Carleton University

COMP 4905 – Honours Project

P2P Messaging Application

Michael Tran

Professor Louis D. Nel

Abstract

One negative aspect that is commonly overlooked regarding communication between devices is when there is limited access to a network. This project will investigate multiple technologies including Bluetooth and Wi-Fi Peer-to-Peer in order to allow for communication between devices without a network. The application I created utilizes Bluetooth in an effort to create a peer-to-peer network. Additionally, it enhances the limited capabilities of Bluetooth by extending the range of its network. The results for this project have been successfully obtained through the creation of an Android application that demonstrates an interconnected network. This project provides a messaging application that uses Bluetooth to create a peer-to-peer network allowing multiple devices to communicate with one another. In conclusion, the application is one of many solutions towards solving communication between multiple devices when there is limited connectivity to the network.

Acknowledgements

I am very grateful for the completion of this project and it could not have been without the wonderful courses offered at Carleton University. I am thankful and indebted to Professor Louis D. Nel for guiding me and assisting me throughout the project.

Table of Contents

List of Figures

List of Tables

Main Body

Author Introduction

This section will provide some background information about myself, my interests and the reason why I decided to choose a Bluetooth Mobile Application as my Honours Project.

Firstly, I do not have much experience with Mobile applications and that is one of the main reasons why this area of study seemed fascinating. This project allowed me to gain more knowledge about the usage and development of Mobile applications, specifically Android. Additionally, this project would provide me with experience that would allow me to use my knowledge in an interview for companies that are doing work in the mobile area. I am interested in working in the car industry, so this project allows me to explore Bluetooth technologies which are present in many car applications and software.

Another reason why this Honours Project is of interest to me is because it allows me to explore the area of peer to peer networking. The peer-to-peer networking intrigues me because, in many applications, it removes the need of additional hardware, specifically servers. If this network model is utilized more often, then the disconnection of one node would not disrupt any other part of the network. I believe that if everyone uses peer-to-peer networking more often then services could be provided quicker to users.

Gaining knowledge in the mobile and networking field are a few reasons why I wanted to work with a Bluetooth Mobile Application topic as my Honours Project.

Motivation

This Honours project will investigate and develop a Mobile application on Android that allows two Android devices to communicate between one another. The objective of this project is to investigate the possibilities of Bluetooth and/or Wi-Fi peer-to-peer when using peer-to-peer communication. One of the reasons this application would be useful is because it assumes that users will not have any connection to a network, whether it be to Wi-Fi, or to their service provider. In these cases, users of mobile phones cannot communicate with one another using Short Messaging Service or messaging applications, an example being WhatsApp.

Creating an application that uses Bluetooth allows almost every user with a mobile device to use the application because, “it is rare when you find a mobile/smart phone that doesn’t contain Bluetooth technology” [Anon., n.d.]. An example of when this application would be useful is when population is too dense and service providers cannot reach customers, for example, during concerts or rallies. Another example is when users are on vacation and are in another country where they do not have service, but still want to communicate to each other through their cellular device. The previous examples provide cases where an application like this would dominate and, although the examples provided may be rare cases, they provide enough desire to investigate into an application that uses this technology.

Additionally, this project is only investigating the networking aspects through a messaging application but can, and should, be thought of more generally than that. The application can be expanded to be used for voice or video and can also be multiplatform, ranging from devices like PCs, to any Bluetooth or Wi-Fi compatible device. If everyone uses this application, it can remove the need of service providers, such that nobody would need to pay for monthly bills to communicate with one another using their devices.

Currently there is an application that is trying to impact the mobile industry in the same manner as this Honours Project. The application is called FireChat and uses Bluetooth as well as Wi-Fi to send and receive messages to other users. One of the features for that application is being able to send private messages or pictures and innovating a multi-hop and store-and-forward technology that is used when the offline network becomes large enough. The main marketing aspect for this application is being able to use it without the need of any internet connection or mobile phone coverage. Many of the key features of FireChat is relevant and similar to that discussed within this Honours Project. Thus showing that this Honours Project is feasible and can be used in real world applications.

The scope of this project will solely focus on mobile applications within the Android operating system. The project will be developed using Java and will be ported to real devices to do testing. This project will not use other libraries that could provide the same functionality desired for this project. The development of this application will not have a beautiful user interface and will not be tested thoroughly for odd cases. The main reason of building this application is to try and get multiple devices to be able to communicate with one another using Bluetooth or Wi-Fi. The project will begin to explore the different technologies and determine what is feasible for this kind of application.

The advantages of having a peer-to-peer network is that processing power is shared across all devices and does not rely on a central server. Additionality, the ability to extend a network proves to be useful for devices that are always moving as the considered technologies only have a limited range. Some of the disadvantages of a peer-to-peer network is that the communication will be slower due to being routed through multiple devices. Another problem is if a node disconnects and this same node was the critical node that would have relayed the message to others then messages could be lost.

The main objective for this project is to try and implement a peer-to-peer network so that devices will be interconnected, and processing power is shared across all devices. The biggest goal that this project is trying to achieve is the ability to allow connected devices to communicate with one another even if they are not in the same range as all the devices. In **Fig. 1**, 3 devices are available however, at most, only 2 devices are in range of each other. The middle device will act as a bridge for the two outer devices so that all devices can be connected and communicate with one another.

