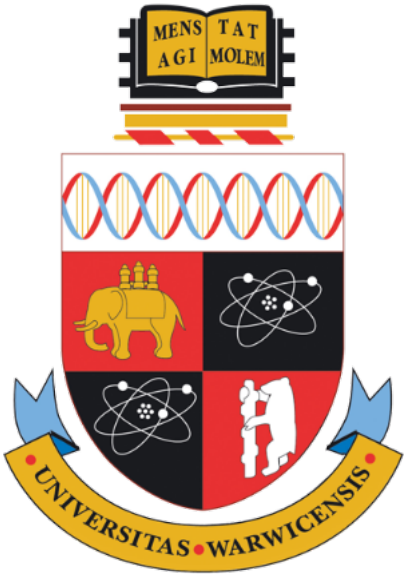


Investigating The Use of Ad-Hoc Networking in Emergency Situations to Share Location Data

Spreading information to emergency services during an emergency service when communication resources are limited

Final Report

Frederick Brown
u1716717

Supervisor: Dr. Matthew Leeke
Department of Computer Science
University of Warwick
2019-20

Abstract

In today's ever more connected world, disaster scenarios result in situations in which communications channels can be disrupted when they are most needed. It is critical that research must be done into systems which can help to allow survivors to communicate important information to the emergency services. This project investigates the feasibility of such a system to be deployed using existing, popular, and readily available technologies, such that it can be used to enhance responses to such a disaster. It proposes a system which utilises the benefits of Ad-Hoc Networking to allow survivors to pass information to the Emergency services increasing their knowledge base and therefore react better to the situation at hand.

Keywords: Bluetooth, Ad-Hoc, MANET, Emergency, Disaster, Epidemiology

Acknowledgements

Over the course of this project, I have received immeasurable guidance and advice from those around me. Firstly, Dr. Matthew Leeke has guided me through my project and has given me incredible assistance throughout my project and has pushed me to further my abilities in Computer Science, as well as the bounds of this project.

Secondly, I would like to acknowledge the support of my friends who have taken an interest in my project throughout its course and acted as sounding boards for my ideas, giving suggestions and helpful criticism, especially Alistair Robinson who has been invaluable to me being able to write my Final Report to the level that I have.

I would like to also acknowledge the work of the people involved in the BlueZ project. Their work and example code increased the speed at which I understood the concepts involved in the project, which enabled me to implement my design.

Lastly, I would like to thank my family for supporting my work this year and throughout my degree, as well as pushing me to achieve everything I have done, and will do.

Glossary

MANET - Mobile Ad-Hoc Network
VANET - Vehicular Ad-Hoc Network
VM - Virtual Machine
BCS - British Computing Society
WSN - Wireless Sensor Network
AODV - Ad-Hoc On-Demand Distance Vector
DTN - Delay Tolerant Network
DAG - Directed Acyclic Graph
BLE - Bluetooth Low Energy
Node - Synonym for "device"
AP - Access Point
DHCP - Dynamic Host Configuration Protocol
PAN - Personal Area Network
DDOS - Distributed Denial of Service
OOP - Object Oriented Programming

Contents

List of Figures

Listings

Chapter 1

Introduction

The purpose of this project is to investigate how MANETs can be used in a disaster scenario to enhance the response of the Emergency services. It aimed to investigate ways in which survivors could be better connected using existing technologies so they would have some way to exchange information. Utilising this idea, the project looked at what information would be needed to help the emergency services to help people. Building on this premise, the project then looks at extended, more advanced behaviours for nodes. The benefit of this is that it adds more utility to the system and it shows that it works in a variety of different scenarios and use cases.

Distilling this problem to what is important is a challenge in itself. How do the emergency services gather information when there is no available infrastructure for them to communicate with survivors over? There are some existing systems which to help the emergency services, such as one developed by FEMA [3], a branch of the US Government focused on disasters. Their systems, such as *Mobile Emergency Response Support* and *Mobile Communications Office Vehicles*, focus on providing a connection with the outside world. While this is incredibly useful, and crucial, it still leaves out connecting survivors who are displaced or trapped.

The project looks at several parts of a system such as this. It investigates the technologies which could be used, data privacy, packet routing, threats to the system as well as different ways this project could be implemented.

1.1 Motivations

In society today, we have become accustomed to being highly connected. Our telecommunications systems are advanced enough that they connect people all over the world. A very important use of these resources is contacting the emergency ser-

vices. During a disaster, such as the 2017 hurricanes in Puerto Rico, the lack of telecommunications services contributed to a 62% increase in mortality [4]. Many remote areas of the islands were hurt most, with many without cellular data services for up to 41 days. Another example of this is Hurricane Katrina, where the high winds of the hurricane destroyed many antennas used by cellular providers [5]. This causes an information gap for rescuers, which makes it difficult for them to know how to help most effectively. Emergency services must move quickly, as survivors have a higher chance of survival if rescued within the first 72 hours after a disaster [6].

Having a method of communication, whether automatic or manual, will help ensure some form of connectivity for survivors so they know people are coming to help them. This underlines the importance of ensuring connectivity between people. In 2016, the United Nations passed a vote which made internet access a human right [7]. Having some form of communication in unsteady times is a human right, meaning systems need to be in place to ensure this.

During the course of the project, the world was struck by a Coronavirus pandemic. In response to this, large numbers of companies and academic institutions turned their focus towards helping those in need. One such response was by Apple and Google [8]. They worked together to create a contact tracing system. The aim of this is for phones to communicate over Bluetooth LE and exchange unique keys to identify each device. This information is stored by each device and can be used to determine if a user has come into contact with anyone who has tested positive for COVID-19. This is an important tool to gather information about social interactions and allows better tracking of the spread of disease. This product is very similar to the system developed in this project. It shows that the design is a viable project with a real world use case. This product shows it would be possible to enact at scale and would provide utility to citizens using it.

1.2 Project Aims

When outlining the project, the different aims of the project should be considered:

- Conceptualise emergency information dissemination system
- Create proof of concept emergency information dissemination system
- Investigate issues about data privacy
- Investigate how a system would perform at scale

- Possible enhancements to the system which could be made in the future
- Different ways this project could be used

These aims should be considered throughout the project. With each decision made, it should be made with these in mind. The goal of the project is to investigate the possibility of a system which could enable message dissemination in an emergency scenario. Using these aims helps to guide the project and keep it on track.

1.3 Stakeholders

The primary stakeholders of this project are people who will benefit from this system immediately. The emergency services would benefit from having more information about survivors so they can tailor their response. Survivors would also benefit from this kind of system as it could enhance survival rates during a disaster. This is especially true for areas of the world which are prone to natural disasters, such as being on a fault line, those who live in tropical weather locations, and areas which are prone to man-made disasters, such as wars or terrorist attacks. All of these groups of people would benefit from a system which feeds back information about survivors who may have been displaced or confined to a certain area.

Chapter 2

Research

This chapter will attempt to illustrate the research that was done for this project. It helped to give ideas for the different areas of the project. This section will go over a number topics which are relevant to the problem area, such as related work, networking protocols to build the system on and the security practices to use to ensure users information is protected.

2.1 Related Work

Natural and man-made, disasters may damage existing infrastructure or put a strain on it, for example, an earthquake might destroy cellular infrastructure preventing survivors from using it. This makes an emergency network difficult to implement as this centralised infrastructure could be unavailable. This contributes to disasters being difficult to navigate due to degradation to communications networks [9]. A solution to this is to utilise a distributed network. Having a system which uses different means to operate over could enable more survivors to be saved [10], due to it providing additional avenues of communication. For this type of system, both academic and commercial solutions to similar problems are have been investigated to identify overlapping problems to analyse their solutions.

2.1.1 Academic

HIRO-NET

In an academic context, this, and many adjacent problems, have been attempted to have been solved. One such example is HIRO-NET [11]. It is comprised of 2 tiers:

local meshing using BLE to connect survivors devices together, and an upper level to connect these local mesh networks over a wide geographic area. In the solution proposed, a node will start off as a client searching for a valid server device. After a client timeout expires, the client will become a server for other clients to connect to. By creating these local client-server piconets, they can join together into larger, local scatternets. Within these networks, if a device still maintains an outside network connection, other devices can route data packets to the connected device so they can be sent over the internet. Server nodes are used to route these packets within the scatternets so they are handed to the correct device. Furthermore, proposed within this paper is the use of more expensive, heavy equipment, such as drones and mobile satellites, to connect and join up these scatternets and piconets over an even larger geographic area. HIRO-NET is a highly complicated system, requiring a lot of complex operations. In principle, the idea of using the client-server model in BLE is an interesting use for this project as it is a widely used technology, and is low power and efficient. This is perfect for a disaster scenario where resources will be constrained. Other aspects of the project, such as connecting large mesh networks, may be unworkable for this project as it runs on the assumption that devices have limited resources and are coming in and out of the network. HIRO-NET requires a more permanent, slow-moving network topology than that which this paper is investigating.

Message Dissemination

A much more simple approach to the problem area is to focus more on the emergency services, and providing them with as much information as possible [12, 13]. This approach strips away a lot of other aims and focuses purely on information dissemination. This approach takes some information, such as geographical information, and sends it to some central node. This allows the information to be collected and aggregated. In one example [13], the author proposes using Bluetooth or WiFi Direct. These systems could be used and could play a significant part in implementing such a system being proposed. The author suggests using WiFi Direct as it has a larger range and is more widely available on devices. Furthermore, the author states that WiFi Direct can provide a large speed increase over the other two options, which has large benefits for information dissemination.

Building on this, other similar systems have also been constructed using WiFi Direct. In [14], the author proposes using multi-group communication to send emergency information to other devices which it is connected to. This is done by forming groups of devices and broadcasting information to all of them.

VANETs

More recent, and applicable, examples of Ad-Hoc networks is in the automotive industry. Over recent years, cars have become more complex and have increased their computational power. We can consider how they could be used to form networks or connect when they come into contact with each other. There have been several studies [15–17] which have investigated the properties, and use cases, for these types of networks in this problem area. Shown in [15], VANETs can communicate with each other when they come into contact with each other, by driving past one another. When this contact occurs, they could form a connection and exchange information. Using similar data to other examples already discussed, such as geographic data, traffic jams and other real-time events can be monitored by transmitting this information to a central authority. In the context of VANETs, they might have access to cellular communications so could transmit this data to a server far away, but the concept would still be able to work well regardless and nodes would be spreading information about who they are, where they are and any other relevant information. In addition to this, there are other issues, such as security and routing [16, 17], which this project also shares. They investigate using novel ideas, such as geographical routing as well as other, more established routing protocols, such as AODV [16, 18]. The VANET problem area is crucial to this project. Although they occur in separate domains, and have different aims, the underlying problem areas are very similar. Both have to consider movement as a factor when designing systems and algorithms, as they will have to be dynamic and not rely on a fixed network topology. This is especially true with contact communications, where exchange happens when 2 devices come into contact, before going their separate ways. Another trait they both share is that they occur in an ongoing scenario where conveying location is important to understanding where nodes are in respect to each other to coordinate a response, such as an autonomous car taking a different route or the emergency services putting more rescuers in areas where they know there is a greater density of survivors.

2.1.2 Commercial

Looking at examples in a commercial context is a critical aspect to understanding a solution to this project. The main reason being is that many similar products will come with the added benefit of actually working at scale with many users. It vindicated the different choices made, if similar.

A major area for investigation is location tracking devices. Examples of these include Tile and Trackr [19, 20]. These two companies produce devices which allow a user to track devices they attach them to. This could be a bike, wallet or keys, for

example. These devices use BLE to communicate with devices running a required app. They attach to items like a tag or a sticker. They will communicate to any device in the network to relay data about itself. The devices (which are internet-connected devices, like phones), send this information to a server which will collate this information so that their last known location is viewable to the owner. This is especially useful when the object is lost and the location is unknown by the owner. A system such as this has numerous overlapping attributes with the sort of system needed for the problem area this paper is investigating. These example commercial systems take the established client-server model and use it in a way such that it acts as a decentralised system of interacting nodes. These use 2 tiers of nodes: the small, lower-powered tags and more powerful devices, such as phones. These can be partitioned into server and client nodes respectively. A network made up of such devices is could be what a potential emergency network would look like: a heterogeneous network where devices, from fitness trackers to powerful phones, can communicate with each other for short periods.

Other similar applications to this are in fields such as healthcare [21, 22] where BLE is being used to transmit information to a device nearby. Although it is an example which is slightly removed from the problem area, it does display that WSNs can be used, and relied on, by critical systems which users rely on to survive.

2.2 Networking Protocols

Considering [23], there are several different factors to consider when choosing a network protocol, such as something not being able to be fast and cheap, or that different problems need different solutions. For this reason, different protocols need to be considered. Aspects to consider include:

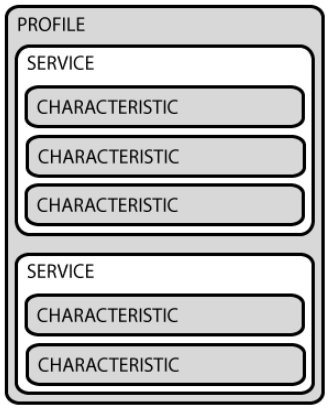
- Number of devices shipped
- Range
- Data transfer rate
- Ease of use (Both for developers and users)
- How applicable is it to the problem area?

Using this list of requirements, three different technologies were identified. Each one of them is used in several different commercial products and has ways in which developers can use them. As they are already used in commercial products, they

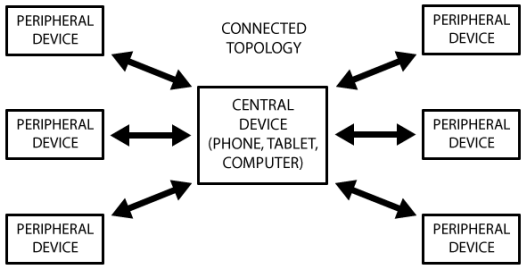
have industry support, which means they are technologies which are suitable for the project and problem are at hand. This section will discuss them further and discuss the benefits and drawbacks of each.

2.2.1 Bluetooth

Bluetooth [24] is a well-established link layer protocol with 2 major versions. The larger, more established version is Enhanced Data Rate (**BEDR**) [25]. This is an enhanced version of the basic Bluetooth which enables faster connection, delivery and higher transmission speeds. The downside to this is that connections are still bulky and it uses a lot of power.



(a) BLE GATT Server Structure



(b) BLE Client-Server Model

Figure 2.1: Diagrams to display how BLE works [1]

Low Energy

To overcome this, the alternative to BEDR is Bluetooth Low Energy (**BLE**) [26–28]. This is a newer variant of Bluetooth (released in 2010) that focuses on lower-powered devices. A major difference is that BEDR has to keep sending packets to maintain a connection, whereas BLE does not. It will enter a "sleep" mode, except for when the connection is initiated, and will only wake when it is needed. This hugely reduces the amount of packets send, thus reducing the amount of power used by a device, but does this at the expense of a higher data rate. It means it can be used by devices with very few resources, such as a single button cell battery. It also still adheres to the client-server model, but not in a usual way.

BLE devices use the Generic Attribute Profile (GATT) and Generic Access Profile (GAP). GATT defines how they communicate, GAP defines how they connect and advertise services. This means they have specific roles within their relationship. Figure 2.1, shows the general structure for how clients (peripherals) and servers (central devices) communicate with each other within GAP. The client will be the one to initiate a connection with a server. It will issue other requests for data and may accept responses. An example of a client-server interaction would be a smartphone and a fitness tracker. This will receive and respond to requests made by a client. The server may contain information that the client wants, and can communicate with up to 8 clients at once.

Initially, a connection will be made. The client will scan for a suitable advertising server within range. This is typically <100m. The suitability of a server will be based on the information in the advertising packet. This includes a unique identifier (UUID) as well as the services it provides once connected. The client will then issue a connection request. Once connected, they can then exchange data packets.

The format of data exchange between a client and a server is defined by GATT. A server will have registered several profiles to advertise their services. Within these profiles, are defined services. The structure of this is shown in Figure 2.1a. If we consider Object-Oriented Programming (OOP), you could liken the Service to be a class and a characteristic as a function or variable. A service will encapsulate different characteristics together. If we use a profile for smartwatches, it may have services for *messaging*, *health*, *wrist position* and *battery level*. *Wrist position* and *battery level* will be simple, only 1 characteristic to provide a single value. *Messaging* and *health* are more complicated and will comprise of many characteristics. Furthermore, within characteristics there are methods, enabling a client to read a value from it, write a value to it, and request for it to send a response whenever its value changes (useful in the *battery* service).

This provides a simple API for a client to connect to and communicate over. It provides clients with transparency of what each server can do and allows them to connect if they need to. This is ideal for this project as bespoke behaviour will be needed on both the client and server sides. BLE is efficient, allows for quick connections and doesn't need to exchange large amounts of data. This is very effective for the emergency services problem space where resources are constrained and being quick is ideal.

2.2.2 WiFi

WiFi is a widely-used communication standard, mainly used for providing wireless internet access to devices in a specific area [29, 30]. It works but having an access point (AP) which devices connect to. These will communicate with connected devices and will can allow these connected devices to pass data packets which will be sent to a specific address using TCP/IP [31]. The access points can also initiate communication with a connected device and send it packets which have been addressed to a device which it is connected to. This main version of WiFi is known as *Infrastructure* mode, and is heavier, and more complex than would be needed for this project.

Direct

Building on *Infrastructure* mode, a variation exists called WiFi Direct. In a similar way to how BLE and BEDR are compared, Direct is a lightweight version of WiFi which focuses on low powered devices. It does not require the presence of an AP which allows a nearby device to communicate on an Ad-Hoc basis [32].

There are 3 types of nodes in this, Group Owners (GO), Group Members (GM) and Legacy Clients. Legacy Clients are those which support older versions 802.11, but can still partake in the network as a GM. These are not preassigned roles, but ones which are decided after connection. During group formation, devices will select a GO using leader election, whereby they select an *intent value* and communicate this value to all other devices. The device with the highest *intent value* is the GO. This will act in a similar way to how an AP would. All other nodes are GMs. The GO will assign an address to the GMs for them to be known by, using DHCP.

The GO will be in charge of routing traffic between nodes in the network, as well as advertising the existence of the network to devices outside the network. This works the same way as it would do for any 802.11 AP. Much like how a Bluetooth piconets can join together to for scatternets, multiple Direct groups can be used in conjunction with each other by having a node in both groups. This could be the GO of one group and GM in another, or a GM in both groups. This enables multi-group communication.

There are various examples of solutions in our problem area that utilise this technology, such as [14]. This illustrates that it is a promising technology which could feasibly be used to accomplish the goals of this project.

2.2.3 ZigBee

Although WiFi Direct and BLE are the most prevalent technologies in this area, mainly owing to their popular predecessors and commercial use, other options should also be considered. ZigBee is another IEEE 802 standard (802.15.4 [2]) used for short-range, low power data communication. A common application for this is in smart home appliances. Examples of this include the Amazon 2^{nd} generation Echo Plus, Phillips Hue and IKEA, among many other examples [33]. The smart home problem are overlaps with our problem area significantly. Both work in areas with many devices wanting to communicate with each other. In both situations, a medium is needed to communicate over quickly and which allows finite, or limited, resources to be used efficiently.

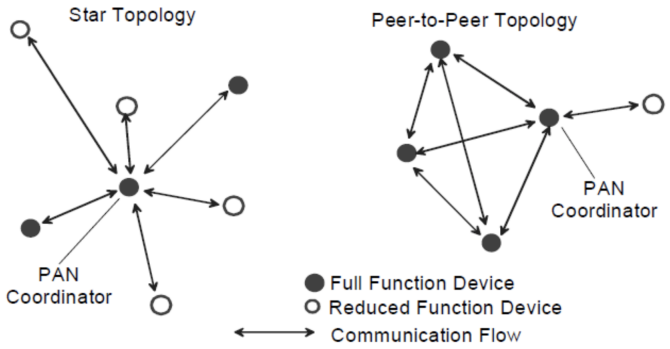


Figure 2.2: Different topology configurations of ZigBee [2]

A ZigBee network has a main *coordinator* which is responsible for organising and maintaining the network [34], called a PAN. All communication goes through this and all devices in the network are managed by the coordinator. Depending on what is required by the application of the technology, it can form 2 different topologies: star-based or peer-to-peer. These are shown in Figure 2.2. This project is focused on the peer-to-peer topology. In this, most devices are equal coordinators, meaning they can all talk to one another, providing they are in range of each other. Each PAN has a unique identifier [2] which could be used to identify where packets have originated from and what route they took to reach their current location.

This technology is ideal for the emergency network problem area. Most nodes can be the same type of node and be fundamentally equal. This can allow for more complicated functionality of the network at the application layer. Furthermore, there is another type of node, this could be used for devices which are extremely battery constrained, such as those on low power, and would still enable them to participate

in such a network. The only difficulty with ZigBee being that not many popular portable devices support it, meaning in such a disaster scenario, it would be infeasible as there isn't the industry support for it yet.

2.3 Security

Security is a highly important aspect of this project. The ramifications, both legal and social [35], can be significant if it is ignored. This project centres largely on collecting and passing on sensitive information, such as location data, to other devices and network participants. For this reason, it is crucial to consider how to protect this data and ensure both confidentiality and integrity. If these are not maintained, data could be harvested and used against network participants that share their data. For example, an adversary could collect the data, and target houses where they know inhabitants are not residing. This enables post-disaster looting, which was a major problem post-Hurricane Katrina in 2005 [36].

We must have a method by which we can judge the effectiveness of provisions taken in this project, to better ensure a users security is dealt with effectively. A model to do this is **STRIDE** [37]. It stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of privilege. This is used to model threats to the project, and allows us to reason about different threats and how to overcome them.

A method by which data can be protected, ensuring confidentiality and preventing tampering, is by using a form of encryption. There are 2 ways of encrypting data, using symmetric and public-key encryption. Symmetric key encryption is where the key is the same for both encryption and decryption [38], a similar concept to using a key in a physical lock. Common examples of symmetric key encryption algorithms are AES, DES and SAFER [39,40]. Symmetric key encryption relies on key exchange, which can be problematic. A method for securely exchanging a symmetric key was proposed by Diffie and Hellman [41]. However, because this is a public-key exchange algorithm, an adversary might gain the key by listening in [42]. This may leave legitimate users open to an impersonation attack as the adversary would appear to be a legitimate user, because they would know the key. This fails the repudiation and spoofing constraints in STRIDE as a user will not be able to verify from whom the key is being sent, and if encrypted messages are from a legitimate user.

An alternative to symmetric key encryption is public-key encryption, such as RSA [43]. This form on encryption uses 2 keys, public and private. A message can be encrypted using one key, and decrypted by using the other. For example, Brian may choose to may his public-key, available to the public. If Lisa wants to send

Brian a message only he can see, she encrypts the message using his public-key and sends him the ciphertext. Once he has received this, Brian can use his private key to decrypt the message and read it. This method of encryption ensures that the data is private and hasn't been tampered with, but doesn't ensure the contents are valid from the user it may say its from. For this reason, this algorithm is vulnerable to similar impersonation attacks that symmetric keys are vulnerable to, even if it is a more feasible concept to practice in concept.

To overcome this, integrity of a message has to be ensured. This means there has to be some way to prove that a message was sent from a specific individual. For this, digital certification [41] can be used. Using RSA as an example, when Lisa wants to send Brian her message, she could include her digital certificate along with her message. That way, when Brian decrypts the message he can see that Lisa was the one that sent it. A form of digital certification can be used in a much more complicated way to better verify the integrity of a message, but this example illustrates the effectiveness of the general concept. This does not fully meet the spoofing or repudiation constraints of STRIDE, but a more complex data exchange protocol could use a combination of RSA and digital certification to ensure this, by encrypting the digital certificate with the private key belonging to Lisa.

Consideration should be given to the Denial of service aspect of STRIDE and how it may relate to this project and how a DDOS attack could be avoided [44]. A MANET could be vulnerable to a flooding attack [45], where a bad actor could send so many packets and so much data, that it may prevent a device from partaking in the network. One method to avoid this may be to implement filtering [46] similar to the way a WiFi router may filter out certain internet traffic using a firewall. To accomplish this in a decentralised model, assumptions would have to be made about the contents of packets which should be received, or how many packets are reasonable for a server to receive from a client before it ends its connection and starts to ignore subsequent connection requests from the client. On a resource-constrained device, this may take important computational resources away from doing the job it is meant to do. Further options include verifying devices which connect using a predefined handshake with a digital certificate. This would mean a device could tell which devices are going to be "good", and which devices are not.

