CLOUD PLATFORM FOR THE MANAGEMENT OF MOBILE AD-HOC NETWORKS

Submitted by:

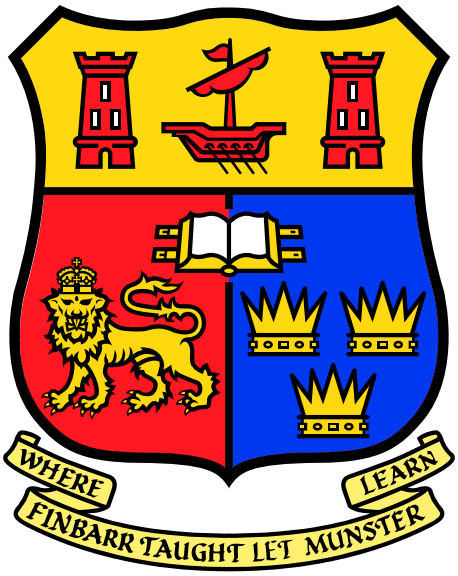
YEJIA YANG

Supervisor:

DR. DAN GRIGORAS

Second Reader:

DR. AHMED ZAHRAN



MSc in Computing Science

School of Computer Science & Information Technology
University College, Cork

September 1, 2022

Abstract

With the rapid growth of mobile devices, mobile ad-hoc networks(MANET) have become very popular. One of the characteristics of MANET is that nodes can communicate with each other in a MANET without infrastructure support, there is no access point(AP), each node can act as a router and a host, and nodes are at the same level. Nodes can communicate with each other by forming a multi-hop route. It is important to construct networks quickly in an emergency. However in a traditional MANET, due to the mobility of the nodes and the scarcity of battery resources, data transmission is likely to be interrupted by the constant changes in topology. Managing mobile ad hoc networks is a difficult task. While managing MANET, we should ensure the characteristics of the mobile ad-hoc network. In addition, the cloud can provide reliability and robustness for traditional MANET. The focus of this paper is on the implementation of the MANET model using Bluetooth technology and the design of two cloud services to ensure the reliability and timeliness of communication as well as the timely update of the routing of each node using depth-first and breadth-first algorithms.

Declaration

I confirm that, except where indicated through the proper use of citations and references, this is my original work and that I have not submitted it for any other course or degree.

Signed: _____

Yejia Yang
September 1, 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Mobile ad-hoc network(MANET)

1.1.1 Wireless Networks

Wireless networks can be broadly categorised into infrastructure-based and infrastructure-less wireless networks (ad hoc wireless networks)[Basagni et al. 2004]. The former networks rely on the access point(AP). In short, it is a wireless switch for terminal users to enter the Internet. This network is used in shopping malls, airports, homes and hospitals, almost everywhere. In terms of another kind of wireless network that does not rely on fixed infrastructure, we name it an ad-hoc wireless network. An ad-hoc mode can be used to connect wireless devices directly together without the need for a wireless AP.

1.1.2 Mobile Ad-Hoc Network

A mobile ad-hoc network is an autonomous collection of mobile devices (laptops, smartphones, sensors, etc.) that communicate with each other over wireless links and cooperate in a distributed manner to provide the necessary network functionality without a fixed infrastructure[Hoebeke et al. 2004]. A mobile ad-hoc network or MANET is a distributed network without infrastructure. Each node can act as a router, and communication between nodes is coordinated through network protocols and algorithms so that the entire network does not become inoperable due to the collapse of one node. Nodes can access other's radio range, and then they can communicate directly and are responsible for dynamically discovering each other[Hoebeke et al. 2004].

For communication between nodes that are not directly connected, the intermediate nodes act as routers and are responsible for forwarding Data. There can be multiple intermediate nodes, and finally, the Data is sent to the destination. However, mobile devices often have some common problems, such as battery energy-constrained limited memory. Moreover, the dynamic topology is unpredictable because mobile devices can freely join and leave a MANET. Although the dynamic network topology is one of the

characteristics, there are other significant characteristics, such as multi-hop routing, self-organisation and self-administration, etc.

1.1.3 MANET Applications and Scenarios

With the ever-growing portable devices and the progress of wireless communication, this development does not depend on the population. MANET is becoming increasingly important nowadays because of its wide application. The flexibility, mobility resilience and independence of fixed infrastructure result in application diversity. The most typical application is the military, and the military battlefield is also the origin of MANET development. Modern war is a scientific and technological war. Therefore, military equipment will include some network equipment. The front-line soldiers, vehicles and command headquarters form an information network, which can ensure communication requirements even in high-speed movement.

In another scenario, a group of young children go to kindergarten without their parent's company. Parents want to know their kids are safe and monitor their activity, this group of kids also can communicate with other children without infrastructure support[Mori et al. 2011], so wearable devices like AirTag can help parents know their kids' location at any time. In the commercial sector, MANET is very important to form a transmission network to perform emergency rescue tasks quickly. Such as flood and earthquake disasters, the AP may have been damaged, and it is impossible to repair infrastructure or install networking hardware. As a result, a group of medical teams has to rush to the disaster area to help victims. In addition to carrying necessary medical equipment, you can also use laptops to communicate effectively with the hospital remotely. However, these characteristics also bring some challenges, such as limited security, interference and lower reliability, the same limitations traditional wireless networks face.

Typical communication applications cannot be used in crowded network places, such as zoos, and amusement parks. In a congested network, normal communication applications cannot be used. Instead, family or friends can set up group chats to send messages to each other to send their location in an Internet-less environment so that even if they get separated in a crowd, they can quickly meet up with friends and family members.

1.1.4 MANET Challenge

In early research on MANETs, the main focus was on issues such as multi-hop routing and wireless channel access, and a friendly environment was envisaged. However, the security of the communication channels between nodes has become a primary concern due to the potentially threatening nature of the real environment. Although network security has been an active research topic in wireless networks, for a unique network model such as MANET, network security design remains challenging. The ultimate goal of MANET security solutions is to provide security services such as authentication,

confidentiality, integrity, anonymity and availability to mobile users[Yang et al. 2004].

In contrast to networks with dedicated routers, each node in a MANET not only can act as a router but also is a host and forward packets to other peers, allowing both legitimate and illegitimate users to access the communication channel without clear traffic monitoring and access control mechanisms, thus, blurring the boundary between the internal and external networks[Bakshi, Sharma, and Mishra 2013]. Secondly, with the existing Ad Hoc On-Demand Distance Vector (AODV) routing protocol, malicious users can deliberately violate the protocol specification by forwarding Data to other nodes, making it clear that the optimal path does not exist under malicious attacks, or intercepting Data, modifying Data, discarding Data.

Although these insecurities are also present in other wireless networks, cryptographic research covers the application, transport, network, link, and physical layers. Security is designed for each of these four protocol layers in MANET. For each mobile device, security protocols prevent potential threats while reducing device performance, and there is an elasticity in the trade-off between performance and security.

Despite the many challenges, mobile ad-hoc networks still show some advantages and have many application scenarios. First, the network model is significantly suitable for places without fixed infrastructure or unreliable and untrusted infrastructure, and the network can be deployed rapidly. Secondly, the devices in a MANET also can access the internet through cellular and Wi-Fi. Finally, the hybrid network mode makes up for some defects of MANET. Consequently, because the MANET model has the characteristics of flexibility, lack of infrastructure, easy deployment, automatic configuration and low cost, the research field is more and more extensive. Moreover, MANET with cloud support will be more and more important in future work.

1.2 Cloud computing

1.2.1 Definition

In Google's documentation, cloud computing is described as the provisioning of computing resources as an on-demand service over the Internet. Using Cloud service can eliminate the need for companies to purchase, provision or manage resources themselves, and they only pay for the resources they use[google 2007]. Cloud computing is a subset of distributed computing that consists of a group of loosely coupled computers that form a super-virtual computer, often used to perform large tasks by breaking down vast amounts of Data into countless smaller programs through a network "cloud".

1.2.2 Cloud Computing Service

Cloud computing service refers to the unified management and scheduling of a large number of computing resources connected through the network to form a pool of computing resources to serve users on demand. There are four types of computing services: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service

(SaaS) and Data storage as a Service (DaaS)[Dillon, Wu, and Chang 2010]. Cloud computing compares to traditional IT, where companies install and manage applications and maintain their own local Data centres by purchasing hardware, system software, and development tools. Instead, the cloud service provider owns, manages, and maintains these assets in cloud computing, while the customer only pays on demand. In other words, the resources and services on the cloud used by the user are unlimited. Therefore, the main advantage of any "as-a-service" solution is economics: for enterprises, choosing a cloud provider is the most economical and stable way to manage and maintain their own business as business volumes increase, without the need to purchase expensive private server equipment, etc. In addition, one of the benefits of the cloud is high flexibility that allows users to access cloud services anytime, anywhere, with an Internet connection and to scale up and down services as needed. Another benefit is security, where the cloud provider's security team is the leading expert in the field and provides depth and breadth of security mechanisms.

1.2.3 Cloud Computing Challenge

However, there are some limitations and challenges to using cloud services.

- The user's control of Data and technology is less flexible. For example, for IaaS cloud services, users cannot control the infrastructure layer; for Pass cloud services, users cannot control the infrastructure layer and platform software layer; in the face of SaaS cloud services, users lose control of the infrastructure, platform software and application software layer[Dillon, Wu, and Chang 2010].
- The second disadvantage is the increase in the number of links that may lead to data leakage. Cloud, disaster recovery centres, offline backup media, networks, cloud terminals, accounts and passwords, etc., all may be called information leakage points. However, compared with traditional IT systems, such hidden risks as storage equipment damage, fire in the server room, earthquake, flood, and other natural or artificial disasters will be reduced[Dillon, Wu, and Chang 2010].
- Highly dependent on the availability of the network. In an unstable network environment, the user experience is poor, but this is not the problem of cloud computing itself. Furthermore, with the network becoming increasingly widespread, in some cities with full wireless WiFi coverage, the network will no longer be a problem.
- Cloud consumers must consider the trade-offs between computing, communications and integration[Dillon, Wu, and Chang 2010]. While migrating to the cloud does reduce infrastructure costs, it raises the cost of Data communications. The cost of using cloud resources and Data integration can be high, there are proprietary protocols and interfaces between different clouds, and consumers may consider a specific cloud provider. In addition, splitting and merging Data for cloud services not only adds additional financial costs but can also severely impact system performance[Dillon, Wu, and Chang 2010].

1.3 Bluetooth Wireless Technology

The development of the Bluetooth industry standard began at the end of 1998. The companies like Ericsson, IBM, Intel, Nokia, and Toshiba formed the development team, and their purpose was to promote a worldwide solution for short-range wireless communication operating in the unlicensed 2.4 GHz ISM (industrial, scientific, medical) band[Bisdikian 2001].

1.3.1 Bluetooth

The Bluetooth wireless technology is a specification for short-range(generally within 10m), low-cost, low-power and small form-factor that enables user-friendly connectivity between portable and mobile personal devices. In addition, these devices can connect to the Internet via Bluetooth[Bisdikian 2001]. But now, a new generation of Bluetooth has a more comprehensive radio range of up to 100 meters. Moreover, Bluetooth provides a special connection between fixed and mobile devices, and one Bluetooth device can connect to serval devices. Another critical project for Bluetooth wireless technology is to enable ad hoc connections between individual devices, allowing individuals to form collaborative groups, for example, during meetings, to exchange data without relying on the infrastructure to support their communications.[Bisdikian 2001].

1.3.2 Ad-hoc radio connectivity

In the an-hoc systems, there is no difference between radio units; that is, there are no distinctive base stations or terminals and ad-hoc connectivity is based on peer communication[Haartsen 2000]. There is no fixed infrastructure to maintain and support the connectivity within mobile devices. Thereforem, in Bluetooth, a large number of ad-hoc connections are likely to coexist in the same radio range without any mutual coordination. According to the previous ad-hoc network scenarios, the ad-hoc model can provide a single network which the devices within the range can find. However, in Bluetooth, each device has its radio range; that is, the device has an independent network, and the network can overlap in dense areas.

There are two forms of Bluetooth network topologies. Here the term “scatternet” and “piconets” is proposed. In terms of Bluetooth Piconet, a piconet is a collection of Bluetooth devices that can communicate with each other and is formed in an ad hoc manner without any infrastructure assistance, and all devices are at the same level and have the same permissions[Bisdikian 2001]. In terms of Bluetooth Scatternet, a scatter ad-hoc environment consists of multiple piconets, because the maximum number of Bluetooth connections is only seven(total number is eight), and there are a limited number of devices in each piconet. In addition, the Bluetooth standard allows the same device to have multiple roles, a node can be a master node in a piconet and a slave node in one or more other piconets, allowing the connection of multiple piconets, and nodes with multiple roles will act as gateways to neighbouring piconets. This interconnection allows devices that are not directly connected and may not be in each

other's transmission range to communicate. This communication via intermediate nodes has the limitation that a node can only be active in one piconet at a time, and as each switchover incurs costs in terms of switching delays, scheduling, resynchronisation, etc., this paper will propose a multi-connection management approach to solve this problem.

1.3.3 Bluetooth Challenge

When it comes to the challenges facing the development of Bluetooth technology, the first concern is security. Given that the network is self-organising, there is no access point; there is no centralised mechanism for security management like the WAN, where MAC address filtering and other security mechanisms are used to prevent malicious access[Bouhenguel, Mahgoub, and Ilyas 2008]. If Bluetooth only connects a wireless keyboard, mouse or earphone to a computer, then security may not be as important. However, in the two decades of Bluetooth development, Bluetooth devices such as wearables, smart homes and mobile devices have had to implement security measures to protect user information. The increasing benefits and conveniences come with several risks and vulnerabilities, for example, transmitted Data becomes vulnerable to attacks on Bluetooth networks and private information such as contact addresses, emails or names stored on mobile phones or Bluetooth watches are compromised.