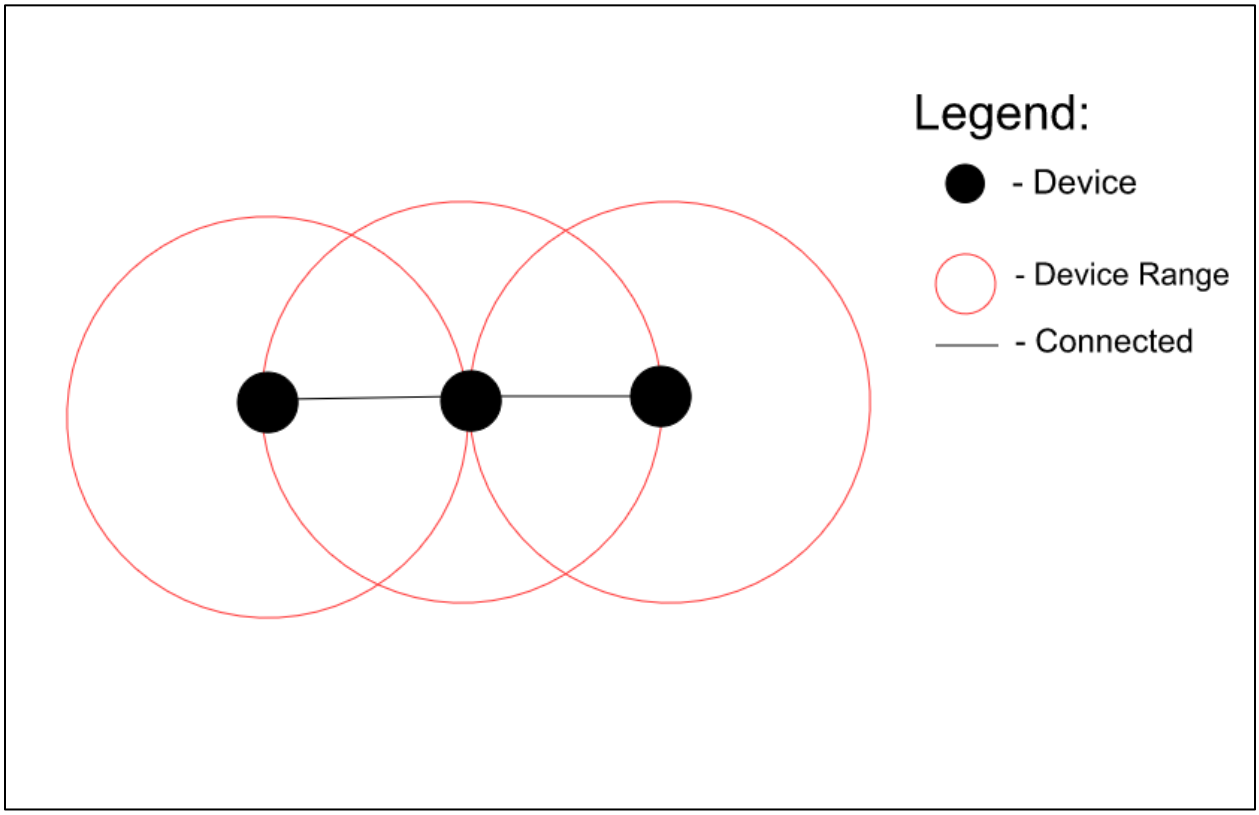

FIGURE 1 – Multiple Devices Connected when out of Range

Wi-Fi Peer-to-Peer API

Wi-Fi peer-to-peer is also known as Wi-Fi direct and is a fairly new technology that is very similar to Bluetooth. One of its main features is allowing users to connect with each other without a need for a wireless access point. An advantage to using this technology is that it provides users with quicker speeds than Bluetooth, as it allows users to communicate with one or more devices simultaneously at "typical Wi-Fi speeds, which can be as high as 250Mbps" [Wi-Fi, n.d.]. Another advantage to using Wi-Fi Direct is that the range of the technology can reach up to 200 meters. The above information about Wi-Fi Direct is very pleasing and it is clear that this technology is something that should definitely be observed for this project.

After doing some more research on Wi-Fi Direct, specifically on Android applications, it seems as if it relates more to a Client Server architecture where one device is acting as the group owner (or server) and all the other devices are connecting to it. This is where many problems may occur causing Wi-Fi Direct to be inferior on hosting a true peer-to-peer network.

If a device acts as a group owner and contains two clients, then all messaging must be routed through the group owner and be passed back down to the clients which resembles a Client Server architecture (See **Fig. 2**). This becomes problematic when the group owner disconnects because once the owner is disconnected, the two clients will have to communicate with one another and renegotiate their place in the network [ThinkTube, n.d.]. Renegotiation between two devices can cause connection errors and loss of data. Additionally, once a device becomes the new group owner it is possible that the previous group owner cannot connect since they may be configured differently now.

Another problem with the group owner in Wi-Fi Direct is that, if a device is out of range from the group owner, then that device cannot be connected with all other devices. This proves to be a problem because if that same device was in the range of another device that is on the network then in a peer-to-peer network, the device should be able to connect but that is not

the case for Wi-Fi Direct (See **Fig. 3**).