2.4 Routing

The main approach of this project is to investigate MANETs, to enable the emergency services to obtain more information about the situation at hand. The variation in solutions that could be found is an important element to consider, as this could

be used after earthquakes, hurricanes or even man-made disasters, such as a during and after a terrorist incident. For this reason, several different approaches to packet routing should be investigated. For this, 2 different areas of packet routing will be examined: established and biologically inspired algorithms.

2.4.1 Established Algorithms

AODV

A popular routing algorithm used in MANETs is AODV [18, 47]. It uses a combination of dynamic source routing and distance vector routing to discover efficient paths for packets to travel from a source to a sink. The algorithm sends a series of Route Request (RREQ) packets to its neighbours, attempting to find a specific sink node. These packets will be passed on by nodes until it reaches the sink (or until it cannot travel any further). When the sink is reached, it will send back a Route Reply (RREP) packet to the source. This will take the exact path it took to reach the source, with each node making note of the next node after it in the path in a routing table. Within this routing table is a destination ID, next hop to make and timeout, as well as other information. This is used to enable the same route to be used in the future to ensure fast routing of packets. A disadvantage of this algorithm is that is floods the network with packets, which increases communication complexity and wastes computational resources.

TORA

In addition to AODV, TORA [48] is a different approach to routing in an Ad-Hoc network. It uses a DAG to determine which nodes to route packets through, onto the destination of the packet. A source node will send out QRY packets to neighbours, similar to RREQ packets in AODV. These are relayed until it reaches nodes which are adjacent to the destination node. These packets will broadcast a UPD packet, containing an id and distance (height) from the sink node. These are propagated back. Each node will maintain the height of the node and will broadcast an updated UPD packet reflecting its height. This propagates back to the source node. Once UPD packets have all travelled back to the source node, a DAG has been formed. The direction of a link is based on the height of the nodes. If a node **i** has a link to node **j** and $\mathbf{i}.height > \mathbf{j}.height$, the **j** is downstream of **i**. So, the link will be: $\mathbf{i} \rightarrow \mathbf{j}$. Using this method uses fewer control packets when compared to other methods as it only reacts when the topology changes. Additionally, partitions in the graph are easily detected. A disadvantage of TORA is that it relies on a stable and steady

network, much like AODV, to make it worthwhile to use. This makes it unsuitable for this project as the network topology of this project is unstable, with nodes coming in and out of the local network frequently.

Geographic Routing

In some situations, it may be important to consider routing based on physical location. For this, Geographic routing could be used [49]. The premise of this strategy is to pass packets to a neighbour which is physically closer to the target. This can be used for position-based routing or for Geocasting, where data is sent/received to/from a region of nodes. An example of when this is useful is monitoring an earthquake. Data may need to be received from a specific location containing sensors to give readings for that area. The basis for this type of routing is knowing the location of a node. For this, GPS (or equivalent technologies) or localization protocols could be used (using signal strength to estimate the distance to a transmitter). There are 2 halves to this form of routing: Greedy Forwarding (GF) and Greedy Perimeter Stateless Routing (GPSR). GF is simple, send a packet to the neighbour which is both feasible and closest to the sink, by knowing the location of itself, one-hop neighbours, and the sink. GPSR is similar but more robust. It is a method which makes the assumption that nodes are arranged in the form of a planar graph. Where GF would fail if there were obstacles, GPSR would turn to its failsafe. It will use GF to go as far as possible. When this fails, the righthand rule is used for the first edge. This is where the next edge traversed is the one sequentially counter-clockwise about the current node from the edge which has just been traversed.

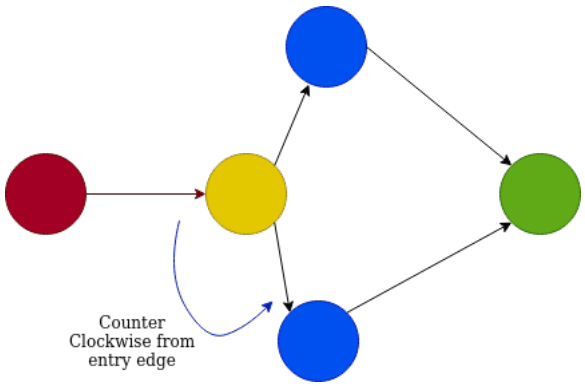


Figure 2.3: Example which displays the Right hand rule when choosing between 2 connected nodes, both equidistant from the goal

This concept is illustrated in Figure 2.3. From here, perimeter routing is used to

route on progressively closer faces of a planar graph. Each of these faces should cross an imaginary line from where perimeter routing starts, to the destination. Each time checking if there is a GF option to revert back to. This algorithm is very specific and could be useful in some emergency contexts, as they are very location sensitive. One use could be to route information about a specific area, to devices in a certain area about some impending escalation in danger.

Flooding

Flooding is a simple, but effective, means for routing. This is the method whereby a node will pass a message to all of its neighbours. They will then pass this message on to all of their neighbours and so on. This continues until a timeout is reached, or until the destination node is reached. It is widely used as a part of many other algorithms, such as AODV and TORA. It is an effective means of getting a message to a destination. A negative to flooding is that it puts a strain on a network, and can waste computational resources. But, due to its effective nature, there are many attempts to adapt the algorithm to be more feasible in the real world [50, 51].

Although it is a flawed approach, flooding can be an extremely potent tool for spreading information in a network where you want to avoid issues with routing, due to the nature of the data being sent. Aside from resources being wasted, it avoids a lot of heavy, and complicated issues which other routing approaches run into, such as issues with quality of service (QoS) where packets get lost and have to be re-sent. If an approach is taken to improve flooding, to take into account these issues, it can be an effective method of routing critical data. It can also be a way to avoid QoS issues other routing algorithms suffer from.

2.4.2 Biologically Inspired Algorithms

An alternative method for routing packets is to look to nature. Some of the methods that have been investigated have not just been honed by humans, but also by evolution, done to ensure a species survives and thrives in its environment. For this reason, they are an interesting area of reason to be considered.

Swarm-based Routing

There are a few iterations of swarm-based routing algorithms that have been invented. This is due to there being numerous swarm-based species which have been studied, such as ants and bees, as well as different interpretations of their behaviour. These are better than many other established routing algorithms [52], such as AODV.

They tend to find the best path through a network from a source to a destination. Algorithms such as Ant Colony Optimization (ACO) include some element of randomness so that it deals well with a changing topology. This is beneficial to MANETs, which may have nodes passing in and out of a network topology frequently, so it cannot rely on using the same path frequently. Using an algorithm such as ACO means those new, potentially better, paths can be discovered. Furthermore, it includes features, such as timeouts on established paths, to force this path exploration principle.

An alternative routing solution is based on using studying bees [53]. It contains a scouting stage where each node in the network is "scouted". Scout packets will flood the network through one-hop neighbours until they reach a sink node. A "back bee" packet is then sent towards the source. This process is done to evaluate the efficiency of routes between nodes. It shares several similarities with other algorithms, such as AODV. The difference is that multiple paths are forged between a source and a sink. Each node discovered is checked to calculate the route efficiency of using that node. These routes are reinforced by other packets travelling through the network. Additionally, a major difference is that intermediate nodes make no choices over routing, this is all done by the source node.

Social and Epidemiological Routing

Social and Epidemiological routing is based on how disease spreads and how humans interact [54]. In many ways, it is very similar to flooding. It takes flooding and considers the likelihood that a node will transmit the packet it wants to send, using probability. It uses a set of social characteristics to determine if the behaviour of the node is "good" and will cooperate, or whether it is "bad" and will only care about its resources. Studies have shown how powerful these routing methods can be [55, 56]. They can reduce the number of packets that are sent and will increase the probability that packets are going to be delivered. Using a social-based routing protocol can be shown to reduce the number of packets sent, when compared with Epidemic routing [56], and can increase the likelihood that a packet will be successfully delivered. This could increase the performance of the network.

Epidemic routing often takes place in an Ad-Hoc scenario, much like social-based routing. Similar to flooding, it attempts to re-create the spread of disease. A device holding a message is said to be a *carrier* which will *infect* other devices. Low latency for message delivery and high percentage of messages delivered are key aims for the algorithm [57,58]. A carrier will come into contact with other nodes that it has never met before. Unlike in social routing, this presents a multitude of different issues, as

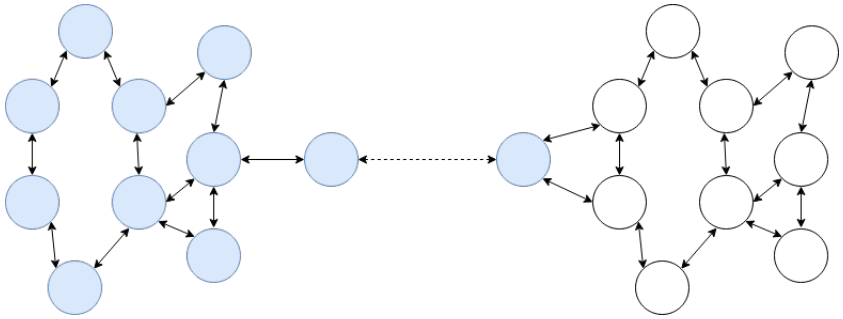


Figure 2.4: Epidemic routing example where *carrier* moves between 2 groups of devices

no choices over its intentions can be made. If the network is partitioned, a *carrier* may move between different portions of the network, illustrated in Figure 2.4. This will help spread messages across a network, regardless of if it is connected or not. This is incredibly useful in the context of an emergency, where the network may be extremely fragmented, with nodes moving around. As with flooding, the "infection" process could yield similar performance issues with messages being passed around disconnected portions of the network without any chance of reaching a destination node.

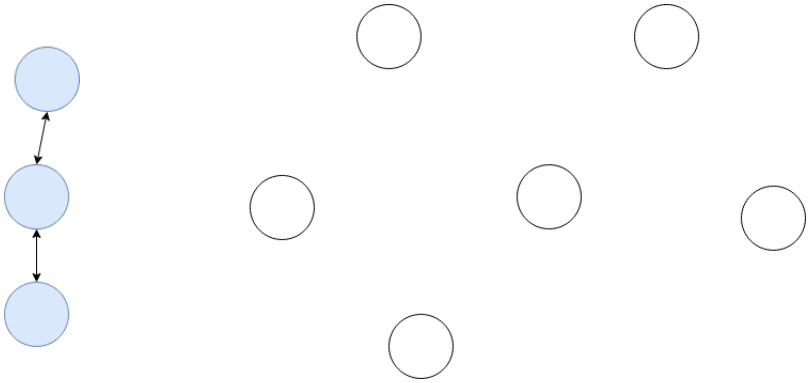


Figure 2.5: Social Distancing example where distance between nodes stops *carriers* infecting other nodes

An unavoidable constraint on the project is in the physical space, where nodes may stay away from each other. This could create an extremely fragmented network where messages cannot be passed between areas of the network that aren't connected. It may lead to a large decrease in the delivery rate of messages, or at least increase the

latency significantly. In a medical context, this is referred to as *Social Distancing*. It is a method to combat wide-scale disease spread in a population [59–61]. By doing this, [61] states that it can reduce infectious contact by 38%. Other studies [59] state that it can reduce infections by more than 90%. Performance issues could be significant. For two or more nodes to connect and exchange information, they have to be nearby and within range of each other. This is the same with the spread of disease, people have to be close together to allow the disease vector to travel between them to infect them. By implementing *Social Distancing*, this example in which two or more nodes are in range becomes less likely, as they are taking measures to be a greater distance apart. If this happens less, communication events are less likely to happen, so a *carrier* will be able to infect fewer devices, thus reducing the spread in the network. This concept is illustrated in Figure 2.5, where nodes staying out of range of each other prevents connections with *carrier* nodes, so will not become infected.

2.5 Reflection

Research in this project has been undertaken extensively and has covered various and highly important areas of the project. The different elements which make up the project have been identified and researched, as have their issues. An example of where this is done is the investigation into the effect of social distancing on Epidemic routing.

The project has identified several different technologies which could be used in similar projects. As time goes on, the use case for some of these technologies, and newer ones, maybe far more significant than they are today. An example of this is the growing uptake in ZigBee, a communication medium that would have been ideal for this project had its consumer adoption not been confined purely to smart home devices.

In future work, more research should be done to investigate the new approach to this problem area, as well as improving the different areas of the project with newer, cutting edge approaches that may be discovered.

Chapter 3

Ethical, Social, Legal and Professional Issues

This chapter will go over the different issues which could arise in the project. It will investigate if the project can cause issues for its users, as well as issues which could affect the project in the long run, such as issues over the legality of how it could work in the problem area.

3.1 Ethical Issues

Having a project with several different objectives can cause ethical issues to arise. These may have negative effects which are not present immediately. If not properly investigated, these issues can have an impact on a project or company. Projects must consider the downsides to their actions and do their due diligence on the effects they might have. Even if an action will have a large, positive increase in one area, it may have severe consequences in another.

This is the reason for doing work on making sure that any ethical issues which occur during the project are dealt with properly. One such issue that appeared during the project was in the area of data privacy. Through investigation, encryption was used to hide sensitive data in messages communicated between two devices. This helps to protect users from adversaries and prevents their data being used for harm.

3.2 Social Issues

Social issues are those which affect the lives of people using the product or service. In this case, it would affect survivors sharing their information. There are not any obvious social issues for which this project should act on combating. The service will work the same way for all users, regardless of any distinguishing factors, such as race, gender or age as transmission of data is automatic and doesn't discriminate between users. Underlying structural issues remain, which could be investigated further in the future, such as the effectiveness in many countries where users may not have the required hardware to run this system even if they need it. Future work should be done to ensure they are not forgotten.

3.3 Legal Issues

The heart of this project is the transmission of sensitive data between devices. This could bring up a variety of legal issues if handled incorrectly. Data could be stolen if communications are to be intercepted by a 3rd party. This information could then be used in compromising ways. To combat this, encryption is used to protect the contents of messages between devices, so that only they can see the data contained within the messages.

A further issue, addressed earlier, is the proper storage of data on a device. Steps should be taken, if implemented on a commercial, large scale product, to protect this data from the user so that the only users who can access this information are the emergency services. As this relates to the handling of data, any commercial implementation of the project should be wary of *GDPR* [62], as well as other data protection regulations in other countries. *GDPR* has been written to apply to all data collection, not just commercially by firms such as Google and Facebook. A project such as this must ensure packet data (data in motion) and data stored on devices (data at rest) is used with these regulations in mind.

3.4 Professional Issues

This project has adhered to the BCS Code of Conduct [63]. This project has strived to produce a research project which provides a new approach to the problem area which it operates in, increasing the need to ensure the conduct of the project is in line with rules. Moreover, the project paid close attention to the University of Warwick *Research Code of Practice* [64].

Chapter 4

Project Requirements

Defined in the *Progress Report*, the requirements are what has guided and shaped the project. Those requirements are detailed below, as well as supplemental requirements which became obvious as the project went along. The requirements followed the MoSCoW method for prioritizing requirements [65].

4.1 Functional

FR1 - **Must** use a widely used MAC layer protocol such as Bluetooth or WiFi.

FR2 - **Must** use data privacy techniques, such as encryption, to prevent data falling into the wrong hands.

FR3 - **Could** monitor battery life and make choices based on its predicted life expectancy.

FR4 - **Could** use a system of authentication to verify nodes, such as using a trust-based system or digital certificates.

FR5 - **Must** use the concept of sources and sinks, where sinks collect data and don't transmit information packets, and sources send and receive information from other sources.

FR6 - **Must** have a way to identify devices within the network. This ID should be used within packets sent by each device about themselves.

FR7 - **Could** use a location discovery service such as GPS to get location data.

FR8 - **Must** create solution to fragment messages to be sent over BLE due to the message size limit (20 Bytes).

4.2 Non-Functional

NFR1 - **Must** be fully documented and maintainable. This means that the project is easily extensible and can easily be implemented on other platforms.

NFR2 - **Must** be easy for a user to connect to and use. This is vital for the project as time is an important factor in an emergency situation and so the less time a user has to worry about how it works, the quicker they can use it to get help.

NFR3 - **Must** be created so it can be applied to a large population of devices. This project works optimally if there are a large number of devices to connect so design choices should consider the need for this project to work at a large scale.

NFR4 - **Must** be able to be used by different types of devices such as Phones, Tablets and PCs. Much like **NFR3**, this project should be designed so it can be easily transported onto other devices so that they can also participate.

NFR5 - **Must** use secure communications so bad actors cannot steal and weaponise the information.

NFR6 - **Should** require as little user interaction as possible to increase efficiency.

4.3 Constraints

The project was constrained by the scale at which it could be tested at. 3 Raspberry Pis were obtained for the project. This allowed testing to be done on a variety of scenarios, but prevented large scale testing. For this, a large number of physical devices would have had to have been obtained. This is unreasonable to expect a project of this size to have those resources, and so the project could only be considered at scale theoretically. However, a version of this could have been simulated, but it

was chosen not to due to the time constraints of the project.

An additional constraint of the project is that most major platforms require different implementations. The project used Raspberry Pis of different variations, using different CPUs and amounts of memory, but they all used the same OS and shared several properties. The ideal testing solution would be to have an implementation on each major mobile platform [66]. This would mean that testing could be performed rigorously across a truly heterogeneous network of devices. This would be more similar to what the network would look like in the real world, but developing extra solutions in for each platform is out of the scope of the project and is considered a future enhancement.

Chapter 5

Design

In this chapter, the paper presents the system model for this project. It will then moves on to discuss the design and the choices that were made, as well as challenges with the design that were encountered throughout the project. It then moves onto discussing the design of more complex behaviours, what challenges they presented and the designs that were produced to overcome these.

5.1 System Model

This section will present a model of the system which was used in this project, shown in Figure 5.1. After doing research, described in Chapter 2, a number of main elements were identified, such as:

- **Devices**: Elements of this project which hold all the hardware and software components which enable it. Contained within a device, is a method to generate data about a device, store data about other devices and execute code to form connections to transfer data.
- **Clients**: Component of a device. This will be an element of a device which will connect to a server program in order to facilitate data transmission between devices. This is the aspect which will drive behaviour and decisions in most devices. A Client device will seek out connections with servers to transfer their information.
- **Servers**: Component of a device. The bridge between different client devices, enabling them to gain information about other devices in the network. Will not attempt to connect to other devices, but will inform clients that they are

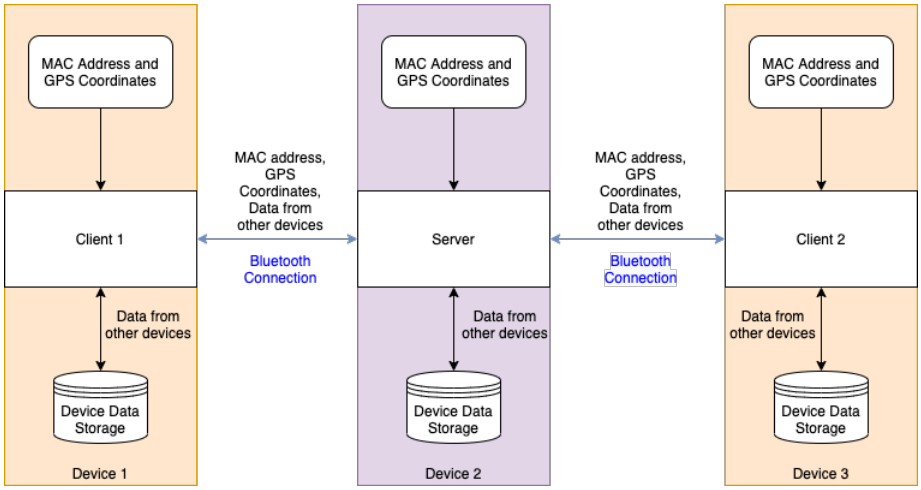


Figure 5.1: System model with inputs and outputs for each node

open to forming a connection, allowing a client to request a connection. This can be done with up to 8 client devices at once.

- **Transmitted Data**: This is the output from one device and an input into another. This transfer takes place over a Bluetooth connection, with two connected devices. Information about devices, and those they have previously encountered, is exchanged.

- **Persistent Data Storage**: A method to store data about other devices. This is an input and output for clients and servers. They will store information about devices that they have gathered, and will also retrieve this information to pass on to other devices.

- **Data Generation**: An input into a client or server. This is information about a device which will be sent to other devices. This information is passed onto from the receiving device to other devices.

This system can also be encapsulated in a formal manner. At time t, the set of client devices can be represented as C_t and the set of server devices can be represented as S_t, which are related as follows:

Definition 5.1.1 ***Emergency Disaster Network***
The network can be represented as an undirected graph $G_t = (N_t, E_t)$ *where,* $\forall t$, $N_t = C_t \cup S_t$ *and* $E_t = \{(c, s) | c \in C_t, s \in S_t\}$ *such that* $\forall s \in S_t$, $|\{(c, s) \in E_t | \forall c \in$

$C_t\}| <= 8$ *and* $\forall c \in C_t, 0 <= |\{(c,s) \in E_t | \forall s \in S_t\}| <= 1$. *This will form a forest where each tree represents a group of clients connected to a single server.*

This uses an element of time because the set of clients and servers could change over time. This needs to be reflected in the system model as it is an important factor in decision making around the design of the project and allows the paper to fully encapsulate the scope of the problem.

5.2 Initial Design

This section details the initial design work done for the project. It will discuss the early assumptions and choices made in the project, as well as changes to those assumptions and what that meant for the project. It will also detail the threats to the project, how those can be thought of formally and how these can be overcome.

5.2.1 Preliminary Thoughts and Assumptions

When design work started, it was done to create a system which was independent of the communication medium. This reason for this type of system was to make it applicable to as many different communication mediums as possible. The choice was made to focus on one medium so that design and implementation could work better together. In the future, the approaches and design work in this project could be applied to other mediums easily. For example, ZigBee is capable of having a star topology, such as those seen in Bluetooth, so the same design ideas could be used for both. Due to this, Bluetooth was chosen to be the medium for which design work would be done. The reason for this being the availability of BLE devices in the general population. This made it an obvious candidate for this project as it would enable widespread adoption, something this project requires to be effective.

5.2.2 Challenges of the project

To accomplish this a different approach had to be taken. This meant adhering the to the client-server model which Bluetooth operates over. For this, it meant having to design two separate halves to the system, one for clients and one for servers. An overall program flow, for both clients and servers, is illustrated in Figure 5.2. Changing to this new model meant that the interaction hierarchy had to also be altered. Initially, the system design was that every device could connect and pass information. Each device was thought to be homogeneous. However, due to the

client-server model, it meant that connection requests could only be made by the client, as does all other communication. A client will communicate with a server, telling it the information it wants to read or be notified of changes to, as well as writing information to the server. This specific set of interactions meant the method of communication had to be set within those bounds.

By using the client-server model, the distributed nature of the project had to be maintained. This problem area does not allow for many forms of established networks. Allowing devices to move about between servers is crucial to the success of the system. This behaviour is captured in these definitions:

Definition 5.2.1 ***Mobility of Devices 1***
At time t, $c \in C_t$ *and* $s_1, s_2 \in S_t$ *where* $|\{(c, s) \in E_t | \forall s \in S_t\}| = 0$
By time $t + i$, $i > 0$, *c could move in such a way that, at time* $t + i$, $(c, s_1) \in E_{t+i}$ *where* $c \in C_{t+i}$ *and* $s_1 \in S_{t+i}$
By time $t + k$, $k > i$, *c could move in such a way that, at time* $t + k$, $(c, s_1) \notin E_{t+k}$ *and* $(c, s_2) \in E_{t+k}$ *where* $c \in C_{t+k}$ *and* $s_1, s_2 \in S_{t+k}$
By time $t + n$, $n > k$, *c could move in such a way that, at time* $t + n$, $(c, s_2) \notin E_{t+n}$ *where* $c \in C_{t+n}$, $s_2 \in S_{t+n}$ *and* $|\{(c, s) \in E_{t+n} | \forall s \in S_{t+n}\}| = 0$

Definition 5.2.2 ***Mobility of Devices 2***
At time t, $(c, s) \in E_t$, $\exists c \in C_t$, $\exists s \in S_t$.
At some time $t + j$, $c \notin C_{t+j}$ *or* $s \notin S_{t+j}$, *therefore* $(c, s) \notin E_{t+j}$.

