Alexandra Riddell-Webster

A System to Help Prevent Crashes in Rowing Boats

Computer Science Tripos – Part II

Murray Edwards College

2023

Declaration of Originality

I, Alexandra Riddell-Webster of Murray Edwards College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

I am content for my dissertation to be made available to the students and staff of the University.

Signed

Date
April 20, 2023

Acknowledgements

This dissertation owes a huge amount to Matthew Ireland, for supervising me. My UTO, Jon Crowcroft was invaluable. Thanks to Cambridge University Boat Club, in particular Patrick Ryan, Mike Taylor and Rosa Millard, for allowing me to put strange boxes on boats, advising me on communication over water and helping me evaluate the system. I also thank Duncan Barnes for discussing GPS and electronics on rowing boats with me.

Proforma

Candidate Number:	TODO
Project Title:	A System to Help Prevent Crashes in Rowing Boats
Examination:	Computer Science Tripos – Part II, May 2023
Word Count:	TODO
Code Line Count:	TODO
Project Originator:	Alexandra Riddell-Webster
Supervisor:	Mr Matthew Ireland
University Teaching Officer:	Dr Jon Crowcroft

Original Aims of the Project

The aim of this project was to design an accessible system for helping prevent collision avoidance in rowing boats. The goal of this system was to warn rowers and coxes when they approach obstacles, be they other boats or items in the river.

Work Completed

This project has created such a system, propagating knowledge about obstacles through a mobile ad hoc network (MANET) of rowing boats using the Epidemic routing protocol. Users were warned when approaching a known obstacle by means of a buzzer and LED light. The MANET was evaluated, with the percentage of packets delivered, bandwidth, latency and time taken to propagate messages after partition analysed.
All of the success criteria of the project were met.

Special Difficulties

None.

Contents

1 Introduction **7**
- 1.1 Motivation . . . 7
- 1.2 Background . . . 7
- 1.3 Related Work . . . 8
  - 1.3.1 Networking . . . 8
  - 1.3.2 Delay Tolerant Routing . . . 9
  - 1.3.3 Safety in Rowing . . . 9

2 Preparation **10**
- 2.1 Starting Point . . . 10
- 2.2 Choice of Hardware . . . 10
- 2.3 Software Development Methodology . . . 11
- 2.4 Choice of Programming Language . . . 11
- 2.5 Summary of Research . . . 11
  - 2.5.1 MANET . . . 11
  - 2.5.2 Routing . . . 12
  - 2.5.3 Epidemic . . . 12
    - 2.5.3.1 Anti-Entropy . . . 12
  - 2.5.4 Media Access Control . . . 13
  - 2.5.5 GPS . . . 13
  - 2.5.6 Serial Communication . . . 13
- 2.6 Requirements Analysis . . . 13

3 Implementation **15**
- 3.1 Open Source . . . 15
- 3.2 System Design . . . 15
  - 3.2.1 Application . . . 15
  - 3.2.2 Networking . . . 16
  - 3.2.3 Packet Design . . . 17
- 3.3 Point to Point . . . 17
- 3.4 Epidemic . . . 17
  - 3.4.1 Discovery Messages . . . 17
  - 3.4.2 Anti-Entropy . . . 17
- 3.5 GPS . . . 17
  - 3.5.1 Satellite Error . . . 18
- 3.6 MicroPython . . . 18
- 3.7 User Interface . . . 18
  - 3.7.1 Hardware Choice . . . 18
  - 3.7.2 Use . . . 19
  - 3.7.3 Testing . . . 19
- 3.8 Repository Overview . . . 19

4 Evaluation **21**
4.1 Evaluating the MANET . . . 21
4.1.1 Latency . . . 21
4.1.2 Bandwidth . . . 23
4.1.3 Percentage of Packets Delivered . . . 23
4.1.4 Partition Testing . . . 25
4.1.4.1 Asymmetric . . . 25
4.1.4.2 Symmetric . . . 25
4.2 Evaluating the System . . . 26
4.2.1 On Water . . . 26
4.2.2 Qualitative Evaluation . . . 27