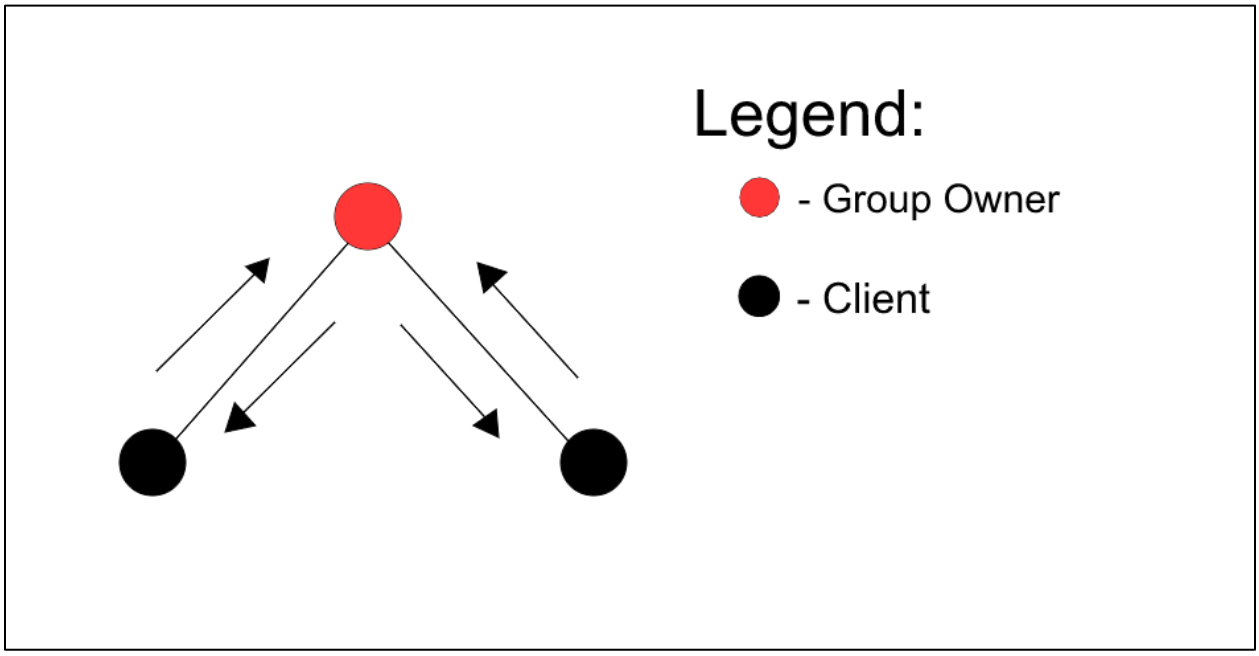

FIGURE 2 – Wi-Fi Direct Architecture resembling Client-Server

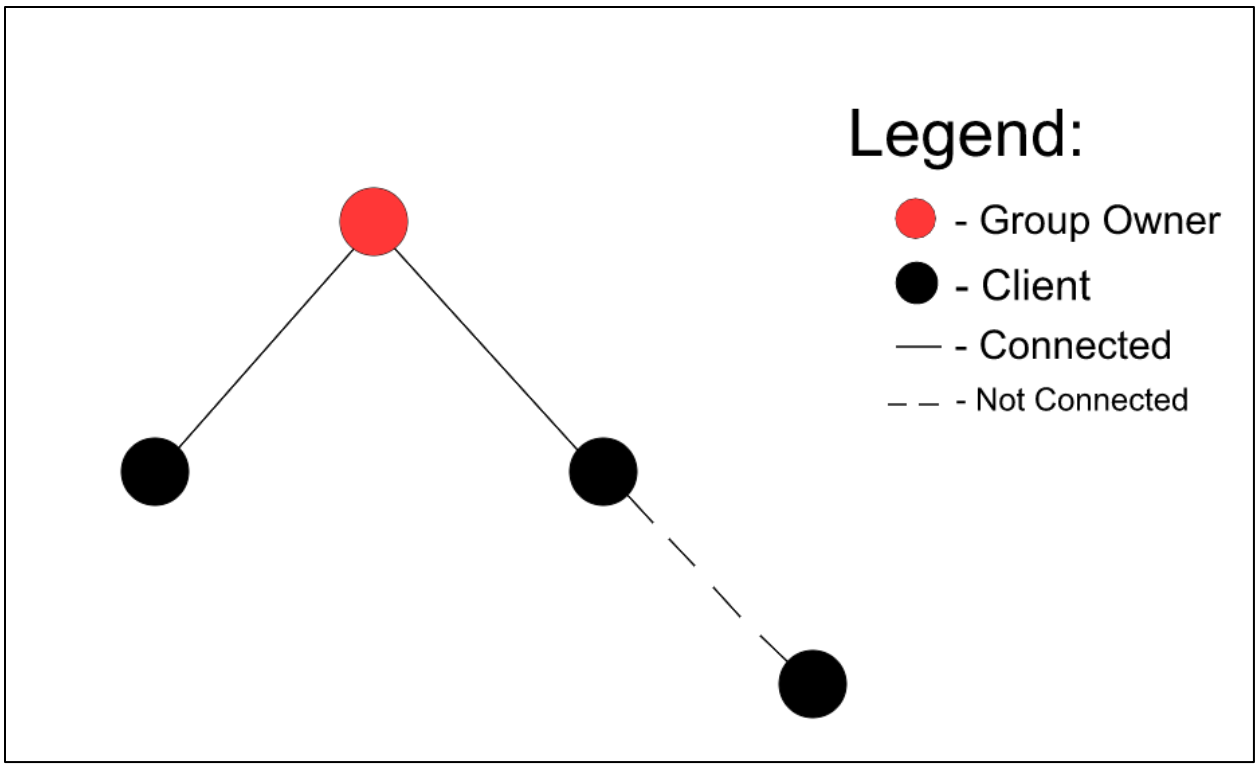

FIGURE 3 – Device Not Connected to Network due to being Out of Range of Group Owner

If Wi-Fi Direct supports concurrent operations, then it is possible that it would be one of the best approaches for this project. The benefit of concurrent operations is that it would allow multiple devices from different ranges to be connected with one another rather than having one singular group owner that is responsible of who is connected and who can join. This would allow any device to connect with any other device and have no restrictions on who is in the range of the owner, thus eliminating the problem shown in **Fig. 3**. However, if concurrent operations were implemented, it may also have limitations, an example being the amount of processing power a mobile device would need and how many concurrent connections it can have at a time.

Wi-Fi peer-to-peer terminology is not entirely correct as it is incorrect. The way this technology was worded makes it very misleading and gives users the misconception that this technology can support larger mobile interconnected networks. The name is not incorrect as it does allow few devices to be connected with one another in a simple manner hence its name. At its best, Wi-Fi peer to peer is a “protocol for forming hierarchical groups, and mostly used to connect just two devices” [ThinkTube, n.d.].

Although this technology gives the perception that it has numerous benefits like being able to achieve high speeds and having greater bandwidth and range than Bluetooth, it will not be used for this project. As stated earlier, this project will investigate this technology but the main objective for this project is to try and form a meshed network. Due to lack of documentation and resources on concurrent operation, Wi-Fi Direct, as is, will not prove to be able to achieve this objective. The main problem with Wi-Fi Direct is that devices must be near the group owner in order to connect and that all messages are routed through the group owner. It is evident that Wi-Fi Direct is not sufficient and that the objective for this project will not be achieved when using this technology.

Bluetooth

Bluetooth, unlike Wi-Fi Direct, has existed for a long time and has had multiple provisioning, which started with Bluetooth 1.0 up to Bluetooth LE and 5, and is now equipped in new mobile devices. The research portion of Bluetooth will pertain to Classic Bluetooth (Bluetooth v2.x), which is more power intensive and does not use Bluetooth Low Energy.

The notable differences in Bluetooth versus Wi-Fi Direct is the range, throughput and bandwidth. The range for Bluetooth is approximately 30 meters while the application throughput is listed as 0.7 to 2.1 Mbps. Other important details about Bluetooth can be found on **Table 1**. These measurements, while not the best compared to the previous technology, are still manageable and could lead to promising results. The range of the technology is not a great concern as this project assumes that the population is dense, and the mobile devices will be near one another, thus range is not an issue. If Bluetooth can support multiple devices, then it is possible that 30m is enough to connect with other devices to build an interconnected network.

Bluetooth networking is generally referred to as piconets and thus uses a “master/slave model to control when and where devices can send data” [Sparkfun, n.d.]. This type of model, evident in its naming, allows up to 8 connected devices where 1 is acting as the master and the other 7 devices are acting as the slave. The master node must do most of the processing power and must coordinate communication throughout the network. The device that is set up to be the master node can send data to any slave while slaves can only transmit and receive from their master. It is important to note that slaves “can’t talk to other slaves in the piconet” [Sparkfun, n.d.]. This is similar to the major concern that was present in Wi-Fi Direct where the master node resembles the group owner and devices outside of this range would not be able to connect. Comparable to the problem noted with Wi-Fi peer to peer, this would cause a problem that is represented in **Fig. 3**.