If we take these properties (Definition 5.2.1 and 5.2.2), and define the mobility model of each node as a random mobility model [67], we can define how nodes interact with each other. It is important to maintain this to exact maximum utility from the system in the problem area. It defines that a client node may move in such a way that it can move between disconnected trees in the network at different times, but also states that any node may drop out of the network at any point, defined in Definition 5.2.2. This property allows a device to visit, and connect to, multiple servers over multiple periods, but also captures the danger that a node may leave the network. This allows the system to work in a distributed manner.

A significant design challenge was how to deal with Bluetooth message sizes. These are limited to a maximum of 20 bytes per message. This forced the design to incorporate consideration for either splitting up messages to send, or piecing them together when received. This was a challenge to design for as considerations, such as the ordering of messages, had to be accounted for when designing the message format. Although this problem did not fundamentally alter the structure of the whole system, it was still a significant technical challenge for the project to overcome.

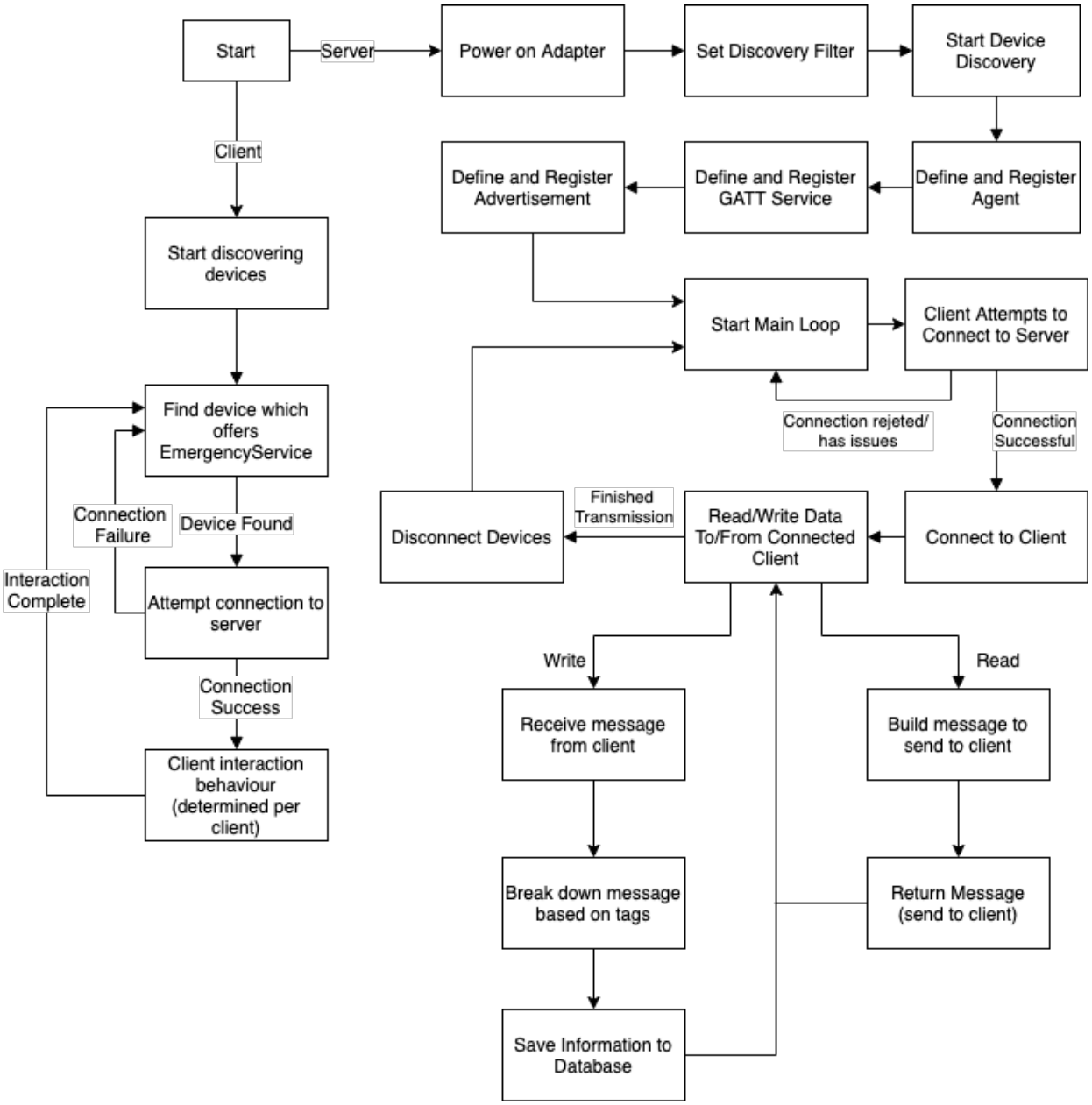


Figure 5.2: Program flow for all types of node

Lastly, a major challenge of the project was to encapsulate the different types of client devices that were needed. The goals of this were to develop 3 different behaviours for clients: normal, secure and emergency. The details of which are illustrated within Figure 5.3, Figure 5.5, and Figure 5.4. The *normal* behaviour is the base actions that most devices should take. This was the base requirement for the project. In this behaviour, a client should connect to a server and exchange messages, then disconnect and move on. *Emergency* and *secure* behaviours were extension behaviours which aimed to encapsulate additional complex behaviours for the client. These took the *normal* behaviour and tailored the communications and actions to be for more specific use cases, such as secure communications and communications with a device belonging to the emergency services.

5.3 Design Detail

In this section, this report will describe the design of the system from the perspective of the client and server, the two most important parts of the system. This design will detail how these two parts interact with each other, the medium of communication and what should be done with those communications as a consequence. It also looks at the threats to the project, both formally and informally. This section should help the reader to understand the choices made in this project, why they were made, and how the system works as a whole together.

5.3.1 Client

The client has three main elements to it: device discovery, connecting to servers and client behaviours. The first of which is its initial setup. As can be seen in Figure 5.2, it will start to discover devices within its range. This is crucial, and will repeat itself several times throughout the operation of a client. This is key as the main role of a client device is to locate and connect to server devices. Once it finds a server it will attempt to connect to it.

After the client has finished searching for servers to connect to, it will find all of the devices within its range. For each of these devices, it will attempt to connect to them. The client will have the advertising information from each server, which they will make available so clients can connect to the right servers. A client will inspect the list of services offered, which is part of the advertising information from the server, and will start a connection with the server if it offers the right type of service. This service is the one which this project has designed. After this, the client will either form a connection with a server or not. If it does not, it will either go

back to looking through the list of devices in range, or will start scanning for devices in range again. If a connection is formed, the client will start to interact with the server.

Once a connection is formed, the client may start interacting with the server. This means it may issue read and write commands, as well as setting up a notify service. This project only utilised the reading and writing elements of server functionality. This allowed the client to read information from the server and write information to it. This protocols used for this communication were defined in the *client behaviours* on the client-side. These defined in what order reads and writes occur, as well as what information is to be written and what is expected to be read. It also defines when the interaction is complete.

Once the client has finished communicating with the server, it will terminate its connection with the server. This allows the client to find other servers to communicate with, and a new client to communicate with the server. The client will either revert to discovering devices in range, or will attempt to communicate with servers that it discovered previously, but hasn't attempted to communicate with yet.

5.3.2 Client Behaviours

Normal

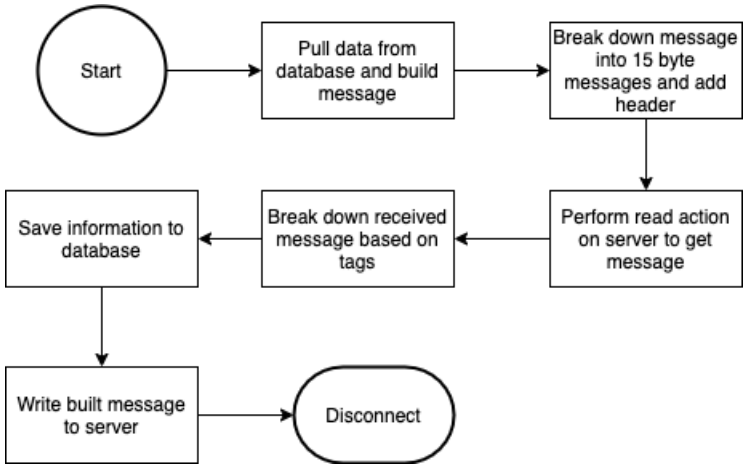


Figure 5.3: Normal client behaviours when interacting with a connected server

This is the most basic of the client behaviours, illustrated in Figure 5.3. The other client behaviours both build off the functionality and protocol defined here. The

client will start by getting information from its data storage, such as a Database. This will retrieve information about devices that has been collected previously. Using this, and other inputs such as information about the device, a message will be built by the client to be written to the server. The message is then broken down into 15 byte sections and a 4 byte header is added to each one, which includes a sequence number and a separator character. Once this is done, the client will read from the server (collecting each message fragment to form the complete message), break down the message it receives, and store the information collected in its data storage. The message to write has to be built before the read action as to avoid writing the same information back to the server. The last step for the client is to write its message to the server. For this, it will write each of its message fragments using the servers write action. Once this is complete, the client will disconnect from the server.

Emergency

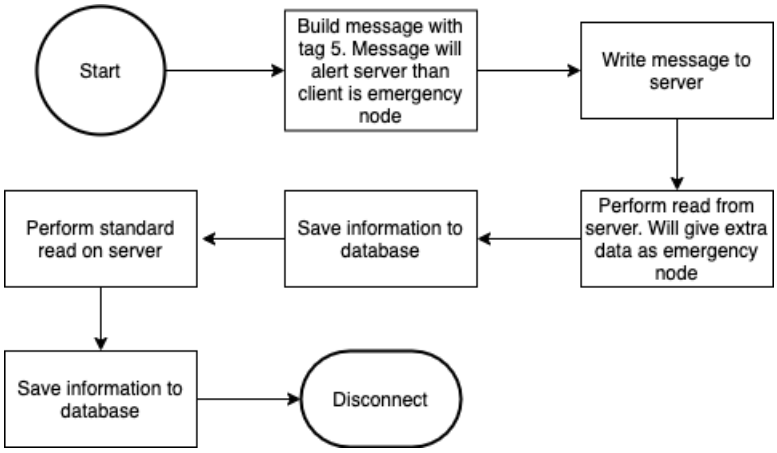


Figure 5.4: Emergency client behaviours when interacting with a connected server

The thought behind the emergency behaviour was that the emergency services will not need to send much data to a normal device. An emergency services device does not need to send devices its location, their only function is to collect information which can then be used to help people. This is why the interactions in this behaviour protocol are generally one-sided and consists mainly of the emergency client requesting information.

The emergency client action is more simple than the normal client behaviour. This behaviour is shown in Figure 5.4. The client will write a message which contains tag **5** the server. This will alert the server that an emergency device is communicating

with it and it should prepare to send special messages. The client will then perform a read action and will receive a message from the server which will contain the MAC address, last seen location, and time stamp of a defined number of devices. The amount of data about devices in a message is specific to the implementation and depends on the data rate of the communication. The client will then perform a normal read on the server to get other information before ending the connection.

This extra read message is important because it gets different information from the server. It will get information about the server, as well as various independent devices and their locations and timestamps about those locations. This helps to transfer information about a wider variety of devices, useful to the emergency services, as well as when that information was recorded. This additionally helps as it establishes if the information may be stale.

Secure

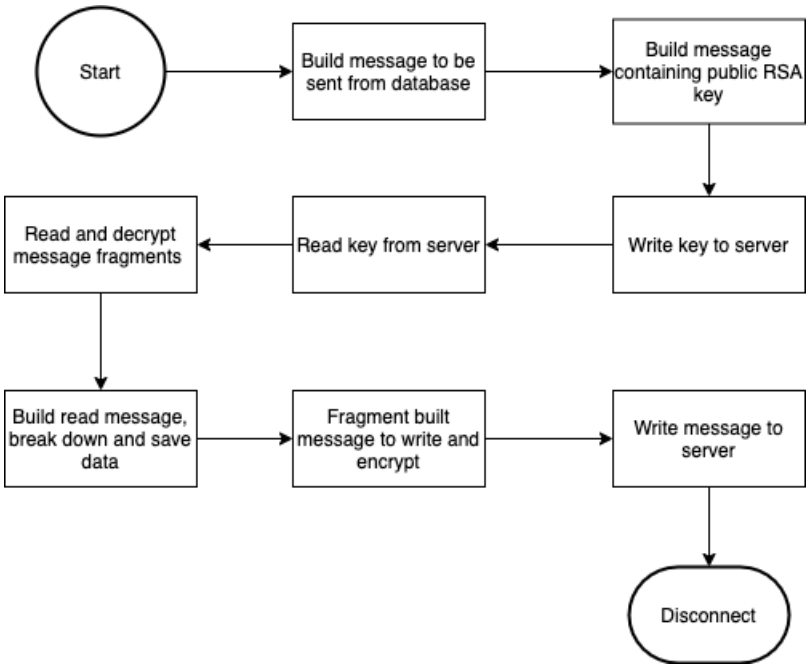


Figure 5.5: Secure client behaviours when interacting with a connected server

Establishing how to transfer information securely is very important. In the problem area this project investigates, it is a very likely possibility that there are adversaries which will want to capitalise on people in vulnerable situations. For this reason,

there must be a method whereby devices can communicate securely. By using the behaviour defined in Figure 5.5, a client can have an encrypted conversation with a server. It will communicate the same information as in the normal client behaviour, but the information will be encrypted.

Once a connection with a server is established, the client will pull information to send to the server, as well as information about the device, and will build a message to send. This message is then put aside, and a message containing the devices RSA public-key is created. This message will be very large so is fragmented the same way other messages are. This key is then written by the client to the server. The next action the client takes is to read the server's public RSA key. It will read each fragment and put it together. The client will then perform a read from the server, as it would do in the normal client behaviour. This message will be put back together. The message will have been encrypted with the devices public key, so the client will use the devices private key to decrypt it. Once it is decrypted, the information is stored in data storage. The message which was created by the client is then taken and encrypted using the servers public-key. It is then fragmented and written to the server. Once this is complete, the client will disconnect from the server.

This protocol for secure communication works well. If we evaluate it using the previously discussed STRIDE model, the important elements are Tampering, Repudiation and Information disclosure. This method helps to prevent tampering: if an adversary does not have the private key of one of the participants, they cannot decrypt and alter messages. It also fulfils Information disclosure as this method ensures the confidentiality of data. The downside to this protocol is that it fails the Repudiation constraint. A method to fulfil this in the future would be to use a digital signature in some way to prove the identities of both users. This would enable mutual authentication.

5.3.3 Server

The server is the most complicated part of the system, it has to concurrently perform many tasks. There are numerous parts and a server has to implement several different sets of functionality at once. The behaviour of the server is described in Figure 5.2.

The first job of the server is to turn on its adapter. This allows it to implement Bluetooth functionality. After this, it turns on device discovery. This is not necessary for most device functionality, but is helpful for a user as it alerts them to connection successes. The server then has to define and register the an agent. An agent is used to define connection criteria in Bluetooth. It helps to implement several security features that are used by Bluetooth, such as entering a specific code to connect to

some devices. This project prefers to implement its security at a later part of the system. In this project, it lets any device connect.

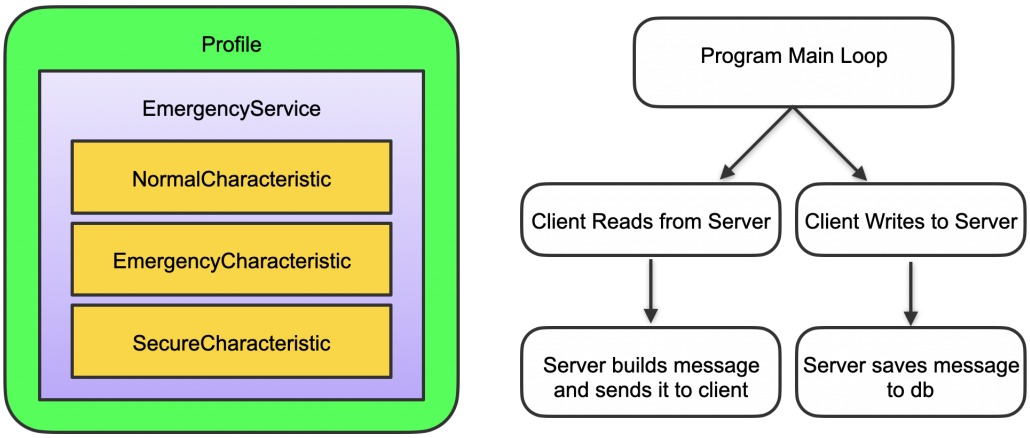


Figure 5.6: GATT profile which defines server functionality

The next job for the server is to define and register the GATT service. This is the part of the server within which read and write actions will be defined. The GATT service for this project is shown in Figure 5.6. This has an overall profile which contains only 1 service, EmergencyService. This service has 3 characteristics to it. Each of these characteristics are a direct link to the equivalent client functionality. Each characteristic implements the read and write functions that are needed for each different type of characteristic. Designing the GATT server this way allows clients to connect the desired GATT service characteristic. Originally, the design only included 1 characteristic to communicate with. This created quite a confusing design and added to the communication complexity. Abstracting the behaviour to separate characteristics is more similar to OOP design principles and makes the design easier to understand and more extensible.

Once the GATT server is registered, the server will register its advertisement. This will cause the device to broadcast information about what services the device offers so that clients may connect to the server offering complementary functionality. Using this reduces the number of connections that a client would have to make and is also used to alert devices that a server is available to connect to. This works similarly to beacons in WiFi [30].

This point signals the start of functionality in the server. It will move into its main functionality loop, where a construct will loop indefinitely, with signals attached to it which listen for changes to the device. It will also keep the device alive and allow the GATT server to stay alive.

If a client attempts to connect to a server and does not, the server will continue to operate and wait for a successful connection. If the connection request is successful, a connection will be formed. From this, the connected client and read/write data from/to server. This leads to 2 strands of behaviour.

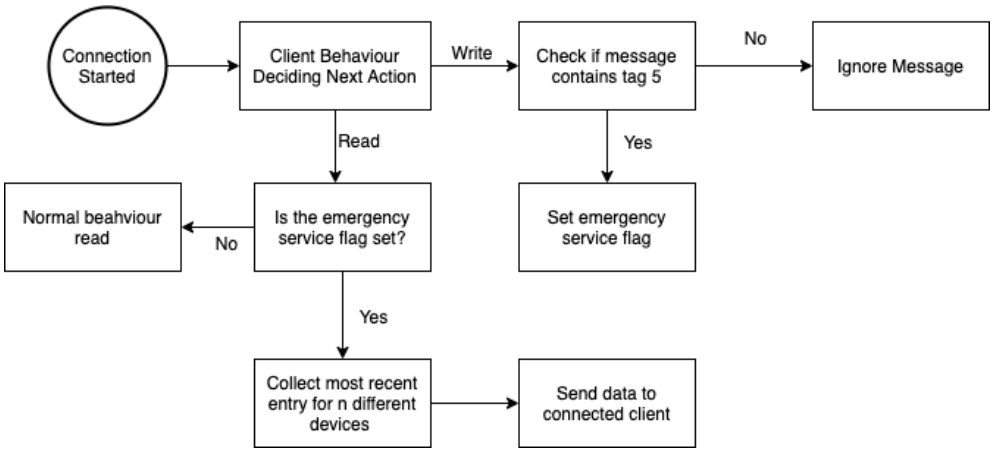


Figure 5.7: Behaviour of emergency characteristic

If a client writes to the server, it will write individual fragments of the message to the server. This server will build this over multiple write operations. Once it has received a complete message, it will break down the message. This is done based on the different tags in the message which equate to different types of information in the message, such as MAC address or device location. This information is then sent to the data storage for the device.

When the client wants to read information from the server from its NormalCharacteristic, it will call the read function on characteristic. The server will build a message and will send fragments to the client. The client will keep calling the read function until it has received a full message. Each time, the server will get a message fragment and send it to the client the full message has been sent.

The design of the other characteristics is more complicated. The design of the EmergencyCharacteristic can be seen in Figure 5.7. This example shows the different paths that could be taken based on the action of the client. If the client writes to the characteristic, it will check if it has tag 5 in the message. If it does, set a flag for emergency service devices. This will mean, when the client next reads from that characteristic, it will be sent back a special message for emergency services devices. This message will contain more information than usual to help the emergency services better. If no flag is set when a read is called, normal read behaviour will occur.

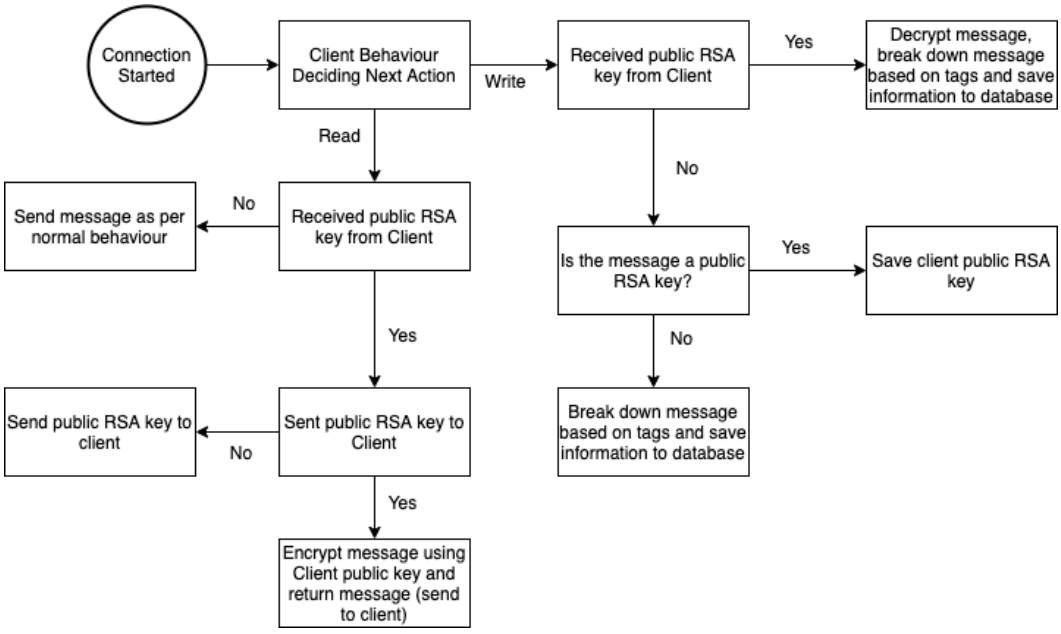


Figure 5.8: Behaviour of secure characteristic

The SecureCharacteristic behaviour is shown in Figure 5.8. In this, several choices are based on whether the server has sent/received a public RSA key. If the client writes information to the server, it must investigate if the message contains an RSA key, if it does, it should be saved for future writes and a flag should be set to show that it has been received for that client. If an RSA key has already been sent, the message which has been sent should be decrypted, the broken down by tag and have the contents saved in data storage. If the client attempts to read from the server, the client needs to check if it has received the clients public RSA key. If it has, it should check if it has sent one. If it has not received it, it should proceed like NormalCharacteristic does with a read. If the server has received the clients public RSA key, but has not sent its own, this should be done. Otherwise, a message should be constructed with data from data storage and encrypted, before being sent to the client.

5.3.4 Message Format

The format of the messages in the system is based on the FIX messaging protocol [68], and this version is called EFIX. This works by using a system of tag-value pairs. Each tag has individual meaning and specification for the associated value. In the example

1=(57.0461, -10.9456)|2=AB:CD:EF:GH:IJ:KL|

Figure 5.9: Example EFIX message

provided in Figure 5.9. This example contains 2 tags. The first tag is for a location. The tag (a number) is set equal to the value of the pairing. Between tags there is a separator. The FIX specification allows the use of many different types of separator, but the — character was chosen. The reason for this being other separators were used for other jobs, such as between a sequence number and the contents of a message fragment. The other tag-value pair indicates that the value is a MAC address. As can be seen from the example, this is true. This example message could be broken down successfully based on its tags and the information used. This messaging standard was utilised for a few reasons:

- **Security**: Devices know what information they are looking for so do not pay attention to rogue tags which have not been agreed initially.
- **Simplicity**: It uses a system of tags which each have meaning. This is a simple process to follow and means it is easy to break down messages into specific information categories to access data quickly.
- **Extensible**: To add a new category of data, new tag has to be agreed. This can be done as many times as necessary.