Although pins are sent to both parties to establish identity when Bluetooth devices are paired with each other, there is a man-in-the-middle attack, which in cyber security means that when user A sends a message to user B, an intruder forwards the message so that both parties think that the other sent it, and the intruder can either eavesdrop or tamper with the message. So in Bluetooth pairing, an intruder can forge Pin passwords to pair with both parties. There are also issues such as denial of service attacks. There are a number of ways to deal with Bluetooth vulnerabilities.

1. Turning off Bluetooth when not needed reduces the risk of potential attacks.
2. Bluetooth devices should have long random alphanumeric PIN codes to make them more difficult for intruders to access.
3. All Bluetooth devices in the network or piconet should verify with each other that they are legitimate devices and be authenticated.

Another issue is that Bluetooth operates in the crowded 2.4 GHz to 2.4835 GHz ISM band, which is often used by other technologies such as the IEEE 802.11b/g wireless LAN standard, making Bluetooth vulnerable to interference[Bouhenguel, Mahgoub, and Ilyas 2008]. Therefore, Bluetooth uses frequency hopping algorithms to reduce the interference it receives when communicating.

1.4 Analysis Of The Project Topic

The project name is "**Cloud platform for the management of mobile ad-hoc networks**" it can be decomposed into two parts, one about cloud management, and the other is to realise an ad-hoc network. As we know, There are many kinds of research on MANET, and a significant proportion of these have been done on various types of simulators, mainly due to the MANET protocol being very cumbersome and challenging to implement. Therefore, there are differences between the simulated environment and situation and the actual results, as implementing the MANET network in the first step is necessary. Mobile ad-hoc network, the keyword is mobile, Android platform supports Bluetooth network stack, this function can realise exchange Data within Bluetooth devices through a wireless way. Within a radio range, mobile devices can establish channels to transfer any kind of Data with each other without infrastructure support.

From my point of view, in Bluetooth, the device communication method is similar to Java socket, where socket communication is a transmission mode based on the TCP/IP network layer. All devices need to allocate the IP address. In comparison, Bluetooth communication relies on the MAC address, the benefit is that there is no need to assign a MAC address, as the device already has a unique one.

The implementation of Bluetooth communication is not network-dependent. One device acts as the server and the other as the client for the connection, the exact connection steps I will describe in Chapter 3. Although the official Android documentation and the official demo briefly demonstrate how two Android devices communicate, in the context of the project, MANET's communication mode is not only a direct connection but also a multi-hop communication, which would be too limited if the Data transferred is only text. In this project, I will use Bluetooth to implement device discovery and multi-hop communication, extend the official demo and support text, voice and image transfer. In conclusion, using Bluetooth to implement a MANET network is feasible.

1.4.1 Challenge

With the above analysis, it seems somewhat ambiguous in terms of what the cloud can do to support the MANET. For traditional MANETs, there are many challenges to managing MANETs, as the mobility of mobile devices and the scarcity of battery resources lead to dynamic changes in the network structure. For example, if routing tables are not updated immediately, then Data transmission will be interrupted, and Data will be lost. Also, because topology frequently changes, tracking down a particular node becomes difficult, and the mobile device can quickly come out of or into the radio range of various other MANET.

Consider a scenario in which some devices in a MANET suddenly shut down due to battery dead and the communication link is broken, since the local routing table of the devices has not been updated in time, the communication between the devices will result in Data loss because the link does not exist (broken link).

In the second scenario, when the number of non-directly devices increases, it is slow to determine membership information and update the routing table by transmitting

messages between intermediate devices. As a result, it takes a lot of time to update the routing table between members once a device leaves the MANET. In some cases, the link interruption as a bridge can be considered a splitting of the MANET since the devices are not paired with other devices, even if they are in the same radio range.

In the last scenario, Data will likely be discarded because of the mobility of the devices in multi-hop communication. Also, when the number of hops increases, the message latency is higher, the possibility of losing Data is higher, and the reliability of the message is lower, which is a chain reaction.

In the real-world environment, the situation is variable, and in general, mobility is both an advantage and a disadvantage, in addition to battery resources being a common limitation. Managing devices in a mobile ad-hoc network without the support provided by the cloud is complicated, unreliable links lead to Data loss, the ever-changing network structure makes routing updates more challenging, and constantly updating routes greatly increases battery consumption, which is clearly not feasible.

1.4.2 Project Solution

To address the shortcomings of traditional MANETs, it is essential to take advantage of the cloud. First, flexible management makes this model more efficient and better applied in the real world. Each device must register with the cloud before using the cloud services. The already registered nodes can log in directly to get the device information and the latest routing information. The cloud can effectively manage device information, including the routing table and neighbour table of each registered device, especially in the case of more mobile devices. Therefore, it can effectively reduce the delay in routing updates. Secondly, MANET splitting and merging are based on devices' physical location and connectivity paths' availability. The cloud will also re-identify the MANET members while updating the routing information. If there is a communication link between two MANETs, i.e. there is an intermediate device as a bridge. The cloud will automatically merge the MANET information, including reassigning the owner ID, the number of members, the member list, etc.

In the same way, within a MANET, if some device communication links are broken, then the cloud will assume that the MANET has been splitting. On the other hand, there may be cases where some devices are leaving the current MANET during communication, and data may be dropped. Cloud services ensure the reliability of transmitted data, even if the communication link is interrupted due to device mobility or battery insufficiency.

The involvement of the cloud can be an excellent solution to the previously mentioned scenarios. Therefore, I have proposed two cloud services, which will be detailed in Chapter 3.

1. Using graph algorithms to form network topology to update routing tables in real-time for each mobile device in a faster and more accurate way. Even if some devices are not connected to the network, they can be updated by neighbouring connected devices, which greatly increases the speed of routing table updates,

and also when some devices lose their signal, the cloud can update the network structure in a short time, reducing the probability of routing table errors.

2. Use the cloud Database to store transmission Data. Although this does not require any logic to be used, it is very effective in multi-hop communication to ensure that messages are ready to be delivered to and returned from the destination.

In context, the term "communication" is mentioned several times, communication is an important part of the project, and in order to implement multi-hop communication, I have defined a communication protocol, the purpose of which is to standardise the transmission format, which includes fields for the message body, time, acknowledgement, routing path, and whether to upload to the network and distinguish the type of transmission to facilitate correct handling in the method of READ and WRITE. Peer-to-peer transmission messages do not need to carry information other than the message, so the communication protocol is defined to enable multi-hop communication. The following sections(3.4) will expand detailed information about the fields and their meanings of the fields.

Finally, because Bluetooth is used to implement mobile ad-hoc networks, the common functions of Bluetooth, such as scanning, pairing, unpairing, getting $BluetoothAdapter$ information and other methods, are encapsulated to improve the readability of the code and easy to debug errors(Refer to 3.4).

1.5 Project goal

This project focuses on how to implement a network model like MANET, and after some preliminary literature reading, the technology used was identified, namely the implementation of MANET using Bluetooth. Bluetooth technology was chosen for two reasons, the first being that Bluetooth is relatively mature and has been developed over a long period, and almost all mobile devices have Bluetooth modules. Even if the device does not have a BLE module, standalone Bluetooth modules are cheap. The second reason is that the Bluetooth connection does not require infrastructure support, which is perfectly suitable for realising the MANET.

In the Bluetooth-MANET model, the mobile device can leave or join a certain radio range anytime and turn off or on the Bluetooth settings. The intuitive operation makes the mobile device more flexible and allows the user to automatically access useful information such as member lists without being aware of it. On the contrary, if the user is to learn complex operations and become familiar with cumbersome functions, once the user experience is terrible, this model should hardly be useful and challenging to promote. According to the reasons analysed, this model allows the program to interact with the user in a user-friendly way and to be used well even without instructions, e.g. the user does not need to know how to get non-neighbouring nodes behind the program and does not need to know how to send messages in a multi-hop communication method.

In addition to implementing the Bluetooth-MANET model, the model is fragile without the support of the cloud, especially in terms of communication and routing. Bluetooth devices can only sense paired devices, i.e. directly connected devices. Non-directly connected devices need to send broadcast messages to each other after the communication channel has been established to update the routing table. It is clear that if there are not many devices in a MANET area, the routing table update delay is within estimable limits, and then updating the routing table in time is difficult due to the dynamic network structure. Besides, the dynamic network topology makes multi-hop communication unreliable without the support of cloud services; in other words, to ensure that messages are correctly transmitted to the destination, then both the transmission path and the intermediate nodes must be reliable and obviously challenging to achieve.

Finally, on the issue of routing in multi-hop communication, I suppose that it is similar to the RIP routing protocol, the route selected based on the number of hops is not necessarily the optimal route. In the absence of cloud involvement, devices send their neighbour information after establishing a communication channel with each other and update the complete routing table after multiple broadcasts (Refer to 3.3.2 for details). If devices can connect to the cloud, one of the cloud services will generate paths based on a depth-first algorithm of the graph Data structure(Refer to 3.2.1 for details), returning the latest routing information to those devices.

The Bluetooth devices used in this project consisted of five Bluetooth-enabled Android phones, and the cloud platform used is AWS Elastic Beanstalk and AWS' cloud Database-Relational Database Service (RDS). The framework of SpringBoot and MyBatis is used to complete the functionality of the cloud service, and the compilation tool for mobile is Android studio, the programming language I used is Java.

The ultimate goal of this project is to:

1. The device is regularly turned on in scanning mode to upload the latest connection Data.

2. Dynamic update of the network topology promptly based on the depth-first algorithm(DFS) of the graph Data structure and update of the routing table by using the breadth-first algorithm(BFS) simultaneously.

3. To achieve three communication modes, point-to-point communication, multi-hop communication and direct communication with the cloud.

4. To implement three types of communication Data, including text Data, image information and voice information.

5. To implement message reliability to avoid the possibility of message loss and high latency.

The project is evaluated as follows:

1. Evaluate the message latency time for three Data types (text, image, voice) from one to four hops, and additionally test the impact of connecting to cellular Data

and Wi-Fi to use cloud services on message latency over a four-hop communication distance, respectively.

2. First, five mobile devices are arbitrarily placed within a radio range, and the changes to the cloud Database routing table are observed. Afterwards, two devices are arbitrarily selected, placed in an area 20 metres away, and then monitored for updated Data from the cloud Database.

Managing mobile ad-hoc networks is an arduous task. While retaining the self-organizing characteristics of mobile networks, its cloud management provides reliability and robustness. To realise multi-hop communication, the communication protocol is defined in this project.

Chapter 2

Literature review

2.1 Related Work

When I reviewed similar studies and papers on cloud management MANET, there were few papers on cloud-managed MANETs, and a 2018 paper proposing a cloud-MANET-IoT framework for communication between smart devices[Alam and Benaida 2019]. There are four specific steps to implement this model:1. Form a MANET; 2. Access to the ad-hoc network; 3. Register the smart device in the MANET; 4. Register the MANET smart device in the cloud. Finally, the devices in the MANET membership can communicate. Smart device-to-smart device communication in the Internet of Things framework is a novel approach. The MANET model can be very effective in saving power and spectrum efficiency. The article's focus is that smart device users will discover devices through cloud services, using Markov chain models to analyse information such as device signals to minimise useful information in big data. Cloud services can also handle video, images, text and audio.

In most mobile devices, the operating system does not allow for creating Wi-Fi ad-hoc networks. Instead, the Android operating system provides a network sharing service, known as a hotspot, where a device can create a hotspot to which neighbouring devices can connect, which is not an ad-hoc network because the created hotspot requires infrastructure support. An ad-hoc network is an infrastructure-free network.

However, the theory proposed in this article is more inclined to devise discovery. The communication between devices does not involve cloud management MANET. In addition, I also consulted a 2014 article on cloud management of ad-hoc networks. This article describes in detail how the cloud manages MANET, and includes some operations and monitoring of MANET[Alshareef and Grigoras 2014]. The authors modified the system configuration to enable MANET mode, but the Wi-Fi mode and ad-hoc model can not run simultaneously, and not all Android phones can be enabled in ad-hoc mode. The authors have another article on achieving communication reliability in mobile ad-hoc networks. They proposed a model that uses cloud services to manage register, save, pause and resume sessions between MANET member nodes, thus saving energy.

The goal of this model implementation is to resume communication properly even after a communication interruption, and the authors designed checkpointing techniques to capture the session's progress and allow it to resume[Alshareef and Grigoras 2017]. Honestly, I tried to follow the procedure of that 2014 paper to implement the enable MANET mode, however, I checked a lot of information to modify the configuration file, and even after modifying the system file, the device is still in the regular network state. This motivated me to look for other solutions.

Generally, mobile devices have three off-grid communication tools: Bluetooth, Wi-Fi Ad-hoc mode and Wi-Fi Direct. Firstly, Bluetooth has a shorter transmission distance and slower data transfer rate. Secondly, Wi-Fi Ad-hoc can only be supported by particular models of mobile devices, which has limitations. And I tried to follow the steps of the 2014 paper on MANET cloud management to implement it, but after searching through the configuration files that needed to be modified, some devices do not have the paths mentioned in that paper. For this reason, not all devices can be enabled in MANET mode by modifying the configuration file. So after a week of trying, I decided to change direction.