TABLE 1 – Bluetooth Specifications
Taken from: http://www.rfwireless-world.com/Terminology/Bluetooth-vs-BLE.html

Specifications	Bluetooth
Network/Topology	Scatternet
Power consumption	Low (less than 30 mA)
Speed	700 Kbps
Range	<30 m
RF Frequency band	2400 MHz
Frequency Channels	79 channels from 2.400 GHz to 2.4835 GHz with 1 MHz spacing
Latency in data transfer between two devices	Approx. 100 ms
message size(bytes)	358 (Max)
Application throughput	0.7 to 2.1 Mbps
Nodes/Active Slaves	7

At first, this looks to be problematic because the objective of this project is to create an interconnected peer-to-peer network. However, after conducting more research it seems that it is possible to create a network that could potentially produce the goal of this project. This network would be in the form of a scatternet which is a “type of ad hoc computer network consisting of two or more piconets” [Wiki, n.d.]. To make this work, a node from the piconet must also be connected to a different piconet. The node that is connected to multiple piconets would then relay the communication between the separate piconets to all other nodes (See **Fig.**

4). The management of the messages and the connection to the piconets must be developed separately because “Bluetooth protocol does not support this relaying” [Wiki, n.d.].

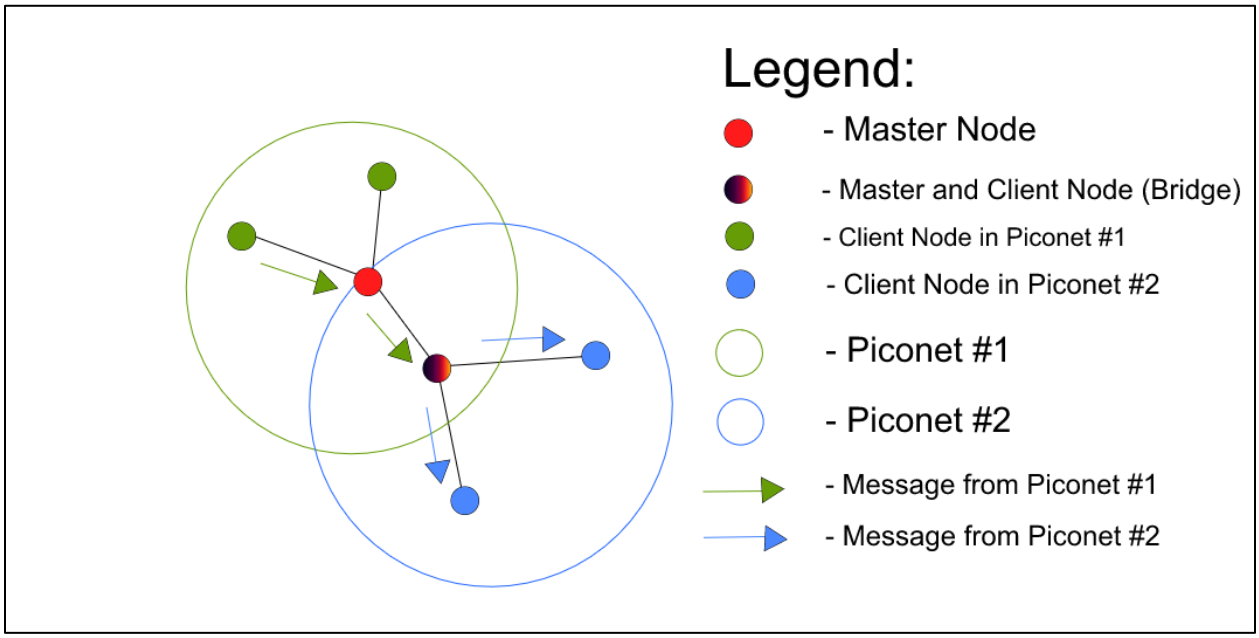

FIGURE 4 – Bridge node Relaying Message from Other Piconet

One of the concerns of using the scatternet implementation is that all the devices in the piconet will still technically rely on one master node. This problem can lead to potential issues if users are mobile and become disconnected, but the ability to connect to any piconet will theoretically reduce this problem. This project will overlook the need to always be connected to a master node which means that the device cannot leave the range of the master node otherwise it will be disconnected. This is overlooked because if the implementation is done correctly then it is expected that all nodes, once disconnected, would be able to find other piconets and thus be reconnected to the network. This was something that Wi-Fi Direct did not support as everyone had to be within range of the group owner.

It is evident that after conducting the research on Bluetooth, it will be the technology that will be used to implement this messaging application. Bluetooth is not comparable to Wi-Fi peer-to-peer when strictly looking at range and throughput, however throughput and range are

not a main concern for this project. The main goal is to be able to allow users to connect to a network, such that there does not need to be one main owner that is responsible for everything. Using Bluetooth, I believe that this goal can be achieved in the form of scatternets. The assumption for this project is that the scatternet will be large enough such that the range of the Bluetooth will not matter since the device would be in range of one to many piconets.

Methodology

As stated, this project will be developed in Android and will not provide any glamourous user interface, but rather focus on the ability to implement a peer-to-peer network where devices will be able to message one another. All the implementation will be done in Java using the Android Studio Integrated Development Environment. The technology that is being used for this project will be Bluetooth, for its ability to form scatternets.

One of the main hurdles in this project was understanding how to develop in Android and learning a lot of the basic techniques. There was a huge learning curve in understanding the graphical user interface. The first key objective for this project was being able to create a screen that had an input box where users can type a message. Once the send button is clicked, the message would have to populate onto the screen to have a visual representation of a message being sent (See **Fig. 5**). This was trivial and was easily accomplished, however there were not any networking implementations attached at this time.

Once this simple activity was done, the next step was creating an easy main activity that would allow the user to decide to either be the master (host) or the slave (client). This was done by presenting two simple buttons laid out vertically that say “Host” and “Client” (See **Fig. 6**). After either button is clicked, a Lobby activity is then initiated with slight differences between each, dependent on whether the device is a host or a client.

