log_2 n + \ln n + O(1) \) rounds to infect the whole population.
In this algorithm describes each node \( u \) asks a chosen communication peer \( v \) for any new information that the peer has. In this mode, the susceptible nodes are the initiators of the communication.

![Pull algorithm](Image)

Fig. 5: Pull algorithm

Due to the fact that only a few nodes are infective at the beginning, the expected relative number \( \# \text{infective}(t) \) of infective units for round \( t \) is:

\[
\# \text{infective}(t) = 2 \cdot \frac{\# \text{infective}(t-1)}{n} - \left( \frac{\# \text{infective}(t-1)}{n} \right)^2
\]

If \( \# \text{infective}(t) \) is in the interval \([n / 2, n]\), the number of susceptible units, that is less or equal to \( 1 / 2 \), squares for the next \( \log \log n + 1 \) rounds, up to the whole population is infective:

\[
\left( \frac{1}{2} \right)^{2^{\log \log n + 1}} < \left( \frac{1}{2} \right)^{\log n} = \frac{1}{2^n}
\]

Thus, the expected number of rounds for a pull-based SI algorithm is \( \log n + O(\log \log n) \), that is much better than the above results for push-based algorithms.

### 3.3.3. Push & Pull algorithms

In this algorithm describes each node \( u \) chooses a communication peer \( v \), sends to the peer any new information it has; at the same time, the node asks its peer for any new information that the peer has.

![Push & Pull algorithm](Image)

Fig. 6: Push & Pull algorithm

If the number of infective units is in the interval of \([1, n / 2]\), the push mechanism ensures that after \( \log_3 n \) rounds at least half of the population is infective. If the number of infective units is in the interval of \([n / 2]\), the pull mechanism ensures, and that in \( O(\log \log n) \) rounds the second half of the population gets infective. Assuming, that communication partners are not chosen uniformly at random but by a weighted function, push&pull algorithms will even get better results. Within the first rounds, one unit will infect the highest weighted unit via a push operation as the chance, to choose the highest weighted unit as communication partner is very high. As soon as the highest weighted unit is infective, most of all other nodes will get infected via pull operation by choosing this unit as communication partner as shown [14].

### 4. ALGORITHM

A more efficient technique is the Min-Counter Algorithm introduced by Scott Shenker [14]. The idea behind the algorithm is that each unit \( P \) gets a counter \( C_R(P) \) for each rumor \( R \). This counter is increased if and only if all \( C_R(Qi) \) of the communication partners \( Qi \) from \( P \) in the previous round were at least as big as \( C_R(P) \). If \( C_R(P) \) gets bigger than \( C_{\text{max}} \), \( P \) would stop spreading rumor \( R \) (fig. 7).

![Shenker’s Min-Counter Algorithm](Image)

Figure 7: Shenker’s Min-Counter Algorithm

Analysis show that the whole population can get informed in \( \log_3 n + O(\log \log n) \) rounds with a probability of \( 1 - nO(1) \) using a push&pull algorithm. Therefore only \( O(n \log \log n) \) messages need to be send.
VI. RESULTS
Vahdat and Becker [12] implemented the Epidemic Routing protocol using the Monarch Wireless and Mobility Extensions [15]. They simulated 50 nodes moving randomly within a 1500m × 300m area and examined the behaviour of their protocol with respect of various settings in transmission range (250 m, 100 m, 50 m, 25m, 10m), message hop limit (8, 4, 3, 2, 1 hop(s)) and message buffer size (2000, 1000, 500, 200, 100, 50, 20, 10 messages).

The results showed that given pairwise connectivity and enough buffer size (2000 equals infinite size, because the whole experiment used only 1980 messages) and time, an epidemic algorithm can guarantee eventual delivery of 100% of messages (fig 8). One exception has to be mentioned: the experiment with a transmission range of only 10m did not deliver all messages until the end of the experiment (within 200,000 seconds). This is due to the fact, that it is hard for the hosts to meet each other because the covered area of one host is rather small (cp. column ‘coverage floor’ in table 1). Given more time, 100% of messages would be delivered according to the random movement.

They also showed that using restrictions on the different parameters, messages are still delivered with high probability increasing delivery latency just by a little factor. For instance, limiting the hop count to three hops still ensures delivery of all messages with the average latency increasing by 33% (fig.9). When limiting the buffer size to 2.5% of the originated messages, the number of delivered messages decreases to 79.9% (fig.16). But using a buffer for 5–25% of messages was sufficient to deliver almost all messages with reasonable latency.

![Figure 8: Message delivery with respect to transmission range][12].

<table>
<thead>
<tr>
<th>Table 1: Characteristics of Epidemic Routing [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
</tr>
<tr>
<td>250 m</td>
</tr>
<tr>
<td>100 m</td>
</tr>
<tr>
<td>50 m</td>
</tr>
<tr>
<td>25 m</td>
</tr>
<tr>
<td>10 m</td>
</tr>
</tbody>
</table>

![Figure 9: Message delivery with respect to message hop limit at 50m transmission range][12]

![Fig. 10: Message delivery with respect to Message buffer size][12]

<table>
<thead>
<tr>
<th>Table 2: Delivery rate and latency with 50m transmission range and a limit of 4 hops at various buffer sizes [12].</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer size</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>500</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

[12]: http://ijpaper.com/
CONCLUSIONS
We develop techniques to allow message delivery in the case where a connected path from source to destination is never available in mobile ad hoc networks. While existing ad hoc routing protocols are robust to rapidly changing network topology, they are unable to deliver packets in the presence of a network partition between source and destination. For a number of compelling application classes, including mobile sensor networks and disaster recovery scenarios, nodes can be spread over wide geographical distances. Such wide dispersion makes it unlikely that a connected path can always be discovered, making it virtually impossible to perform message delivery using current ad hoc routing protocols. Thus, we introduce Epidemic Routing, where random pair-wise exchanges of messages among mobile hosts ensure eventual message delivery. The goals of Epidemic Routing are to maximize message delivery rate and to minimize message latency while also minimizing the total resources (e.g., memory and network bandwidth) consumed in message delivery. Through an implementation in the Monarch simulator, we show that Epidemic Routing delivers 100% of messages with reasonable aggregate resource consumption for scenarios where existing ad hoc routing protocols are unable to deliver any messages because no end-to-end routes are available.

REFERENCES
[9]. L. Isaksson, and M. Fiedler, Optimization of the Random Backoff Boundary of the Bluetooth FHSS Technique, COST 279, 11th Management Committee Meeting 2004, Ghent, Belgium