Finally, Wi-Fi Direct is more user-friendly than Wi-Fi Ad-hoc mode, and almost all mobile devices can support this mode. A paper on developing a mobile Ad-hoc network using Wi-Fi Direct seems to fit my project requirement., Besides browsing the official Android documentation, Wi-Fi Direct can provide a peer-to-peer communication mode without the need for infrastructure. With one device as the Owner and all other devices as members, the device can connect to other devices anywhere and anytime, enhancing mobility. The paper focuses on establishing connections and forming Wi-Fi P2P networks based on Wi-Fi Direct, information exchange, and the use of Wi-Fi P2P to form a novel dynamic multi-hop ad-hoc network. However, when I implemented this model using Wi-Fi Direct, the switching of the Owner and members in P2P networks may cause unstable peer discovery as well as communication collisions, which can cause unreliability of multi-hop communication, and although I did not practice multi-hop communication, I observed that the P2P network structure is a star, and we know that the topology of a MANET is dynamic, it may be a star, a net or a chain. Besides topology, the number of members is also a bottleneck and should be controlled. It can be seen that Wi-Fi Direct does not meet some requirements.

To make the model unrestricted by device model, I noticed another article on using Bluetooth to implement ad-hoc networks in the final analysis and then jumped back to choosing Bluetooth technology. This article presents a Bluetrees-scatternet network model, in which two decentralized network formation algorithms are presented that take into account role and link constraints[Zaruba, Basagni, and Chlamtac 2001]. The first algorithm is based on the designation of a node 'blue-root', which initiates the construction of a 'blue-tree', i.e. the resulting decentralised network topology will be a tree that spans the entire network[Zaruba, Basagni, and Chlamtac 2001]. In this scheme, the number of roles assigned to a node is limited to two, i.e. a node can only be a master node, a slave node or a slave node in two different piconets or a master node in a piconet and a slave node in a neighbouring piconet. The second algorithm speeds

up the process of forming a decentralised network by selecting more than one tree to form the root of the tree and then merging the trees generated by each root[Zaruba, Basagni, and Chlamtac 2001]. The protocol is divided into two phases. In the first stage, a subset of the selected nodes is used as initialisation nodes to initiate the construction of subtrees, similar to the first algorithm. These two algorithms inspired me; in my project, the devices in a MANET constitute a graph data structure, and each vertex can build a tree structure, which is the "blue-root" model proposed by the author.

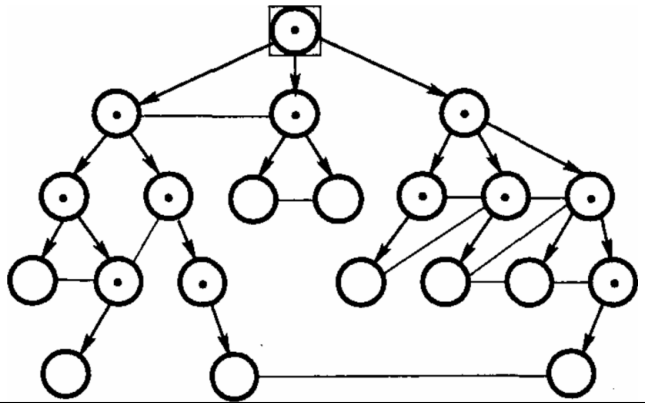


Figure 2.1: A rooted bluetree[Zaruba, Basagni, and Chlamtac 2001]

In terms of communication, in browsing the official Android documentation[2021], the official demos gave me good ideas, although, at this point, Bluetooth communication technology is also in point-to-point mode. In my case, Bluetooth devices can be one-to-one or many-to-many, which means that Bluetooth communication channels can be established and managed simultaneously, and there will be no communication conflicts within the maximum number of connected devices. Although low-power Bluetooth (BLE) can support multiple connections, mobile devices only support the classic Bluetooth mode. In this stage, almost all Bluetooth communication codes are one-to-one to establish the communication channel. In multi-hop mode, the message transmission process is constantly disconnected and reconnected, which not only increases the possibility of data loss, but the delay time will also increase with the number of hops. Bluetooth communication is similar to Java socket, after referring to Java network programming multi-connection management documentation based on a similar implementation model, the Bluetooth multi-connection problem is quickly solved. For Bluetooth, one device can be paired with seven devices, and member devices can be paired with other devices. In short, these devices are directly or indirectly connected to the Owner. From a topology point of view, all of the above can be formed in a Bluetooth ad-hoc network. What's more, the connection between Bluetooth devices does not require infrastructure support. Therefore, it makes sense to implement a mobile ad-hoc network based on Bluetooth.

2.2 Brief Description Of The Project

I have to mention that Bluetooth is very different from other wireless networks, such as Wi-Fi or hotspots, in that devices only need to input a password to join the network. In Bluetooth-MANET, when a device wants to join the MANET, it does not display a list of available networks but rather displays nearby devices based on the characteristics of Bluetooth. When a device wants to leave the current MANET, unpairing or moving away from the MANET's radio range is recognised as a leaving operation. Obviously, the action of joining or leaving a MANET is related to the physical distance and the pairing status. In terms of MANETs, each device has a certain radio range named piconet, i.e. each device can be considered as a MANET with only one member, so the joining and leaving of devices can be equated to the merging and splitting of MANETs as well. It is easy to understand based on the characteristics of Bluetooth. Regarding the project itself, I have implemented and explained it in two parts, and one is the operations and functionalities between devices, such as enabling Bluetooth, pairing operations, unpairing operations, etc. The other is how the MANET is managed in the cloud, which includes the creation, merging and splitting of MANET and guaranteeing the reliability of messages.

Simply, the process of implementing a Bluetooth self-organising network consists of the following steps:

1. The devices start scanning each other, and when the scanning is finished, the neighbouring devices that are paired and in the scan list are uploaded to the cloud.

2. The cloud forms a topology graph based on the neighbour table and then generates all paths from the topology.

3. When the devices get all member information in the MANET, the communication channels are established, and then devices can send messages.

4. Transmission reliability. Messages can be uploaded as a backup during transmission as long as any nodes are connected to the cloud, and the backup message is resent as long as the message is lost.

Chapter 3

System Design

The project code is divided into two parts, the mobile side and the server side; the development tool used for mobile is Android Studio, and the HTTP tool for interacting with the cloud is rxhttp. The development tool used for the server side is IntelliJ idea, and the Java framework I used is SpringBoot+MyBatis.

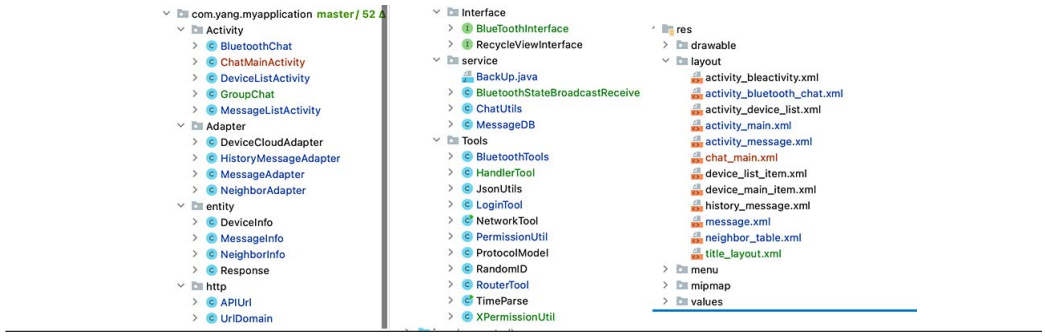


Figure 3.1: Android Program Structure

3.1 System Architecture

The Bluetooth-MANET-Cloud system is composed of two components: the public cloud and the mobile ad-hoc network. There are two limitations of the Bluetooth-MANET model. The first one is that the member devices cannot access the non-neighbour device information between them in the case that the communication channels are not established with each other or without cloud support. The other is the limitation of mobile devices themselves that resources are scarce, especially memory and battery resources, and complex computations are not suitable for processing through mobile devices. In contrast, resources in the cloud are unlimited and allocated on demand, which is ideal for complex computation and logic processing.

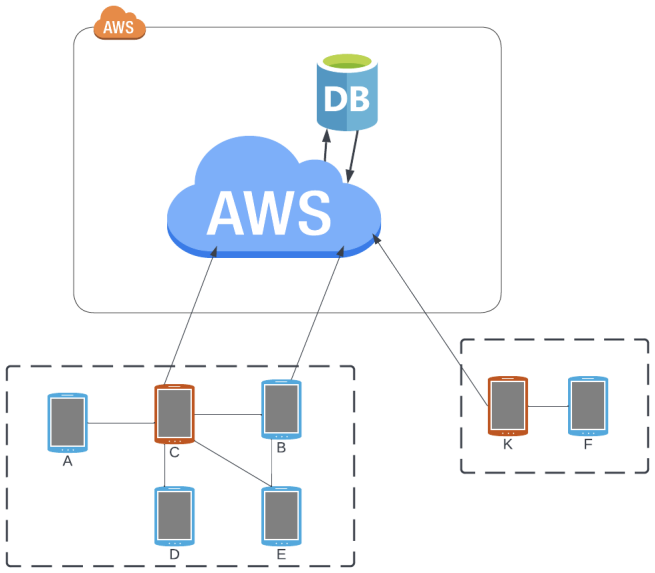


Figure 3.2: The Bluetooth-MANET-Cloud model

The model will be described in detail in three parts:

The first part is the cloud service, which contains two services: network topology formation and traversal communication links and storage of unread information. The focus will be on the former. For the whole project, the topology is dynamic. Therefore, it makes sense to update and form the graph structure in time, whether for routing or selecting a reliable communication link for each device.

The second part is about the operations of Bluetooth mobile devices, although Bluetooth also has a radio range, the devices are independent individuals, and communication is only possible after successful interconnection and pairing. The device can request a connection without access to the network. In the absence of cloud support, peer-to-peer mode mobile devices are unable to obtain more than two-hop devices in the same MANET members unless communication channels are established(Refer to 3.3.2).

The third part is the communication protocol. In order to guarantee the reliability of the multi-hop communication of MANET, even devices that are not directly connected or lose the connection can communicate with each other, the service defines a communication object that the communication thread can catch the keyword to identify which type of the message and how to handle the different message.

Finally, the design of database tables is as follows (including cloud database and Android local database):

- Device Info Table: UUID, Username, MAC, Password, Status.

- MANET Table: MANET_UUID, Owner UUID.
- MANET Membership Table: MANET_UUID, Member UUID.
- Message Table: Message UUID, Message body, Send Time, Read Time, destination, Source, DataType, etc.
- Neighbour Table: Source Name, Source MAC, destination Name, destination MAC, Timestamp.
- Router Table: Source, destination, Path, Hop.
- Neighbour Info Table(local DB): Neighbour MAC, Neighbour Name, Hop, Path, Connection Status, etc.

3.2 Cloud Services

3.2.1 Bluetooth Graph Topology Formation

In this part, I will detail the algorithms and protocols that form the topology of the Bluetooth network. Only by knowing the network structure can we analyse the message transmission path. Taking into account the mobility of the devices, the topology also changes depending on the location of the device and the strength of the signal. Generally, the topology of the mobile ad-hoc network changes at any time, and the common structures are mesh, chain and star. However, regardless of which structure, the formation of topology is the same way.

Before forming the network structure, the device needs to enable Bluetooth and scan the available devices, then upload the information of neighbour nodes after the scanning, such as MAC address, device name, signal strength, etc. In fact, Android provides a function called “BluetoothAdapter.getBondedDevices”, which can directly get the neighbouring devices that have been paired already. However, considering the actual situation, the paired devices may not be in the radio range, or the Bluetooth setting is turned off, so Android’s function is inaccurate. Scanning available devices periodically can avoid inaccurate information caused by the above situations. In addition, besides the periodic scanning operation between mobile devices, the routing table in the cloud will be updated regularly as well according to the latest timestamp. It is the necessary condition for the cloud to determine whether links are reliable or not. In the Bluetooth-MANET-Cloud model, each mobile device scans every five minutes, and the routing table updates every two minutes. The steps of the whole progress are below:

- Operation and steps on the device side:

1. Setting up Bluetooth and start to scan.
2. Finding devices that are either paired or available in the local area.

3. Device request pairing or Connecting devices.
4. Filter paired and active devices.
5. Upload the information to the cloud.

- Operation and steps on the cloud side:

1. Query all neighbour information in the neighbour table.
2. Generate graph structure and recursively traverse all nodes.

In mobile devices, not all devices are able to access the cloud through Wi-Fi or cellular data. Even if the devices do not have the ability to use the API, as long as any neighbour devices take advantage of the cloud, the information in the database is still reliable. If there is the worst case, that is, all devices cannot directly communicate with the cloud, another advantage of the Bluetooth-MANET-Cloud model is that after the devices have established communication channels (threads), devices will periodically send their neighbour information to update the local routing table.

It's noticeable that mobile devices do not need to do complicated work, they just scan regularly and find active neighbouring devices. The benefit of having complex work done in the cloud is that it saves resources on the device itself, especially battery and memory.

