Abstract

Mobile phones have become a global product, serving the need for communication. There are currently many routing algorithms that tackle with routing in ad-hoc networks. However, applying these algorithms to mobile phone devices incurs problem due to their reduced battery power, bandwidth and processing power.

In this paper we propose BlueMesh, an ad-hoc routing protocol over Bluetooth for mobile devices. It is self-starting and free from any type of central administration. The BlueMesh protocol utilises a proactive, table driven algorithm creating a mesh network that distributes routing tables across all nodes to share knowledge of the best end to end routes. We also develop a text messaging and Tic-Tac-Toe applications to operate over BlueMesh

1. Introduction

The last decade has seen a rise in the development of telecommunications, in particular with portable computers and mobile phones, with the increasing demand for flexibility and portability whilst ensuring optimal connectivity.

This has bought much interest in MANET (Mobile Ad-Hoc Networks) [1], in particular the development of wireless routing applications in static and dynamic topologies. Furthermore, MANET describes two types of routing algorithms, proactive and reactive (discussed in Section 2.1). Proactive characterises algorithms that store routing tables of the whole network at each node at the expense of network traffic. On the other hand, Reactive algorithms reduce network traffic by only storing local connectivity nodes (within one hop) and discover the routes on demand at the expense of discovery latency. MANET studies have largely been based upon Wi-Fi technology as the infrastructure; however we propose to develop our protocol over the emerging Bluetooth technology.

2.0 Background

2.1 Current Algorithms

In order to develop an algorithm for use over a Bluetooth mesh network we researched the current algorithms that are applied to MANET’s [1].

Routing algorithms are classified into Proactive and Reactive algorithms. Proactive algorithms are characterized by having pre-established route tables for nodes within the network before routes are needed which is formed through periodic broadcasting.

DSDV [3] is a well established proactive algorithm in which nodes entering the network...
broadcast themselves and exchange route tables with local nodes. A newly entering node would have an empty table but when connecting to a neighbouring node it is able to create its own view of the topology.

This routing algorithm also uses sequence numbers with each route entry in order to differentiate between stale and fresh routes but more so preventing the formation of loops. All the nodes of the network maintain their own sequence numbers which are even. However if a broken link is discovered then the neighbouring nodes set the broken link’s sequence number to odd and broadcast the update which triggers an immediate propagation.

The algorithm’s main feature is that nodes maintain a global knowledge of the network through periodic broadcasting. However this proves to place a heavy upon the network. Firstly periodic broadcasting means that packets are being sent even if there are no topology changes, but even more so, nodes maintain information of the whole network even though the node may only intend to communicate with a small selection of the network. Furthermore this affects the efficiency of the algorithm’s scalability as the overheads increase with network size.

OLSR [4] is another proactive routing method that also operates by periodically transmitting throughout the network. However, the algorithm utilizes MPR’s (Multipoint relay’s) that are selected nodes in the network to distribute network messages. Thus minimizing the flooding in the network by only utilizing MPR’s to forward through the network.

The concept of reactive algorithms such as AODV [5], DSR [6], B.A.T.M.A.N [7], is to initiate route discovery only when needed and by dynamically establishing the route. In AODV nodes initiate RREQ (route requests) that contain the destination address of the node they wish establish a connection with. The node broadcasts the RREQ to their neighbours who either satisfy the request or re-broadcast the RREQ. All nodes that re-broadcast the RREQ set a forward pointer to where they received the message from in order to initiate a reverse path for RREP (route reply) to establish a connection.

This protocol reduces the traffic overhead in the network by not periodically broadcasting through the network and consequently not storing a route table for the whole network. Although this reduces the traffic on the network, the protocol suffers from discovery latency. Furthermore this latency may cause unnecessary route requests to nodes that are not present in the network thus causing time inefficiencies.

In conclusion we have decided to develop our own proactive algorithm that maintains distribution a route table of all nodes in the network. We recognize that it places an increased traffic overhead on the network. However due to the mobility of ad-hoc networks, in particular of mobile devices, it is time efficient to have a pre establish route for nodes rather suffer problems of route discovery and reverse route formation. Furthermore with regards to the performance of our applications, we felt that reactive algorithm would result in unacceptably long wait for transmissions.