These points illustrate the utility derived from using this standard. It enhances the value of the project and provides a simple, but powerful communication protocol over which this project runs. This enables effective communication among devices, as well as enhancing the security of the data. These are aims stated at the start of the project which this medium helps to achieve.

5.3.5 Routing

As discussed previously in Chapter 2, there are a number of different routing algorithms which could be used by this system. There are some, more established routing algorithms for Ad-Hoc networks, such as **AODV** and **TORA**. These are widely used for a number of WSN applications. Neither of these algorithms quite fit this problem area as they rely on a number of homogeneous nodes in a network where the topology does not change very frequently or drastically. This makes them inappropriate for this system. An interesting approach to this project would be to use **Flooding** to

spread packets across the network. The idea of using a store-and-forward algorithm is appealing in this project due to the impermanence of the network. For this reason, a routing method similar to Flooding, **Epidemic routing**, is most appropriate for the problem area.

Epidemic routing will allow the system to use the random mobility model and maintain Definitions 5.2.1 and 5.2.2 to route packets efficiently across the network. The basis of this type of routing is that devices attempt to spread their *disease*. Each time they connect to another device, they try and infect them with their *disease*(information about the node), as well as other diseases of which they are *carriers* of. The aim of each node, therefore, is for their *disease* to infect an emergency client.

Another important aspect of this method of routing is that it anticipates devices moving around physically, it is the method by which disconnected trees become infected with different diseases. This is how information spreads, by devices moving, physically, into the range of other devices and connecting, before exchanging information. This concept is illustrated in Figure 2.4. This type of routing has several benefits, such as being highly flexible and delay-tolerant.

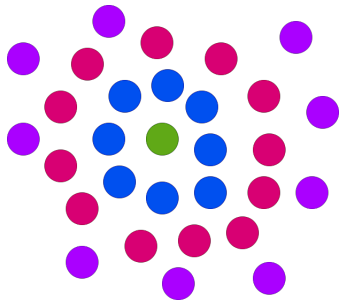


Figure 5.10: Illustration of timeout in Epidemic routing

Chapter 2 discussed optimisations to the Epidemic routing algorithm. In this system, one optimisation that is implemented is a timeout. This is done implicitly by having an upper bound, T, on device information exchanged between two devices. The value of T is a detail for implementation. Each device will exchange the T newest diseases it has contracted, plus its own disease it is trying to spread. The effect of this timeout is shown in Figure 5.10. In this figure, the green dot is the node for which a given disease originates from, call this *patient zero*. From *patient zero*, it will infect a new wave of devices which will become carriers, shown in blue, and so on. The timeout of the message at this stage will be T, as this disease will be the latest which these devices have been infected by. As carriers pass this disease on, the timeout will decrease as it will no longer be the newest disease that a device has been

infected with. This decrease in timeout can be described as: $timeout = T - (\text{number of hops from } patient\ zero) - 1$. In the example shown in Figure 5.10, if $T = 2$, the timeout will expire at the purple nodes. This will be the 3^{rd} hop and it will infect 2 waves of carriers before reaching its timeout. Using this mechanism will prevent a message from going round the network indefinitely, wasting computational resources.

Proof 5.3.1 ***Social Distancing***
If we take Definitions 5.1.1, 5.2.1 and 5.2.2, we want to show it is feasible to have a fully disconnected graph. Suppose we start with a graph, defined in Definition 5.1.1, at time t where $\forall c \in C_t, |\{(c, s) \in E_t | \forall s \in S_t\}| = 0$, $c_1 \in C_t$ *and* $s_1 \in S_t$.
If we move to time $t + 1$, *there is* $c_1 \in C_{t+1}$, $s_1 \in S_{t+1}$ *where* $(c_1, s_1) \in E_t$. *By the mobility model in Definitions 5.2.1 and 5.2.2,* c_1 *and* s_1 *can move in such a way that* $(c_1, s_1) \notin E_{t+1}$ *where* $\forall s \in S_{t+1}, (c_1, s) \notin E_{t+1}$.
We can reach some time $t + k$, $k > 1$, *where* $\forall c \in C_{t+k}$ *and* $\forall s \in S_{t+k}$, $(c, s) \notin E_{t+k}$ *therefore* $|E_{t+k}| = 0$. *This means there are no clients and servers which are connected because they moved in such a way that it is not possible for there to be an edge between any client and server.*

An issue with Epidemic routing is the effect of social distancing. This was also discussed in Chapter 2. This is the process by which people will stay away from each other during the outbreak of an infectious disease. As Epidemic routing models such an outbreak, the practice of social distancing also applies. An example of where this could occur in the problem space is in a low-density population area. There may be active clients and servers, but they may not be positioned in such a way that they can form connections, or that there are so few connections that the desired effects of the routing do not happen. It stifles communication and information does not spread throughout the network. The effect of this can be viewed formally in Proof 5.3.1. This illustrates the effect of Definitions 5.2.1 and 5.2.2 in such a way that, if nodes are partitioned in a certain way there could be no edges (which represent connections between clients and servers) at some point in time. This underlines the effect of social distancing on the chosen type of routing; There can be a point whereby no clients and servers can be connected which means it is impossible for information to flow in the network. It is a weakness of the algorithm which cannot be solved. This algorithm is very suited to high-density urban environments and performs optimally in this situation. Formally, this situation will be akin to the graph at time t in Proof 5.3.1. In low-density urban areas, it will still perform to its specification, except for the fact there will be a greater degree of latency between the discovery of a survivor and the information being collected by the emergency services, as well as a greater risk of social distancing occurring.

5.4 Reflection

Overall the design of the system has been thorough. Important elements can be described theoretically and have been formalised for clarity. The system model covers the system from an abstract viewpoint and takes a top-down approach. This meant that further elements of the project could be investigated from both a practical and theoretical approach when needed.

The detailed design section goes through the most important parts of the project and describes, with the aid diagrams, how each part works. This helps to convey the important concepts. On top of this, it looks at the weaknesses of the design. This is important to look at so they can be investigated in more detail in the future.

Future work for this project could focus on making the secure communications protocols more secure. As mentioned before, they could be amended to meet the STRIDE requirement of Repudiation so that both participants in the information exchange can be sure of the authenticity of the other.

Chapter 6

Implementation

During the implementation phase of the project, previous research and design was utilised to create a proof of concept of all the work previously done for this project. To enable this to happen, various factors had to be considered, such as:

- Network Protocol
- Platform
- Programming Language

Together, all of these factors combined to create a proof of concept of the system, described in Chapter 5. This system can be used as a working reference design for future works in this problem area. The project aimed to investigate a system which could disseminate information in an emergency scenario using Ad-Hoc networks, and this proves that it can be accomplished.

6.1 Choices

6.1.1 Network Protocol

This was a major choice that impacted both the implementation and design of the project. Essentially, this was a choice between Bluetooth Low Energy, WiFi Direct and ZigBee. These were the three different network protocols which could realistically be considered as they are the main, established standards for Ad-Hoc networking. Several factors were considered. A major aspect which counted against ZigBee is the lack of major mobile phone support. None of the top 3 best selling phones of

2019 [69] support it, but they all support BLE and WiFi. This meant that ZigBee was an infeasible option at this moment in time.

Deciding between WiFi Direct and BLE was harder, both have their merits. On the one hand, WiFi has a larger range and higher throughput than BLE. On the other, BLE consumes less power and is available on a wider variety of devices, such as low powered sensors and tracking devices. This means the system can be implemented and run on a larger variety of devices. For this reason, BLE was chosen over WiFi.

6.1.2 Platform

For this project, it was decided that Raspberry Pis were to be used to implement the project. The reasoning behind this decision was quite simple: they are cheap, available, and run Linux. These are important for this project.

Linux is widely used among many low power sensors due to its developer friendliness, as well as being the basis for Android, one of the 2 major mobile operating systems. With an embedded system, the OS is important as the API will usually be very different between platforms, as they each have a different take on how to implement it. The Linux API is widely used by developers to create their own Bluetooth devices, so it has plenty of resources to draw from.

The price and availability of the device is another important point. Part of the aim of this project is to run this system on all types of devices. Running it on high powered devices, with plentiful RAM and cores, is a trivial exercise as it will work well on them. The challenge is for it to run well on devices which do not have vast computational resources. These devices will be what makes this system worthwhile as mobile devices will be more likely to be mobile due to their form factor. Mobile phones tend to have fewer computational resources than a home PC, and have the form factor that this project needs devices to have to function properly.

6.1.3 Language

Another important detail was the implementation language. For this, a variety of factors had to be considered, such as developer knowledge of the language, language suitability, and library support for Bluetooth and BLE. For this reason, 2 languages were considered: Python and C.

Python is a very easy language to use, but is very powerful at the same time. It has a variety of supporting libraries for Bluetooth development, such as bluepy, as well as support for other ways to interface with Bluetooth on Linux, by using DBUS.

As well as this, there are a lot of examples which can be studied to gain knowledge about how to complete implementation.

C is a very powerful language, but difficult to get right. Unlike Python, it is at a much lower level and requires more thought for a variety of issues, such as memory management. Additionally, there are fewer packages for technologies such as Bluetooth. The main way to interface with Bluetooth is using DBUS, but a lower level version of its API which does less work for the developer. This makes it harder to implement the project. However, using C does allow the developer the ability to implement more complicated features and can enable the code to be more efficient if written correctly.

Comparing the two languages, they each have their merits: C is the more powerful of the languages and allows more control over the finer details of the project whereas Python does more for the developer which allows an implementation to be quicker and increases the readability of the code, an important detail for a proof of concept like this project. It was decided that Python were to be used for this project. The major deciding factors were the ease and speed of implementation due to the plethora of libraries available.

6.2 Implementation Detail

This section will present the details of the implementation of the system. It will look at all aspects, such as how the designs of the client and server were implemented and how they communicate with each other, and the different methods by which that is accomplished.

6.2.1 Client

Describe how the client was created. Talk about libraries and approach to designing it.

The client section of the project was designed in a way so that new behaviours could be added in the future. This led to a very modular approach. The *bluepy* library was used for this part of the project because it focused on forming connections with other devices. The client side had very different needs to the server side, so using a different for the client side allowed it to have bespoke functionality for itself.

The main section of the client side is a `client` class. This class contains all data which pertains to a connection, such as methods to read and write data to a connected server, as well as starting and ending a connection. There were also more complex methods which wrote to the server and read messages back.

Starting the client

When starting the client, the user has the option to specify which type of client they want to use: normal, secure, emergency. If no argument is given, the normal client behaviour is used. After this has been selected the main method, `start_client`, is called and the client behaviour function is passed in. This is the main method where searching for devices and running client behaviours occurs. The `client` object will be created, being instantiated with the clients' current location and its MAC address. This code uses a static location because the Raspberry Pi does not have GPS, however, the system does have functionality for changing the location in the `client`.

```
def start_client(func):
  cli = Client("52.281799, -1.532315", bluezutils.get_mac_addr(dbus.
   SystemBus()))
  print("Starting Client")
  while True:
    devices = cli.discover(5.0)
    for dev in devices:
      print("Scan Data: {}".format(dev.getScanData()))
      have_service = False
      if len(dev.getScanData()) >= 3:
        for uno in dev.scanData.keys():
          print("getValueText ",dev.getValueText(uno))
          if dev.getValueText(uno).lower() == cli.SERVICE_UUID.lower
   ():
            have_service = True
      if have_service:
        func(cli, dev.addr)
```

Listing 6.1: Main client function

This function will start an infinite loop whereby it will perform a number of steps. Each of these are repeated for as long as the program runs for:

- The client will spend 5 seconds discovering nearby devices.
- For each device that is found:
  - Check that the client has enough data about the server
  - If it does, go through this looking for its UUID list (list of services and characteristics it offers). Otherwise, move onto the next device
  - If it offers the required emergency service, set a flag `have_service`

– If `have_service` is set, call the client behaviour for the client along with the address of the server which offers the required service.

The code for this is shown in Listing 6.1. As well as this function running, a signal handler is registered for the client. This is used to handle any behaviour which may end the execution of the program so that it ends safely.

Writing Messages

```
1
2  def split_message(message, delim=chr(5), size=15):
3    print("Splitting message, size: {}".format(size))
4    byte_arr = []
5    message_len = len(message)
6    if int(message_len/size) == 0:
7      print("Small message size")
8      byte_arr.append(message)
9    else:
10     print("Breaking up message")
11     for i in range(0, int(message_len/size)):
12       j = (i+1)*size
13       byte_arr.append(message[i*size:j])
14     if message_len%size is not 0:
15       byte_arr.append(message[(i+1)*size:(i+1)*size+message_len%size
       ])
16   if delim:
17     byte_arr.append(delim)
18   return byte_arr
```

Listing 6.2: Method for splitting messages into variable sized segments

Sending a message works by setting a message to send as a variable in the `client` class. If there is a message set, and the method `send_message` is called, the method will move the message into a buffer and split it up into 15 byte segments. The way this was done is shown in Listing 6.2. This method takes a delimiter and a segment size. The terminator is sent at the end of a message to tell the other device that the whole message has been sent. This is used when writing and reading messages. It gives a point for a server or client to move on from the message and use the received data for something. The segment size is variable to allow for other uses of this method, or if the size needed changes. This came in handy during this project because the segment size used to be 16 bytes. It was reduced to 15 bytes to allow for a larger header on each message.

Once the message is broken down, the client will need to write each segment to the server. For this, it loops through each segment in the buffer. For each, it adds a header to the segment. This allows the server to know if it has received each part of the message. The method of writing to the server is *write with response*. This means that, when a message has been written to the server it will reply with a short message alerting the client it has received the message. This helps to increase the quality of service because the client does not write the next segment until the previous one has been written to the server.

```
def to_byte_array(value):

  # Convert string into some sort of char array
  char_arr = list(value)
  ret_list = []
  # For each member of the char array, get the ASCII code for each
   character
  for char in char_arr:
    ascii_v = ord(char)
    # Take each ASCII code and create a dbus.Byte object with it and
    add it to another array
    ret_list.append(dbus.Byte(ascii_v))
  # Once byte array built, return
  return ret_list
```

Listing 6.3: Method for converting string into byte array

The method by which Bluetooth transports data is by sending bytes of information between devices. For this reason, any data that has to be written or read has to be converted into a byte array. The method by which this is done is shown in Listing 6.3. In this method, a string is passed in and it will be converted to a list of characters. For each character, it will each be converted into its numerical representation and passed into a constructor to make a `Byte` object. The `Byte` object used here is from the DBUS library, but still works as a `Byte`. This reason for using this is that its a requirement for the server, but does not affect how the client works. This reduces code reuse later.

Once the segment has been converted into a byte array, it is written to the relevant characteristic that was decided when the `client` object was instantiated. This process will continue for each segment in the buffer, with the sequence number being incremented each time. Also in `client` is a secure writing method which is used when encrypted messages are being sent (`send_secure_message`). This is discussed in Section 6.2.2.

Reading Messages

Another key part of client-server interactions is the ability for the client to read data from the server. In `client`, the ability to read messages is contained in `read_message`. The first thing the method does when called, is to check a server is connected. If there is, it proceeds with the rest of the method. The main idea of the method is that the client will keep reading from the server until it receives a message which contains the message terminator. This tells the client that the message has been fully read from the server.

The client will repeatedly read parts of the message from the server, each time converting them from a byte array to a string. For each part read from the server, the sequence number and segment are separated. The sequence number is checked to see if it is the first message that has been received. If it is, first a flag, `first_mess`, is set. This indicates the first message has been received. Then it continues with the normal behaviour. If the segment is not the terminator, the segment is added to a buffer. If it is, the message is complete and the buffer is joined together. The check for the first message is done in case there was a problem during the last interaction between the server and client and it starts sending segments from halfway through the buffer on its side.

When the message has been fully read and the terminator has been received, the segments in the buffer are joined together in a string and that is returned. This allows the message to be processed elsewhere however is needed. In `client`, there is an additional read function called `read_secure_message` used for secure data exchange. This is discussed in Section 6.2.2.

6.2.2 Client Behaviours

The main aim of designing the client behaviours was to make them extensible and modular so that they could be added to later on. Also, they were implemented in such a way that it would be easy to create more client behaviours later without changing the structure of the client in a meaningful way. By abstracting this out of the main parts of the program, it lowered the code complexity and reduced the scope of bugs and programmer error.

Each client behaviour is a global function in the file `blueclient.py`. Each function encapsulates the bespoke actions that need to be performed for each different type of node. Originally, all of this behaviour was intertwined, using a system of complex tokens being sent between the client and server to help decide what type of behaviour was needed on both sides. To reduce this complexity, a function was made for each different client behaviour and a characteristic was made on the server side

to correspond with each. This means, each behaviour will connect to and interact with a part of the server which is made to interact with it. This further reduced the complexity and increased the modular nature of the project.

Normal

```
def add_to_db(db, broken_down_msg):
  now = datetime.datetime.now()
  coords = broken_down_msg['1']
  addresses = broken_down_msg['2']
  values = []
  if len(coords) == len(addresses):
    for i in range(0, len(coords)):
      values.append((addresses[i], coords[i].strip('()'), now))
  db.insert(values)
  print("Added values to DB")
```

Listing 6.4: Function which adds information from a dictionary about a message, to a database

The `normal` client behaviour is the basic way in which a client communicates with a server. The client will start by building a message to send. It will select 50 records from the database. This is the maximum number of device location records that can be sent between devices in a message, can also be referred to as the timeout. It then adds the location and MAC address of the device. Using this data, the function `build_message` is used to create an EFIX message string containing all the data. This function is shown in Listing 6.5. This message is set in the instance of `client` passed into the function.

```
def build_message(locations, addresses, filter_addr = None):
  message=[]
  if filter_addr:
    print("Addresses to filter: {}".format(filter_addr))

  for i in range(0,len(locations)):
    if filter_addr and addresses[i] in filter_addr :
      print("Filtering address: {}".format(addresses[i]))
      continue
    message.append("1=({})|2={}|".format(locations[i], addresses[i])
   )
  true_mess = ''.join(message)
  print("Built Message: "+true_mess)

```

```
14    return true_mess
```

Listing 6.5: Function used to build an EFIX message string containing MAC addresses and locations

Using the address passed into the function, the client will connect to the server at that MAC address. It will then connect to the normal characteristic on the server device. The client will then read a message from the server using the previously described `read_message` function. The message read will be taken and broken down based on the tag-value pairs in the message. This is done by the function `break_down_message`. This function is displayed in Listing 6.6 and further discussion of it is in Section 6.3. The information gained from breaking down the message is taken and inserted into the database. The method `break_down_message` will return a dictionary of the key value pairs in the message. This can be taken and passed into the method `add_to_db`, along with a reference to the database object.

```
1  def break_down_message(message):
2    print("Breaking down: {}".format(message))
3    ret_dict = {}
4    save = ''
5    tvps = message.split("|")
6    for tvp in tvps:
7      tvp_no_equals = tvp.split("=")
8      if len(tvp_no_equals) is 2:
9        save = tvp_no_equals[1]
10     elif len(tvp_no_equals) > 2:
11       save = "=".join(tvp_no_equals[1:])
12     # Check if entry for tag exists
13     if tvp_no_equals[0] in ret_dict:
14       # If it does, get list associated with it and add element to
     it
15       ret_dict[tvp_no_equals[0]].append(save)
16     else:
17       ret_dict[tvp_no_equals[0]] = [save]
18   return ret_dict
```

Listing 6.6: Function used to build an EFIX message string containing MAC addresses and locations

`add_to_db` is tailor made to deal with this data and insert it into the database. it is shown in Listing 6.4. As can be seen, the current datetime will be recorded and the device coordinates and address will be separated out. When the message is broken down, the ordering of elements is maintained within the lists containing the coordinates and addresses. There is an `if` statement to check if the number of coordinates and address is the same, as a sanity check to make sure nothing has gone

wrong. If this passes, a loop goes through each pair of addresses and coordinates and inserts them into a database, which is passed in as a parameter, along with the current datetime for each pair.

Once the data from the read message has been dealt with, the pre-built message is written to the client. This message was created previously containing data from other devices, as well as the current device.

Secure

The `secure` client behaviour builds on much of what is in the `normal` behaviour. As described in Chapter 5, the encryption process follows much of what is done in the base behaviour, but adds some extra functionality on top. When the `client` object is instantiated, it will generate a 1024 bit RSA keypair, which this behaviour needs.

Once the secure behaviour function is called, it will build a message in the same way as the `normal` behaviour. It will then connect to the device which belongs to the passed in address and will connect to its secure characteristic. This will ensure the server knows it is a secure client and will act accordingly. The client will start by creating a message containing its public-key, using an EFIX message with tag 3. This is used for sending an RSA public-key. This message will be written to the server. The server will notice that this is a public RSA key and will send their public RSA key back when the client next reads from the server. This will ensure that both the client and server have each others public RSA keys.

When each device holds the other public-key, communications can be secured. The originally generated message is set as the next message to send to the server. The client will then securely read a message from the server using the function `read_secure_message`. This function works in much the same way as a reading a normal message, but with a few changes. To understand this, we have to look at how messages are encrypted and sent.

```
1 def encrypt_message(public_key, message):
2   key = RSA.importKey(public_key)
3   cipher_rsa = PKCS1_OAEP.new(key)
4   return cipher_rsa.encrypt(str.encode(message))
```

Listing 6.7: Function used to encrypt messages

```
1 def decrypt_message(private_key, ciphertext):
2   key = RSA.importKey(private_key)
3   cipher = PKCS1_OAEP.new(key)
4   return cipher.decrypt(ciphertext).decode()
```

Listing 6.8: Function used to decrypt messages

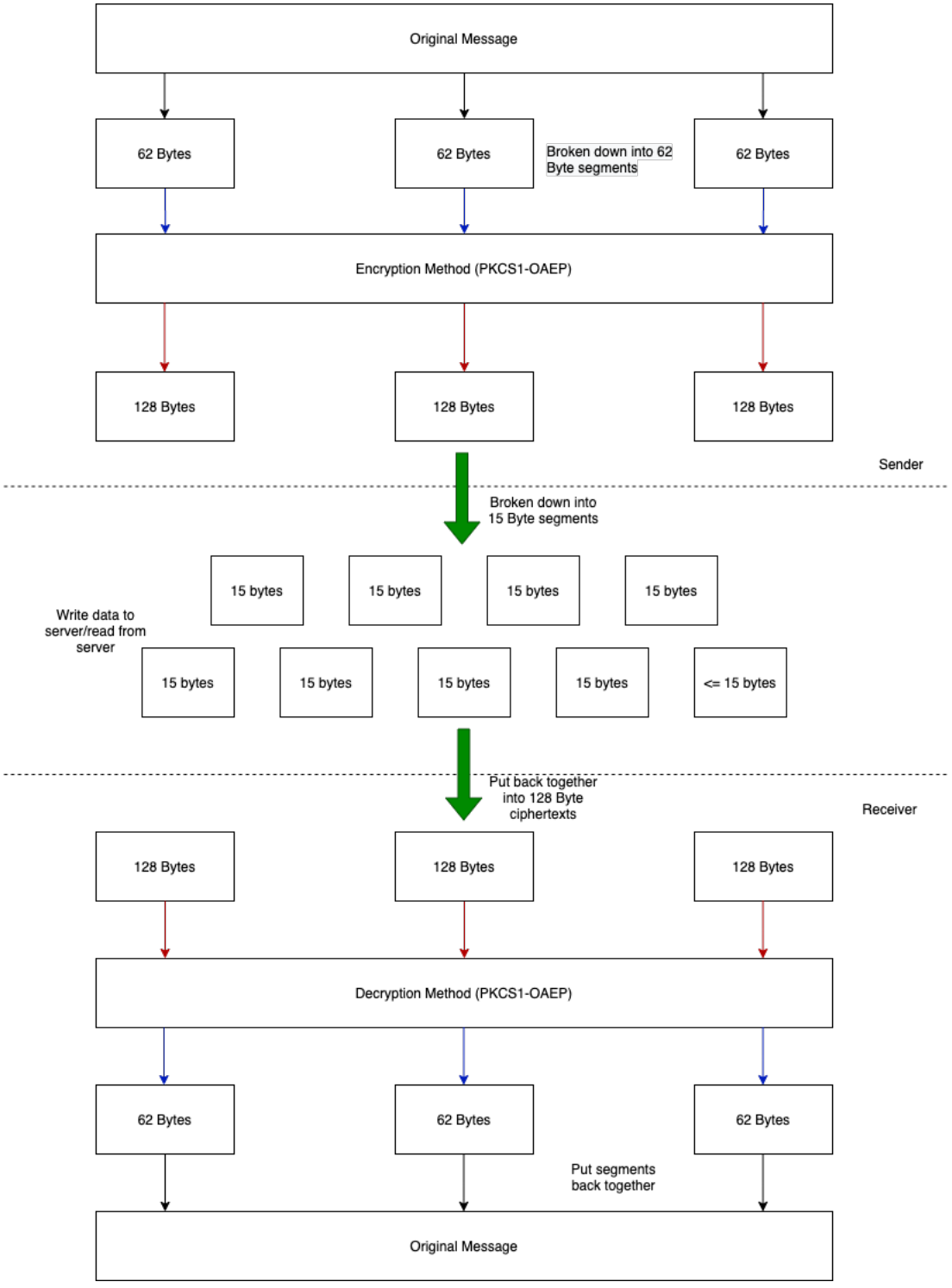


Figure 6.1: Illustration of how the encryption functions

A limitation of the library used for RSA encryption, PyCryptodome, is that encrypting and decrypting data is limited to certain sizes of data. This size is: (number of bytes in the key) - 2*(hash output length) - 2. This creates the equation: 128-64-2=62 bytes. If a message is more than 126 bytes, it cannot be encrypted using this method. The way this challenge was overcome was by splitting the message up into 62 byte segments and encrypting those using the other devices public key. This gave us a ciphertext which is 128 bytes long. This ciphertext is too long in itself to send to the server, so it too is segmented and sent as segments usually are. This is how the function `secure_write_message` works. The functions for encrypting and decrypting data are shown in Listings 6.7 and Figure 6.8.