To explain the topology more clearly, each node to any destination can be considered a tree structure, and topology can be regarded as an undirected graph. Trees and graphs are nonlinear data structures, and a Graph is more abstract and complex than a tree. It can be considered that a tree is the basis of the graph, and the tree is a graph in a simpler sense. Generally, a binary tree has two branches, left and right; the root has children, and the children have children and so on. In the Bluetooth MANET model, the graph can be considered several trees that have more than two children nodes. Based on the same logic, iterate through all the paths. These paths are routes for each node to all members within the same MANET. In the project, the data structure is defined as $TreeNode(< name, String >, < children, List < TreeNode >>)$, in this class, a variable of type list is used to represent the child nodes, therefore the ADD function(see Algorithm 3.1) contains three parameters as input: $addNode(< name, String >, < node, TreeNode >, < children, List < TreeNode >>)$

Algorithm 3.1 Add Children

```
If node.children is null:
    Exit new TreeNode(name,children)
EndIf
For _node in node.children:
    If _node.name equals name:
        Add children in _node
        Exit node
EndFor
For _node in node.children:
    Recursive call addNode(name,_node,children)
EndFor
```

Moreover, whether the topology is complex or simple, the graph data structure will generate all the tree structures of each node. The path, or the shortest path, can be found from the root node to any child node based on the depth-first traversal(DFS) algorithm(see Algorithm 3.2), there are three parameters as inputs to the algorithm: $TraversalPath(< node, TreeNode >, < list, List >, < target, String >)$.

Algorithm 3.2 Traversal Path(DFS)

```
Declare static global variables to store paths:
Initialise: List<List> allPath and List currentPath
If node.children = null or node.name equals destination:
    If node.name equals destination:
        Add list to allPath and currentPath
        Exit
    EndIf
EndIf
For _node in node.children:
    Add _node.name to list
    Recursive call TraversalPath(temp, list, destination)
    Remove temp.name from list
EndFor
```

Here a simple topology is defined below:

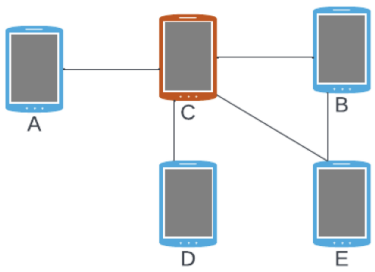


Figure 3.3: Topology

In this undirected graph, we can get a list of sets:$[A\{C\}, B\{E, C\}, C\{A, B, D, E\}, D\{C\}, E\{B, E\}]$.

For instance, starting from node C as the root, recursively all nodes, as shown in the figure, the dashed line indicates the nodes that have been visited, HashSet in the DFS algorithm to record the nodes that have been visited, ensuring no repeated traversals or endless recursions and consider the actual situation, the more edges, the more nodes are passed and the more hops. The distance from node C to any node is only one hop. Also, each recursive traversal of an edge is a path, i.e., $C \rightarrow A : [C, A]$,$C \rightarrow B : [C, B]$,$C \rightarrow D : [C, D]$,$C \rightarrow E : [C, E]$. The list of$[C, B, E]$ will be discarded because the edge from B to E has been visited.

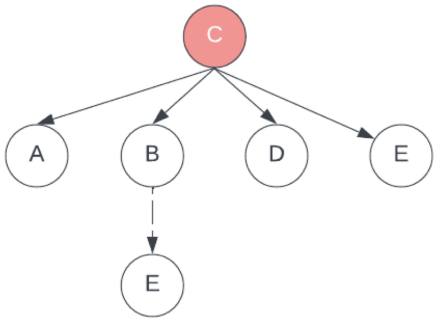


Figure 3.4: Node C Tree Structure

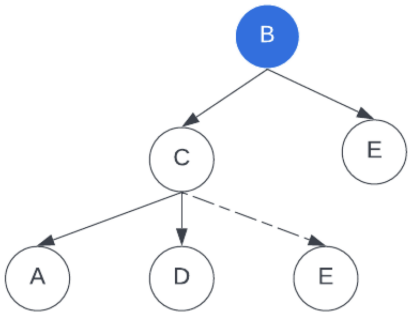


Figure 3.5: Node B Tree Structure

Similarly, when node B is the root, the structure of the tree is three-layer which indicates the maximum hops is 2. The paths from node C to other nodes are, $B \rightarrow C$: $[B, C]$, $B \rightarrow E : [B, E]$, $B \rightarrow A : [B, C, A]$, $B \rightarrow D : [B, C, D]$.

Consequently, the algorithm generates all routes between nodes.

$$\begin{bmatrix} -1 & A & B & C & D & E & \dots \\ A & -1 & 2 & 1 & 2 & 2 & \dots \\ B & 2 & -1 & 1 & 2 & 1 & \dots \\ C & 1 & 1 & -1 & 1 & 1 & \dots \\ D & 2 & 2 & 1 & -1 & 2 & \dots \\ E & 2 & 1 & 1 & 2 & -1 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & -1 \end{bmatrix}$$

The cloud can form a graph fastly, however, each device that uploads the latest neighbour node might not occur simultaneously, checking the latest data of the neighbour table regularly will bring additional overhead by frequent database operations. Fortunately, limited by the maximum number of Bluetooth connections, even if the chain structure, the depth-first traversal efficiency is considerable and these operations are implemented by cloud services, and the device itself will not increase the cost or energy consumption.

3.2.2 Reliable Communication

Generally, messages might be lost in the process of transmission due to multiple-hop communication will pass through more than one intermediate node, and the link may be broken for some reasons, such as battery dead, turning off Bluetooth settings and so on, because of this, the cloud provides reliability during the process of message transmission. Android provides a set of message processing mechanisms called Handler. It can be used to send and process messages. When the source device sends a message to the destination device, the WRITE method is used for listening to the message from a

sender, as a sender, the device needs to implement two procedures in sequence. Firstly, use the API to upload the message to the cloud, wait for the return message, return it successfully, and then mark the UPLOAD field as 1; otherwise is 0. Secondly, store the message in the local database. However, two situations need to consider:

- The source device can connect to the cloud. The device directly uploads the message to the cloud.
- The source device is unable to connect to the cloud. Any intermediate node can upload messages to the cloud as long as it connects to the Internet.

As a receiver, the steps of processing the message are the same as those of the sender and use the READ method to process received messages, and then the receiver returns an ACK message to the source device. The roles of the receiver and the sender are interchangeable, then follow the sender's steps.

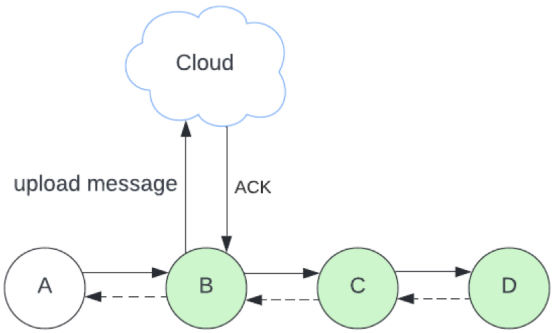


Figure 3.6: Multi-hop Transmission

As shown in the figure above: If node B successfully upload the message, in spite of node C still being able to connect to the cloud, in order to avoid wasting bandwidth by repeatedly uploading messages, according to the UPLOAD field, node C just send the message directly. After the message is sent to node D's destination, the destination returns the feedback to the source device. The ACK message also should update to the cloud to ensure the timeliness of the message.

For the cloud, the cloud service regularly checks for undelivered or non-returned messages and sends them to the receiver and sender separately. Peer-to-peer communication way dramatically reduces latency in multi-hop communication, and messages are automatically downloaded as soon as they are connected to the Internet. Perhaps an unread message identified in the cloud has already been delivered, and the device's local database ignores the message to avoid duplicate acceptance, which can cause message inaccuracy.

It is also necessary for the device to periodically check the local database and then send unread messages to the cloud, as some messages are not uploaded because all intermediate devices, sender and receiver, are not connected to the network, which is the worst-case scenario. There is also a physical break in the link during resending, and only when the device is connected to the network again can it upload unread messages and download read messages to update the local database and the cloud database.

3.2.3 Scheduled Task

Technically, although this is also a cloud service, this service can stand alone and the timing service is used to check the routing table and neighbour table for expired information. For instance, a device in the MANET is offline or has left the current MANET, but the device is not connected to the network, resulting in the neighbour table information still existing. The network topology is generated from the neighbour table, so inaccurate information will produce inaccurate topology as well as incorrect routing information.

Springboot framework provides a useful annotation @EnableScheduling, that is, to enable scheduled task support, annotation @Scheduled to declare a task, including $cron$, $fixDelay$, $fixRate$ and other types. In the program, the task is set to execute once every five minutes and the system will check the most recent update time of each routing information, the time difference is greater than 300 seconds will be considered expired information and deleted.

3.3 Operations of MANET

Due to the nature of Bluetooth, the creation, splitting and merging of Bluetooth MANETs is not actively triggered by devices. As mentioned earlier, compared to a wireless network like Wi-Fi, although both Bluetooth and Wi-Fi belong to the wireless communication network standard, Bluetooth belongs to the WPAN wireless personal area network, i.e. point to point. In contrast, Wi-Fi belongs to the WLAN wireless local area network, a network mode where multiple terminals transmit simultaneously. When multiple Bluetooth devices are connected point to point, the range that these Bluetooth devices can scan can be considered a signal range. Therefore, it is very understandable that when a device is placed within this signal area, it can be considered as joining, and vice versa. Similarly, devices that leave the previous network are automatically considered by the cloud to join another or new MANET at the end of the following scan.

3.3.1 Creating a MANET

Within a specific range, the devices enable Bluetooth settings, and the devices can scan each other, connect and pair with each other. For this area, the scannable paired devices belong to the same network, both logically and physically.

- If some devices can connect to the cloud:

The device information is uploaded directly to the cloud. Still, the topology is not complete until the devices update their local routing tables by transmitting to each other over socket channels and then uploading to the cloud. Therefore, the cloud MANET table should update more than once to ensure its members are contained entirely.

- If all devices are unable to connect to the cloud:

The devices cannot upload neighbour information, so the cloud cannot form a topology. But within this physical area, the devices transmit neighbour messages through the socket channel and update the local routing table; only the cloud database does not have this MANET information, which does not affect the communication of member devices.

- If all devices are able to connect to the cloud:

Device information is uploaded directly to the cloud for fast topology and routing table generation, and the cloud MANET table inserts the new MANET information without waiting for socket transfers between devices to update local routing tables.

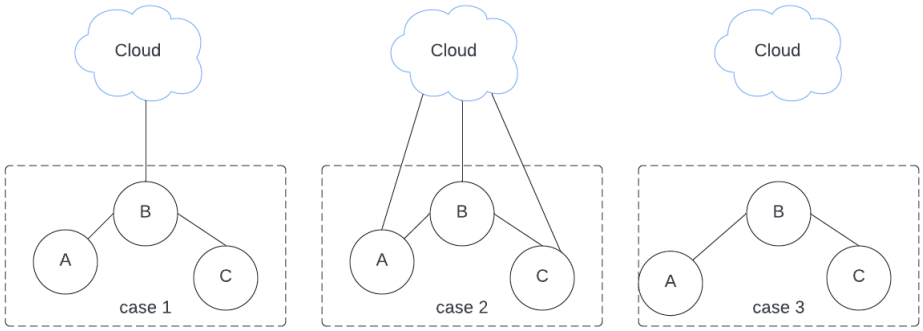


Figure 3.7: Create A MANET

The above scenario describes a situation where devices are all clustered in the same area, and each device's radio range can cover each other so that a MANET can be formed. However, another scenario to consider is where a device cannot be within scanning range of all devices and in the Bluetooth-MANET model, it is still considered to be the same device as the other devices in a MANET, for example, in an undirected graph, a vertex is far away from other vertices, but there are still edges at that vertex. Devices are considered to be in the same MANET if they are connected. Nevertheless, the further the physical distance and the more hops, the higher the latency will be.

In particular, MANET-related information, such as MANET UUID, number of members, owner ID, etc., are automatically created by the cloud service. The UUID may

change as the network structure changes. In summary, creating a MANET forms a network structure, the devices do not necessarily access the cloud, even if the cloud database does not store the information of the MANET which still does not affect the communication between devices.

3.3.2 Creating a MANET without Cloud Support

This part is an extended description of case 2 in 3.3.1. The worst-case scenario is that none of the devices paired with each other is connected to the internet and cannot use cloud services. In the Bluetooth-MANET model, communication threads are automatically created when the pairing of devices is complete so that there are available communication links between all devices. Using Figure 3.2 as an example, for device C, information from neighbouring nodes A, B and D can be broadcast by device C to A, B and D, and then device A updates its routing table: $[A,C]$ $[A,C,D]$ $[A,C,B]$, similarly the routing table for B is updated as:$[B,C]$ $[B,E]$ $[B,C,D]$ $[B,C,A]$, and the routing table for device D is $[D,C]$ $[D,C,A]$ $[D,C,B]$. This is the result of the first round of broadcast updates. For device B, its neighbour nodes are C and E. Device B sends messages to C and E. Device C only need to update one userful information, that is $[C,B,E]$ and device E updates the routing table: $[E,B]$ $[E,B,C]$ $[E,B,C]$ $[E,B,C,A]$ $[E,B,C,D]$. At this stage, device C acquires new routing information, i.e. device E. Then device C broadcasts the message again to its neighbour nodes. Eventually, at least three broadcasts must be made before the routing information for all devices is fully updated.

The order in which the communication links are established is random and broadcast constantly between devices. Updating routes is fast when there are few devices in the area, but in a MANET where the devices are crowded, updating the routing table by broadcast will be slower and when the update is complete, a device joins or leaves this radio range, then other devices will need to update the routing table again. To avoid the broadcast frequency being too high and affecting the performance of the device, as well as to save energy in the device, the broadcast frequency can be changed to once every five minutes. In essence, without the support of cloud services, updating the routing table via broadcast is not only slow but also challenging to get the latest routing information in time.

3.3.3 Joining a MANET

Devices can join a Bluetooth MANET regardless of whether the devices are connected to the cloud. The process of joining an existing network has three steps.