### 2.2 Difficulties of Bluetooth in Ad-Hoc Networks

Wireless networks benefit from several advantages such as the reduction in cables as data is transmitted through the air. Furthermore they provide devices with full functionality whilst enabling the mobility of the device. However these advantages cause difficulties in the routing of ad-hoc networks. By allowing devices mobility, causes routing problems in the network as nodes can be de-partitioned from part or whole of the network. This causes problem for the routing within the whole network as constant movement of nodes increases the difficulty of finding a stable route for synchronous connection. This will be a difficulty to our project as each Bluetooth device (mobile phone) only has a radio range of 10 meters. This means that in order for a mesh network to form, one node from each piconet must be within 10 meters of a node in another piconet in order to establish a mesh network. Although this is a significant difficulty, the BlueMesh software is intended for use over small spaces such as buildings, lecture theatres as so to reduce the movement.

In routing with Bluetooth, a device discovery results in the Bluetooth address of phone nodes that can then be used for direct connection. However, the address has no significance on the location of the device. This information would allow us to discover the shortest measurable route rather then shortest route through hops. However this feature will not impact the routing of our protocol as the maximum distance between nodes is 10 meters thus the advantages of finding shortest measurable distance are less.

Utilizing Bluetooth on mobile phones means that we will be restricted in terms of battery power, bandwidth and processing power which would hinder the performance of J2SE on the device. However SUN Microsystems have developed a J2ME (Java Micro Edition) (Section 3.3) which contains compact libraries for on developing ubiquitous devices.
3. Bluetooth

3.1 Bluetooth History

Bluetooth was first established in 1998 through the SIG (Special Interests Group) [9] that now has over 9,000 members who are leaders in telecommunications, computing, music and industrial automotive. Members include Ericsson, IBM, Intel and Nokia who drive the development for wireless technology, market and implement the technology through their products.

Bluetooth radio nodes operate in the industrial, scientific and medical (ISM) band at 2.4GHZ. Its primary intention is to replace short cables which are made possible by its ability to reach up to 100 meters. Although most standard Bluetooth products come with Class 2 Bluetooth which has a range of 10 meters.

Bluetooth’s true potential is realized as it allows wireless connectivity with an array of electronics such as: a PC with its keyboard, mouse and printer, mobile phones with headsets and most recently its integration into next generation games consoles such as the Playstation 3 and Microsoft xbox 360.

3.2 Bluetooth Network Formation

Bluetooth nodes are capable of establishing local one hop networks called piconets but more importantly connecting several piconets together forms a larger network called a scatternet.

When a node wishes to connect to other devices it initiates an inquiry to its neighbours. On receiving the inquiry the node can connect with any of the devices using its Bluetooth address and establishing a connection creating a Piconet [Figure 1a].

The master of the Piconet is selected as the initiating node in the network and then the devices found become slaves to the master. The master can have up to 7 active slaves however it is possible to have up to 255 parked nodes that can be initiated when needed.

Scatternets are formed by connecting two or more piconets through nodes that exist in both piconets [Figure 1C]. These nodes are known as interpiconet nodes and can be a slave in one piconet but also a master in another

![Figure 1. Piconets with a single slave operation (a), a multi-slave operation (b) and a scatternet operation (c).](image)

Unfortunately the Bluetooth specification does not specify any routing protocol for this type of network. However the aim of our protocol is to remove the idea of a piconet or scatternet and instead create a wireless mesh network where by any node can connect to any node in the network through intermediate nodes.

3.3 Development Platform

Mobile devices are characterized by having smaller CPU’s of either 16 or 32 bits. Furthermore they are limited by memory and battery life. Hence it would be in practical to implement either the J2SE or J2EE on micro devices. Instead Sun Microsystems developed J2ME (Micro Edition) to target small devices with limited hardware.

The J2ME is a collection of libraries form the standard edition with the technologies to develop on smaller devices. The J2ME contains configuration of CLDC (Connection Limited Device Configuration) or CDC (Connection Device Configuration), profiles and optional packages for development on the targeted devices.

CLDC is the smaller of the two but the most important to our software development as it is targeted directly to smaller devices with limited hardware capabilities. The CLDC provides developers with java.io, java.lang, java.util and java.microedition which are limited subsets of the J2SE.

4.0 Bluetooth Routing in Ad-Hoc Networks

We propose Bluemesh, a proactive routing protocol designed for mobile ad-hoc networks over mobile phone devices. Due to its proactive nature, the network will benefit from pre-established route tables. These are established by nodes exchanging route information with one another and only maintaining the single shortest route to each device.
4.1 Device Discovery

On initialising BlueMesh, the device begins a search for Bluetooth devices. This search will find all phones within 10 meters and returns a list of the devices. The list stores the unique BD_ADDR (Bluetooth hardware device address) of all the devices found. However, these addresses are meaningless to the user, so the friendly name of each device will be resolved. The friendly name is given to the device by default as the model number, but these are usually edited by the user to their name or nickname.