When reading these messages, each 128 byte ciphertext is added to a buffer until the whole ciphertext has been read by the client. This is known as it will receive a message with a header and just the terminator. Once this has been received, the ciphertext is joined together and is then decrypted using the devices private key. Once this segment has been decrypted, it is added to another buffer. This buffer is used to hold the decrypted segments of the message being transmitted.

This process is highly complicated, and probably the biggest challenge of the implementation. It is easier to comprehend using the diagram in Figure 6.1. This diagram illustrates how a message is taken, segmented, encrypted and segmented again before being sent, then the receiving device reversing that to reveal the message that has been sent once they have received all the required parts. This then means the receiving device can break down the message as it does in the `normal` client behaviour.

Emergency

The `emergency` client behaviour is far more simple than both the `secure` and `normal` behaviours. This function starts by creating a message using tag 5. The message, when written to the server, will tell it that the next message that will be read, will be an emergency message read. This means it will need extra information in the message that it reads.

The client will then connect to the server using the address passed into the function. After this has succeeded, the client will connect to the Emergency characteristic. It will then send the message created previously. After it does this, it will do its emergency read. For this, normal read functions are used, except the contents are special. This message will contain information about the server, as well as data about the latest unique devices it has encountered, such as its MAC address, location and time the server heard about it. This gives the emergency client more information

about a device as it will know when that server node last heard about each device it tells it about.

The client will then do a normal read as well. This is done as a server may have heard about multiple devices recently, which will give the emergency client more information about the devices in the nearby area and how they are moving about. This information is taken and broken down by the client, before adding it to its database.

In the future, it may be beneficial to add another aspect of functionality to this where it has a "data dump" mode, whereby an emergency client could request all of the data that a server node has. This would require the server to change several different parameters, such as increasing the size of the header and decreasing the size of the segments. This would allow an emergency node to gain far more information about everything the server has seen in its lifetime.

6.2.3 Server

The server side of the project is a much more complicated system than the client side. It has to do far more underlying jobs than a client needs to do. It is structured so that it has a main loop which runs continuously and has a variety of other methods attached to it. These methods will print information about devices which are trying to connect, as well as other servers in the area. A major part of the server side is DBUS, a message bus technology which is used by services to allow developers to interface with other services over. On Linux, the Bluetooth service uses this. To manipulate this in a meaningful way, messages have to be passed to it over DBUS to implement complex functionality, such as registering an advert or a GATT profile. This concept is illustrated in Figure 6.2. A few pyhton modules were used to implement this: `dbus-python`, `PyGObject`, and `pycryptodome`. To run these, other libraries needed to be installed on Linux for them to work: `libgtk2.0-dev` and `libdbus-1-dev`.

The Bluetooth service will allow users to register customised parts of its system, such as adverts and agents. This allows the user to tailor the behaviour of Bluetooth to suit the application at hand. However, to register something to the Bluetooth service, it has to fulfil the API. This means that a base class has to be created which implements each function needed for the API, as well as tagging each one with a `@dbus.service.method` tag, which defines the inputs and outputs of the function below it, as well as the interface that the method is defined in. Once this is implemented, another class can build on this and offer bespoke functionality, such as the advert for the service being provided. Many of the base classes that have been

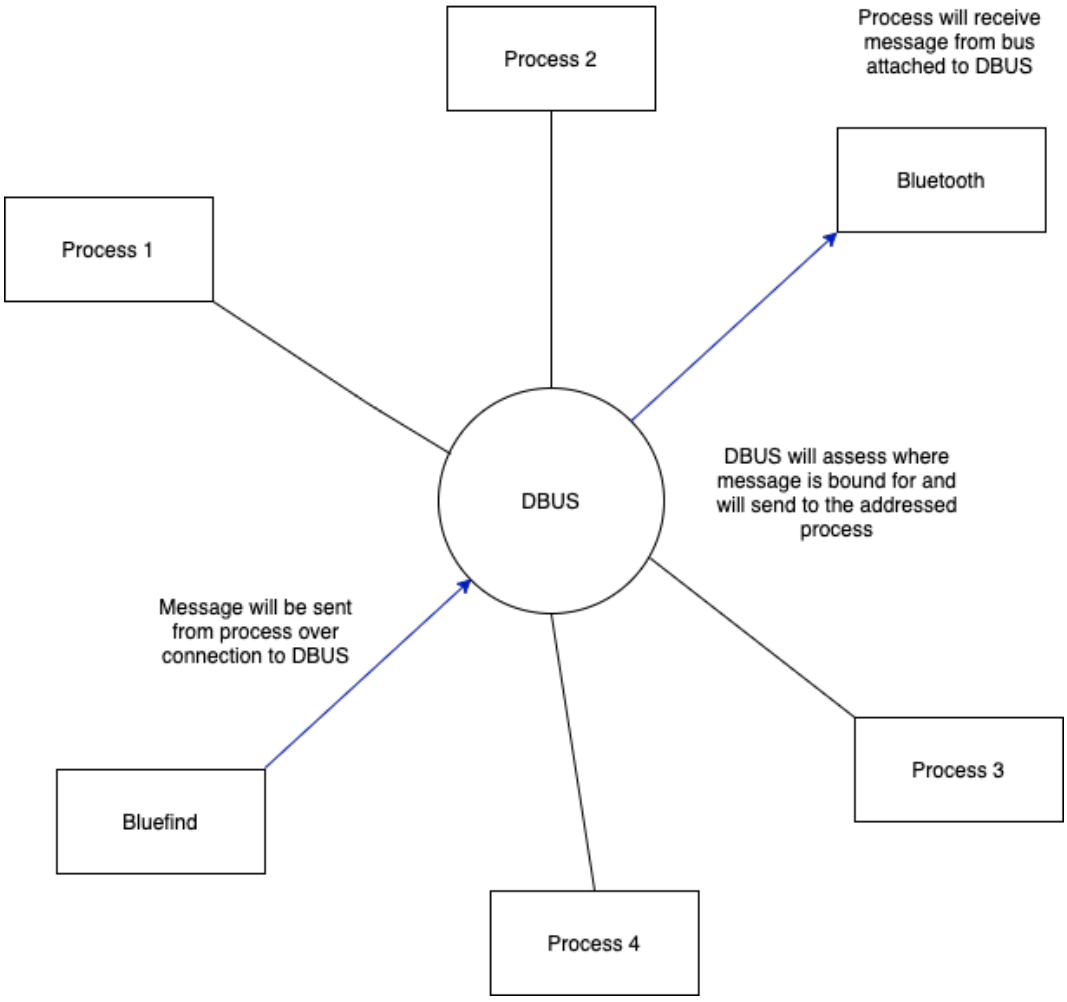


Figure 6.2: Illustration of how processes communicate using DBUS

implemented in this project were shown in example code provided by BlueZ [70], the project which the Bluetooth service belongs to. Using this code, the bespoke implementations were built on top of them.

The main function, contained in the file `bluefind.py`, starts by setting a signal handler. This is used to deregister the advert and agent when the program quits to ensure a clean exit. It will then go through and setup the system bus required to communicate with the devices Bluetooth service and it will register the agent, advert and GATT profile. These will all be used by the Bluetooth service to perform its jobs, such as advertising its services, allowing devices to connect and performing data communications with connected devices. It will then start the main loop and keep the program alive, allowing devices to see, connect and interact with the Bluetooth service running the bespoke functionality for this project.

Advertising

```
class EmergencyAdvertisement(Advertisement):
  def __init__(self, bus, index):
    Advertisement.__init__(self, bus, index, 'peripheral')
    # Change this when UUID is finalised
    self.add_service_uuid('FFF0')
    self.add_manufacturer_data(0xffff, [0x00, 0x01, 0x02, 0x03, 0x04
   ])
    self.add_service_data('9999', [0x00, 0x01, 0x02, 0x03, 0x04])
    self.add_local_name('EmergencyAdvertisement')
    self.include_tx_power = True
    self.add_data(0x26, [0x01, 0x01, 0x00])

    def register_ad_cb():
      """
      Callback to be called whent the advertisement is registered.
      """
      print('Advertisement registered')

    def register_ad_error_cb(error):
      """
      Callback to be called when there is an error in registering
      advertisements.
      """
      print('Failed to register advertisement: ' + str(error))
```

Listing 6.9: Class which describes the device advertisement

The basis for an advert is the `Advertisement` class. This class implements the standard class `dbus.service.Object`. Within `Advertisement`, it defines a variety of

functionality needs, such as returning the path of the advert, returning the properties of the advert and being able to add different pieces of information to the advert, such as a UUID for the service.

Using this base class, a class called `EmergencyAdvertisement` was created. This implements the `Advertisement` class. This class contains information for the advertisement, such as the UUIDs that it wants to advertise and the name of the device to be displayed. This class is shown in Listing 6.9. As can be seen, there are 2 callbacks also defined. These are used when registering the advertisement to say whether it has been registered successfully or not.

Agent

The `Agent` is an important part of the Bluetooth system as it defines many of the key security components in Bluetooth, such as entering a code to enable pairing or confirming that the pin code was correctly entered and that a connection may be formed.

The standard Agent for the Bluetooth service requires many of these things as standard. For this project to work without any user interaction, devices have to be able to connect without a user having to enter a code. For this reason, an Agent has to be registered to allow devices to just connect and not have to authenticate its connection using this method.

```
1
2  @dbus.service.method(AGENT_INTERFACE, in_signature="ou",
    out_signature="")
3  def RequestConfirmation(self, device, passkey):
4    """
5    This function will confirm if the connecting device pass key is
6    correct. As we don't care about this, it is always approved and
7    trusted.
8    """
9    print("RequestConfirmation (%s, %06d)" %(device, passkey))
10   trust_device(device, self.bus)
11   return
```

Listing 6.10: Function to verify the correctness of the passkey provided

```
1
2  @dbus.service.method(AGENT_INTERFACE, in_signature="o",
    out_signature="s")
3  def RequestPinCode(self, device):
4    """
5    This function will give a simple, easy pin code. The program
    doesn't
```

```
6       use pin codes and so this function is here to fulfil the spec of
        the
7       interface.
8       """
9       print("RequestPinCode (%s)" % (device))
10      trust_device(device, self.bus)
11      return "0000"
```

Listing 6.11: Function to provide a requested pin code

Much like with advertising, the `Agent` class was implemented to the specification of the API, except for each authentication method, the functionality was changed to allow any connection to be made. An example of this is in Listing 6.10 where it shows the `RequestConfirmation` function. This will trust the connecting device and will not even check the passkey provided if it provides one. Additionally, Listing 6.11 shows a function whereby it will automatically provide a pin code. This is so if it requests a pin code from a device, the pin code it is expecting is standard and always the same.

The Agent is an important part of the project. Although it does not add a lot in terms of functionality, it allows devices to easily connect and disconnect. Considering the security of the system, it is not ideal as it is reducing the authentication of the client. As described before, in a future version, this kind of authentication could be moved to a higher level and use some sort of digital authentication to authenticate a connecting device further.

GATT Profile

As discussed in Chapter 5, a GATT profile has a structure which could be likened to that of an OOP software project. The implementation details of this are similar. Each part of the GATT server needs a base class derived from `dbus.service.Object`. This is so DBUS can use it and register it as a profile. For this, classes for the profile (called application here), service, characteristic and descriptor are needed. We do not need descriptor, but it is needed for the API.

Each base class in this part of the project implements similar functionality. Each class will implement functions to return a string containing the path of the object when it is registered. Additionally, all classes implement a function which returns a dictionary containing information, such as paths, about all objects within it and information about itself. This also includes objects nested within other objects, so the Bluetooth service can register all parts of the profile that are needed.

The characteristic and descriptor will implement some extra functionality that the other classes do not implement. This is due to them holding values which clients

might hold an interest in. For this reason, both of them implement the functions `WriteValue` and `ReadValue`. These are the main functions with which a service or descriptor communicate with a connected device using. These allow them to give information to a server (`WriteValue`) or get information from the server (`ReadValue`). Additionally, the characteristic class has the ability to set up a "signal". A client can subscribe to this and the server will tell them when a certain value changes. For example, this would be useful for a blood pressure monitor connected to another device. Instead of the connected device constantly reading, it will be alerted to any changes in blood pressure.

Using these base classes, an `EmergencyService` class was created and implements the `Service` base class. This implementation adds the bespoke characteristics into its `init` function so that they are registered to the service. Additionally, it sets a UUID for the service which is used when advertising the service. The rest of its functionality is implemented in the base class. The `Application` class is used when registering the profile. It is used more as a way to easily register services and does not have much more of an effect than that.

Characteristics

This is where the most useful GATT functionality lies in this project. Although the other parts of the server are needed, the characteristics are where the key work is done. This project implements 3 characteristics: **NormalCharacteristic**, **SecureCharacteristic**, **EmergencyCharacteristic**. Each one corresponds to a specific client behaviour.

```
    if (dev in self.write_states.keys()) and  int(sequence_num) is
  len(self.write_states[dev]):
      self.write_states[dev].append(message)
      if str(chr(5)) == message:
        # If it is in message, join up message
        print("End of message")
        full_message = ''.join(self.write_states[dev])
        print("Message Written To Server: {}".format(full_message))
        # break down message
        message_parts = bluezutils.break_down_message(full_message)
        print("Keys in Message: {}".format(message_parts.keys()))
        bluezutils.add_to_db(self.db, message_parts)
        print("Processed whole message from {}".format(dev))
        del self.write_states[dev]
      return sequence_num
    elif int(sequence_num) is 0:
      if str(chr(5)) == message:
```

```
17            print(message)
18          else:
19            self.write_states[dev] = [message]
20          return sequence_num
```

Listing 6.12: Section of function which shows dealing with new segment

NormalCharacteristic is the base from which the others are built. It has variables for its MAC address, database connection and location. Much like in the client side, static coordinates are used due to the limitations of the platform being used. This class implements the `Characteristic` class and overrides two functions already implemented: `WriteValue` and `ReadValue`. Both of these methods use a buffer to hold partially received, or sent, messages for each connection. These can do this for multiple devices at once. In `WriteValue`, when something is written to it, it will split the sequence number and segment. If this is not the first segment received for a message, it will be added to the buffer for the device. However, if it is the terminator character, it will not add it to the buffer and will form the whole message from the buffer before breaking it down and storing the values in the database. This is the same as for the client side. This is shown in Listing 6.12. Also shown is that, if the sequence number is 0, meaning it is the first segment to be received for a message, it will create a new buffer for the connected device.

`ReadValue` is split into two halves. One half deals with a new client reading from it, and the other half deals with a client asking for another section of an already partially sent message. If it is the first segment being read by the client, the message to send will be generated and broken down, like on the client side, before being stored in a buffer alongside a value, pointing to the next segment to send for that device. Every time a device reads the next segment, the message will be formed and the segment will be found. The pointer will then be incremented so that the next read knows which segment to send. Once the whole message has been read, the buffer is cleared. Before the message is returned from the function, it is converted into a byte array, done using the function shown in Listing 6.3.

SecureCharacteristic follows the same protocol shown in Figure 6.1. It holds an RSA keypair and will use two buffers for a device to hold the encrypted and decrypted parts of messages. In `WriteValue`, a dictionary `local_place` is used to hold the segments of ciphertext that are transmitted. Once a complete ciphertext has been written, the private key is used to decrypt it. It is then stored in the global buffer for that device. Once the full message has been received, it will break it down based on its tags and the data will be added to the database. The difference between this `WriteValue` and that implemented in **NormalCharacteristic** is that this will put segments into its lower buffer first, before decrypting and placing in a higher

buffer whereas **NormalCharacteristic** will just place all segments into one buffer. Also, it will have a public RSA key written to it which it will store. This is so it can be used by `ReadValue` when it needs to encrypt a message for a client to read.

`ReadValue` will first send its RSA public-key to a device if its own has first been written to the characteristic already. Once this exchange is complete, the device will create a message for the client to read. It will create a global buffer for this and will break it into 62 byte segments. This is the global buffer for reading. If the local buffer is empty, a segment is taken from the global buffer, encrypted and then segmented into 15 byte segments before being stored in the local buffer. Every time a read is performed, the segment to send will come from the local buffer. When both the global buffer and local buffer have been read, a terminator will be read by the client and the buffers will be cleared ready for future interactions.

```
def build_generic_message(message_struct):
  """
  Takes a dict where keys are tags, and each has a list with
  values for that tag, e.g
  {
    1 : [...],
    2 : [...],
    3 : [...],
  }
  """
  message = ""
  for key, value in message_struct.items():
    for item in value:
      message+="{}={}|".format(key, item)
  return message
```

Listing 6.13: Generic message building function

EmergencyCharacteristic works in a very similar way to **NormalCharacteristic**. In `WriteValue` works in the same way. It will keep adding segments to a buffer until it gets sent a terminator. Once this is received, the message received is joined together from the segments in the buffer. This is then broken down. Instead of adding this to the database, it will set a flag if the message found contains tag 5. This is to indicate that when `ReadValue` is called for the first time, it will generate a message which contains the usual information about locations and addresses. Additionally, it will use tag 6 to specify the datetimes on which each location/address pair was received. This means it is a bigger message and uses a different, more generic message builder function called `build_generic_message`. This function, shown in Listing 6.13, will create a message using a dictionary of lists where the dictionary keys are the keys for each value in its list. The flag set earlier is then flipped to

reflect that the emergency message has been generated. The `ReadValue` function will then work as it does in **NormalCharacteristic**, it will place this message in a buffer and segments will be read from it until its empty. If the flag is not set for an emergency message, this function will call to build a message as it does in **NormalCharacteristic**.

Overall, this section has shown how the client and server were designed to be modular. They have been designed so that future work can be done to extend and improve the functionality to make it even more useful in the future. It has also described the implementation of the complex aspects of the project, such as encryption. This shows the technical achievements of the project, and how the system has been carefully taken from the design stage and implemented as a working system.

6.3 Message Format

The type of message sent between devices in this system is called EFIX, a variant of the Financial Information eXchange protocol. It uses a system of tags and values to specify context for values. In this system, there are several different tags, as well as helper methods which are used to deal with these messages. The tags used in this project are:

- **1**: Data is the coordinates of a device
- **2**: Data is a MAC address of a device
- **3**: Data is a public RSA key
- **4**: Previously used tag but now unassigned
- **5**: Flag used during **EmergencyCharacteristic**
- **6**: Data is a datetime string

By using these tags, both sides of the data exchange know what the context is for each tag and so easily know what to do with the value of the tag-value pair. This provides extra security as a device will ignore any tags which it is not looking for.

Within EFIX, there are protected characters. These are ones which are used for jobs and so cannot be used by tags or values. These characters are "|" and "=". The reason being is that they are needed to split up the message and separate parts, such as the tag from the value and separating different pairs.

```
def break_down_message(message):
  """
  When provided with a FIX-type message, the function will split
  it up and will store the values in a dictionary, which will then
  be returned.
  """
  print("Breaking down: {}".format(message))
  ret_dict = {}
  save = ''
  tvps = message.split("|")
  for tvp in tvps:
    tvp_no_equals = tvp.split("=")
    if len(tvp_no_equals) is 2:
      save = tvp_no_equals[1]
    elif len(tvp_no_equals) > 2:
      save = "=".join(tvp_no_equals[1:])
    # Check if entry for tag exists
    if tvp_no_equals[0] in ret_dict:
      # If it does, get list associated with it and add element to
   it
      ret_dict[tvp_no_equals[0]].append(save)
    else:
      ret_dict[tvp_no_equals[0]] = [save]
  return ret_dict
```

Listing 6.14: Function to break down a message based on its tags

Helper methods were used to breakdown the message into a form which could be used by other parts of the system. One such method is shown in Listing 6.14. This function takes in a message and splits it up based on the delimiter. This character is "|" in EFIX. This is the character which separates tag-value pairs from each other. Once the function splits the message, it will be in a list, each item in the list being a tag-value pair. A loop will go through these and will split them on the "=" character. This will split up the tag and value. Then, if there is both a tag and value, the value will be added to a dictionary of lists, `ret_dict`, which will hold all the values for a certain tag. This will then be returned from the function. Using methods like these shows it has scope for expansion in the future and is beneficial to the project.

6.4 Reflection

The implementation of the project had various peaks a troughs. Originally, the implementation was attempted in C using the DBUS API. A lot of time was sunk into this approach but it was felt that it was not moving quick enough, hence the

switch to Python. Although that avenue did not bear fruit, it did provide extensive knowledge in how DBUS works and helped to allow the project to move quickly later on.

The client side of the project fits in well with the server side. Each complement each other well. This is due to the extensive design work which was done to allow this when implemented. Using different libraries for each side was difficult at first, because it meant having to almost produce two different projects, but the libraries chosen do have huge merits. Both specialise in the areas in which they are used. `bluepy` cannot do most of the stuff that using DBUS can accomplish, and vice versa. It meant that the overall result was a more flexible and powerful system.

The biggest challenges of the implementation were getting started and encryption. This is a very difficult area to understand due to the lack of documentation and examples to work from. This led to countless hours reading source code line by line trying to understand how Bluetooth worked on Linux. This was especially true for DBUS. Encryption was a large challenge too that was overcome in this project. Originally, a crude implementation was done which limited the number of addresses and devices which could be transmitted such that the total size was less than, or equal to, 126 bytes, the max size for using a 2048 bit RSA key. The new approach allows a smaller RSA key, 1024 bit, to be used meaning encrypting data is quicker, and allows more data to be included in a data exchange between two devices. This is a significant technical achievement of the project.

There are many different avenues for future work in this area. Knowledge gained from this project could be applied to re-writing the system in C or C++. This is because they are more performant languages (if used correctly) and allow various other optimisations, such as making it easier to have a smaller header on each message by treating it all as 1 number. If the header was decreased by 1 byte, it could have been considered as a 16 bit number. This would mean a maximum of 65535 segments could be sent for a message, much higher than the current limit of 999 messages than is achieved with the 3 byte header.