5 Conclusion **28**
5.1 Summary . . . 28
5.2 Achievements . . . 28
5.3 Reflections . . . 28
5.4 Future Work . . . 29
5.5 Other applications . . . 29

Bibliography **29**

Appendices **31**
A – Guide to Building a Node . . . 31
B – Evaluation Plan . . . 32
C – Progress Report . . . 36
D – Project Proposal . . . 39

Introduction

During this project, I built a system to help prevent crashes in rowing boats. The system is implemented in hardware, cumulating in a series of boxes that can be attached to boats. The project is split into two parts. First, the mobile ad hoc network (MANET) allows nodes to communicate a dynamic collection of known obstacles, then the application layer warns users when they are approaching an obstacle and allows the user to add obstacles. The system will be available online, fully documented, after my graduation. This will include a 'how to' guide [Appendix A] for the construction of a node so any rowing club can use my project as a collision avoidance tool.

1.1 Motivation

Crashes between rowing boats and obstacles or other boats injure rowers and cause equipment damage. Unfortunately, crashes are common. Figure 1.1 shows a boat on the Thames, having collided with a bridge at a regatta in March 2023. Some boats have coxswains, responsible for steering a boat, while others are coxless, where rowers who face away from the direction of travel are responsible for steering boats. This project is designed for use in both coxed and coxless boats. My project has a very personal motivation, as a friend was hit by a larger rowing boat while in a single three years ago, causing a severe concussion that resulted in two years of intermission from their studies.

Figure 1.1: An eight colliding with Barnes Bridge [1]

1.2 Background

The Defence Advanced Research Projects Agency (DARPA) Packet Radio Network (PRNET) [2] was the first wireless data network, using store-and-forward routing over packet radios. Jubin and Tornow lay out the state of PRNET in 1987, nearly 15 years after research began on it, in their paper 'The DARPA Packet Radio Network Protocols'. The work on PRNET fed into DARPA's Survivable

Radio Network (SURAN) [3]. The connection between the military and ad hoc networking still exists today, with MANETs used in conflicts [4], as well as autonomous vehicles [5] and disaster relief scenarios where previously existing infrastructure is destroyed [6].

Each node in a MANET is free to move in any direction, so the topology of the network changes in an unpredictable way. Rowing boats frequently move at speeds greater than 15 km/h, making the topology of any network of boats unpredictable and dynamic. Every node in a MANET must forward traffic, making it a router. In addition to this, the network may move between rivers, meaning no existing infrastructure would be available. These form the primary challenge for my network; routing messages through the network without pre-existing infrastructure, giving each node enough information to pass traffic to other nodes.
This is made more difficult by the small CPU and memory on the Raspberry Pi Pico [7], limiting the processing power and information each node has.

While routing corresponds to many hops across a network, medium access control considers only one. Media access control protocols control access to the transmission media to prevent collisions between packets. This project implemented media access control to prevent packets colliding and subsequently being corrupted.

The Open Systems Interconnection (OSI) model for computer networking provides a standardisation for communication over a network. The OSI model consists of seven layers: Application, Presentation, Session, Transport, Network, Data Link, and Physical [8]. Based on this model, my project uses the libraries provided with hardware for the physical and data link layers. It focuses mainly on the networking layer, sending messages between nodes.

1.3 Related Work

1.3.1 Networking

There is precedent for using MANETs on rowing boats. I spoke to the team behind the broadcasting and associated telemetry for the Oxford Cambridge Boat Race. They use a 15-node MANET to get the video and telemetry from the rowing boats. They also take GPS readings for the location of the boat, recording the location of the boat up to 20 times a second. Figure 1.2 shows the equipment they attached to the crews for the 2023 Boat Race.

Figure 1.2: Equipment placed on the stern of the Cambridge boat [9]

1.3.2 Delay Tolerant Routing

Delay tolerant routing assumes that networks will lack connectivity, with partitions between nodes. Vahdat and Becker's paper 'Epidemic Routing for Partially-Connected Ad Hoc Networks' [10] introduces a replication-based, delay-tolerant routing protocol where messages are passed onto nodes that do not have a copy of the message. Vahdat and Becker's paper is the key paper used to implement routing within this project.