1. Enable Bluetooth and start to scan.
2. Discover the device, and click to pair it.
3. Wait for the destination device to accept the pairing request.

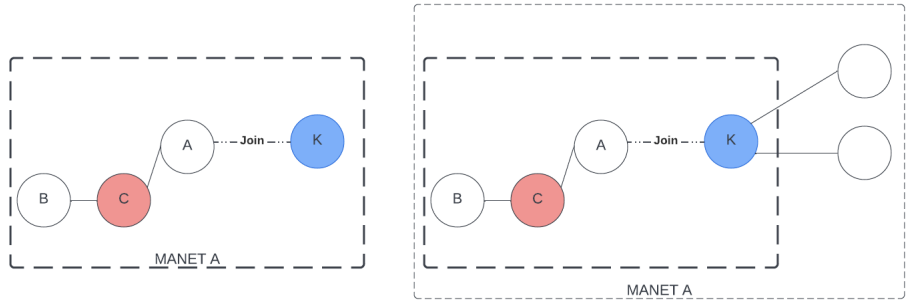


Figure 3.8: Joining a MANET

Physically, the newly joined device is a member of the MANET. For non-directly connected devices, devices are unaware of the existence of this new member until the information about this new member is uploaded to the cloud or broadcast after all communication channels are established. Therefore, three cases need to be taken into account to update the topology in time.

1. If the device and its neighbour connect to the cloud, both devices can upload the pairing success information to the cloud, and each time the cloud gets the information, it can be used to analyse the membership in a MANET and update the routes.

2. Suppose one of the devices is connected to the cloud. In that case, the pairing information can also be appropriately uploaded, and the cloud regenerates the routing and classifies the members of the MANET based on the latest data.

3. Suppose neither device is connected to the network. In that case, the cloud still maintains the previous topology, the neighbour table of two devices will only show one-hop nodes until the socket channel is established, and all devices are in interconnected states(direct and non-direct), so this MANET's member still can update the routing table via broadcast without network support.

The second case, shown in the diagram(Figure3.8), is when this newly added device has devices paired with it, in which case the joining operation is multiple devices becoming members of the MANET.

3.3.4 Leaving a MANET

As in the previous section logic, when the device leaves the current MANET, the virtual chains connecting this device are disconnected in the physical structure. However, depending on the latest uploaded neighbour nodes, the cloud must re-update the topology. There are three aspects to consider when the device leaves the MANET.

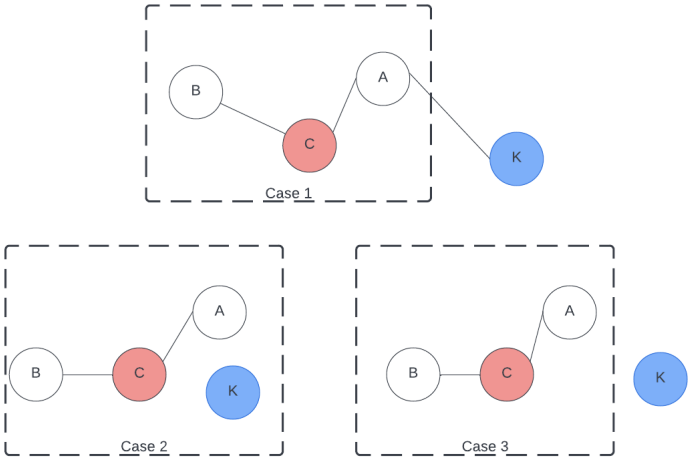


Figure 3.9: Leaving a MANET

1. In the first case is that the device doesn't cancel the pair but is simply out of range of the network in terms of distance.

2. In the second case, the device is unpaired and is considered to have left the network regardless of whether the device is within or outside the network's scannable range.

3. In the third case, when the cloud detects the presence of specific devices in other networks, this is also considered as leaving the original MANET.

In the same logic as for the join operation, the leaving node is also separated from the previous MANET if there are only nodes connected to this node. In an undirected graph structure, two vertices delete the joint edge, and if there is no other pathway associated with the edge, two undirected graph structures are formed at this point.

3.3.5 Splitting and Merging a MANET

In the previous two subsections, refer to the joining or leaving. In the Bluetooth-MANET model, the mobility of the device determines the operation, i.e., join or leave a MANET. Generally, when a mobile device joins a MANET, on the one hand, it can be regarded as if the two MANETs are merged into one, and on the other hand, the active behaviour of this device is to join. Similarly, when a device leaves a MANET, it is not only a device leaving operation but also splitting a MANET with only one member. Hence, naturally, a logical division.

When the device is not connected to the cloud that does not affect the distinction between physical and logical MANETs, this is due to the nature of Bluetooth, in which,

even though the devices are within the same radio range of each other, they are not paired with each other and so are not classified as being within the same MANET. This shows that there are two necessary conditions for the splitting and merging of MANETs, the first being that the devices are within the same radio range and the second being the existence of a communication link between the devices, i.e., whether devices are paired.

However, for the cloud, the number of the MANETs that are analysed by devices uploading information about their neighbours, i.e. there are several MANETs if the algorithm can generate several graphs.

In the program, I designed a function to distinguish the number of graphs using a queue and the typical algorithm called Breadth-first search (BFS); this is a non-recursive method similar to DFS. Generally, BFS uses a queue (first-in-first-out) to get extra memory. BFS will eventually find the destination state, but DFS may be lost in parts of the graph where there is no destination state and never return. As the example of splitting operation, in the MANET A, there are four sets:$[A\{C\}, B\{K,C\}, C\{A,B,K\}, K\{C\}]$. In the MANET B, there are three sets: $[D\{E,G\}, E\{D\}, G\{D\}]$.

Source	destination	Source	destination	Source	destination
A	C	C	B	D	G
B	K	C	K	E	D
B	C	K	C	G	D
C	A	D	E	\	\

Table 3.1: Neighbour Table

Algorithm 3.3 Classification(BFS)

```
Initialise List anslist
set = all device name
While size of set is not 0:

    Initialise queue and sameGroup
    Add set(0) to queue
    While queue is not empty:
        For i to size of queue:
            temp_name = pop queue
            Add temp_name to sameGroup and Remove temp_name from set
            List tmp = get temp_name in pairedMap
            For name in tmp:
              If sameGroup contains name:
                skip
              EndIf
              If queue not contains name:
                Add name to queue
              EndIf
            EndFor
        EndFor

     EndWhile

     Add sameGroup to anslist

EndWhile
```

For demonstration purposes, I convert the result to JSON format. In the cloud service, the information of MANET will be updated automatically when the cloud classifies MANET, which will generate a new MANET UUID and update the member list, and finally, return to the mobile device in each MANET.

```
"data": [
  {
    "membership": [
      "A",
      "B",
      "C",
      "K"
    ],
    "group": 1
  },
  {
    "membership": [
      "D",
      "E",
      "G"
    ],
    "group": 2
  }
]
}
```


Figure 3.10: Group Devision

- Operation of Splitting

When some devices leave the scanning range, even if some of them are paired with members of the original MANET, devices in different ranges are treated as split into several new MANETs.

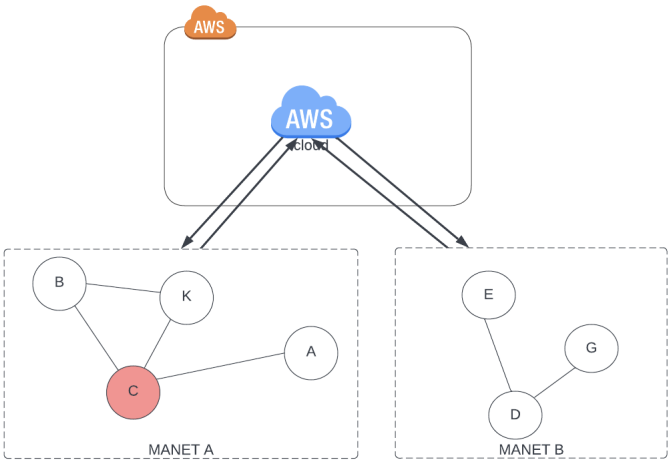


Figure 3.11: Splitting a MANET

- Operation of Merging

When some mobile devices access the scanning range, two or more Bluetooth MANETs will merge into one as long as the nodes acting as bridges are paired with each other. In other words, the two or more piconets are combined into one scatternet.

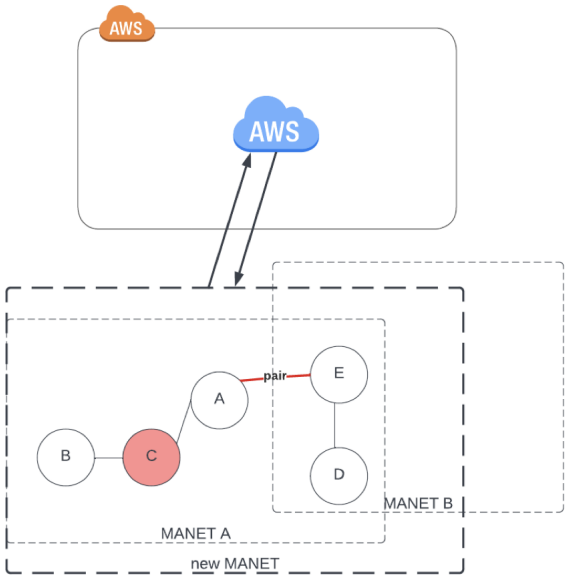


Figure 3.12: Merging a MANET

In conclusion, the conditions for MANET operations are triggered by the active actions of joining or leaving.

3.3.6 MANET's Owner

The device needs to register its device information to the cloud before using the cloud service. The registration operation and login operations are automatic. The device will register information in the device table when it first accesses the cloud service, including device MAC address, device name, default password, device UUID and other information. After the registration is completed, the device will automatically log in when opening the application again. The whole execution process does not need to be triggered by the user, and the user can change personal information by himself after the device information is automatically registered. In addition, the device's role is determined by the topology. The device with the highest number of connections is the Owner of MANET, all other devices are members, and the Owner device must be able to connect to the network by default. As an intermediate node, it can effectively communicate with the cloud and ensure the reliability of messages.

3.4 Bluetooth Service

This section introduces the functionality of Bluetooth and encapsulates the Bluetooth service to make the program look neat. The Bluetooth service is divided into two parts, the device scan module and the Bluetooth operations module. The device scan module defines four interfaces, ACTION_FOUND, ACTION_ACL_CONNECTED, ACTION_DISCOVERY_FINISHED and ACTION_STATE_CHANGED, which are overridden by MainActivity class and DeviceActivity class, the benefit is to reduce code reuse and only require registration of the services. The methods encapsulated in the Bluetooth Operations module include initialising the Bluetooth module, enabling/disabling Bluetooth, turning on/off device discovery, getting a Bluetooth device and adapter, pairing (create a bound), unpairing(cancel a bound), setting a PIN password etc. Equally, the Bluetooth module only needs to be initialised once and the Bluetooth methods can be called from any Activity class.

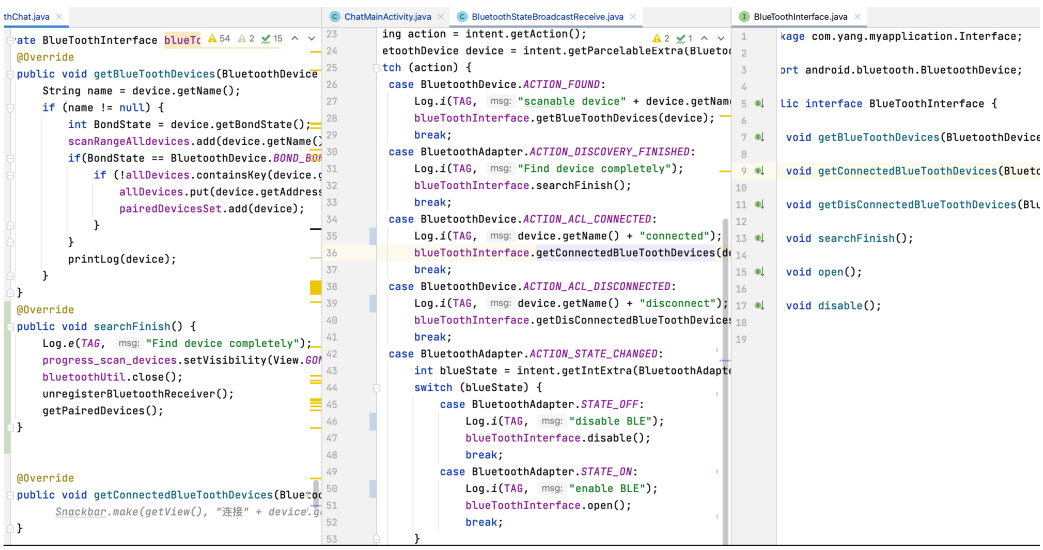


Figure 3.13: Bluetooth Service

In MainActivity class, simply override the getBlueToothDevices method to get the available devices and searchFinish method to unregister BLEservice.

3.5 Communication Service

In Bluetooth, before the two devices establish communication, there are four major steps necessary for Bluetooth communication:

1. Setting up Bluetooth and Scanning for other Bluetooth devices.
2. Querying the local Bluetooth adapter for paired Bluetooth devices.

3. Establishing RFCOMM channels/sockets.

4. Connecting to a remote device.

Finally, transferring data over Bluetooth maybe need another step that manages multiple connections.

For Bluetooth-enabled devices to transfer data between each other, devices must firstly form a communication channel through a pairing process. Then, each device can act as a server and client, depending on which one requests a pair. When communication ends, the two devices are still paired, and to re-establish communication channels, they can automatically reconnect as long as they are in the range of each other and neither has removed the binding.