Chapter 7

Testing

Testing is very important for a software project. It provides a way to verify that objectives have been met for project requirements. Additionally, it helps to debug issues in the project and ensure that changes alter the functionality in a way that is intended. The unit testing framework used by the project is **PyTest**. This was chosen because it is a simple, but powerful, testing framework. Alongside this, **TravisCI** was used for integration testing. It is a widely used service with integrates seamlessly with GitHub, the remote version control used by the project.

7.1 Unit Testing

Tests were written to test different parts of the project to ensure they function correctly. This task proved tricky in some places, for example where code was written to perform Bluetooth operations. This is because it is platform-specific and could not be run on a continuous integration platform like TravisCI that uses virtual machines. However, efforts were taken to cover as much of the codebase as possible. The benefits of unit testing and how it works are further discussed in [71, 72]. In these books, they discuss the benefits of user testing during development and how these can bring benefit to the project.

Unit testing was used along with TravisCI and GitHub to automatically approve merging in the repository to ensure that merging branches is safe and does not alter established functionality in the program.

Without unit testing, other methods had to be devised to ensure Bluetooth code is rigorously tested. With this, a set of scenarios were drawn up, along with success scenarios. This had to be performed and verified manually. The set of test cases is displayed in Figure 7.1. Each of these tests are run when merging branches, and

Number	Test Details	Success Criteria
1	Start up server	Print device address, register advertisement, register GATT application, local device scanning starts
2	Start up client (any mode)	Print device address and start scanning local devices
3	Start up server and client (normal) then disconnect	Server and client startup as in **test 1** and **test 2**. Client locates and initiates connection with server. From this, they will each exchange messages and store information in their databases. Disconnection then occurs for both devices without errors.
4	Start up server and client (secure) then disconnect	Server and client startup as in **test 1** and **test 2**. Client locates server and performs RSA key exchange. Encrypted message exchange occurs using other devices public RSA key. Messages are decrypted using private keys and data stored in databases. Disconnection then occurs for both devices without errors.
5	Start up server and client (emergency) then disconnect	Server and client startup as in **test 1** and **test 2**. Client locates server and *write*s message to GATT server. Server then allows for Emergency *read*. Client then performs second, regular *read* operation to get extra data. All of these messages broken down and data stored in database. Disconnection then occurs for both devices without errors.

Table 7.1: Client/Server Test Criteria

individual tests are run Ad-Hoc when changing parts of the project it is testing.

7.2 Success Measurement

In the *Progress Report*, the success criteria of the project was defined clearly. This defines success as being able to have multiple nodes connected to a central node, passing information in a repository node (emergency node). Due to the implementation of the project, the central node (server node) couldn't be the repository node, this has to be a client node. Overall, the project has met the success criteria defined previously, as well as other functionality, such as encryption to enhance the security of communications.

Furthermore, the success of the project was judged based on the requirements

for the project in Chapter 4. This sets out further, in more depth, targets for the project which allow for a deeper assessment of the success of the project.

This combination of unit testing, which can be run automatically and quickly, as well as broader, more complicated testing, allowed the project to stay on track and helped to identify issues when they arose. This approach led to comprehensive coverage of the different system elements and provided markers to measure progress and verify the success of the project.

Chapter 8

Project Management

	Start Date	End Date	Timeline	Status
Project	30-09-2019	23-04-2020		
Initial Research	30-09-2019	07-10-2019		Done
Specification	30-09-2019	10-10-2019		Done
Routing Algorithms Research	11-10-2019	19-11-2019		Done
MAC Protocol Research	11-10-2019	19-11-2019		Done
Data Security Research	11-10-2019	19-11-2019		Done
Progress Report	11-10-2019	25-11-2019		Done
Implementation	26-11-2019	19-02-2020		Done
Presentation	20-02-2020	04-03-2020		Done
Final Report	11-10-2019	23-04-2020		Doing
	30-09-2019	23-04-2020		

Figure 8.1: Project Timeline

The running of a project such as this, over a significant time period, requires significant planning. Each different aspect needs to be accounted for. This chapter illustrates the timeline of the project, the tools which are used by the project to manage it, as well as the risks which may arise concerning the management of the project.

8.1 Project Timeline

The project timeline, as seen in Figure 8.1, is represented as a Gantt chart. It planned the dates between which each task took place. All parts of the project have been completed. This plan was created at the start of the project to outline all the different tasks that needed to be completed to accomplish the goals of the project. The research tasks took differing amounts of time. The routing algorithms and data security research took less time than was originally planned for, but the research in

the MAC layer protocols took longer. This is mainly because the different Bluetooth APIs were implemented in different ways, on different platforms and in different languages. This meant this continued while implementation started. After this was completed, the implementation and presentation were completed on time.

8.2 Project Tools

The project was implemented in Python3 [73]. This language was chosen due to the speed at which components could be created, as well as the ease at which modules can be added, as well as the portability of the code. Furthermore, the code is more readable, which is important as the implementation is a proof of concept.

An important part of the project is Bluetooth. To use this, a software medium was needed to use this. For this, 2 different packages were used. The first one is Bluez [70], used for server devices. This is an API to allow programmatic control over Bluetooth. It uses DBUS [74] to facilitate communication between the program and the lower level Bluetooth system. This is specifically for Linux machines. This API has many example programs which also made it very attractive and meant for a quicker learning process. These examples heavily influenced the structure and style of the server code. For the client side, another package was used, BluePy [75]. This API allowed client side devices to connect to server devices dynamically.

For encryption, the package PyCryptodome [76] was used to perform RSA encryption. The reason for using this package is that it is safer to use a more established encryption library as there is a better guarantee that it is correct and works.

The downside of using these packages is that performance is reduced due to the overheads of using Python3, but the speed of implementation and readability of the code means that Python3 was the best choice of language for the project. In the future, it would be beneficial to consider using a lower level language, such as C++ as greater emphasis can be put on improving the performance of the project while maintaining functionality.

The project used Git with GitHub for version control management and offsite backup. This was used due to project members familiarity with the systems and the industrial reputation they have. This allowed the project codebase to be stored securely as well as being used, maintained and tested on multiple systems easily. Furthermore, a Kanban board was used for the repository to document tasks which were completed and which still needed to be done. This allowed the project to be managed easily and for project requirements to be broken down to smaller, more manageable tasks which could be accomplished. This board was created and used on GitHub using the *Projects* functionality in a repository.

8.3 Risk Management

A risk posed to this project was that all machines which hold the codebase could be simultaneously destroyed. To mitigate this risk, all code is stored on **GitHub**, as well as maintaining a copy on an external hard drive. The key to protecting against this is to distribute copies across multiple locations to spread the risk.

Additionally, illness could have affected members of the project which may have prevented access to resources needed for the project. To overcome this risk, communication was be maintained with DCS and the tutor associated with the project to ensure they have all the information needed in case any event occurred which would have prevented the completion of the project, or its associated milestones.

Lastly, a task may take longer than is planned to complete. This risk has arisen during the project when part of the research took longer than planned for. The impact of this was mitigated by having generous time allowances for tasks and starting implementation in parallel with finishing the research stage.

Chapter 9

Evaluation

This chapter will evaluate the project as a whole. It will look back over the different aspects of the project and will reflect on what has been done. It will discuss how effective the project was at fulfilling the requirements set out at the start and how well the project has been managed. Finally, there will be an overall evaluation of the project as a whole. This will reflect on the content of the design and implementation work and how well the project improves the current efforts to improve current work in the problem area.

9.1 Functional Requirements

Reflecting on the requirements of the project, the project fulfilled all of its **must** functional requirements, but did not fulfil the **could** functional requirements. The reason for this was that the **could** requirements ended up being quite domain-specific. For example, using location discovery services depends very much on the device which the program is being implemented on. Using Raspberry Pis, this couldn't be done as they have no GPS capabilities. This was the same as with monitoring battery life, this depends on the APIs provided to the program on the platform, but could be implemented easily in the future. Using an authentication method, such as a digital certificate, would benefit the project, but ended up being out of the scope of the project. This would be a beneficial addition in the future to protect against a variety of different attacks.

9.2 Non-Functional Requirements

The project fulfilled all of the Non-Functional requirements, except **NFR4**. This is a requirement is difficult to prove, however, the underlying design would satisfy this. The reason for this being that each different platform has a separate implementation of the Bluetooth stack, so it would have meant implementing the project on each platform. This is infeasible due to the time constraints of the project and so the current implementation can be used as a reference design for other implementations on those platforms. However, the project can be run on any Linux device which uses DBUS and is capable of BLE.

9.3 Project Management

The project used a variety of tools which were at its disposal effectively. The project was carefully planned out and executed as closely as possible to the project timeline. A number of risks were for seen and documented initially. This meant, when they did occur, a response could be quickly executed as it had already been considered. Such planning allowed the project to be delivered on time.

The project utilised many software development tools at its disposal. Examples such as Git and GitHub were used to manage the versions of the project, allowing code to be rolled back and securely stored as a backup. Using these tools meant developing on different machines was much easier, a critical part of the development process in this project. Integrated with this was project monitoring, which use Kanban boards to keep track of tasks which needed completing.

9.4 Legal, Social, Ethical and Professional Issues

Overall, this project has done work to monitor the different issues which may threaten it. Investigation work has been done into its legal threats, such as *GDPR*, as well as the other professional issues which could have arisen, such as not sticking to the BCS code of conduct and not performing research following rules set out by the University of Warwick.

In the future, work should still be done to monitor all of these different issues and threats and make sure that the project adheres to legal frameworks. This is especially important if this product were to be taken to market and used commercially where some of these issues serious and not just theoretical.

9.5 Author's Evaluation

This project was undertaken to investigate ways in which the emergency services could respond in an emergency disaster and be more effective at finding and rescuing people. The current solutions to this problem, as well as associated work, is detailed in Chapter 2. The aims of the project, stated in Chapter 1, set out the direction for which the project was to go. These aims were to design and implement a system whereby emergency messages could be disseminated to those who can do something with the information. Other aims were to investigate other issues surrounding this problem, such as data privacy, performance at scale, and future work. This project has fulfilled these aims fully over the course of this report.

9.5.1 Design

This project aimed to conceptualise a system which could effectively disseminate emergency information from those in need of help, to those which can provide it. This report looked at the challenges going into this problem area and formalised them, so they could be critically assessed and so an effective system could be produced. Built from the choice of network protocol (which influenced the design). This was unavoidable as each different protocol had distinct attributes which separated them from each other, such as fundamental communication structures which could not be altered. However, as was said in Chapter 5, the design created was done so that it could be easily adapted to other systems. Formalising the problem enabled an abstract, protocol-agnostic approach than would have been available had it not been done.

The project designed a system which is modular and extensible, to allow for future work and contributions to the problem area. This was done with future developments in research in the area so that new solutions could be conceptualised and designed easily, using this fundamental work. It also proposes a routing algorithm, implemented in the modular client behaviours and server characteristics, which suits the problem area at hand. Epidemic routing takes the difficulties of the problem area, the fragmented network, and turns it into a feature of the system. It enables this to be a strength which allows messages to be routed to the emergency services quickly. Furthermore, it gives them vast flexibility in their approach to an emergency while providing them with information about survivors.

Finally, a strength of the design is that it can be taken from a concept to a working system. This is not an easy thing to do. The fact that it has been done shows that it is laid out clearly and thought through fully. By thinking about implementation

during the design, it allowed choices to be made which allowed easier implementation later.

9.5.2 Implementation

The project implementation had its ups and downs, but overall it was a success. Building on the earlier design work, choices had to be made about how the design was going to be implemented. The early choice was to go with C as the implementation language, but it was found that this would not be feasible and that Python would be better suited to this proof of concept system.

The project used a variety of tools and libraries to implement this design. Different libraries were used for both the client and server portions of the project as it was felt that they each had different attributes which would lend themselves to the different libraries.

Overall, the project implementation was a success. It built on the work established in the design and made it work in practice as well as in theory. This allowed the design to be tested in the real world and to have it exposed to situations that it may be in. By using the testing frameworks laid out before, this could be verified and the implementation could be said to work. This is also documented by the fact most requirements were fully met.

9.5.3 Challenges

This project had many different technical challenges to overcome to reach this stage. One such challenge was with Bluetooth. Conceptually it is quite a simple protocol, but there are several layers and intricacies which need to be understood for a system like this. Another challenge the project had to overcome was maintaining the distributed nature of the design. Building a distributed system on a platform which forces a very specific interaction model meant that a large body of work had to be devoted to thinking about how information can be passed between nodes in the client set.

Lastly, a major challenge was implementing secure communications between devices. This, as documented in Figure 6.1, was a complex process which took a number of iterations to solve. This could be said to be the most complex technical challenge overcome during the project implementation. Although not a novel contribution, it was a crucial part of the project showing that encrypted communications can be achieved in a system such as this and critical to this project for the future. It would be a significant part in any real-world application of this project.

9.5.4 Final Words

Overall, this project has been a significant undertaking involving numerous different complex, interconnecting parts. Knowledge was taken from varied different areas from within the field of Computer Science, as well as drawing on elements from Epidemiology. Bringing all of these elements together has created a system which helps to solve a significant problem area, and provides what I believe is a novel contribution to the field. Hopefully this project could be used to help create systems similar to this, to be used at scale, which can help real people in very real situations that can have a significant impact on their survival. This is a problem area which I believe, using today's advanced modern technology, can and should be approached more frequently.

This project approaches both the practical and theoretical challenges that the problem area presents and gives solutions to them all. It also critically assess the pitfalls of the project, formalises some issues, and attempts to discuss them and overcome them using solutions drawing from Computer Science and beyond.

Chapter 10

Conclusion

This chapter will look at the future work which could be done on this project, as well as a summary of the work that has been done during the project. This looks back on the project as a whole and brings everything together.

10.1 Future Work

For all the technical achievements of this project, there are different avenues which could be explored in the future surrounding this. These are areas which could have been included in this project but, for reasons such as time or scope, were not fully explored.

One such area is security. The project details, and implements a system whereby communications can be secured by encrypting messages using public-key encryption. This is important as it allows the user to send messages without adversaries being able to intercept and read the contents of the message. This project solves this part, but leaves open the opportunity for impersonation, or for legitimate conversation with an adversary. This could lead to a user connecting with an adversary and sharing their location information, as well as that of others who have trusted them with their information. A method whereby this could be overcome is by using a digital authentication scheme. This was suggested earlier in the project, but was not implemented in the project because of the time constraints imposed on it. By using digital authentication, clients and servers would be able to have more confidence that they know who they are talking to and that the devices they are communicating with is a legitimate user of the network.

This project only implements the design on one type of device. Using the Raspberry Pi and Linux has allowed this project to implement a significant working proof

of concept system which shows that the design of the system works. In future work, it would be beneficial to implement the system on a wider variety of platforms. This would require it to be implemented using different languages and using different libraries and APIs. This is a significant undertaking but would allow further testing of the design using a more realistic networks and could even help to refine and evolve the design.

Something that could not be done due to the time constraints of the project was to develop a simulation of the network. This could allow testing at a much greater scale than this project could ever do. To fully test this system, it has to be tested on a network representative of what may be found in an actual emergency situation. This could be done using some sort of simulation. This would, like using a greater variety of devices and platforms, allow for different feedback to help improve and optimise the design of different factors such as packet routing and how the client behaviours work, as well as investigating the feasibility of the system further.

10.2 Summary

In conclusion, this project is a novel take on solving the problem area defined at the start of this report. It deals with numerous aspects of the problem at hand, gives many formal solutions for problems and several solutions. It investigates the performance of the system and gives both theoretical and practical concepts of what a real-world system such as this could look like. It is built on widely available technology and is its basis is from both academia, as well as a commercial setting. The project also investigates what a future version of this project could look like with additional work and improvements. It illustrates that it is possible to have a system which can utilise Ad-Hoc networking to send location data during an emergency, and in doing this proposes a feasible system, based in theory and proven in commercial settings as well as through a significant working proof of concept system.

Bibliography

[1] K. Townsend, "Introduction to bluetooth low energy." [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/

[2] "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, 2016.

[3] D. of Homeland Security, "Disaster emergency communications." [Online]. Available: https://www.fema.gov/disaster-emergency-communications

[4] N. Kishore, D. Marqués, A. Mahmud, M. V. Kiang, I. Rodriguez, A. Fuller, P. Ebner, C. Sorensen, F. Racy, J. Lemery *et al.*, "Mortality in puerto rico after hurricane maria," *New England journal of medicine*, vol. 379, no. 2, pp. 162–170, 2018.

[5] K. Banipal, "Strategic approach to disaster management: lessons learned from hurricane katrina," *Disaster Prevention and Management: An International Journal*, vol. 15, no. 3, pp. 484–494, 2006.

[6] C.-M. Feng and T.-C. Wang, "Highway emergency rehabilitation scheduling in post-earthquake 72 hours," *Journal of the 5th Eastern Asia Society for Transportation Studies*, vol. 5, no. 3281, pp. 3276–3285, 2003.

[7] H. R. Council, "Promotion and protection of all human rights, civil, political, economic, social and cultural rights, including the right to development," *UN General Assembly*, vol. A/HRC/32/L.20, June 2018.

[8] *Contact Tracing: Bluetooth Specification. v1.1*, Apple Inc. and Google LLC, April 2020.

[9] B. Allyn and J. Beaubien, "Getting aid to bahamas is a logistical nightmare as support systems 'do not exist'." [Online]. Available: https://www.npr.org/2019/09/10/759431554/

getting-aid-to-bahamas-a-logistical-nightmare-as-support-systems-do-not-exist?t=1570629766473

[10] T. Frank, "Cell phone service must be restored quicker after hurricanes." [Online]. Available: https://www.scientificamerican.com/article/cell-phone-service-must-be-restored-quicker-after-hurricanes/

[11] L. Ferranti, S. D'Oro, L. Bonati, E. Demirors, F. Cuomo, and T. Melodia, "Hiro-net: Self-organized robotic mesh networking for internet sharing in disaster scenarios," in *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2019, pp. 1–9.

[12] X. Wu, M. Mazurowski, Z. Chen, and N. Meratnia, "Emergency message dissemination system for smartphones during natural disasters," in *2011 11th International Conference on ITS Telecommunications*. IEEE, 2011, pp. 258–263.

[13] A. A. Shahin and M. Younis, "Alert dissemination protocol using service discovery in wi-fi direct," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 7018–7023.

[14] Y. Duan, C. Borgiattino, C. Casetti, C. F. Chiasserini, P. Giaccone, M. Ricca, F. Malabocchia, and M. Turolla, "Wi-fi direct multi-group data dissemination for public safety," in *WTC 2014; World Telecommunications Congress 2014*. VDE, 2014, pp. 1–6.

[15] B. Zheng, P. Wang, F. Liu, and C. Wang, "Cooperative data delivery in sparse cellular-vanet networks," in *2016 6th International Conference on Digital Home (ICDH)*. IEEE, 2016, pp. 128–132.

[16] H. Kaur *et al.*, "Analysis of vanet geographic routing protocols on real city map," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2017, pp. 895–899.

[17] S. Hu, Y. Jia, and C. She, "Performance analysis of vanet routing protocols and implementation of a vanet terminal," in *2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC)*. IEEE, 2017, pp. 1248–1252.

[18] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*. IEEE, 1999, pp. 90–100.

[19] tile, "tile." [Online]. Available: https://www.thetileapp.com/en-gb/

[20] trackr, "trackr." [Online]. Available: https://www.thetrackr.com/

[21] A. H. Omre and S. Keeping, "Bluetooth low energy: wireless connectivity for medical monitoring," *Journal of diabetes science and technology*, vol. 4, no. 2, pp. 457–463, 2010.

[22] S. Silva, H. Martins, A. Valente, and S. Soares, "A bluetooth approach to diabetes sensing on ambient assisted living systems," *Procedia Computer Science*, vol. 14, pp. 181–188, 2012.

[23] R. Callon, "Rfc 1925: The twelve networking truths," *1st April*, 1996.

[24] M. B. Shoemake, "Wi-fi (ieee 802.11 b) and bluetooth," *Coexistence Issues and Solutions for the*, vol. 2, 2001.

[25] S. Bluetooth, "Bluetooth core specification 2.0+ edr, the official bluetooth membership website, 2004 http://www. bluetooth. org/technical," *Specifications/adopted. htm.*

[26] R. Heydon and N. Hunn, "Bluetooth low energy," *CSR Presentation, Bluetooth SIG https://www. bluetooth. org/DocMan/handlers/DownloadDoc. ashx*, 2012.

[27] J. Decuir *et al.*, "Bluetooth 4.0: low energy."

[28] S. Bluetooth, "Bluetooth core specification version 4.0," *Specification of the Bluetooth System*, vol. 1, p. 7, 2010.

[29] D.-J. Deng, K.-C. Chen, and R.-S. Cheng, "Ieee 802.11 ax: Next generation wireless local area networks," in *10Th international conference on heterogeneous networking for quality, reliability, security and robustness.* IEEE, 2014, pp. 77–82.

[30] I. S. Association *et al.*, "Ieee std 802.11-2016, ieee standard for local and metropolitan area networks—part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 2016."

[31] W. R. Stevens, *TCP/IP illustrated vol. I: the protocols.* Pearson Education India, 1993.

[32] W.-F. Alliance, "Wi-fi peer-to-peer (p2p) technical specification," *www. wi-fi. org/Wi-Fi_Direct. php*, 2010.

[33] Z. Alliance, "Smart home." [Online]. Available: https://zigbeealliance.org/market-uses/smart-home/

[34] K. Gill, S. Yang, F. Yao, and X. Lu, "A zigbee-based home automation system," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 422–430, 2009.

[35] N. Patel, "It's time to break up facebook." [Online]. Available: https://www.theverge.com/2018/9/4/17816572/tim-wu-facebook-regulation-interview-curse-of-bigness-antitrust

[36] "Looters take advantage of new orleans mess." [Online]. Available: http://www.nbcnews.com/id/9131493/ns/us_news-katrina_the_long_road_back/t/looters-take-advantage-new-orleans-mess/#.XZus_CV7lTY

[37] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017, pp. 1–6.

[38] M. Agrawal and P. Mishra, "A comparative survey on symmetric key encryption techniques," *International Journal on Computer Science and Engineering*, vol. 4, no. 5, p. 877, 2012.

[39] S. Heron, "Advanced encryption standard (aes)," *Network Security*, vol. 2009, no. 12, pp. 8–12, 2009.

[40] J. Savard, "Safer+." [Online]. Available: http://www.quadibloc.com/crypto/co040407.htm

[41] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[42] N. Li, "Research on diffie-hellman key exchange protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 4. IEEE, 2010, pp. V4–634.

[43] F. J. Aufa, A. Affandi *et al.*, "Security system analysis in combination method: Rsa encryption and digital signature algorithm," in *2018 4th International Conference on Science and Technology (ICST)*. IEEE, 2018, pp. 1–5.

[44] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, p. 39–53, Apr. 2004. [Online]. Available: https://doi.org/10.1145/997150.997156

[45] M. Chhabra and B. Gupta, "An efficient scheme to prevent ddos flooding attacks in mobile ad-hoc network (manet)," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 10, pp. 2033–2039, 2014.

[46] H.-X. Tan and W. K. Seah, "Framework for statistical filtering against ddos attacks in manets," in *Second International Conference on Embedded Software and Systems (ICESS'05)*. IEEE, 2005, pp. 8–pp.

[47] N. H. Phong and M.-K. Kim, "Enhancing reliability on wireless sensor network by aodv-er routing protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 6. IEEE, 2010, pp. V6–32.

[48] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of INFOCOM'97*, vol. 3. IEEE, 1997, pp. 1405–1413.

[49] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, "Geographic routing in city scenarios," *ACM SIGMOBILE mobile computing and communications review*, vol. 9, no. 1, pp. 69–72, 2005.

[50] C. Chi, D. Huang, D. Lee, and X. Sun, "Lazy flooding: A new technique for information dissemination in distributed network systems," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, p. 80–92, Feb. 2007. [Online]. Available: https://doi.org/10.1109/TNET.2006.890125

[51] A. Goel, A. G. Kannan, I. Katz, and R. Bartoš, "Improving efficiency of a flooding-based routing protocol for underwater networks," in *Proceedings of the Third ACM International Workshop on Underwater Networks*, ser. WuWNeT '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 91–94. [Online]. Available: https://doi.org/10.1145/1410107.1410125

[52] T. L. Lin, Y. S. Chen, and H. Y. Chang, "Performance evaluations of an ant colony optimization routing algorithm for wireless sensor networks," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2014, pp. 690–693.