1.3.3 Safety in Rowing

ROWCUS [11], a company based in Switzerland, has attempted to solve the problem of crashes in rowing boats. While ROWCUS has similar goals to my project, the technical methodologies are different, using radar rather than GPS location to detect proximity to obstacles. Additionally, ROWCUS does not network nodes together, instead using individual nodes. ROWCUS has "decided not to pursue the commercial deployment of ROWCUS", in a statement on their website [11].

Preparation

2.1 Starting Point

I had no previous hands-on experience with microcontrollers, although I had worked with single-board computers. I therefore dedicated a small amount of time over the summer to learning about microcontrollers and MicroPython, completing basic tasks such as flashing an onboard LED. While this was useful, my project is implemented in CircuitPython, so it would have been more beneficial to have explored this.

My project implemented a modified version of the Epidemic routing protocol. I had some experience with networking and routing protocols prior to starting this project. This was composed of the Part IB Networking module and two weeks' work during an internship on a MANET with a different routing protocol, much further abstracted than this project. During the course of my project, I took the Part II Principles of Communications course, further expanding my knowledge.

2.2 Choice of Hardware

Hardware was ordered before the start of the Michaelmas term, in order to guarantee its availability for the project proposal. The main factors in my hardware choices were size, weight, power consumption and cost. Items needed to be relatively small to allow them to fit on a rowing boat. If the components were overly costly, it may prohibit other rowing clubs from producing their own networks.

I chose to use a microcontroller to run the system due to the reduction in power consumption and costs it would offer over a single-board computer. The Raspberry Pi Pico was chosen due to its large peripheral set, with support for both UART, SPI, and I2C protocols. This allowed for greater flexibility throughout the project.

For GPS and radio, AdaFruit boards were chosen due to the strong community surrounding the hardware, with the AdaFruit boards being supported by open source libraries that allow rudimentary operations to be performed.
The AdaFruit RFM69 radio has an SPI interface and 500m range, appropriate for the use case as rowing boats will take around 2 minutes to cover 500m. Additionally, I chose to use the 433 MHz industrial, scientific, and medical (ISM) band as it is free to use without licencing, allowing me and other rowing clubs to use it without incurring additional costs.
The CD-PA1616S GPS has benefits similar to the RFM69, with community support and libraries for basic communication. It was additionally chosen as it includes a patch antenna and a relatively short cold start time – these were important for the use case, as a delay in the collision avoidance device starting up reduces the overall utility of the system.

2.3 Software Development Methodology

Most software development methodologies are designed for use within a team of programmers working over a long period and are not suited for a part II project. I therefore integrated key components from several software development methodologies, particularly the waterfall and agile methodologies.

The waterfall model of software development was used as it lends itself to the structure of a dissertation, with the five stages:

Requirements → System Design → Implementation → Testing → Maintenance

From the waterfall model of software development, I took the focus on planning and documentation, particularly as I wanted others to be able to reproduce my results.

Agile software development is based on 12 principles that highlight responding to change and generating working software [12].
I integrated the tenets "Simplicity – the art of maximizing the amount of work not done – is essential" [13] and "Welcome changing requirements, even late in development. " [13] into the project.

I broke the work to be done down into blocks, giving each block concrete deliverables to mark the end of each block. This helped prevent scope creep and keep me on track for each part. I also tested each component of the project as it was built, ensuring that it worked and minimising debugging that needed to be done on the final product.

2.4 Choice of Programming Language

This project used CircuitPython, a branch of MicroPython, a version of Python for microcontrollers. CircuitPython was used as it can easily be run on the Raspberry Pi Pico. Additionally, it allowed me to use Pylint to statically analyse code. This was particularly useful as the code was run on an external Raspberry Pi Pico, so running the code to find small errors would have been an inconvenience. Additionally, the existing AdaFruit libraries I used in the project were written in CircuitPython, so keeping the whole project in the same language reduced complexity.