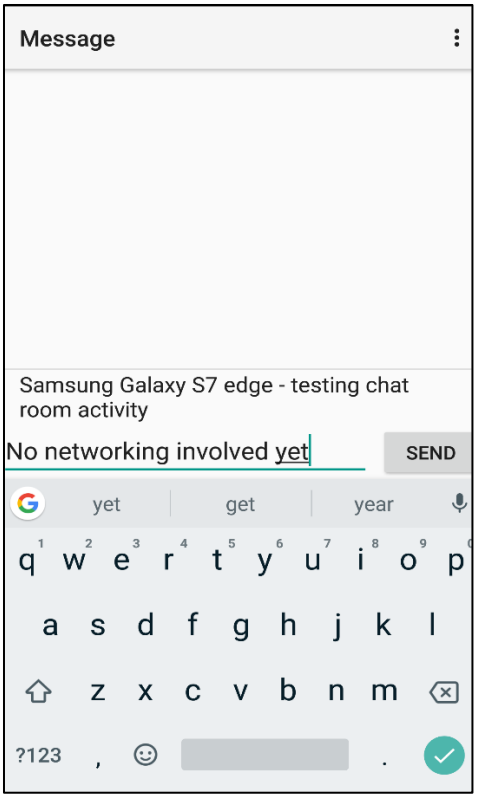

FIGURE 5 – Chatroom Activity Screen

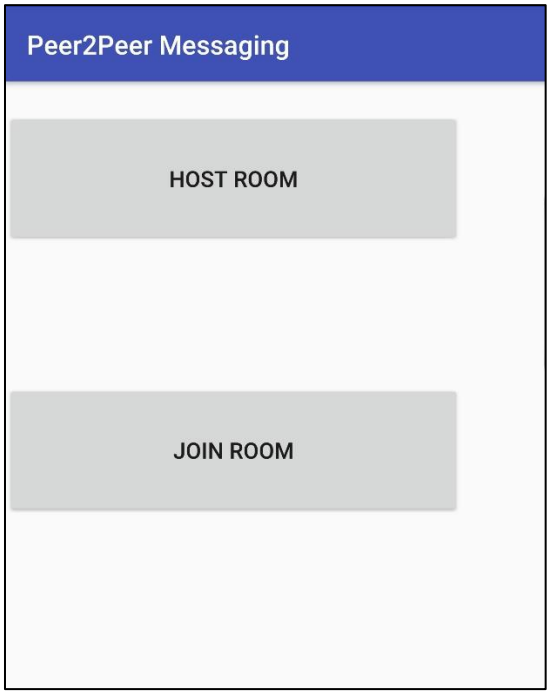

FIGURE 6 – Main Screen to let User Host or Join a room

The differences between the host and the client when in the Lobby activity is that the host device will become discoverable to allow other devices to find it. Currently, the implementation for this is to allow the device to be discoverable up to 300 seconds or 5 minutes. Additionally, the host device will have a button that will let the host decide when to begin the chat room. This button is present because it allows the host to decide when everyone has been connected to the piconet and when other devices can no longer join.

Before the Client gets to the Lobby activity, another screen is launched before it. This activity starts the discovery method and lists out all the discoverable devices in range of this device. There is currently a shortcut for this implementation which is that it will only list the device name, which can be problematic because not all devices will contain a name. This means that if the device is the host and does not contain a name, then it will not appear in this screen, meaning the client would never be able to connect to that host. The reason for this shortcut is because, when the devices are messaging one another, the project will use the device name to display who sent the message. A device name like “Samsung Galaxy S7” is better than viewing a MAC Address that might be confusing. Also, another thing to note is that, for the scope of this project, the behavior of two devices with the same name is unknown, as testing has been limited due to time restrictions.

Once the list of devices in range have been populated, the user will have to choose a device to try and attempt a connection with that particular device. Once the user clicks the device name, the Lobby activity will be displayed and if the name of the connected device appears then that means a connection has been established. If the connected device name does not appear, then there was a problem connecting to that device. Once a connection has been successfully established, the master node will then relay the device name to all participants in the piconet. Once the other devices receive the device name, their list will be updated to account for the newly connected device. At this point, the client will not be able to do anything else and it is up to the host device to start the chatroom.

Once the users are within the chatroom activity, the design is very simplistic where users can type a message and then click a send button. As stated earlier, the messages will be prepended with the name of the device as shown in **Fig. 5**. As well on this screen, at the top right, there is an option menu that will give the device functionality to create another piconet and start a chatroom for that piconet (See **Fig. 7**). This functionality is implemented in the similar way as the host activity except it is located within the chatroom activity. The reason these settings are here is because it will allow a slave node to act as a master node for a different piconet thus producing a scatternet network. The location of the buttons is not ideal in terms of user interface design, however as previously stated, this project will not focus on the layout and user interface of the application.

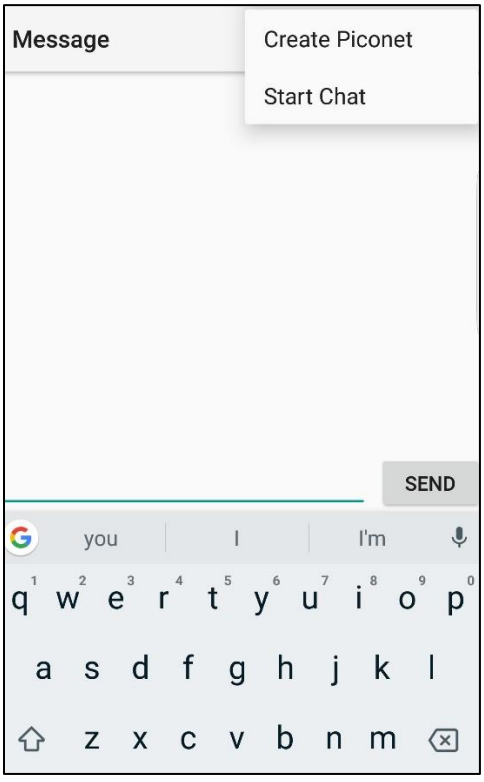

FIGURE 7 – Top right corner has Options to Create another Piconet

The option of creating another piconet will put the device into discoverable mode and let other devices connect to it. Unfortunately, due to the user interface, the device that is the host of the new piconet will still be in the chatroom activity, as that device is still participating in another piconet. This means that the user who created another piconet cannot see who is in the newly created network, which is a downfall to the implementation. Also, since this device is still in the chatroom activity, there needs to be a button that would initiate the chatroom for the client devices and that is why the start chat button is present. Once the start chat button is clicked, the client devices of the new piconet will then switch its screen to the chatroom activity and, in theory, all devices should be able to communicate with one another using a scatternet topology. A flaw to user interface design is that if the host decides to click start room after it has already been initiated then all clients will lose their current chatroom activity and a new chatroom screen will be created. This would produce an undesired behavior as all the clients will lose all the messages that were on their screen.