Generally, direct communication in Bluetooth is a standard way. I found a few documents that talk about the multi-hop way in Bluetooth. In the direct connection, the destination or source device only needs to parse the transferred bytes simply. For parsing a file or image, it needs to be read and stored in both input and output streams, which is why direct communication is simpler, as the byte arrays transferred are a complete file, and these byte array contents do not carry other information. Otherwise, the destination device must know the length of bytes contained in the message body and the byte length of other information to parse so that the probability of error will be high.

There are two challenges in implementing multi-hop communication using Bluetooth. The first is to ensure the integrity of the message. The bytes carried contain not only the message body but also the destination device information, It is similar to the TCP message format. The other is about the timeliness of messages. Even if the communication link is interrupted during transmission, it is necessary to ensure that messages can reach the destination.

In the following subsections, I will describe how to manage multiple connections, what the communication protocol contains, and several ways of communication.

3.5.1 Create a Connection Thread

The official documentation for Android already provides simple peer-to-peer text communication. Based on the reference code given in this documentation, I have encapsulated a communication tool class named ChatUtils for ease of invocation and readability. The tool class ChatUtils, which manages communication contains three methods ConnectThread, AcceptThread, and ConnectedThread corresponding to the whole connection process, i.e., request connection, accept the request, created communication thread, the two devices can communicate with each other normally.

When two devices are ready to connect, one device acts as the server and the BluetoothServerSocket method in the server is used to listen for connection requests. If the device is connecting for the first time and has not been paired before, the Android framework will automatically display a pairing request dialogue to both devices.

```
private static final String NAME_SECURE = "BluetoothChatSecure";
private static final UUID MY_UUID = UUID.fromString("188c5bda-d1b6-464a-8074-c5deaad3fa36");
private static final UUID MY_UUID_SECURE = UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");
private static final UUID MY_UUID_INSECURE = UUID.fromString("8ce255c0-200a-11e0-ac64-0800200c9a66");
```

Figure 3.14: UUIDs

In the AcceptThread method, two connection methods are provided, a secure connection ($listenUsingRfcommWithServiceRecord$) and an insecure connection ($listenUsingInsecureRfcommWithServiceRecord$). The UUID uniquely identifies the service it is connecting to and the UUID must match to accept the connection. In the connectThread method, the client device must first obtain the remote device object BluetoothDevice and call its internal method $createRfcommSocketToServiceRecord(UUID)$ or $createInsecureRfcommSocketToServiceRecord(UUID)$ to get a BluetoothSocket, where the UUID is the same as in the AcceptThread method.

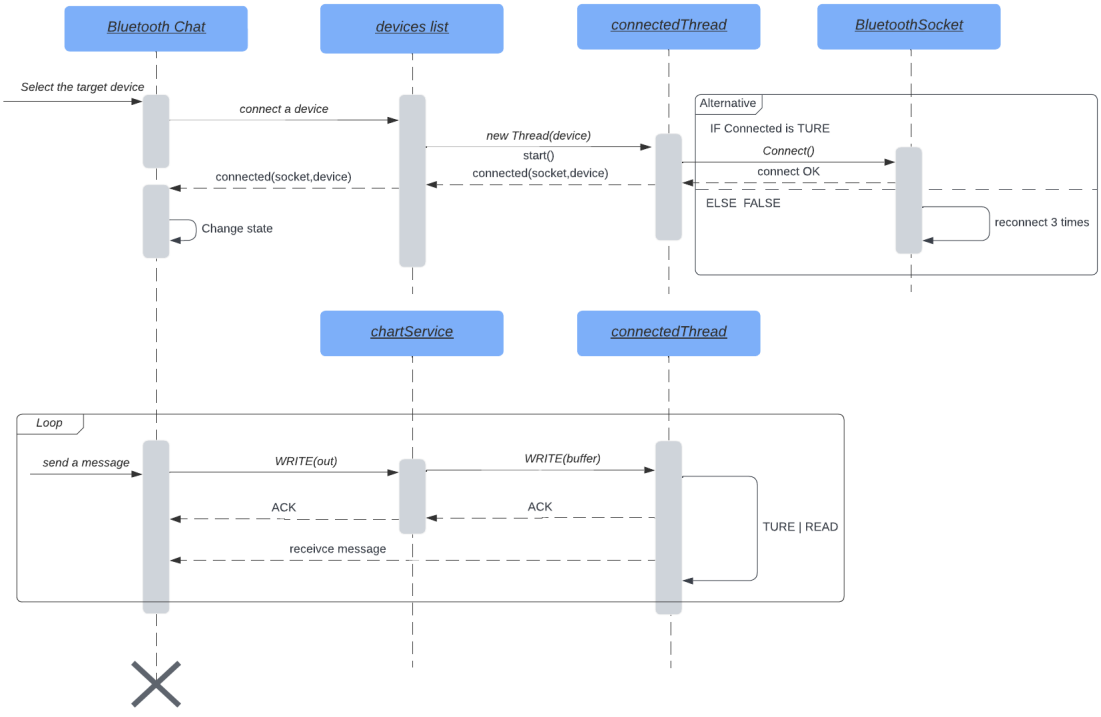


Figure 3.15: The process as call sequence diagram

When the entire connection process is completed, the ConnectedThread method handles the reading of the message and the DataOutputStream object created is used

to write the data; likewise, the DataInputStream object handles the reading of the data. In the methods for writing to the data stream, to make it easier to read the data type, the DataOutputStream provides writeInt and writeUTF methods, the former for storing the data type and the latter for storing the message object, which needs to be converted to JSON string format. Correspondingly, the DataInputStream provides readInt and readUTF methods to read data.

3.5.2 Manages multiple Connections

In addition to analysing how to achieve multi-hop communication, another more important issue to be solved is managing the multiple communication channels.

Almost all of the reference code implements peer-to-peer text communication. At the beginning of the project, each time the data is forwarded, the next hop node is reselected for connection. That is, disconnects and reconnects are executed repeatedly with each data forwarding. This unreasonable way increases the message latency time and sends messages when the communication threads are accidentally interrupted by multiple connections. And when the user continues to select the same user to communicate, another reconnection process is carried out. If the intermediate node has to handle messages from other devices, there is only one communication thread, and the messages are obviously lost. Furthermore, Bluetooth devices may refuse to connect after multiple disconnects and reconnect operations, resulting in an unstable link.

Based on this creation of a communication thread (see the previous subsection3.5.1 for details), a device connected to two devices at the same time will create two communication threads without defining client and server roles. In order to achieve reasonable multiple connections, a thread management class GroupManager is encapsulated for managing multiple connection threads. This class will be called automatically after a device scan for available devices, reducing the number of manual steps. In addition, the first time the management thread is created, the connection time is approximately 5 seconds, depending on whether the other device is Bluetooth enabled or not, and the connection process will be repeated three times, with three failed connections meaning the link is broken. It is worth noting that to avoid the repeated creation of communication threads leading to crashes when called, the global variable of the HashMap type $deviceConnections$ is used to store as well as the existence and status of the threads. The global variable $chatSockets$ of the ChatUtils array type is used to store communication threads.

Algorithm 3.4 GroupManager

```
Declare static global variables: chatSockets
Initialise: HashMap deviceConnections, HashSet seen
For user in DevicesList<device_name,device_MAC>:

    If seen not contains device_name:
       Add device_name to seen
    EndIf
    Initialise: flag is TRUE
    For _socket in chatSockets:
       If device_MAC equals _socket.MAC:
          flag is FALSE
          Exit
       EndIf
       If flag is TRUE:

          Initialise: temp is new chatSocket
          If temp.state equals STATE_NONE:
            call start()
          EndIf
          Add temp to chatSockets
       EndIf
    EndFor

EndFor
```

3.5.3 Communication Protocol

To realise multi-hop communication, the source device can still communicate with the destination device even doesn't connect to it directly. Because of the feature of MANET, the communication protocol is defined. Shortly, the protocol format contains the following structure: message content, source, destination, send time, read time, ACK(acknowledge), message ID, routing path, and a flag on whether the message is uploaded to the cloud. Additionally, for simplifying the feedback message, the protocol format only carries fields of message ID and receive time. Because of the limited bandwidth of Bluetooth transmission, the message body occupies more bytes, especially for pictures and voice messages. Therefore, the returned message only includes these two useful messages to reduce latency.

Parameter	Info
Content	The message body.
Source	The field contains source name and source MAC address, MAC address is important, The bluetooth MANET use MAC address to create a connection.
destination	The field contains destination name and destination MAC.
Send Time	The field records the sending time from the source device.
Read Time	The field records the reading time of the destination device.
ACK(acknowledge)	There are two attributes 1 and 0, 1 means the message is delivered, whereas 0 is undelivered status. Note: 1 or 0 doesn't mean the destination or source device read the message or the feedback which is from the destination.
Message ID	Unique ID (primary key)
Hop	The distance from source to destination. • 1 hop : directly communication • More than 1 hop: multi-hop communication
Routing path	The field is a list which contained all the necessary paths from source to destination. The path is form by the cloud.
Upload	This field indicates whether the message has been uploaded to the cloud.
DataType	The field labels which message type is transferred because handling text messages is not the same as handling images or file types. There are three types that can be transferred: DATA_TEXT, DATA_FILE, DATA_IMAGE.
ImageBitmap	This field stores information about the image.
AudioFile	This field stores voice information in the format of a byte type.

Table 3.2: Communication Protocol Information

```
20      @Column(unique = true, defaultValue = "unknown")
21      private String  uuid;
22      private Object content;
23      public String message;
24      private String sendDate;
25      private String readDate = "END";
26      private int isRead; // 0 no 1 already read by the target
27      private int isUpload; // 0 no 1 already upload to cloud
28      public Bitmap imageBitmap;
29      public File audioFile;
30      public int dataType;
31      private String targetName;
32      private String targetMAC;
33      private String sourceName;
34      private String sourceMAC;
35      private String routeList;
```

Figure 3.16: Message Object Class

3.5.4 Three Data Types

The program needs to handle the message according to its data type, and the listener types are defined as follows:

```
public static final int MESSAGE_READ_IMAGE = 6;
public static final int MESSAGE_READ_AUDIO = 7;
public static final int MESSAGE_READ_TEXT = 8;

public static final int MESSAGE_WRITE_IMAGE = 9;
public static final int MESSAGE_WRITE_AUDIO = 10;
public static final int MESSAGE_WRITE_TEXT = 11;
```

Figure 3.17: Data Types

The six processing types are divided into READ and WRITE processing. For text processing READ and WRITE are relatively simple as text is stored as a string. On the other hand, in Android applications, images are stored in Bitmap type for image processing. However, the local database does not store this data type, so the bitmap data is converted to String type in Base64 format. However, the Base64 format data type to string is often very large, so when taking photos or selecting photos, there should be a limit on the size of images that users can upload and ensure that the image size is under 2M.

The processing of voice messages is a little more complex. The Android multimedia framework provides MediaRecorder APIs that can capture and encode various standard audio formats. Before using the APIs, the permission RECORD_AUDIO should be

introduced firstly. Then, when the user presses the audio button, the steps to record a voice are as follows:

1. Initialise MediaRecorder.
2. Set some audio parameters, such as the audio recording encoding format, frequency recording audio source, the recorded media file output conversion format, and media file output path.
3. Prepare to record.
4. Start recording.
5. Stop recording.
6. Reset and Release resources.

```
File f = new File(fileName);
setFileName(fileName);
FileInputStream fis = new FileInputStream(fileName);
byte[] buff = new byte[(int) f.length()];
fis.read(buff);
Calendar calendar = Calendar.getInstance();
String txtWriteTime = sdf.format(calendar.getTime());
nextrouters = getNextrouters();
int isUpload = 0;
if( buff.length <= 2048){...}
if(buff.length > 2048*5) {...}
chatManager.sendMessage(buff, DATA_AUDIO,currentConnectDevice,
        localName,localMacAddress,isUpload,txtWriteTime,nextrouters);
fis.close();
```

Figure 3.18: File to byte array

The recorded voice is stored in a preset storage location. Then the file should be converted to the byte array before the write method is called to process it. Text, images and voice messages are all converted to byte arrays, and the write method does additional processing depending on the data type.

- Type of TEXT

The text content does not require additional processing and is directly stored in the message object field.

```
} else if (datatype == DATA_TEXT) {   You, Moments ago • Uncommited changes
    dataSent.setContent(bytes);
    String content = new String(bytes);
    dataSent.setMessage(content);
    jsonObject.put("content", dataSent.getMessage());
    writtenMsg = handler.obtainMessage(MESSAGE_WRITE_TEXT,  arg1: -1, DATA_TEXT, dataSent);
    OutData.writeInt(DATA_TEXT);
    OutData.writeUTF(JsonUtils.objectToJson(dataSent));
```

Figure 3.19: TEXT Type

- Type of IMAGE

For image messages, when the system camera or system album is called, the selected image format is the bitmap type in Android. Bitmap-type images can be displayed directly in the interface, the method is $setImageBitmap()$. The method to get the image by calling the camera is $(Bitmap)data.getExtras().get()$, whereas the album method is $MediaStore.Images.Media.getBitmap()$.