GitHub was used to perform version control on the code and dissertation for this project. This also allowed me to fork repositories and submit a pull request to AdaFruit.

I used Visual Studio Code as my main Integrated Development Environment as I have used it in previous projects, and it offered support for CircuitPython via an extension.

2.5 Summary of Research

2.5.1 MANET

A mobile ad hoc network (MANET) is characterised by wireless nodes, a frequently changing network topology and no reliance on pre-existing infrastructure. They are decentralised and therefore have no single point of failure [14]. This project constructs a MANET due to the lack of pre-existing infrastructure and the difficulties associated with setting up and maintaining a base station or similar. Requiring an infrastructure like this would also raise the barrier to entry for many clubs with limited funds and technical skills. Additionally, rowing boats can move between stretches of water, further supporting the use of a MANET for this project.

2.5.2 Routing

Routing protocols find a path from a source to one or more destinations destination within the network. Different routing protocols optimise different parameters and are better suited for different network topologies and applications [15]. Within MANETs, a routing protocol must allow the network topology to change over time. They tend to contain node discovery techniques to allow for this.

Before deciding on Epidemic as the routing protocol to implement for my project, I considered several other protocols. The Better Approach to Ad Hoc Mobile Networking (BATMAN) protocol [16] is designed to route messages through MANETs, broadcasting originator messages (OGM) for node discovery. BATMAN has the interesting addition of a transmit quality (TQ) metric in the OGM packets, allowing the quality of connections between nodes to be factored into the route packets take through the network. While BATMAN does allow messages to be broadcast to all nodes, its primary focus is routing messages from one node to another. Additionally, while it allows for message mobility, it is not delay-tolerant.

Greedy Perimeter Stateless Routing (GPSR) [17] is a location-based routing protocol. GPSR exploits the relation between geographic position and connectivity in a wireless network, where each node tells its immediate neighbours its current location. Greedy forwarding is predominantly used to send packets to nodes that are progressively closer to the destination until the destination coordinates can be reached. Where greedy forwarding fails, GPSR uses perimeter forwarding (forwarding the packets around the perimeter of the region) until greedy forwarding can be used again. This protocol was ultimately deemed to be unsuitable for my project as, similar to BATMAN, its primary focus is on sending messages between two nodes. Additionally, the high mobility of the nodes in the use case means that forwarding packets to a set of coordinates does not mean the message will reach the intended destination, as the node may have moved.

2.5.3 Epidemic

I chose to implement Epidemic in my project [10]. Epidemic routing gains its name from its similarity to the spreading of infections. Each node replicates and transmits messages to neighbours that have not recently been contacted. These neighbours are discovered when each node broadcasts its existence to its neighbours. Epidemic was implemented in this project because it is a delay-tolerant routing protocol and best fits the likely network topology generated by rowing boats, as detailed in the Requirements section below. The networks generated by rowing boats have a high chance of partition, but the nodes are highly mobile. As Epidemic allows any node to carry network information it is best suited to the network. Additionally, Epidemic supports a broadcast functionality, sending messages to every node in the network, which is necessary for the use case as every boat should hear about potential obstacles.

2.5.3.1 Anti-Entropy

The term ‘anti-entropy’, as used in my dissertation, comes from this paper. An anti-entropy session occurs when two nodes come into communication range and exchange messages. Anti-entropy within Epidemic is initiated by the node with the lower ID. This lower ID node initiates anti-entropy by sending a summary vector, representative of the messages they have to another node. This node calculates logical AND between this summary vector and the negation of its own summary vector to produce the messages it has not seen that the other node has. It requests these messages. The node with the higher ID then sends its updated summary vector to the other node, and the node with the lower ID requests any messages it has not seen before. After a successful message exchange, the node is added to a list of recently contacted nodes, preventing anti-entropy from being conducted many times between two sets of nodes. This list is periodically cleared.