A minor defect to note is that for this project, a device becomes a bridge if and only if that device turns into a master node of a new piconet. This is produced by the additional options shown in **Fig. 7**. In scatternets this behavior does not have to be the only case for a slave to become a bridge for two piconets. The other scenario where a slave becomes a bridge is when the slave is connected to two master nodes. For this scenario, a slave can only be connected to at most two master nodes producing a bridge between the two. Unfortunately, this feature is not implemented into the project. An advantage to being a slave bridge in comparison to a master and slave bridge is that the slave bridge will only need to worry about 2 connections rather than at most 8 connections. This means that the amount of processing would be less as the slave node will only need to forward the message to the other device. Unfortunately, this scenario is not implemented due to a design flaw of the user interface. It would become problematic to have a slave node join a chatroom and then seek for another master node while still present in the current chatroom activity. Due to the complicated workflow it has been decided to leave that use case out for this project.

Bluetooth in Android

This section will briefly describe in an abstract manner how to use the Bluetooth Android Application Programming Interface. Details on the exact approaches, syntax and object usage can be found in Android's documentation: https://developer.android.com/guide/topics/connectivity/bluetooth.html.

For Bluetooth devices to communicate with one another, they will need to be paired. In order to pair the devices, one device will be listening while the other is trying to connect. For the device that is listening, it must make itself discoverable so that every device within its range can find it. To do this in Android, there is an Intent that can be called that will specify the number of seconds that will let the device be discoverable for. The client device needs to be able to find nearby devices and this is done by turning on the discoverability. Once a device that is nearby is found, access to that device's MAC Address, name and other accessor methods become available. A broadcast receiver needs to be used and will use an intent filter for when Bluetooth devices are found. When the broadcast receiver is called it means that a nearby device has been located and this is when the device's information is displayed to the user so that users can decide what device they want to connect to.

For the host to start listening, the device will need to have a Bluetooth Server Socket object, and this is done by calling "listenUsingRfcommWithServiceRecord" in the Bluetooth Adapter class. The prementioned method will take in an arbitrary name, preferably the application name, and an Universal Unique Identifier (UUID). When a client connects to the device, the client will need to know the same variables that was passed to the method. The reason these variables are needed is because the system will write to a Service Discovery Protocol that will be used when the client is trying to establish a connection. Once the Bluetooth Server Socket is successfully created it must be in the accept mode which is a blocking call and thus is recommended being done on a different thread. Once a connection is established, this call will return a Bluetooth Socket which will be used to communicate between devices. In this project, the Bluetooth will be using radio frequency communication (RFCOMM)

and this only allows one connected client per channel at a time. This means that the Bluetooth Server Socket should be closed immediately after a connection has been established to eliminate the possibility of forming a connection with another device.

For the client to attempt to connect to the host, it will have to use the Bluetooth Device object that the user is attempting to connect to. The client will get a Bluetooth Socket by calling a method “createRfcommSocketToServiceRecord” in the Bluetooth Device object and will need to pass in the same UUID that was used by the host. Once the Bluetooth Socket object is returned, the ‘connect’ method needs to be called. This connect method is a blocking call and should be done on a separate thread similar to the accept call for the host device. If the connect method returns successfully then that means a connection has been made. Otherwise, the connect method would throw an IOException which means that the connection has failed or that the connection has timed out. Once a connection has been formed, it would best to turn off the discoverability so that this device won’t have to continue searching for devices nearby.

When both the host and the client have a Bluetooth Socket object, they are both able to get the Input and Output stream of this object so that both devices can communicate with one another. This will allow both devices to read and write messages to one another.

This shows how two devices can communicate between one another using the Bluetooth API in Android. For detailed instructions and concrete examples, please refer to the Android documentation.

Networking Design

With the design of this project and moving towards the Scatternet approach, it is clear that a host device can support up to 7 clients. Each client can potentially become a master node of a different piconet that can also support up to another 7 clients. This methodology allows users to connect and message one another without having to be in the range of all the devices.

When a device is put into host mode, it will start a thread that will open a Bluetooth Server Socket that is listening on a secure radio frequency communication Bluetooth channel. In the application there are 7 pre-configured universal unique identifier that were created using an online UUID generator. These unique identifiers are used when each new thread is created to listen for an incoming connection. Once the device is put into host mode, a new thread is opened which forms a new RFCOMM channel that will pertain to the specific UUID. The listening call is a blocking call so that thread will wait for a client to connect. When a client connects, a Bluetooth Socket is created that will be used to manage the connection with the client. Once this happens another thread will start that will be using a different UUID to support multiple connections for a single master node. Each new connection that is created is stored in a collection so that the thread can be easily accessible for write operations. If a connection is lost within that thread, the thread will be stopped and then removed from the list. This means that there can be at most 8 threads per device. There will be 1 thread for this device connected to its master node and then 7 slave nodes connected to this device which is the master node for another piconet.

When a client device tries to connect to a host device, the device will create 7 threads. Each separate thread will try to connect to a different UUID. The reason for this is because the client does not know what channel the host is listening on and that is why the client must iterate through all 7 UUIDs to find a match. Once a match is found, the thread is stored into the same collection in order for it to be easily accessible for managing the Input and Output streams.

When dealing with a Scatternet, the master node will have to manage all the data coming in and relay it to any other network that it is connected to. This is done by using the collections of connected threads. Every time a connected thread receives a message, the device will iterate through the list of connected threads. This device will relay the received message to all the connected threads that is not equal to the thread that received the message. The process to do this is trivial as there can only be, at most, 8 connected threads. This results in the device having to relay to an upper bound of 7 other devices since it is receiving from 1 of those 8. One problem to note is that the write methods to the other threads are in an unsynchronized matter so that means that it is possible that messages on a device can be displayed differently than on another device. For the scope of this project, there was no check for validity and timestamps of messages. This project was solely concerned on being able to transmit messages to multiple devices.

The messages that were being sent to the devices were set up to be very simple for this project because encryption and parsing the messages was not of high priority. Depending on what was being sent to the devices, there was always an integer at the beginning of the message to indicate what was being sent. Each device would receive the message, parse the integer and act accordingly to what the integer number was. For this project there was only 4 integers starting at 1 to 4 and they were: start chatroom, device name, device lost and message respectively.

An integer with 1 at the beginning would indicate that the master node clicked the Start room button and this would mean that all the clients would react by switching their screens to the chatroom activity. This integer was needed so that the client can react to when the host wanted to initiate the chatroom. Since there is a Start room button within the messaging application, this means that if the host decides to click that button again when everyone is already on the chat room screen then all clients' devices will refresh to a new chatroom activity.

Messages that starts with an integer of 2 or 3 are utilized by the Lobby activity so that those screens can be updated accordingly. When the message begins with a 2 that means that a device has been added to the piconet. Every client will react to the number by appending the device name to the collection of devices in the Lobby screen. Similarly, a message that begins with a 3 pertains to a device that has lost connection to the network. When a device is no longer connected to the master node, all the clients in the piconet should update their lobby screen to remove the device name from the list.