Once the list of devices within reach has been created, the device performs a service-search using the BD_ADDR for each. The purpose of this search is to find which devices have the BlueMesh software. The list is narrowed down to only those devices with BlueMesh, as only nodes with this service can utilise the applications of BlueMesh (discussed in section 3.3).

The initialising node now has a DeviceString of the nodes within one hop that it can use to connect to. The format for the DeviceString is as follows:


This is a very simple DeviceString, containing only one single-step device. At the start is the Bluetooth address of the local device followed by the local FriendlyName, separated by a ‘-’ character. This allows the recipient of such a DeviceString to discover who sent it to determine whether or not they can see the device, and therefore whether to take note of the contents. At the end is a ‘?’ character, which indicates the end of the friendly name and the start of the next device address in the route. Since this is the only device in the route, there is a ‘|’ character. This indicates the end of the current route and the start of the next route. As can be seen, the local address and FriendlyName is put at the end of each route. This is because the DeviceString is there for the purpose of the recipient, who will know to hop through the sender to get to the device. This means that each route in the DeviceString starts with the destination and shows how to get to the device device-by-device until the sender of the DeviceString is reached. The end of the DeviceString is indicated by the ‘.’ character.

During the DeviceString generation, it will be checked whether or not the routes stored for each device come from devices that can still be seen directly. If the device can no longer be seen, the route information is removed.

When nodes receive a route path for a node that they already have a path for, they keep the node with the shortest hops to the destination in the route table, deleting the remaining not-to-be-used routes. If this metric is the same within both routes then the current route is maintained.

On completion of the table exchange, the Bluetooth device search begins again in order to maintain a fresh table due to the mobility of nodes.

Through exchanging the DeviceString through the network, nodes gain global knowledge of the topology. The list of friends is stored in friendsList and is abstracted by only containing the destination nodes. This list is maintained in Friends List.

4.2 Friends List

The purpose of friends list is to allow the user to keep track of their Bluemesh friends. The application takes the complete list of Bluemesh devices from the complete device list search and stores any non-friend devices in the category ‘others’. Any devices that are friends are listed under the ‘online friends’ category. Any friends that are not part of the mesh are listed under ‘offline friends’. The user is then able to add a device into their friends list category by highlighting and selecting the appropriate device in ‘others’.

The friends list application uses file storage to store data about friends. The friend’s friendly name as well as the device’s Bluetooth address are stored. To see if any friends are in the mesh, friends list compares its stored friend’s addresses to all other devices in the mesh. If a match is found the device is put into the ‘online friends’ category. If a match cannot be found, the friend device must therefore not be in the mesh so its friendly name is added to the ‘offline friends’ category.

5.0 Applications

Nodes with Bluemesh software are capable of utilizing the built in applications. Providing applications on top of the routing algorithm was necessary in order to provide functionality of the software.

We developed ‘Text Chat’ a text messaging application as this is one of the primary uses within mobile phones and a Tic-Tac-Toe game.

5.1. Text Chat
Text Chat operates directly over the Bluemesh routing algorithm and it does not use the Wireless Messaging API (WMA) profiles. Instead text chat operates through sending text fields to the target nodes.

Once the first device list is complete and the friends have been configured in Friends list, a list of communicatable devices is stored in view devices. Text Chat allows the user to select a device for text chat. This presents the user with a text field form which the user can enter their message. This is then sent to the target node in the packet format of:

```
(T|local_address|destination_address|local_name|msg)
```

The ‘T’ as the start of the messages signifies to the Bluemesh server that the service being used is ‘Text – Chat’. The local address and the local name is the 48 bit hardware address of the node and the local name is the friendly name. Finally the message is contained in ‘msg’

On intermediate nodes receiving the packet, they compare their own address with that of the destination address of the packet. If it matches it processes it into the local text chat. However if the destination doesn’t match then the node check to see if it has the route to the destination in the device string and sends to the destination

The receiving node may not know that they are receiving the text therefore the message window pops up with the user’s text.

The main influence for the graphical user interface was from MSN messenger which is a highly successful messaging service.

The GUI of text chat maintains a list of the current conversation as opposed to the classical SMS (Short Messaging Service) that only shows the current message.

5.2 Tic-Tac-Toe

Implementing games over Bluetooth has traditionally operated when both devices are in the proximity of each other. However in the context of our software, the game selected would potentially operate by hopping through destinations to reach the target node. This provides difficulties as the real time of game playability may be affected by the number of hops towards the target. For this reason we developed, Tic-Tac-Toe.