[53] A. V. Leonov, "Modeling of bio-inspired algorithms anthocnet and beeadhoc for flying ad hoc networks (fanets)," in *2016 13th International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, vol. 2. IEEE, 2016, pp. 90–99.

[54] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: Positive and negative social effects," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 387–401, 2012.

[55] G. K. Wong, Y. Chang, X. Jia, K. H. Wong, and W.-Y. Hui, "Performance evaluation of social relation opportunistic routing in dynamic social networks," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 874–878.

[56] Z. Zhu, S. Liu, S. Du, X. Lin, and H. Zhu, "Relative interpersonal-influence-aware routing in buffer constrained delay-tolerant networks," in *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2013, pp. 4446–4451.

[57] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," 2000.

[58] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Computer Networks*, vol. 51, no. 10, pp. 2867–2891, 2007.

[59] R. J. Glass, L. M. Glass, W. E. Beyeler, and H. J. Min, "Targeted social distancing designs for pandemic influenza," *Emerging infectious diseases*, vol. 12, no. 11, p. 1671, 2006.

[60] M. P. Broderick, C. J. Hansen, and K. L. Russell, "Exploration of the effectiveness of social distancing on respiratory pathogen transmission implicates environmental contributions," *The Journal of infectious diseases*, vol. 198, no. 10, pp. 1420–1426, 2008.

[61] P. Caley, D. J. Philp, and K. McCracken, "Quantifying social distancing arising from pandemic influenza," *Journal of the Royal Society Interface*, vol. 5, no. 23, pp. 631–639, 2008.

[62] C. Tikkinen-Piri, A. Rohunen, and J. Markkula, "Eu general data protection regulation: Changes and implications for personal data collecting companies," *Computer Law & Security Review*, vol. 34, no. 1, pp. 134–153, 2018.

[63] "British computer society code of conduct," British Computing Society, BCS The Chartered Institute for IT, First Floor Block D, North Star House, North Star Avenue, Swindon, SN2 1FA. [Online]. Available: https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/

[64] "Research code of practice," Univeristy of Warwick, University of Warwick, Warwickshire, CV4 7AL, United Kingdom. [Online]. Available: https://www.warwick.ac.uk/services/ris/research_integrity/code_of_practice_and_policies/research_code_of_practice/

[65] S. Bradner, "Key words for use in rfcs to indicate requirement levels," Internet Requests for Comments, Network Working Group, BCP 14, March 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2119.txt

[66] A. Holst, "Global mobile os market share in sales to end users from 1st quarter 2009 to 2nd quarter 2018." [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[67] D. N. Alparslan and K. Sohraby, "A generalized random mobility model for wireless ad hoc networks and its analysis: One-dimensional case," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 602–615, 2007.

[68] "Fix 4.2 protocol specification guide," London Stock Exchange Group.

[69] V. Mishra. [Online]. Available: https://www.counterpointresearch.com/iphone-11-second-best-selling-phone-2019-less-four-months/

[70] "Bluez: Official linux bluetooth protocol stack." [Online]. Available: www.bluez.org

[71] R. Pressman, *Software Engineering: A Practitioner's Approach 7th Edition*. McGraw-Hill, 2010.

[72] I. Sommerville, *Software Engineering Ninth Edition*. Addison-Wesley, 2011.

[73] "Welcome to python.org." [Online]. Available: https://www.python.org/

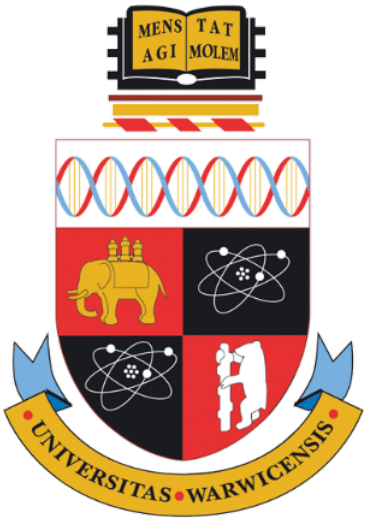
[74] "dbus." [Online]. Available: https://www.freedesktop.org/wiki/Software/dbus/

[75] "Ianharvey/bluepy: Python interface to bluetooth le on linux." [Online]. Available: https://github.com/IanHarvey/bluepy

[76] "Pycryptodone." [Online]. Available: https://pycryptodome.readthedocs.io/en/latest/

Appendices

Progress Report



Using Ad Hoc Networking in Emergency Situations

Spreading information in an emergency situation to help both people in danger and the emergency services

Progress Report

Freddie Brown
u1716717

Supervisor: Dr. Matthew Leeke
Department of Computer Science
University of Warwick
2019-20

Abstract

During a disaster, it is often very difficult to communicate using our devices. Whether this be because infrastructure has been damaged, or there is too much traffic over networks. This can mean people feel isolated and makes it difficult for the emergency services to know the situation on the ground. This project aims to help ease this by creating emergency Ad-Hoc networks during times of emergency. Using hardware already found in the devices people carry around with them day-to-day, information can be passed from person to person and can be sent to the emergency services. It could help improve the efforts of the emergency services by giving them a better picture of an area, helping to use their resources more effectively. This lets them to save more lives.

Keywords: Bluetooth, Ad Hoc, MANET, Emergency, Disaster

Abbreviations

MANET - Mobile Ad Hoc Network
VM - Virtual Machine
BCS - British Computing Society
WSN - Wireless Sensor Network
AODV - Ad-Hoc On-Demand Distance Vector
DTN - Delay Tolerant Network
DAG - Directed Acyclic Graph

Contents

Chapter 1

Introduction

This project is about investigating the application of MANETs for use in emergency situations. This is a technology which lends itself to the low infrastructure needs of a disaster scenario. They can form networks dynamically with low overheads. As mentioned in the specification, these are very interesting in fluid situations where nodes move in and out of a network [1]. Although this is the main area of exploration for the project, there are other areas which are just as important which also need to be considered. A project such as this brings up issues such as data privacy, packet routing and taking actions to prolong the battery life of a device. These are all areas which will have a huge impact on the success of a project such as this.

1.1 Motivations

The modern world is highly connected through our telecommunications systems. These are used to connect people together. One important usage of this, is to quickly contact the emergency services. As discussed in the specification, during a disaster, such as the one seen in the 2017 hurricane in Puerto Rico, not having access to a form of communication can cause great harm to peoples lives. A study estimates that the lack of communications services in the aftermath of the hurricane contributed to an increase in mortality of 62% [2]. Remote areas were hurt most, being without cellular data services for up to 41 days. It can make it difficult for rescuers to know what is going on and how to intervene such that they can make a difference. This commonly leads to high cost and disruption. Also mentioned in the specification was Hurricane Katrina, in this scenario, high winds destroyed antennas highly damaged or destroyed [3], many of which belonged to belonged to cellular providers.

This illustrates the need to develop systems which help to ensure connectivity, even in challenging times. The United Nations voted, in June 2016, that internet access is a human right that should be protected [4]. This puts our current

systems under the microscope and shows how important these services are to people and how access to them can save lives.

1.2 Project Aims

This project is aiming to develop a system which could be integrated into future mobile devices, which allows for the dissemination of emergency information in the event of a disaster. The project will look at the different ways which this can be implemented, what the challenges are that this problem presents as well as peripheral issues which were mentioned earlier.

1.3 Stakeholders

As mentioned in the specification, the stakeholders in this project are those which will benefit from a system such as this being implemented. Those who live in areas which are frequently affected by large scale emergencies will be the ones who will use this more often. Furthermore, the emergency services will benefit greatly from a project like this so they can maintain a clearer picture of what the situation on the ground its, and where their efforts will be most effectively utilised.

Chapter 2

Research

2.1 Related Work

Emergency networks are difficult to implement because of, in many cases, the damage to infrastructure that is caused in an emergency or the strain that puts on the existing infrastructure. Disasters can be difficult to deal with after they happen due to this degradation in communications infrastructure [5]. This can then hamper efforts to rescue survivors as they will struggle to communicate with those who need saving. Having a system in place, specially for this scenario, could save the lives of those that are affected by disaster [6].

An interesting attempt to solve this problem is HIRO-NET [7]. This system has 2 tiers: a local meshing using Bluetooth Low Energy (BLE) to connect devices together using a client-server model, and a way to connect together these local networks. In local meshing, a node starts off as a client and will try and find a server to connect to. After a while, it will timeout and will become a server for nodes to connect to. Doing this means that, when nodes join to network, over time it becomes easier to connections to form. Piconets are joined together to form Scatternets and packets are transported within them. If a Piconet has an active internet connection, the server will declare this. This means that other servers will route packets to this Piconet to be sent out over the active connection. This system is complex but allows for a more regular level of service for users.

A more simple approach then HIRO-NET has a greater focus on providing information for the emergency services [8, 9]. It takes some information, such as geographical information, and transports it to a central node to allow this information to be aggregated. The suggested systems use both WiFi and Bluetooth and how they can both play an important role together. These conclude that, as WiFi has a wider range and is more available, it plays a more important roles than Bluetooth. It also states that using WiFi Direct can provide a large speed boost when spreading information [9]. It means that a connection doesn't need

to happen for a device to spread critical information about itself, which has obvious benefits for the types of situations that this project is investigating.

WiFi is an incredibly important technology as it supports long range transmission. For some standards, transmission can be up to, or close to, 1,000m in certain situations [10, 11]. Bluetooth tends to be used more for short range transmissions [12]. It relies on being in quite a well populated area as its range is limited. Each have their uses. Bluetooth is also considered to be more power efficient [13] than WiFi, but this is disputed for certain use cases [14]. This means it is sensible to consider both technologies for this project. This project is focusing on close range communications in environments where users may not have access to a stable power source. For this reason, Bluetooth may be a better choice than WiFi. WSN systems which use Bluetooth could provide an ideal model to analyse, such as [15]. This example discusses a high performance sensor network using highly optimised hardware. It presents an interesting model for nodes to work efficiently, such as only have 1 outgoing route for traffic. This could be applied in this project by only allowing a node to send data to 1 other node at any point.

A more recent, and very applicable, example of Ad Hoc networks is in the automotive industry. As cars have become more advanced, they have included more powerful computers on-board. Research has been done which looks into how they can be used to form networks made up of these vehicles. A number of studies [16–18] have investigated different aspects of this type of network. As discussed in [16], VANETs can communicate with each other as they drive past each other, communicating information between themselves. This information could be about similar things to the data in this project, such as location data of interesting events, like traffic jams. Although, as also discussed in [16], this isn't a stable network as they may be moving too fast to be able to communicate effectively. It offers a number of solutions, such as using store-and-forward routing algorithms or using Cellular data. VANETs also have a number of other issues similar to this project,security and routing. [17,18] discuss a number of different possibilities for routing, such as novel ideas like geographical routing [18] and more established routing protocols such as AODV [17, 19] and how well they perform in this paradigm.

2.2 Security

Security is an area of this project which needs serious consideration. The legal and social ramifications [20] of ignoring it can be extremely high. As this project is dealing with sensitive location data, it is crucial that a solution is found that will ensure integrity and confidentiality. If it isn't, data could be harvested and used to the detriment of those who survive a disaster. An example of post–disaster problems is looting, which was seen after Hurricane Katrina in 2005 [21].

Firstly, a method of sharing a key with the nodes is needed so they can encrypt information before it is transmitted. A way to accomplish this is to use Diffie–Hellman key exchange [22] so symmetric key encryption may be used. As suggested by [23], this protocol has issues with security and is susceptible to man in the middle attacks.

An alternative to using symmetric key encryption is using public key encryption, such as RSA [24]. In this, each member which wants to receive data will declare their public key, which is available to everybody. Once the message has been encrypted with the public key, only the private key can decrypt the message. As the private key should only be known by 1 users, this means only the correct recipient may view the encrypted data. This will provide messages with the confidentiality that is needed. To provide integrity to the messages, digital certificates could be used [22].

2.3 Routing

As Bluetooth uses a client-server model, routing is usually device-to-device. With this, it forms multiple small groups of devices (Piconets). These can be joined together and packets can be routed between them. Furthermore, other topologies which aren't as stable also exist, and so packet routing needs to be investigated in these forms too. By investigating all of these, a variety of circumstances can be covered.

As MANETs are already in use in a variety of situations, there are a couple of routing protocols that are already used. One popular one is AODV [19, 25], a combination of dynamic source routing and distance vector routing. This algorithm uses a series of packets to build routes, which are stored in nodes throughout the network. These are built using Route Request(RREQ) packets being broadcast across the network, flooding it. When an RREQ packet reaches the desired node, a Route Reply (RREP) packet is broadcast back, along the path which was used to reach the node in the first place. When an RREP packet is received by a packet which transmitted an RREQ packet, it creates a pointer in its routing table. This means it knows where to route any packets for that node in the future.

Another algorithm which is used for routing within MANETs is TORA [26], a type of routing algorithm that uses a DAG to determine which nodes to route packets through to the destination of the packet. If a node wants to route packets to a sink, F, it will broadcast QRY packets, which are relayed until it reaches a node adjacent to F. When it reaches these nodes, they broadcast a UPD packet which contains the id and distance from the sink node. These are propagated back towards the source node. This allows a graph to be constructed and allows routing to be done based on distance. The advantage of this method is that it has fewer control packets compared to other, similar methods as it only reacts

when changes are discovered in the topology. Furthermore, partitions in the network are easily detected, whereas in AODV the source will send lots of RR packets which will flood the network.

An interesting are of packet routing, is swarm-based packet routing. It has been shown to perform better than a number of other, more established routing algorithms [27]. Swarm-based algorithms tend to find the best path in a network, but also include a probability that a different route will be chosen. As a MANET is an inherently unstable network, this is good as it allows new, better paths to be found. It also includes a time parameter to force information to be recalculated if it hasn't be used in a while. This feature also forces the exploration of new, quicker paths in the network. This type of routing is typical of a number of swarm-based routing algorithms, such as ant-colony routing [28].

An alternative type of biological routing is based on bees [29]. This protocol has a scouting stage where each node in the network is ścouted. Scout packets will be sent out to each neighbour and will travel through the whole network. They are passed on each time they visit a node until they reach a sink node. A back bee packet is then sent from this node towards the source. This process is done to evaluate the efficiency of routes between nodes. This is similar to the way that more established algorithms, such as AODV, work, except the scouting it much wider ranging, whereas in AODV, the equivalent stage is only done for 1 node at a time. This is a promising solution for MANETS, but the packet overhead might be too much for some time-sensitive cases.

The algorithms investigated already all look at algorithms which are only really useful in networks which are more stable. Connections are broken infrequently meaning the optimisations which have been discussed, can actually make a difference. Another form of routing caters more for networks which are less stable. Optimistic routing algorithms are common in DTNs and are good spreading information quickly. There are many different examples of this. One example of this is inspired by epidemiology to model how packets can flow through a network [30, 31]. It works by a node passing any information it has onto available neighbours. This is similar to how a disease might spread within a general population. Each node which receives these messages will store them and then carry them forward. If these nodes are physically moving, it allows packets to achieve large multi-hop movements with far fewer exchanges.

An alternative to disease-based routing, is to consider another type of routing also used within in DTNs, social-based routing [32]. This kind of routing considers the behaviours of nodes within a network. This will help improve the choices that a node makes as it considers the likelihood that the other node will forward the packets that it is going to transmit. It uses a group of set social characteristics to determine whether its behaviour is 'good' and cooperative or if its 'bad' and only caring about its own resources e.g only forwarding on messages from a select group of users. Other studies, such as [33, 34], show how

powerful social-based routing protocols can be. They help to reduce the number of packets which are sent, and increase the probability that these packets actually are delivered. As can be seen in [34], the chance of delivery is lower than epidemic routing, but the number of packets which are sent is far lower and the latency is far greater. This could help to reduce traffic and unnecessary traffic in the network.

Chapter 3

Ethical, Social, Legal and Professional Issues

3.1 Ethical Issues

Ethical issues can come up when a project has a number of different objectives, each with negative effects which aren't immediately present. The benefits this objective could bring, if fulfilled, might out weight the immediate negatives for an objective and so due diligence on other impacts may be forgone. An example of this is in a project which collects lots of personal data. An objective of the product might be to increase cash flow of the company. Using this data to provide products such as targeted adverts may seen like a good idea at first, but might create some reputations damage in the long run. These kinds of issues may lead to data leaking or some people may find this an invasion of privacy. These damage had impacted Facebook recently. [35]. In this project, location data is used among a number of nodes. This data should be protected so that it can't be used for anything unintended by nodes which have access to it. This could be through encryption or forms of providing data privacy.

3.2 Social Issues

Social issues are those that have an affect on the lives of groups of people. As discussed in the project specification, these problems could affect how people interact with each other or could limit their access to certain goods and services. It may also relate to certain groups having access to key services over other, marginalised groups. It is difficult to see any of these issues relating to this project but this should be continually reviewed as the project moves onwards.

3.3 Legal Issues

A large part of this project is about sending sensitive data about vulnerable people, issues around data privacy need to be considered seriously. In this project, data is being sent to other devices nearby, so trust issues which may arise. To counter this, encryption could be used to provide data privacy. By doing this, it limits access only to users who should have access to the data. Otherwise, if this isn't protected, malevolent users may use it to cause harm to others, such as terrorists looking to find people to harm.

3.4 Professional Issues

As mentioned in the project specification, the project will adhere to the BCS Code of Conduct [36]. The aim of the project is to produce a research project which can be used in the future. This means adhering to these rules is very important. Furthermore, the project will pay close attention to the Research Code of Practice at the University of Warwick [37] and the departmental rules on ethical consent for any data the project may collect from participants in any potential testing [38].

Chapter 4

Project Requirements

4.1 Functional

FR1 - The system must use a widely used MAC layer protocol such as Bluetooth or WiFi.

FR2 - The system must use data privacy techniques, such as encryption, to prevent data falling into the wrong hands.

FR3 - The system could monitor battery life and make choices based on its predicted life expectancy

FR4 - The system could use a system of authentication to verify nodes, such as using a trust-based system or digital certificates

FR5 - The system must use the concept of sources and sinks, where sinks collect data and don't transmit information packets, and sources send and receive information from other sources.

FR6 - System must have a way to identify devices within the network. This ID should be used within packets sent by each device about themselves.

FR7 - Project could use a location discovery service such as GPS to get location data

4.2 Non-Functional

NFR1 - The system must be fully documented and maintainable. This means that the project is easily extensible and can easily be implemented on other platforms.

NFR2 - The system must be easy for a user to connect to and use. This is vital for the project as time is an important factor in an emergency situation and so the less time a user has to worry about how it works, the quicker they can use it to get help

NFR3 - The system must be created so it can be applied to a large population of devices. This project works optimally if there are a large number of devices to connect so design choices should consider the need for this project to work at a large scale.

NFR4 - The system must be able to be used by different types of devices such as Phones, Tablets and PCs. Much like **NFR3**, this project should be designed so it can be easily transported onto other devices so that they can also participate.

NFR5 - The system must use secure communications so bad actors cannot steal and weaponise the information.

NFR6 - System should require as little user interaction as possible to increase efficiency.

4.3 Constraints

As was discussed in the project specification, the project is constrained by the number of devices which it can be tested with. It is very expensive to test on a number of devices so, for that reason, the project will use a cheaper alternative, the Raspberry Pi. This will demonstrate the applications of this project, but can't replicate a proper use case of this project as it will lack the scale of devices which would be needed.

Another constraint is the types of devices, that can be used. The research will focus on applications in a heterogenous network of devices but it is likely this will only be demonstrated on a homogenous network as more popular devices (e.g iOS and Android [39]), which this type of system would be employed on in the real world, require separate development. This is outside of the scope of this project as it would require more time than is available to it. This would be a stretch goal for the project or could be considered a further extension for the future.

Chapter 5

Project Management

5.1 Project Timeline

As can be seen in fig.5.1, it shows the predicted project timeline in a Gantt chart as well as displaying the dates between which each task will take place. A large block of time has been left for the implementation of the project, about 2 months. This gives time for the project code to be written, tested and iterated on. This is to catch bugs and produce a working system. When this is complete, testing of the system will be done to look at the performance of the project so the project can be analysed. With all of this, a presentation will need to be produced. This will detail the different stages of the project, how the system works, as well as retrospectively looking at the project and discussing what went well and what could be improved in the future. With all of this produced, the final report will be written which will discuss all of this in further detail and will assess whether or not the project has been a success.

	Start Date	End Date	Timeline	Status
Project	30-09-2019	23-04-2020		
First Research	30-09-2019	07-10-2019		Done
Specification	30-09-2019	10-10-2019		Done
Routing Algorithms Research	11-10-2019	19-11-2019		Done
MAC Protocol Research	11-10-2019	19-11-2019		Done
Data Security Research	11-10-2019	19-11-2019		Done
Progress Report	11-10-2019	25-11-2019		Doing
Implementation	26-11-2019	12-02-2020		To Do
Empirical Testing	13-02-2020	05-03-2020		To Do
Presentation	01-03-2020	13-03-2020		To Do
Final Report	11-10-2019	23-04-2020		To Do
	30-09-2019	23-04-2020		
		Burndown		

Figure 5.1: Project Timeline

5.2 Project Tools

The project will use a variety of tools to accomplish its goals. The project will be written in **C/C++**. This is because its a low level language and it allows more fine grained control over aspects of the project. Along with this, **blueZ** will be used to interface with Bluetooth. This is a widely used library for projects such as this, with a number of resources to draw from [40].

In terms of hardware, as discussed in the project specification, the project is going to be written for Linux-based devices such as the Raspberry Pi which have Bluetooth on-board. Bluetooth is a common technology used by a number of devices, so is a useful medium to use for a project like this. Using a widely used technology increases the number of technologies that this project could apply to.

Other tools are needed to monitor the progress of the project. This project has been using **Trello** to keep track of tasks which need to be done and have been done. This is being used because its clear to use and allows tasks to be grouped clearly. Alongside this, **GitHub** has been used to backup code and keep track of project history, in case an older version of the project is needed. This is a widely used product in industry, so is a good choice for this project.

5.3 Risk Management

As mentioned in the project specification, this project doesn't have too many major risks that could have an effect on its performance. One risk, which was discussed above, is losing all physical machines which hold the project. This can be mitigated by storing any code and reports on Github, as well as maintaining a local copy on a hard drive. This provides extra layers of assurance that the project won't be lost.

Another risk is that persons who work on the project fall ill or are unable to do work. This can be mitigated by keeping in contact with DCS and discussing any factors that could lead to a delay in the projects completion because of this reason.

Furthermore, there is a risk that part of the project might overrun its planned length of time. This could be because hardware access has been delayed or there are extra technical difficulties that weren't foreseen during the planning of the project. Generous amounts of time have been allowed for each stage, as well as considerations within each stage. For example, in the implementation stage, there are different stages to development. The first stage is to complete basic functionality of the project before layering more features on top. This way of working means there will always be a working version of the project which meets the requirements.

Chapter 6

Progress

6.1 Defining Project Scope

The project specification focused on 2 directions for the project, it was quite broad. Since then, the scope of the project has been tightened so that the project is more focused on relevant goals.