2.5.4 Media Access Control

While these routing protocols often have mechanisms to minimise the probability of collisions, such as adding a jitter in the sending of discovery messages, none of them contain medium access control. Due to the probability of collisions between messages causing disruption to the sending of messages, particularly as all nodes are broadcasting on the same radio frequency (433 MHz). It was therefore prudent to include media access control in the system. Multiple Access with Collision Avoidance for Wireless (MACAW) is often used by ad hoc networks. MACAW uses request to send (RTS) and clear to send (CTS) messages to minimise the probability of collision. As discussed in the System Design section, MACAW inspired the medium access control in my project.

2.5.5 GPS

TODO GPS and haversine formula - note we assume the earth is a sphere (error as assumes a sphere). Put in the equations :) Perhaps something about the GPX file format?

2.5.6 Serial Communication

A short overview of the different serial communication protocols I used? They were new to me but would they be new to examiners?

2.6 Requirements Analysis

The three requirements I identified as my success criteria were:

The Epidemic routing protocol is implemented within the network	**Achieved**
An evaluation of the network has been carried out	**Achieved**
An application layer has been implemented to allow utility of the network	**Achieved**

All of these requirements have been implemented during this project.

I also laid out several extension tasks. Some of them were drawn from my research into networking protocols other than Epidemic. For instance, I wanted to use a metric for transmission quality – received strength signal indicator (RSSI) in radio communication – to influence whether an anti-entropy session was initiated. Additionally, the GPS location could be used, either by only contacting nearby nodes to increase the probability that an anti-entropy session is successful, or prioritising sending messages about new obstacles to nodes that are near these obstacles. While time constraints have not allowed me to implement these extensions, I have implemented an extension allowing messages to have two priorities, normal and urgent.

These requirements were based on the use case for the project – as devices attached to rowing boats, where the nodes have high mobility. This mobility requires a high degree of flexibility within the network. I analysed the potential topology of networks. This was done by looking at example distributions of rowing boats on lakes and rivers. One potential use case for this network that I am familiar with is the River Thames. I analysed Google Maps and Earth's satellite view of the Thames along a 5.5km stretch of the Championship Course, pinpointing rowing boats and coaching launches, then adding them to a map with potential connections between nodes, assuming the radios have a range of 500m, alongside obstacles. Figure 2.3 shows a satellite image of a single sculler, a potential node in this network. Figure 2.4 shows the annotated map of the Thames, and Figure 2.5 presents the abstracted network topology of these rowing boats.

Figure 2.1: A rowing boat seen on Google Earth [18]

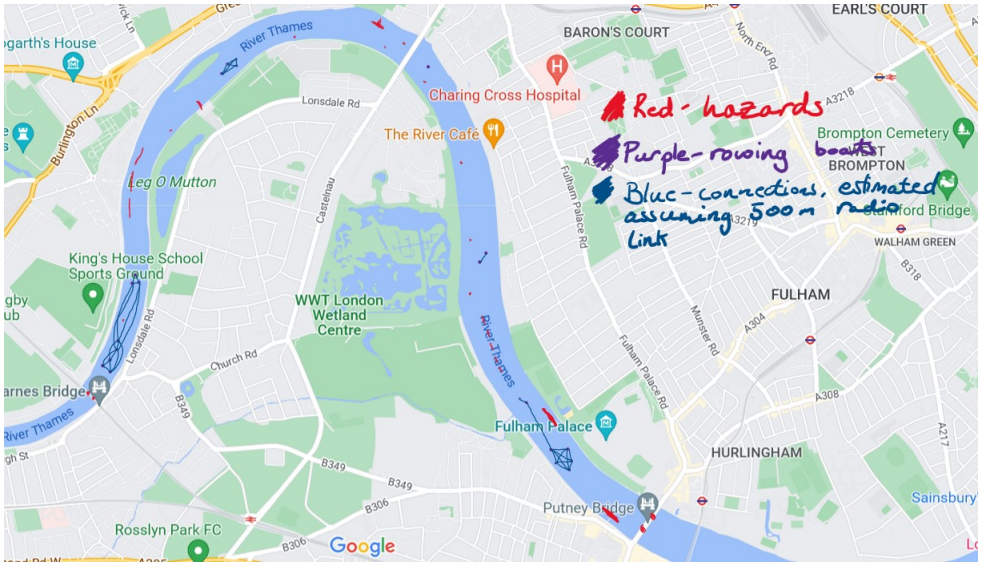


Figure 2.2: Rowing boats, potential obstacles and assumed connections marked on a Google Maps map of the river Thames