It is a turn based [10] game that waits for the move of the other player and repaints the canvas with the new move. The message structure is similar to that of text chat, however the message contains ‘GT’ to signify the service is ‘Tic-Tac-Toe’. Furthermore the packet also contains the player identity and the index of the square they are updating.

Once the message is received by the target node the move updates the target node screen. In turn the receiver can reply in the same fashion.

Every time a move is made the updated state is stored in a hashtable. This allows the user to play other games at the same time or if the users become disconnect they can later rejoin where they left

6.0 Testing

In testing our software we decided to utilise a testing framework as opposed to manual testing. Although manual testing is easy to implement, as the size of the software increases so does time to run the tests manually. Where as testing frameworks allow tests to be run automatically and hence more time efficient.

As we were implementing with the J2ME platform we discovered that the classical JUnit testing framework would not integrate due to a lack of support for java reflection from J2ME. Hence we researched J2MEUnit and JMUnit, which are similar to the JUnit framework and do not use java reflection.

We analysed both frameworks and discovered that although J2MEUnit has an informative emulator to test on, the assertEquals method can only take an Object or a long. In contrary, JMUnit supports 8 different data types if using CLDC 1.0 or 10 different data types if using CLDC 1.1. Although the GUI of the JMUnit only specifies the number of tests passed or failed, when used with NetBeans 5.0 (or higher), detailed test reports are provided. Hence we implemented J2MEUnit with the supporting library of CLDC 1.1.

The types of testing we used on our software were unit, regression and system testing. In order to support these tests we used the ‘Trac’ system to monitor bugs. We found this feature particularly useful in managing bugs in development. Firstly we would run the weekly tests then assign a ticket to any bugs which helped us monitor and take the necessary action to resolve the bugs.

Unit tests were coded throughout the project. Regression tests were run every Monday evening when our group finished a day’s coding. The regression tests ran all of the unit tests in serial. Towards the end of the project, system tests were written which tested that the product met the requirements. The system tests were performed just before submission of the project.
When developing code we ran the software using the Net Beans emulator however, it was necessary to test on mobile devices also as essentially this is the intended use for. Though we encountered several differences when running the software on the NetBeans emulator compared to the mobile device.

We discovered that NetBeans already had many of the optional packages for mobile devices. However the mobile devices we tested on didn’t always have all the packages. This is because the mobile industry does not specify a standard set of Bluetooth libraries that phones must implement.

Further problems were encountered when testing on mobile devices as they do not support any type of fault reports. Hence we developed a console which provided us with debug messages when run on the mobile. When debugging the software we would insert addMessage methods around the source of the bug such as the example below:

```java
device_list_menu.addCommand(getTextChatCommand());
console.addMessage("TextChat Command Added");
device_list_menu.addCommand(getTicTacToeCommand());
console.addMessage("TicTacToe Command Added");
```

Hence the console would show all the message leading up to the line of source code that failed. It was essential to development that we provided a console on the mobile due to the differences of running the software on NetBeans and a mobile device with limited resources and the implication of integrating software to a mobile phone.

### 7.0 Development Methodology

In developing our BlueMesh software we decided to use the Extreme Programming approach. We researched other methods such as the Waterfall method which is characterized in development stages. At the end of each stage the progress is evaluated to see how well each stage has been achieved. Necessary changes are made then the next phase of development begins. Although the Waterfall method is highly defined, the time required to analyze and implement each stage can be significant which we believed may of led to the project exceeding deadlines.

In addition roles defined in the Waterfall method specify that members of the team only have a specific role and may not be needed throughout the development of the project apart from the project manager. We felt this would affect the knowledge taken through development stages but also the team cohesion would suffer.

Extreme programming methodology specifies that roles are not fixed and so members are encouraged to contribute to all stages in the development process. Furthermore we felt that with each member rotating between tasks would improve motivation and productivity.

Development with extreme programming provides continuous feedback on the success of development. This is facilitated by short iterations of design, implementation and testing which results on feedback of the functionality that is measured against requirements.

Extreme programming provides values, practices and principles to guide the development. Values are the fundamental aspect of the project which includes communication, courage, respect and feedback. These are driven by fourteen principles to support the values. Furthermore XP offers primary and corollary practices to drive development to its full potential. These included shared code, incremental design and implementation and paired programming. Although we were speculative of paired programming is became an effective tool in developing code in the critical stages of programming.

Testing is perhaps the most important feature that extreme programming provides guidance on. It emphasizes testing early and often so that bugs can be discovered early. Testing through short code cycles makes it easier to find the bug(s) as opposed to testing after implementation is complete where multiple bugs can be much harder to find in a complex system.