Lastly, a message that begins with a 4 means that it is a message that was sent by a user. The message in the data payload should be updated on the chatroom screen and all the devices within the network should receive the same message. The size of the data will be dynamic as it depends on what the user is trying to send. As stated in **Table 1** the length of the data is limited to 358 bytes when using Bluetooth. In the current design, there is no validation for the size of the message which means that a message that is greater than 358 bytes will result in an unknown outcome.

This design of the data allows the clients to react in a uniform way, as the data is meant to be received by all the nodes on the network. Prescribing the data with an integer at the beginning allows every receiver to easily parse it and determine how to react. Since the receivers do not care about where the data originated from, the receiver will update its screen accordingly and relay the message depending on whether it is a master or a bridge node. The design of the data was meant to be open in this manner as it eliminates a lot of the processing power a device is responsible for. One downfall to this approach is that since the data is transparent and every client reacts to it, there is no way to transmit private messages to a particular node. As stated earlier, encrypting a message will not be within the scope of this project as this project is solely focusing on being able to connect to and transmit messages to multiple devices.

Build Setup

This project was developed for Android devices that are compatible with Bluetooth and is done using Java and Android Studio. The version of Android Studio used was 2.3.3 however it should be compatible with any newer version. This project uses specifically Java 1.8.0_122 however any version of Java 1.8 should be able to work with this project. Lastly, no third-party API was used in the creation of this application therefore no additional set up is needed.

The Android Studio project will be submitted with this report as a .zip file. To begin, extract this file into the Android Studio workplace. Once all the files have been extracted, in Android Studio, click “Open” and locate the “P2PMessaging” project in the extracted directory. When the project has been selected, click “Ok”, at this point Android Studio should begin to build the project and download any dependencies that are required for the project. In the Gradle Console, there should be an output indicating a successful build. As stated earlier, this project does not rely on any third-party API, so the build instructions are simple for this project.

The project is set to be compatible with Android SDK version 21 (Android 5.0) up to version 25 (Android 7.1). These versions correspond to Android Lollipop and Android Nougat respectively. The tests that have been done for this project used real devices running Android 5.0 and Android 7.0 but has not been tested on Android 6.0 however there should be not be any problems. Lastly, the brand of the devices does not matter as the tests were using LG and Samsung devices.

The preferred method on porting this application to devices is by using Android Studio. In the IDE, click the “Run” button and this will display a list of connected devices. Click the intended device for the installation of this application and once it’s installed successfully the application should be executed. It is important to note that that turning on Developer options is mandatory for this process. Lastly, another approach to install this application is through an APK which can be created in Android Studio and then transferred to the device. After these steps have been completed, the application should be able to run on the device.

Other Strategies

This project was done using Classic Bluetooth that matches to version 2.x but there could have been many other strategies to try and implement this project. One of the other strategies that was ruled to be insufficient was Wi-Fi peer-to-peer due to its lack of concurrent operations. If that feature becomes available, then Wi-Fi peer-to-peer would be another approach to tackle the ability of creating an interconnected messaging network. Another interesting approach to this project could have been Bluetooth Mesh.

Bluetooth Mesh is a new version of Bluetooth that was announced in July 18, 2017 that enables many-to-many device communications. This is different than the current Bluetooth that is being used for this project which is described as a Point to Point network. One of the advantages of using this technology is that it "operates on Bluetooth Low Energy (LE)" [Bluetooth, n.d.] allowing devices to save battery. The main advantages of this technology are that "it's also inherently peer-to-peer, allowing all nodes to communicate directly with one another" [Bluetooth, n.d.] and has a publish and subscribe messaging model. One of the most important points about Bluetooth Mesh is that it can manage all the communications for the users and is based on a publish and subscribe model that would easily allow the devices to have more precise messaging. Using this technology would allow the project to grow and add more functionality like being able to create groups easily and only have certain messages reaching certain people. Lastly, Bluetooth Mesh provides a lot of the security implementation which is another factor that the developer does not need to worry about.

Unfortunately, due to the timing of the Bluetooth Mesh release and timing of this Honours Project, this technology could not have been used. However, it is evident that Bluetooth Mesh would have been a great approach for this project as it would have provided many useful features that could have expanded this application into something very versatile.

Results

The final product for this project was a messaging application that is developed for Android using Bluetooth as the networking technology. The application can connect multiple users to one another and can demonstrate a Scatternet topology. A major point to note is that the user interface and the workflow for the application is not perfect however that was not the key objective for this project. The main goal was to try and implement a peer-to-peer network using Bluetooth or Wi-Fi peer-to-peer and that objective has been successfully completed.

Some important issues that occurred during this project is the understanding of networks, how they work and how to implement them using the technology chosen. One example of this is discovering that a piconet can only support 7 slaves. If a slave node could not act as a master node, it could have created a major setback for this project because multiple piconets would not have been able to be interconnected. Luckily, Bluetooth follows the Scatternet topology thus it is possible to be implemented using this technology.

Another frequently recurring issue with this project was how the Bluetooth specifications and application programming interface worked. Having an understanding of how to implement Bluetooth took some time as there were many different protocols and approaches of Bluetooth that could have been used and are currently available. Trying to understand the abilities provided by Android and Bluetooth was difficult and a major component to this project.

As stated throughout the report, there are still many minor bugs that can occur when running the application as time was an issue for this project. However, the core functionality exists, and this can be observed by running through some of the test cases. Due to the nature of the emulator and not being able to support Bluetooth, the test cases of this project were very limited, as obtaining multiple Android devices were difficult. Another difficulty with this project is when the test cases dealt with the out of range devices. Moving devices far away

from one another becomes troubling and unpredictable as this is usually dependent on the environment. For this project, test cases were executed with at most 4 Android devices running different scenarios. The scenarios included all cases where a device was a master node and/or a slave node. Theses scenarios can be observed in **Fig. 8** and the range of the devices should not matter as long as a client is in range of a master node then the client should be connected to the network.

It is evident that this application is able to have devices connect with one another even if some of the devices are out of range. This is a very useful use case in real life as it allows many technologies to be able to communicate with one another. This can be used when people are going on vacation and have adjoining rooms and want to communicate. A quick message at night to say, “Remember to pack your swim trunks” is very useful in this case because they would rather leave a message than potentially wake up their friends during the night. With this application, they would be able to do that as well as connect other people to build greater network.