One option for the project, which was discussed, was a project similar to HIRONET [7], a large network which used local bluetooth meshing to pass IP packets to devices, within a piconet/scatternet, which had atleast one active internet connection. With a system like this, routing protocols such as swarm-based routing algorithms [27–29] can also be used, enabling further improvements in network efficiency. Some of the benefits of this kind of system are:

- Creates a flexible network
- It is more useful to each survivor as they will have internet access; large variety of services to access
- More efficient data efficient; Doesn't constantly flood the network with packets
- Easy to implement 2-way communication between survivors and others; Can route packets to and from nodes as network is more stable

Although this proposal has a number of huge benefits, it also has a number of drawbacks:

- It is difficult to implement and probably beyond the reach of this timeframe. Too much complexity
- Difficult to use in a sparsely populated area; Less useful in many environments

- May need to use multiple different technologies to accomplish this on top of Bluetooth, such as Wifi. Adds to the already large complexity
- Server nodes need to do a lot of work, not good when aiming to conserve battery life

An alternative solution is to implement a solution akin to more common DTNs [8, 30, 32, 34], similar to a number of fault-tolerant networks. In this solution, as has been described before, messages about the status of survivors are passed between devices to a sink, usually the emergency services. These messages will contain location data about survivors which can be collected by the emergency services. The packet routing method will be one common to DTNs, either social-based or epidemiology-based. Some of the benefits of this are:

- Simple. Makes it easier to implement on a number of different devices so more nodes in the network
- Lightweight information. Reduces the packet size so data transmission is quicker
- Able to work effectively in both dense and sparse environments
- Network focused on emergency services, more beneficial to them
- Very tolerant to nodes coming in and out of the network
- Works well in topologies with lots of moving nodes

Like the first proposal, this also has some issues which have to be considered:

- Less efficient. Price of simplicity is that routing is more primitive and so less efficient
- 2-way communication is difficult because there is no constant network of devices with addressing
- More computation is involved: connections formed with other nodes more often, more nodes routing packets

After considering both of these ideas, the direction of the project will be the second proposal, utilising epidemic routing to build a fault-tolerant, lightweight network for spreading information among local, moving nodes. It has a more realistic implementation timetable and has a clearer and better defined use case.

6.2 Timetable

In the project specification, a timetable was defined which displayed the planned activities which were to be accomplished if the project is to be successful, as well as the dates which they need to be completed by. This was done to give the project structure and to help plan for deadlines. Since this part of the project

was submitted, a few changes have been made, as can be seen in fig. 5.1. In this updated timetable, the time for improvements and testing have been rolled into the implementation period. This was done to be in line with principles of test driven development. Also, parts of the timetable have been marked as done. These parts of the project were to be completed in between the project specification and the progress report. These have been completed mostly, as well as some extra research into implementation and working out how the project will work.

6.3 Technical

As well as defining the project scope, work has been done on investigating the technical parts of the project, such as in defining what should be sent within the packets in the network. A principle of the packet structure for the project is that it needs to be small. This is so that as much data as possible can be fit inside as few Bluetooth packets [41] as possible. Only essential information should be included in the packet. Inspirations for a format are the FIX messaging protocol [42], a widely used protocol for transporting financial data, and JSON [43], a widely used data format in industry. The advantage of using a protocol like to FIX is that it has a number of security benefits. It uses a group of specified tags which describe very specific information. This means that a program will only analyse the data for specific tags which it cares about, which reduces the chance of unwanted information being processed. If extra functionality is added, nodes can ignore the extra data included, which allows for backwards compatibility in the future. On the otherside, JSON would be good as it is concise and very flexible. There is lots of support for it as it is very established. Both of these offer a lot of inspiration for a format of exchange to use.

The data to be included within packets has also been investigated. Obviously, including location data needs to be a priority. This can be done by using GPS, a common way of collecting location data. This is available on most devices today to power a number of location based applications. As well as this, an ID should be included in the packet. This will allow the sink to aggregate the data based on the device from which it originated. As well as this, including a timestamp saying when the packet was generated will allow the sink node to determine how old this data is. This way, it allows the emergency services to determine if the location data is accurate and allows nodes to prioritise which packets to forward first.

As well as considering packets, work has been done working towards understanding how the different libraries work. BlueZ [44] is a very useful, and widely used, library for C/C++ but has very little documentation. Much work has been done working towards understanding how it works. As it also contains a number of command line tools, BlueZ has a lot of open source code which uses a lot of the features which this project will use.

Chapter 7

Testing

As discussed in the specification, testing in this project will be used to verify the functionality of the project while changes are made as well as being able to verify that requirements have been fulfilled. The project will use different technologies to accomplish this. A unit testing framework will be used, such as **CUnit**, to test and verify the project. Also, other technologies will be utilised, such as bash scripting. This allows for tests to be written easily for a variety of projects and makes the testing process more flexible. For integration testing, **TravisCI** will be used. It is a robust service which is widely used and works very well with GitHub.

7.1 Unit Testing

A test will be written for each part of the project that is written to verify that it functions as it is needed to. Agile development methodologies will be adhered to by writing failing tests, which are then made to be passing by writing the code for which passes the test. Unit testing and its benefits are further discussed in [45, 46]. In these, they discuss the benefits of Test Driven Development and how unit testing works and is beneficial to a project. Unit testing is very useful and allows the functionality of the project to be verified quickly and easily.

7.2 Integration Testing

TravisCI allows larger tests to be written to incorporate more of the project. This is run on a clean VM which means there is nothing external to the project which could influence its testing. It enables more rigorous testing and fewer outside influences. This approach to testing the system is discussed in [45].

7.3 Success Management

Success for this project will defined by whether a network of Raspberry Pis, running the software written for this project, can pass packets about their location, between each other to a central repository node representing the emergency services. On top of this, there will be other measures of success for sub-targets, such as data privacy. These targets will be included in the requirements section of this report.

Chapter 8

Conclusion

Overall, the project has started smoothly. This part of the project plan has always been earmarked for research into the different areas which the project will touch on. At this stage, a good plan has been developed which will provide a structure for the project going forwards. Obviously, parts of the project will change with implementation, but this is acknowledged in this report. Much of the work which has been done so far has been to define a framework within which the implementation can be done. Being critical, this has been done to a good degree, but some extra work will have to be done during implementation. Not all libraries which will be used have been nailed down and, as a result, how an individual ID will be generated is not 100% certain yet. Over December, January and much of February, implementation will occur. The project is in a good place to start implementation. Some details are fluid still, but overall much has been nailed down and the majority of research has been completed.

Bibliography

[1] J.-Z. Sun, "Mobile ad hoc networking: an essential technology for pervasive computing," in *2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No. 01EX479)*, vol. 3. IEEE, 2001, pp. 316–321.

[2] N. Kishore, D. Marqués, A. Mahmud, M. V. Kiang, I. Rodriguez, A. Fuller, P. Ebner, C. Sorensen, F. Racy, J. Lemery *et al.*, "Mortality in puerto rico after hurricane maria," *New England journal of medicine*, vol. 379, no. 2, pp. 162–170, 2018.

[3] K. Banipal, "Strategic approach to disaster management: lessons learned from hurricane katrina," *Disaster Prevention and Management: An International Journal*, vol. 15, no. 3, pp. 484–494, 2006.

[4] H. R. Council, "Promotion and protection of all human rights, civil, political, economic, social and cultural rights, including the right to development," *UN General Assembly*, vol. A/HRC/32/L.20, June 2018.

[5] B. Allyn and J. Beaubien, "Getting aid to bahamas is a logistical nightmare as support systems 'do not exist'." [Online]. Available: https://www.npr.org/2019/09/10/759431554/getting-aid-to-bahamas-a-logistical-nightmare-as-support-systems-do-not-exist?t=1570629766473

[6] T. Frank, "Cell phone service must be restored quicker after hurricanes." [Online]. Available: https://www.scientificamerican.com/article/cell-phone-service-must-be-restored-quicker-after-hurricanes/

[7] L. Ferranti, S. D'Oro, L. Bonati, E. Demirors, F. Cuomo, and T. Melodia, "Hiro-net: Self-organized robotic mesh networking for internet sharing in disaster scenarios," in *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2019, pp. 1–9.

[8] X. Wu, M. Mazurowski, Z. Chen, and N. Meratnia, "Emergency message dissemination system for smartphones during natural disasters," in *2011 11th International Conference on ITS Telecommunications*. IEEE, 2011, pp. 258–263.

[9] A. A. Shahin and M. Younis, "Alert dissemination protocol using service discovery in wi-fi direct," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 7018–7023.

[10] W. Sun, M. Choi, and S. Choi, "Ieee 802.11 ah: A long range 802.11 wlan at sub 1 ghz," *Journal of ICT Standardization*, vol. 1, no. 1, pp. 83–108, 2013.

[11] A. M. Abdelgader and W. Lenan, "The physical layer of the ieee 802.11 p wave communication standard: the specifications and challenges," in *Proceedings of the world congress on engineering and computer science*, vol. 2, 2014, pp. 22–24.

[12] P. Bhagwat, "Bluetooth: technology for short-range wireless apps," *IEEE Internet Computing*, vol. 5, no. 3, pp. 96–103, 2001.

[13] G. D. Putra, A. R. Pratama, A. Lazovik, and M. Aiello, "Comparison of energy consumption in wi-fi and bluetooth communication in a smart building," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017, pp. 1–6.

[14] R. Friedman, A. Kogan, and Y. Krivolapov, "On power and throughput tradeoffs of wifi and bluetooth in smartphones," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1363–1376, 2012.

[15] H. Chu, Z. Xie, Y. Shao, Q. Liu, and Z. Mi, "Design and implement of wsn based on bluetooth and embedded system," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 5. IEEE, 2010, pp. V5–641.

[16] B. Zheng, P. Wang, F. Liu, and C. Wang, "Cooperative data delivery in sparse cellular-vanet networks," in *2016 6th International Conference on Digital Home (ICDH)*. IEEE, 2016, pp. 128–132.

[17] H. Kaur *et al.*, "Analysis of vanet geographic routing protocols on real city map," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2017, pp. 895–899.

[18] S. Hu, Y. Jia, and C. She, "Performance analysis of vanet routing protocols and implementation of a vanet terminal," in *2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC)*. IEEE, 2017, pp. 1248–1252.

[19] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*. IEEE, 1999, pp. 90–100.

[20] BBC, "Facebook to pay record $5bn to settle privacy concerns." [Online]. Available: https://www.bbc.co.uk/news/business-49099364

[21] A. Press, "Looters take advantage of new orleans mess." [Online]. Available: http://www.nbcnews.com/id/9131493/ns/us_news-katrina_the_long_road_back/t/looters-take-advantage-new-orleans-mess/#.XZus_CV7lTY

[22] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[23] N. Li, "Research on diffie-hellman key exchange protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 4. IEEE, 2010, pp. V4–634.

[24] F. J. Aufa, A. Affandi *et al.*, "Security system analysis in combination method: Rsa encryption and digital signature algorithm," in *2018 4th International Conference on Science and Technology (ICST)*. IEEE, 2018, pp. 1–5.

[25] N. H. Phong and M.-K. Kim, "Enhancing reliability on wireless sensor network by aodv-er routing protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 6. IEEE, 2010, pp. V6–32.

[26] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of INFOCOM'97*, vol. 3. IEEE, 1997, pp. 1405–1413.

[27] T. L. Lin, Y. S. Chen, and H. Y. Chang, "Performance evaluations of an ant colony optimization routing algorithm for wireless sensor networks," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2014, pp. 690–693.

[28] G. Sharvani, N. Cauvery, and T. Rangaswamy, "Different types of swarm intelligence algorithm for routing," in *2009 International Conference on Advances in Recent Technologies in Communication and Computing*. IEEE, 2009, pp. 604–609.

[29] A. V. Leonov, "Modeling of bio-inspired algorithms anthocnet and beeadhoc for flying ad hoc networks (fanets)," in *2016 13th International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, vol. 2. IEEE, 2016, pp. 90–99.

[30] T. Choksatid, W. Narongkhachavana, and S. Prabhavat, "An efficient spreading epidemic routing for delay-tolerant network," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 473–476.

[31] C. S. De Abreu and R. M. Salles, "Modeling message diffusion in epidemical dtn," *Ad Hoc Networks*, vol. 16, pp. 197–209, 2014.

[32] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: Positive and negative social effects," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 387–401, 2012.

[33] G. K. Wong, Y. Chang, X. Jia, K. H. Wong, and W.-Y. Hui, "Performance evaluation of social relation opportunistic routing in dynamic social networks," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 874–878.

[34] Z. Zhu, S. Liu, S. Du, X. Lin, and H. Zhu, "Relative interpersonal-influence-aware routing in buffer constrained delay-tolerant networks," in *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2013, pp. 4446–4451.

[35] N. Patel, "It's time to break up facebook." [Online]. Available: https://www.theverge.com/2018/9/4/17816572/tim-wu-facebook-regulation-interview-curse-of-bigness-antitrust

[36] "British computer society code of conduct," British Computing Society, BCS The Chartered Institute for IT, First Floor Block D, North Star House, North Star Avenue, Swindon, SN2 1FA. [Online]. Available: https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/

[37] "Research code of practice," Univeristy of Warwick, University of Warwick, Warwickshire, CV4 7AL, United Kingdom. [Online]. Available: https://www.warwick.ac.uk/services/ris/research_integrity/code_of_practice_and_policies/research_code_of_practice/

[38] "Ethical consent," Department of Computer Science, University of Warwick, Warwickshire, CV4 7AL, United Kingdom. [Online]. Available: https://warwick.ac.uk/fac/sci/dcs/teaching/ethics

[39] A. Holst, "Global mobile os market share in sales to end users from 1st quarter 2009 to 2nd quarter 2018." [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[40] A. S. Huang and L. Rudolph, *Bluetooth essentials for programmers*. Cambridge University Press, 2007.

[41] MICROCHIP, "Bluetooth® low energy packet types." [Online]. Available: https://microchipdeveloper.com/wireless:ble-link-layer-packet-types

[42] "Fix 4.2 protocol specification guide," London Stock Exchange Group.

[43] D. Crockford, "Introducing json." [Online]. Available: https://json.org/

[44] "Bluez: Official linux bluetooth protocol stack." [Online]. Available: www.bluez.org

[45] R. Pressman, *Software Engineering: A Practitioner's Approach 7th Edition*. McGraw-Hill, 2010.

[46] I. Sommerville, *Software Engineering Ninth Edition*. Addison-Wesley, 2011.

Project Presentation

Investigating The Use of Bluetooth in Emergency Situations to Share Location Data

By Freddie Brown

Overall

- Motivations
- Research
- Design
- Implementation Challenges
- Demonstration
- Project Management
- Evaluation

Problem Area

During Hurricane Katrina, high winds destroyed antennas highly damaged or destroyed. Many of these were used to provide cellular services. [2]

No cellular infrastructure -> Lack of communication

Current Emergency Services have backup systems to communicate between themselves [6]

- Most services are for maintaining communications between branches of emergency services

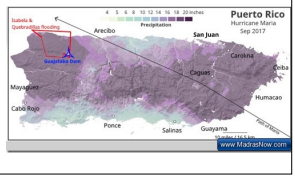
Aim is to provide more information to Emergency Services

Motivations

In the 2017 hurricane in Puerto Rico, the lack of communications services after the hurricane contributed to an increase in mortality of 62% [1]

United Nations voted that access to the internet is a human right that should be protected [3]

Want to create a way to improve the ability of the emergency services to respond to disaster when there is a lack of infrastructure



Existing Systems

HIRO-NET [4]

- Bluetooth mesh networks provide internet access to connected devices

Smartphone emergency message dissemination [5]

- One-way Ad-hoc networks to disseminate geographical information to devices connected to cellular network

Technologies Used

Bluetooth Low Energy (BLE)

Reduces the amount of data that needs to be sent

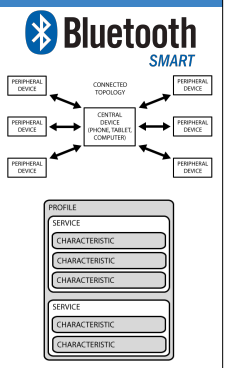
Sporadic data transmission uses less energy using LE vs Classic

GAP - Generic Access Profile

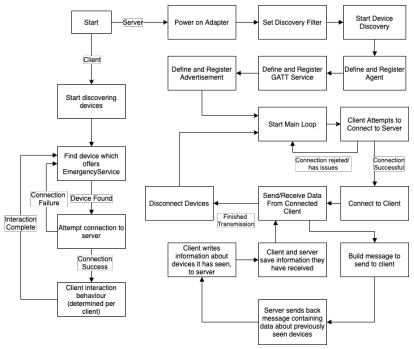
- Performs advertising and decides what type of device (Peripheral/Central)

GATT - Generic Attribute Profile

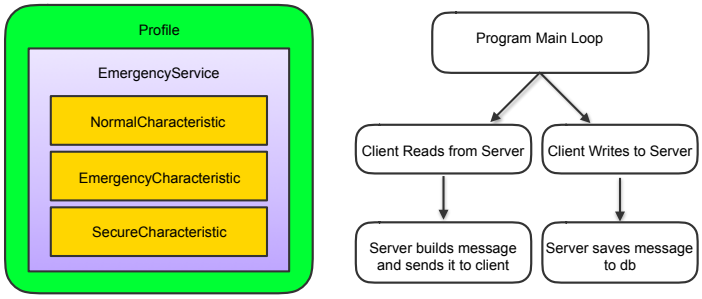
- Defines data transmission (Services and Characteristics)



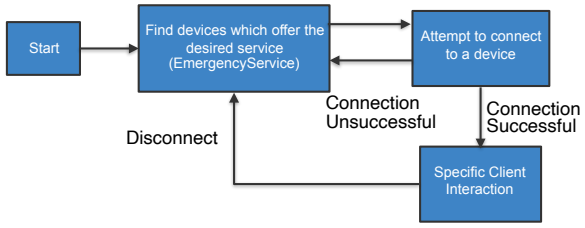
Design: Overview



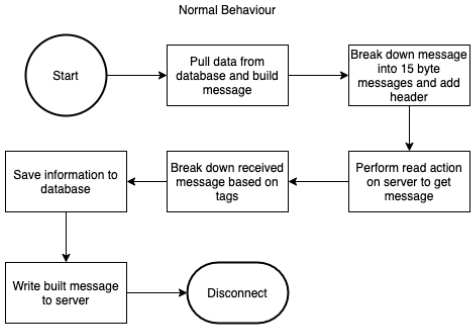
Design: Server



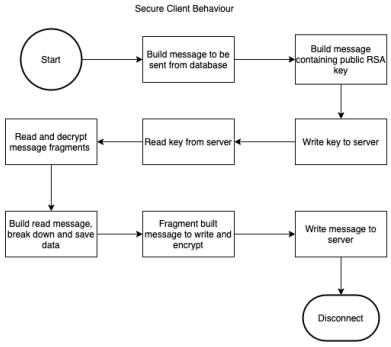
Design: Client



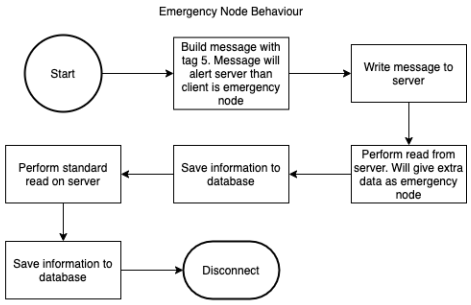
Design: Client Behaviours



Design: Client Behaviours



Design: Client Behaviours



Implementation Challenges

This message is fragmented due to BLE message size limits (20 Bytes)

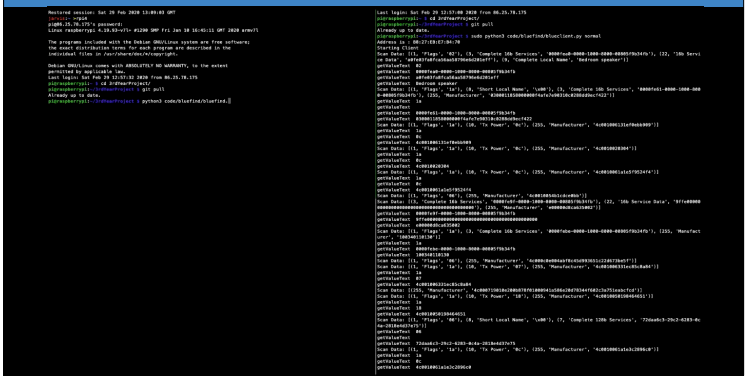
Message format based off FIX
- Financial Information eXchange

1=(57.0461, -10.9456)|2=AB:CD:EF:GH:IJ:KL|

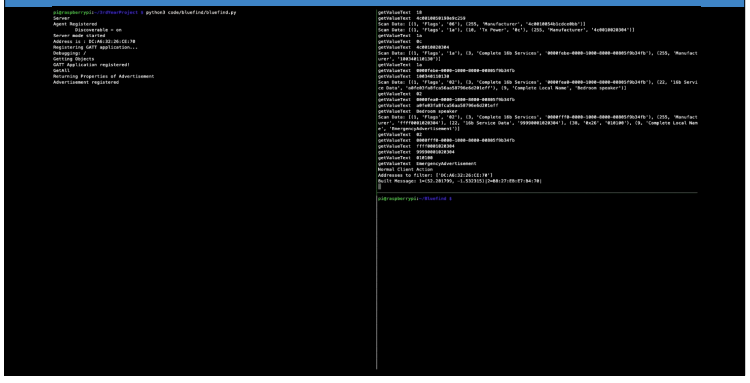
BLE uses Read/Write system to a device
- Had to adjust original model for system

Have nodes as individual devices
- Ensuring system maintains decentralised design
- Even with Master-Slave model in BLE

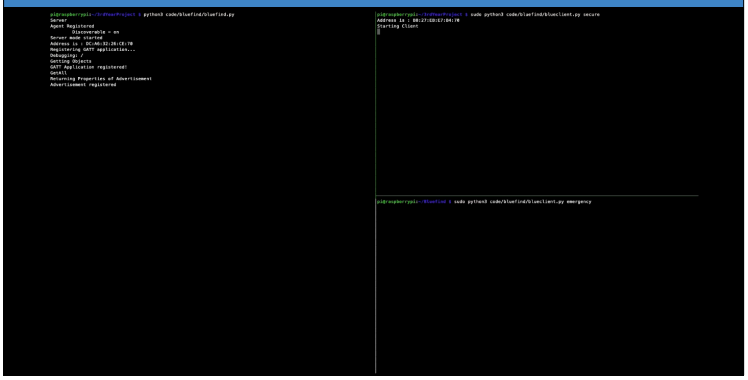
Can We Do Point-to-Point Comms?



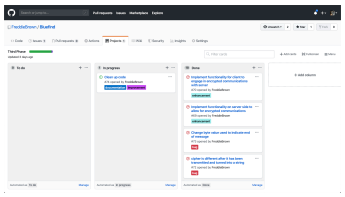
Can We Relay Messages Among Client Sets?



How Complex Can Behaviours Get?



Project Management



	Start Date	End Date	Timeline	Status
Project	30-09-2019	23-04-2020		
First Research	30-09-2019	07-10-2019		Done
Specification	30-09-2019	10-10-2019		Done
Routing Algorithms Research	11-10-2019	19-11-2019		Done
MAC Protocol Research	11-10-2019	19-11-2019		Done
Data Security Research	11-10-2019	19-11-2019		Done
Progress Report	11-10-2019	25-11-2019		Done
Implementation	26-11-2019	19-02-2020		Done
Presentation	20-02-2020	04-03-2020		Doing
Final Report	11-10-2019	23-04-2020		To Do
	30-09-2019	23-04-2020		

Evaluation

- Extended emergency node capabilities
  - Visualisation of data which it receives
- Re-write project in C
  - More complex packet and performance work
- Extend and create API so it can be built into and used in other systems
- Improved Tolerance to Faults

This project has been influenced by both commercial and academic solutions and is a system which utilises the best aspects of both to address an unsolved problem area

References

[1] N. Kishore, D. Marqués, A. Mahmud, M. V. Kiang, I. Rodriguez, A. Fuller, P. Ebner, C. Sorensen, F. Racy, J. Lemery et al., "Mortality in puerto rico after hurricane maria," New England journal of medicine, vol. 379, no. 2, pp. 162–170, 2018.

[2] K. Banipal, "Strategic approach to disaster management: lessons learned from hurricane katrina," Disaster Prevention and Management: An Inter- national Journal, vol. 15, no. 3, pp. 484–494, 2006.

[3] H. R. Council, "Promotion and protection of all human rights, civil, political, economic, social and cultural rights, including the right to development," UN General Assembly, vol. A/HRC/32/L.20, June 2018.

[4] L. Ferranti, S. D'Oro, L. Bonati, E. Demirors, F. Cuomo, and T. Melodia, "Hiro-net: Self-organized robotic mesh networking for internet sharing in disaster scenarios," in 2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM). IEEE, 2019, pp. 1–9.

[5] X. Wu, M. Mazurowski, Z. Chen, and N. Meratnia, "Emergency message dissemination system for smartphones during natural disasters," in 2011 11th International Conference on ITS Telecommunications. IEEE, 2011, pp. 258–263.

[6] https://www.fema.gov/disaster-emergency-communications