### 8.0 Conclusion

The concept of our software was to develop applications over an extended Bluetooth mesh. We initially believed that the central focus of the project would be on the applications. However as our research developed it became apparent that the main focus would be on algorithm for routing within the mesh. The Bluetooth specification defines Scatternets, however it does not specify how to route through these larger networks. Further investigation concluded that routing over Bluetooth networks is very much an area of current research. Thus the scope of the project was much wider then we initially believed and were capable of therefore we concentrated on developing a routing algorithm for a Bluetooth mesh network with just two applications.

The outcome of our final product was successful as it met all the requirements. When running Bluemesh on nodes they were capable of discovering all devices within 10 meters and further they were able
to resolve the list to only those devices with our service, Bluemesh. However we faced difficulties using Bluetooth technology as it proved unreliable. This was because nodes may take several searches to discover nodes within 10 meters and furthermore these nodes may be lost again even through the nodes are static.

Nodes were also capable of transmitting route tables throughout the network providing nodes with a global knowledge of the network. The functionality was successful as both TextChat and Tic-Tac-Toe operated fully whilst hopping through nodes in the network.

However the implementation of Bluemesh over Bluetooth provided us with several existing obstacles. The underlying range of 10 meters for each phone meant that the protocol worked in best dense networks. Our intended use for mobile phones were as routers however this provided difficulties as nodes are highly mobile which meant routes through intermediate nodes expired quickly. For this reason we implemented a proactive protocol so that route tables are continuously updating, in turn the user always has a fresh view of the topology. Although this is the advantage, the network suffers from increased overheads due to continuous broadcasting. The network was further slowed down by limitations of mobile phones and Bluetooth speed. As the number of 1 hop neighbours increased, the time taken to discover and service search also increased substantially. The Bluemesh protocol operates best in static networks as the stability increases the freshness of table.

It is difficult to compare the routing algorithm to others currently implemented as the main research in routing over MANET’s is undertaken on Wi-Fi technology and utilizing more capable devices such as laptops and desktop PC’s. Furthermore, the measurement of performance of current algorithms is measured through specifically designed simulators, measuring performance under various network constraints. Although this would have provided a better analysis of Bluemesh, the development of a simulator for mobile networks is in itself a final year project and thus outside of our scope.

The concept of our project is unique due to routing purpose over Bluetooth but furthermore implementing a text chat application. The most significantly used functionality through mobile phones is that of voice calls and text messaging. However the current industry charges for the use of these services. Hence the underlying technology of Bluetooth signifies that these services could become free of charge which could dynamically impact the industry.

9.0 Future Work

The project tackled the main issue in routing over Bluetooth applications; however this area has a vast range of different challenges for future research.

The primary lead on from this project would be to develop a simulator for a bluetooth mobile mesh. This would allow testing our routing algorithm under a variety of network conditions. However difficulties may be faced in simulating the realism of Bluetooth as through our project we often found that devices would not always find each other first, second or even the third time even though devices are within 10 meters of each other.

As an enhancement application of Bluemesh, VOIP could be developed. This would be an interesting prospect as bluetooth is currently used with hands free set for mobile phones. Currently the Bluetooth SIG [9] defines HV packets for voice packets and DV for data and voice packets transmitted at 64kb/s. However routing voice packets in a bluetooth MANET would be a challenge due to delays in packets and limited Bluetooth range of mobile phones.

A further enhancement can be made to the BlueMesh software in the forms profiles. Users of Bluemesh would be enabled to develop a personal profile including a picture and personal description. This would follow the current rising trend of social networking.

For the development of future applications by external developers, an interface could be developed. This would allow developers to integrate their applications smoothly into Bluemesh.

Perhaps the most prosperous research would be that of Bluemesh utilizing Class 1 bluetooth devices (100 meters bluetooth range). This would extend the geographical size of the network substantially. Conceptually, for the given scenario of the current project such as a library, campus, any node would only be two hops away from any other by using the class 1 device as the primary router.

Perhaps the most natural branch off from our research would to be to develop a reactive algorithm for implementation under the same constraints. This would be inline with current MANET studies proactive and reactive methods. Furthermore this routing method is more suited to mobile networks at the expense of discovery latency.
10.0 Acknowledgements

We would like to thank our project supervisor, Dr. Peter Rodgers, University of Kent, for his guidance throughout the project.

Our thanks also go to John Crawford of the University of Kent for sharing his expertise in the field of networks and communications with us.

References


[2] http://www.keynote.co.uk.chain.kent.ac.uk/kn2k1/CnIsapi.dll
?uni=38134&usr=13204&srv=04&alias=kn2k1&uni=120515
5978&fld=K&noLog=1&NoInWorkset=1&key=1810