When processing transmission data, the bitmap format needs to be converted to base64 format, but this storage format is very bandwidth intensive, so the latency time of image transmission will be a bit higher.

```
if (datatype == DATA_IMAGE) {
    ByteArrayOutputStream imageStream = new ByteArrayOutputStream();
    imageStream.write(bytes);
    String decodedString = new String(imageStream.toByteArray(), Charset.defaultCharset());
    byte[] decodedStringArray = Base64.decode(decodedString, Base64.DEFAULT);
    Bitmap bp = BitmapFactory.decodeByteArray(decodedStringArray,
            offset: 0, decodedStringArray.length);
    String base64 = HandlerTool.bitmapToBase64(bp);
    dataSent.setMessage(base64);
    jsonObject.put("content", dataSent.getContent());
    writtenMsg = handler.obtainMessage(MESSAGE_WRITE_IMAGE,  arg1: -1, DATA_IMAGE,
            dataSent);
    OutData.writeInt(DATA_IMAGE);
    OutData.writeUTF(JsonUtils.objectToJson(dataSent));
    imageStream.close();
```

Figure 3.20: IMAGE Type

- Type of AUDIO

For voice messages, voice information needs to be converted into byte arrays during transmission. Voice messages can be more troublesome to process than the first two data types because the local database cannot store byte data. Therefore, byte arrays are regenerated during each transmission, and data streams are processed.

```
}else if (datatype == DATA_AUDIO) {
    String fileName = ChatMainActivity.getFileName();
    File file = new File(fileName);
    dataSent.setMessage(file.getName());
    dataSent.setContent(file.length());
    jsonObject.put("content", dataSent.getContent());
    writtenMsg = handler.obtainMessage(MESSAGE_WRITE_AUDIO, arg1: -1, DATA_AUDIO, dataSent);
    FileInputStream in = new FileInputStream(file.toString());
    OutData.writeInt(DATA_AUDIO);
    OutData.writeUTF(JsonUtils.objectToJson(dataSent));
    int r;
    byte[] b = new byte[4 * 1024];
    while ((r = in.read(b)) != -1) {
        OutData.write(b, off: 0, r);
    }
```

Figure 3.21: AUDIO Type

3.5.5 Communication Type

The communication type can be classified in three ways, direct communication, multi-hop communication and direct communication via the cloud. For the neighbour device, communication between devices is direct, as there is only one hop distance. For non-neighbouring nodes within the same MANET, the communication between devices is multi-hop. Finally, for nodes where the devices are all networked, the communication between devices consists of the three methods mentioned above.

In the Bluetooth-MANET model, multi-hop communication has always been a research focus. Each mobile device can act as a router, with devices at the same level. To transmit the message from source to destination, how intermediate devices know which device is the next hop, i.e. the routing path, is determined by the cloud's routing algorithm. These three types of communication are listed below.

- Direct communication between MANET members

After selecting the destination, RFCOMM channels/sockets will be established, and then waiting for connecting successfully, the device can send the message to the destination. The distance for direct communication is one-hop.

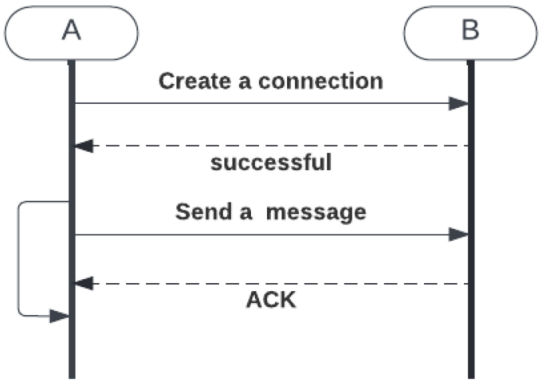


Figure 3.22: The process of the direct communication

- Communication in MANET(multi-hop)

To realise multi-hop communication, the sender must include the full routing path and the encapsulated packet to reach the receiver successfully. The field carrying the route is included in the communication protocol. Therefore the intermediate devices will send the message to the correct next-hop device when it is transmitted, without the need to go back to check the local routing table and find the destination device.

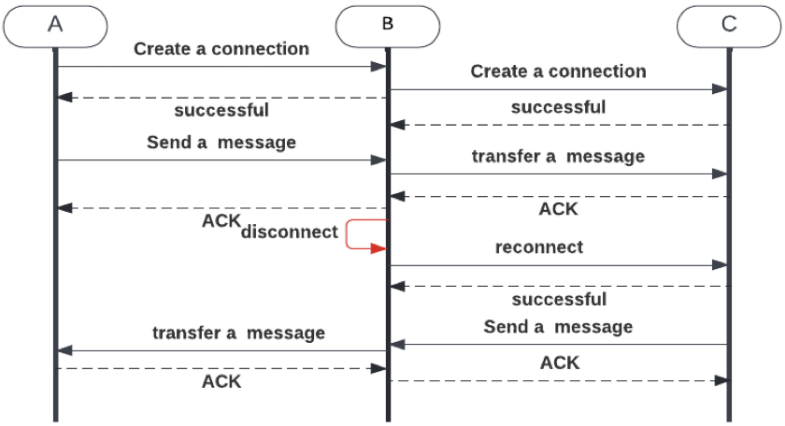


Figure 3.23: The process of the communication (multi hop)

- Communication via the cloud

When a message is not sent to the destination device for some reason, as mentioned above, as long as the message is backed up in the cloud, it can be directly sent to the destination device and the source device.

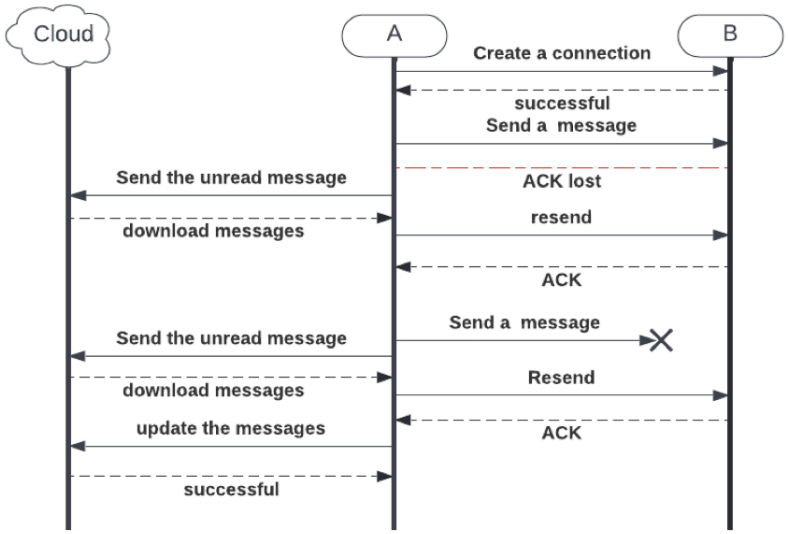


Figure 3.24: Communication via the cloud

Chapter 4

Experimental Results

This section presents the performance results of the model implementation, including communication capabilities, and MANET network topology formation analysis. Each mobile device was placed away from the other devices but within radio range. First model test: three separate communication methods were tested, including direct and multi-hop communication and communication with message retransmission involving the cloud. Each result was an average of 10 experiments conducted. Second model test: Any device is placed out of radio range and the cloud is examined for membership updates, topology and routing updates within the MANET network.

The version information of Android devices:

Device Brand	**Android Version**
Galaxy SM-A750FN	8.0.0
LG Nexus 5	6.0.1
Nexus 5X	8.1.0
Nexus 5X	8.1.0
Nexus 4	5.1.1

Table 4.1: Android Devices for Testing

The android UI:

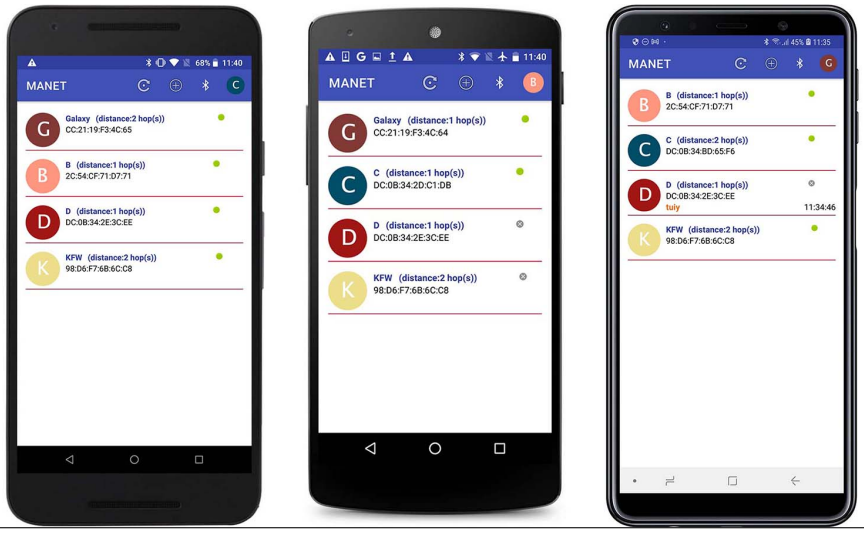
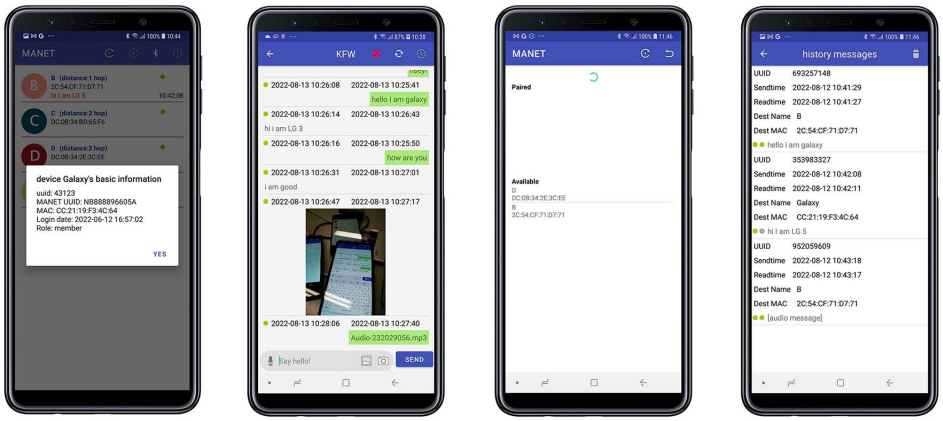


Figure 4.1: Bluetooth Chat UI

The member list displayed on the home page not only shows the device name and MAC address but also marks the hop distance between the device and other member devices. Unread messages are also displayed in the device list without the need to click to jump to the chat interface to view these messages. The green icon on the right side indicates that a neighbour member is online, otherwise, the grey icon means offline status.

Other user interfaces:



device info chat UI search device UI history message UI

Figure 4.2: Other UI

The first UI is the current device's information, including its UUID, which MANET it

in, last login time and its role. Generally, the device with the most connections will be identified as the Owner and the others as members.

The chat screen includes, the name of the sender-receiver, the time sent and the time received, and whether the message has been read (green icons indicate read messages, grey icons indicate unread or undelivered). The input field at the bottom of the screen allows users to send three types of messages, voice, text and images that can be photographed or selected in the gallery.

The third UI is a listing page showing all devices, where the user can tap on a device in the available list to pair (join the network) and tap on a device in the pairing list to unpair (leave the network).

The History page contains all the messages for which data has been transferred. As can be seen in the diagram, the last line of each piece of data has two icons, the first one indicating whether the message has been read and the second one indicating whether the message has been backed up in the cloud.

4.1 Cloud Deployment

Amazon Web Service (AWS) provides services that enable rapid deployment and management of applications. With Elastic Beanstalk services, users do not need to understand the infrastructure to run applications and do not need to start by setting up a runtime environment as they do with Elastic Compute Cloud (EC2) services. Elastic Beanstalk reduces the complexity of administration without compromising choice or control. Users simply upload their applications and can monitor how they are running.

1. Create an account in AWS and select Elastic Beanstalk for the Ireland region.

2. Create the web server environment, select the Tomcat platform and finally upload the local WAR package and wait for the deployment to complete.

To check if the deployment is working, a simple interface is provided in the application to test it, in addition to the environment monitoring provided by the Cloud service itself.

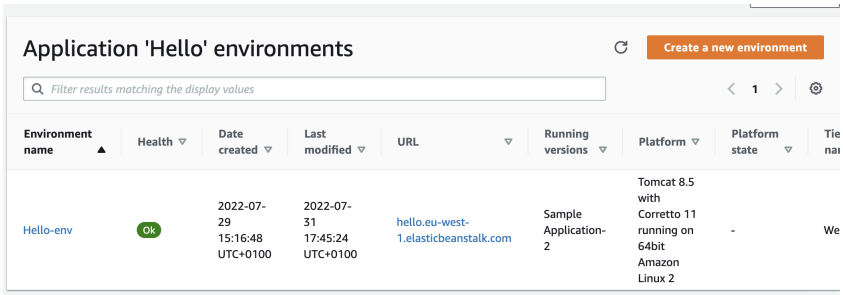


Figure 4.3: Elastic Beanstalk Deployment

Table 4.2 lists the database information that the GET API is designed for viewing for convenience:

Interface URL	API Info
router/test	This is a test API to check how many MANETs currently exist.
router/viewAll	This API allows user to view the routing table and neighbour table, including routing paths, recently updated neighbour nodes.
message/viewAll	This API is to view the history messages.
user/viewAll	This API is to view registered members.
MANET/viewAll	This API is to check the members of the MANET and returns information including the device's UUID and MANET UUID.

Table 4.2: Test API

4.2 Sending Messages Service Results

Testing the latency of three message types (text, voice, pictures) over distances from one to four hops without cloud support. In addition, the latency of the four-hop distance using the cloud service is tested separately for both networks (Wi-Fi and 4G)

- Text Message Test

A mobile device is selected to send an arbitrary string from the source to the target.