Another useful case for this application is the possibility to wipe out the use of a service provider. This would require that everyone uses this application and since the average human has at least one mobile device containing Bluetooth, this could be feasible. This kind of use case may be too farfetched, however in cases when the service provider cannot reach its clients, it seems to become more obtainable. These cases would be ideal during concerts or rallies when the population becomes too dense and the service providers have difficulty reaching their customers. During these use cases, if everyone had this application, they would all be able to communicate with one another and still stay in touch. The greatest part is that they don’t have to be in the same range as one another as they only need to be in range of a person who is already on the network. Thus, the usefulness of this application solely depends on the usage of it by the people in those cases.

The interesting factor to this project is that it allows multiple devices to connect to one another using a technology that would otherwise limit it due to range constraints. The ability to do this is very interesting and it can be used for a multitude of scenarios that uses Bluetooth. Since the world of Bluetooth is continually growing from being implemented into headsets, cars, trackers and many other things, it seems that this project could help in the improvement of the range connectivity issues.

This project shows its importance because it is following along the same direction as where Bluetooth is going with their new implementation of Bluetooth Mesh. The difference with this approach and Bluetooth Mesh is that the latter will have all the implementation embedded into the specifications of Bluetooth. Additionally, the approach that was done in the project allows devices with older Bluetooth hardware to be compatible and mimic a Bluetooth Mesh behavior.

This project has successfully accomplished most of the goals that were set at the beginning of the term. Investigation on the differences of Wi-Fi peer-to-peer and Bluetooth as well as the advantages and disadvantages of using them has been conducted and analyzed for this project. Additionally, the ability to communicate between two or more devices has been accomplished with a simple user interface where users can message one another. All of this was developed in Android using Java and the Classic Bluetooth that the Android API offers. There have many challenges with this project, but many lessons have been learnt like the development of an Android application, networking and Bluetooth specifications. In conclusion this project shows the potential of Bluetooth and how it can have a large impact on communication between multiple devices.

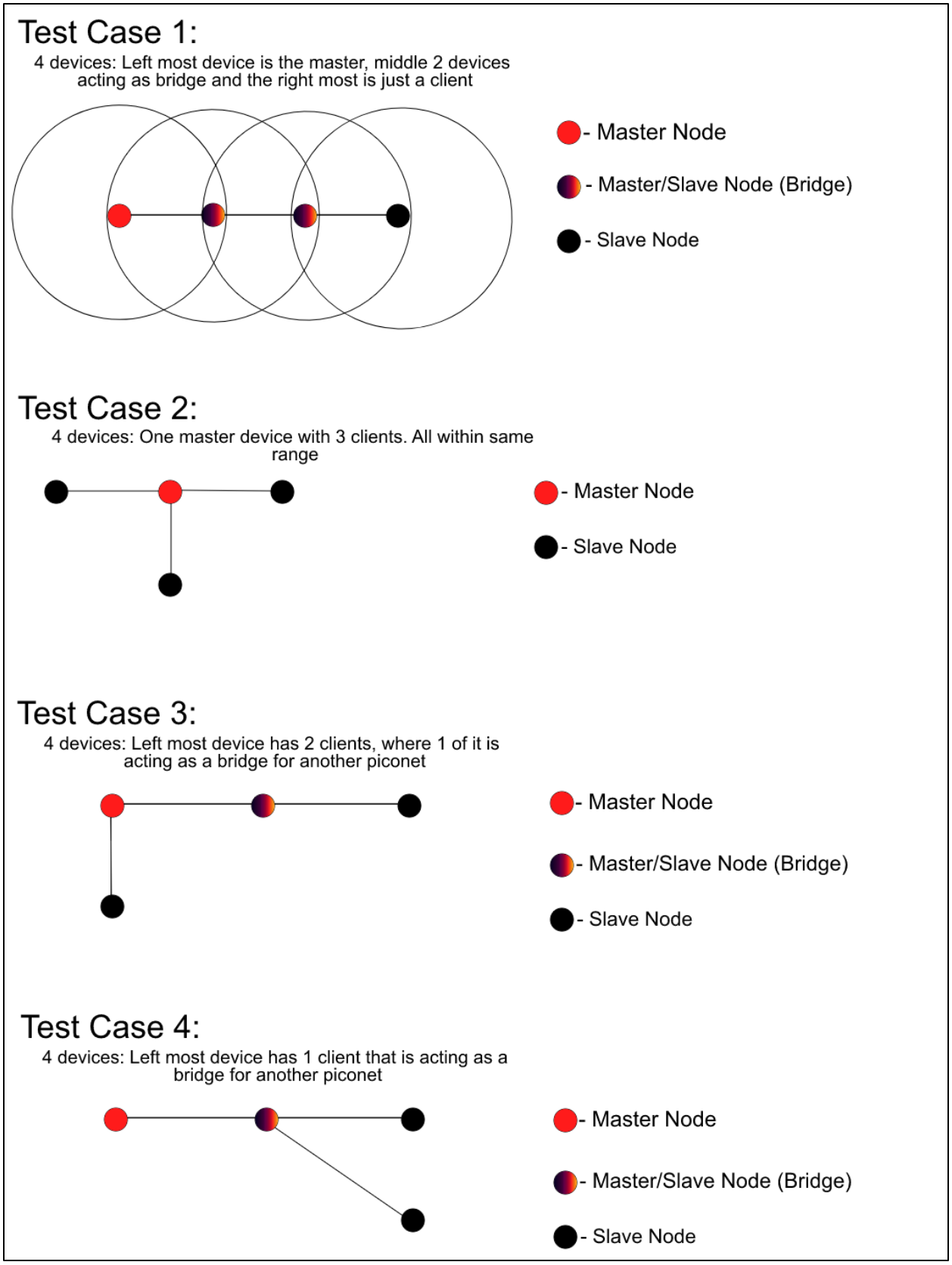

FIGURE 8 – Test Plan with 4 devices

References

Anon. (n.d.) Mobile Phones Smart Phones | Bluetooth Technology Website, https://www.bluetooth.com/what-is-bluetooth-technology/where-to-find-it/mobile-phones-smart-phones

Bluetooth. (n.d.) Blueooth mesh networking FAQS, https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/le-mesh/mesh-faq

Sparkfun. (n.d.) Bluetooth Basics, https://learn.sparkfun.com/tutorials/bluetooth-basics

ThinkTube. (n.d.) Wi-Fi Direct, http://www.thinktube.com/tech/android/wifi-direct

Wiki. (n.d.) Scatternet, https://en.wikipedia.org/wiki/Scatternet

Wi-Fi. (n.d) Wi-Fi Direct, https://www.wi-fi.org/discover-wi-fi/wi-fi-direct