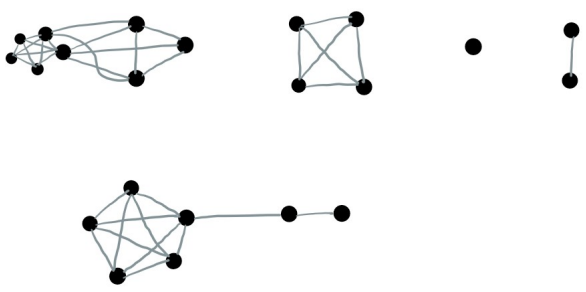


Figure 2.3: The network of rowing boats in an abstracted form

Implementation

Brief paragraph about how the implementation happened - system design then follow the layers with timetable.

3.1 Open Source

Not open source until graduation. Why? Make it clear you want it to be open source all the way and the rules around plagiarism are stopping you

3.2 System Design

Having decided on the system identifying obstacles using GPS coordinates and the Epidemic routing protocol to propagate obstacles through the network, I planned the structure of the software that would run on each node. The Raspberry Pi Pico uses the RP2040 chip, containing two cores [7]. To make best use of the hardware, I decided to design application and networking threads that would run on each core, with global data structures and concurrency control to pass messages between the two layers and allow both of them to access the GPS.

3.2.1 Application

The application thread is responsible for notifying the user when they are too close to an obstacle. It also takes input from the user, generating new obstacles when a button is pressed. The initial state machine is shown in Figure 2.4. The a time to live (TTL) for each obstacle. If the object is permanent, such as a bridge, the TTL is set to -1, indicating that it should never be removed.

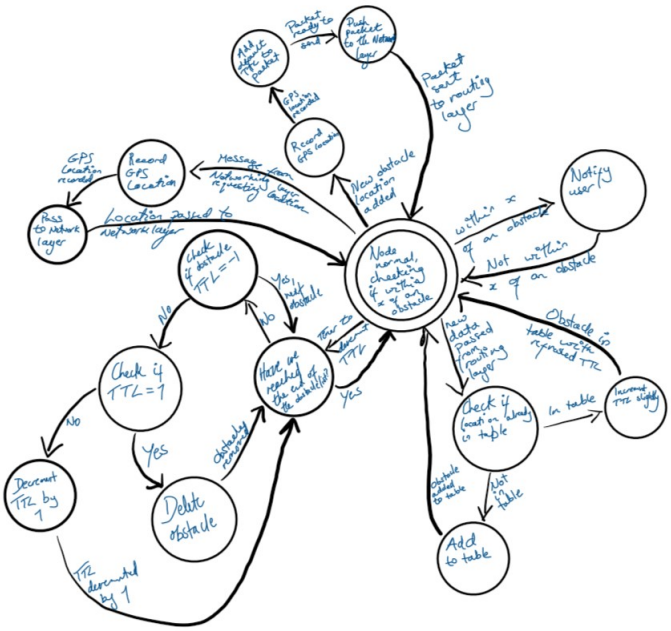
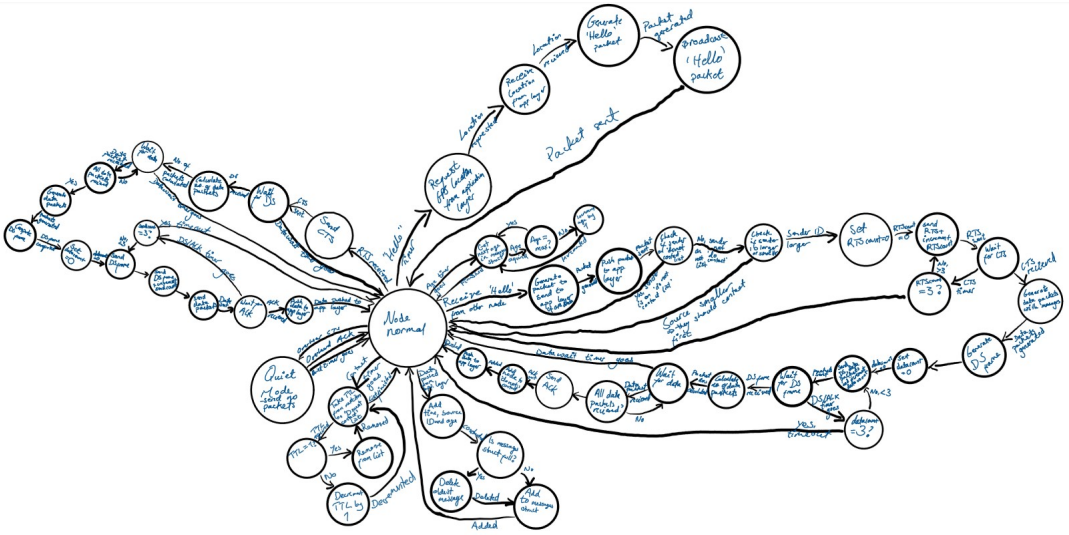