Communication Type	Network Condition	Avg.delay(Seconds)
one-hop(directly)	without the cloud	0.3066
two-hop	without the cloud	1.4335
three-hop	without the cloud	1.4102
four-hop	without the cloud	2.1468
four-hop	Over the Cloud(Wi-Fi)	0.9150
four-hop	Over the Cloud(4G)	1.0140

Table 4.3: Text Message

- Voice Message Test

Select a target device and send a voice message with a file size of approximately 30kb to the target device.

Communication Type	Network Condition	Avg.delay(Seconds)
one-hop	without the cloud	0.8579
two-hop	without the cloud	1.5321
three-hop	without the cloud	1.8994
four-hop	without the cloud	2.3242

Table 4.4: Voice Message

- Image Message Test

The size of the test image is 15.59kb.

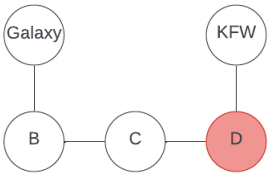
Communication Type	Network Condition	Avg.delay(Seconds)
one-hop	without the cloud	1.4282
two-hop	without the cloud	3.2275
three-hop	without the cloud	4.1831
four-hop	without the cloud	4.1215
four-hop	Over the Cloud(Wi-Fi)	1.1787
four-hop	Over the Cloud(4G)	1.6081

Table 4.5: Image Message

Thus to summarise, the first is that each node needs to determine the status of the link when forwarding the message, if it is not connected it will be reconnected three times within 15 seconds until it is confirmed that the link is broken, ideally the broken link will be reconnected within five seconds. The second scenario is more likely, for each node from the source to the target device, the "WRITE" listening method needs to determine if the transmitted information has been backed up in the cloud, if not it will be uploaded to the cloud using the API, in the absence of a network connection, no upload operation will execute, however, with network support, some unpredictable errors might happen in the cloud, that is, connection timeout, although the end result is the same as if there was no network, the network timeout lasts 10 seconds which seriously affects the timeliness of the message.

4.3 Analysing of MANETs' operation and topology

First, place the devices together in a radio area and wait for the devices to scan each other within range for information about their neighbours, upload it to the cloud and record the resulting routing database.



id	sourceName	sourceMAC	desName	desMAC	MANET_UUID	timeRecord
9536	C	DC:0B:34:BD:65:F6	D	DC:0B:34:2	NB888896605A	2022-07-31 16:40:26
9537	D	DC:0B:34:2E:3C:EE	C	DC:0B:34:B	NB888896605A	2022-07-31 16:40:26
9538	D	DC:0B:34:BD:6B:EB	KFW	98:D6:F7:6E	NB888896605A	2022-07-31 16:42:59
9539	KFW	98:D6:F7:6B:6C:C8	D	DC:0B:34:B	NB888896605A	2022-07-31 16:42:59
9558	C	DC:0B:34:BD:65:F6	B	2C:54:CF:7'	NB888896605A	2022-07-31 16:50:21
9559	B	2C:54:CF:71:D7:71	C	DC:0B:34:B	NB888896605A	2022-07-31 16:50:21
9562	Galaxy	CC:21:19:F3:4C:65	B	2C:54:CF:7'	NB888896605A	2022-07-31 16:52:19
9563	B	2C:54:CF:71:D7:71	Galaxy	CC:21:19:F3	NB888896605A	2022-07-31 16:52:19

id	source	dest	path	hop	MANET_UUID
43254	Galaxy	B	[Galaxy, B]	1	NB888896605A
43255	Galaxy	C	[Galaxy, B, C]	2	NB888896605A
43256	Galaxy	D	[Galaxy, B, C, D]	3	NB888896605A
43257	Galaxy	KFW	[Galaxy, B, C, D, KFW]	4	NB888896605A
43258	B	Galaxy	[B, Galaxy]	1	NB888896605A
43259	B	C	[B, C]	1	NB888896605A
43260	B	D	[B, C, D]	2	NB888896605A
43261	B	KFW	[B, C, D, KFW]	3	NB888896605A
43262	C	Galaxy	[C, B, Galaxy]	2	NB888896605A
43263	C	B	[C, B]	1	NB888896605A
43264	C	D	[C, D]	1	NB888896605A
43265	C	KFW	[C, D, KFW]	2	NB888896605A
43266	D	Galaxy	[D, C, B, Galaxy]	3	NB888896605A
43267	D	B	[D, C, B]	2	NB888896605A
43268	D	C	[D, C]	1	NB888896605A
43269	D	KFW	[D, KFW]	1	NB888896605A
43270	KFW	Galaxy	[KFW, D, C, B, Galaxy]	4	NB888896605A
43271	KFW	B	[KFW, D, C, B]	3	NB888896605A
43272	KFW	C	[KFW, D, C]	2	NB888896605A
43273	KFW	D	[KFW, D]	1	NB888896605A

Figure 4.4: Topology

Following this, select two arbitrary devices to be placed within the radio range

away from the original area, i.e. around 15 metres away, and wait for the devices to periodically update the neighbour table information to the cloud before viewing the routing table here.

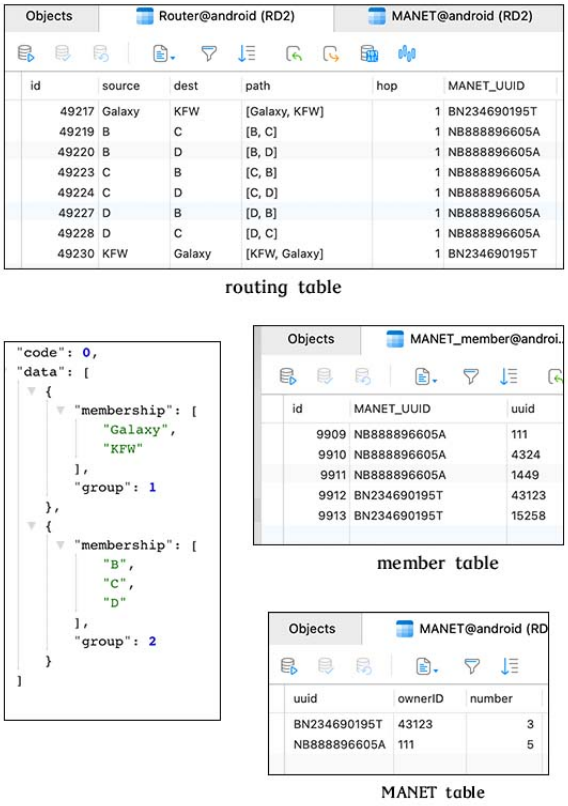


Figure 4.5: Topology

According to the result, the device Galaxy and KFW are placed out of the previous radio range, and thereafter all devices have updated their information, the cloud service redivides the MANET, automatically creates a new MANET record, and updates the MANET member list information as well as the routing information.

Chapter 5

Conclusion

This paper has presented a Bluetooth-MANET model that introduces cloud services. The paper focuses on the design and implementation of the system, as well as the results of the test model. The model proposed in this paper is based on the implementation of Bluetooth features such as discovery between devices, pairing/unpairing or enabling/disabling Bluetooth. As mentioned earlier, unlike other wireless networks, Bluetooth is a one-to-one connection, but with the characteristics of a MANET, the connection structure between Bluetooth devices can be relatively homogeneous, so an important function of the cloud is to use the neighbourhood information uploaded by the devices to produce a network topology and analyse the path between each device. This way even if the devices are not directly connected, they are still considered to belong within this MANET, as long as the devices are in the Bluetooth range. Obviously, if a device leaves this radio range, or appears in another radio range, the cloud will detect that the action of the device is leaving and vice versa. In addition, a device leaving or joining a network is recognised by the cloud, i.e. the device does not actively trigger the joining or leaving action, in fact, the splitting and merging of MANET networks are not actively triggered either, the cloud divides the network based on the neighbour node information.

Another cloud service is the retransmission of unread messages. When devices send messages to each other, any intermediate device can transmit the message to the cloud, ensuring that the message is delivered properly even if a node is disconnected. This is necessary to ensure the reliability of the messages.

The users do not need to do any operation when they use the application, just enable Bluetooth settings and wait for the scan to complete to get the list of members. As the scan is a cyclical operation, the user can also click the update button if they want to update the list at intervals.

In conclusion, the experimental results show that multi-hop communication, as well as direct communication, performs well, especially for text message types. Transfer latency is related to message size, and in the image transfer tests, latency times are higher as the images generally can be quite large, although this can be completely overcome if supported by cloud services.

Chapter 6

Future Work

In future work, I am planning to improve the MANET management service in the cloud, which can automatically monitor and analyse the status of the devices, and smartly generate the best path for each device. On the device side, the connection can establish or re-connect quickly without too much latency.

In terms of the current Bluetooth-MANET model, it is only supported on Android devices currently. The Bluetooth function is the same in IOS devices as in Android Bluetooth. Although the native Android Bluetooth file transfer function is not interoperable with that of the iPhone, the swift official documentation provided by Apple can be used as inspiration for the two different operating systems themselves to communicate via Bluetooth.

Currently, I am still progressing with the IOS device side in terms of the basic operations of Bluetooth, such as scanning and pairing. If the model is to support communication between two different operating systems, the IOS side needs to be consistent with the Android side in terms of both the format of the data received and the data sent. Although I have not yet implemented this step, from my humble perspective, I believe that if IOS devices can be paired with Android devices, communication is possible and only requires a unified communication protocol.

Because of its ubiquity and reasonable price, almost all mobile devices can support Bluetooth, and if this application can also support IOS, then the usefulness of this model will be well promoted.

Additionally, I would like to be able to test this model in a real environment. Taking into account mobility, the communication connections are constantly updated or checked for the latest connection status to ensure the reliability of the communication better.

Bibliography

[2021]. URL: https://developer.android.com/guide/topics/connectivity/bluetooth/connect-bluetooth-devices.

Alam, Tanweer and Mohamed Benaida [2019]. "The role of cloud-MANET framework in the internet of things (IoT)". In: *arXiv preprint arXiv:1902.09436*.

Alshareef, Hazzaa Naif and Dan Grigoras [2014]. "Mobile ad-hoc network management in the cloud". In: *2014 IEEE 13th International Symposium on Parallel and Distributed Computing*. IEEE, pp. 140–147.

— [2017]. "Robust cloud management of MANET checkpoint sessions". In: *Concurrency and Computation: Practice and Experience* 29.2, e3816.

Bakshi, Aditya, AK Sharma, and Atul Mishra [2013]. "Significance of mobile AD-HOC networks (MANETS)". In: *International journal of innovative technology and exploring engineering (IJITEE)* 2.4, pp. 1–5.

Basagni, Stefano et al. [2004]. *Mobile ad hoc networking*. Vol. 461. Wiley Online Library.

Bisdikian, Chatschik [2001]. "An overview of the Bluetooth wireless technology". In: *IEEE Communications magazine* 39.12, pp. 86–94.

Bouhenguel, Redjem, Imad Mahgoub, and Mohammad Ilyas [2008]. "Bluetooth Security in Wearable Computing Applications". In: *2008 International Symposium on High Capacity Optical Networks and Enabling Technologies*, pp. 182–186. DOI: 10.1109/HONET.2008.4810232.

Dillon, Tharam, Chen Wu, and Elizabeth Chang [2010]. "Cloud computing: issues and challenges". In: *2010 24th IEEE international conference on advanced information networking and applications*. Ieee, pp. 27–33.

google [2007]. *What is cloud computing*. URL: https://cloud.google.com/learn/what-is-cloud-computing [visited on 2022].

Haartsen, Jaap C [2000]. "The Bluetooth radio system". In: *IEEE personal communications* 7.1, pp. 28–36.

Hoebeke, Jeroen et al. [2004]. "An overview of mobile ad hoc networks: applications and challenges". In: *Journal-Communications Network* 3.3, pp. 60–66.

Mori, Yuichiro et al. [2011]. "A self-configurable new generation children tracking system based on mobile ad hoc networks consisting of Android mobile terminals". In: *2011 Tenth International Symposium on Autonomous Decentralized Systems*. IEEE, pp. 339–342.

Yang, Hao et al. [2004]. "Security in mobile ad hoc networks: challenges and solutions". In: *IEEE wireless communications* 11.1, pp. 38–47.

Zaruba, Gergely V, Stefano Basagni, and Imrich Chlamtac [2001]. "Bluetrees-scatternet formation to enable Bluetooth-based ad hoc networks". In: *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240)*. Vol. 1. IEEE, pp. 273–277.

Appendix

6.1 Reference Document

1. Bluetooth overview: https://developer.android.com/guide/topics/connectivity/bluetooth
2. Transfer Bluetooth data: https://developer.android.com/guide/topics/connectivity/bluetooth/transfer-data
3. Wi-Fi Direct: https://developer.android.com/training/connect-devices-wirelessly/wifi-direct
4. Bluetooth Chat Video: https://www.youtube.com/watch?v=NR1sDXMzyww&list=PLFh8wpMiEi8_I3ujcYY3-OaaYyLudI_qi

6.2 Reference Code

1. Offical Bluetooth Chat demo: https://github.com/DevExchanges/BluetoothChatAppAndroid
2. Demo extensions and UI: https://github.com/glodanif/BluetoothChat
3. LitePal for Android: https://github.com/guolindev/LitePal
4. A type-safe HTTP client for Android: https://github.com/liujingxing/rxhttp

6.3 My Github

1. Source code for Android: https://github.com/Amber916Young/MANET
2. Source code for Cloud server: https://github.com/Amber916Young/Cloud_MANET_server