Figure 3.1: The initial state machine for the application thread

3.2.2 Networking

The networking thread contains the implementation of Epidemic with media access control. While routing and medium access control are typically handled separately – in the OSI model they are in the second and third layers respectively – they were considered together for my project to prevent work being repeated and use the limited computing resources available most effectively. This means that the implementation of Epidemic will also change. Not only will the node refuse to send any messages after hearing a CTS for a set period of time, or until it hears an ACK, I also integrated the message vector exchange at the start of an anti-entropy session into the CTS and RTS messages. This minimised the number of packets sent over the network, reducing the overall probability of collisions or errors in transmission. This state machine has changed slightly since it was first designed.

Figure 3.2: The initial state machine for the networking thread



While these state machines were broadly implemented, the overall structure of the software changed, with only one thread running due to the limitations of CircuitPython and difficulties transitioning into MicroPython.

3.2.3 Packet Design

After designing the system, I considered the structure of the packets sent by each node. These are influenced by the structures of packets in previous routing protocols I examined. I attempted to keep the structure of the start of the packets Figure 3.3 shows the design for these packets. They all start with the length of the packet so the system knows how much data to expect.

3.3 Point to Point

Before starting to Talk about the width of wire antenna (quarter wave whip antenna) having an impact on the range. Desg=ignate 0x00 as broadcast address, as is common in many networks.

3.4 Epidemic

3.4.1 Discovery Messages

From point to point the move to sending node discovery, hello, messages

HELLO = const(0x00) Payload contains the keys for message dict

3.4.2 Anti-Entropy

RTS = const(0x01) Payload contains the keys for message dict CTS = const(0x02) First byte of payload is the total number of DATA packets, second is the number of the DATA packet, then the messages themselves DATA = const(0x03) Empty payload ACK = const(0x04)

Talk about all the changes you made and perhaps generate a new state machine that shows the final epidemic you used - "history of changed decisions"

Fixed length packets removed issues

What edge cases did we consider and deal with? When messages are identical When messages is empty When one is empty and other isn't

Some nice diagrams for packet structure and how they are sent (how about one of those diagrams with two lines for time and messages sent between them

Justify your choices of timeout etc

Also the TTL started as being time based and became number of hops (moves into extension where one is sent with more hops -¿ will propagate through the network faster (you didn't actually test this...) Debugging when connected to an external power source was actually awful I have never hated antthing so much

Timeouts as well

Talk about generating messages on what typed in, generated messages anyway, generating messages based on node ID and number of messages previously generated, circling round

3.5 GPS

Before integrating the GPS into the system, I ensured that it worked on its own. It took a little while to get this to work, having moved through several serial communication protocols. In the end, I used UART as the asynchronicity seemed to work best with an unpredictable signal. To ensure that the GPS was mapping to the correct place, I ensured that the coordinates it fed me were near my location.

To test the tracking on the GPS, I wrote a short program to generate GPX files. I then ran this while holding the device. For comparison, I tracked a 'walking' activity on my smart watch. Figure TODO shows my first attempt at this.

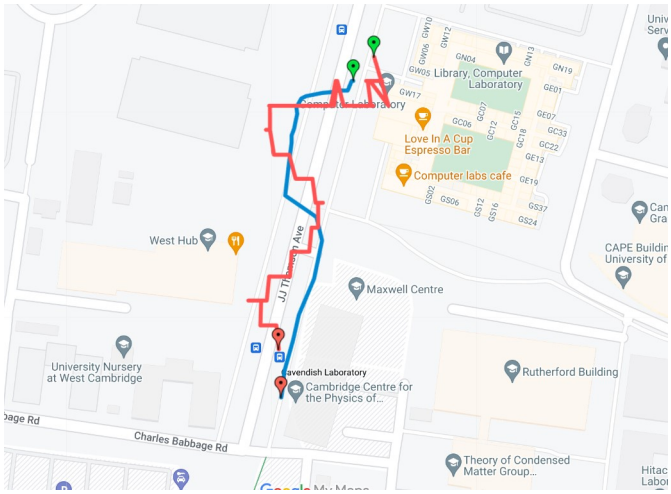


Figure 3.3: The first attempt at tracking a walk with GPS. The red line is the GPX track generated by the board, the blue line is the GPX track generated by my smartwatch [19]

As can be seen from Figure TODO, the track generated by the system's GPS is jagged and not very accurate. On examination of the GPS coordinates generated by the board

Took a little while to get this to work - why? Not picking up satellites, adding a button battery, changing the Serial interface used Precision with the cool tracking haversham formula

3.5.1 Satellite Error

I found that the GPS location moved about even when I was stationary. At first, I believed this to be an error in the library, perhaps related to the precision of floating point numbers. After attempting to debug this and talking to the gentleman who took GPS measurements for the Oxford Cambridge Boat Race, I discovered that this is a limitation of the technology, and that as satellites move across the sky they generate noise in the readings. In their solution to this problem, the Boat Race sets their own base station up at the end of the course, uses an external antenna and a reads from the GPS up to 20 times a second. They then feed this data through a Kalman filter to give the final value for location. Implementing a Gaussian or Kalman filter to remove some of the noise from the GPS would be possible in this system and is further work. In the meantime, it is necessary to accept a small quantity of error in the reading.

3.6 MicroPython

I ensured that microPython can multithread on the Raspberry Pi Pico. CircuitPy thon does not have this capacity. Debugging with Matthew - images and perhaps screenshots? only one thread running due to the limitations of CircuitPython. Therefore tried to port to MicroPython Had one on MicroPython and one on circuitpython and tried to get the two talking Can't multithread in CircuitPython (what is needed to natively (natively?) run the libraries for the AdaFruit modules

Turns out getting circuitpython to port in micropython is like 8 levels of dependency nightmare Trying to get circuitpython libraries to work with micropython using Blinka

3.7 User Interface

3.7.1 Hardware Choice

In order for the MANET and GPS to be useful, a rudimentary user interface was built. The plan initially contained only a buzzer and button, but I added an LED, to allow the device to be more

accessible to those who may not be able to hear the buzzer. I therefore chose to use a button that had an LED built into it. This kept the design of a node simple, meaning it is easy to construct and less likely to get water in. The button was carefully chosen to ensure that it contained the LED. It was also chosen to be waterproof to fit the use case, as it may get wet in rowing boats. It was chosen to be a momentary, non-locking, relatively large button to allow the user to press it when they encounter an obstacle, then move on.
A 10k resistor was chosen to be used in series with the buzzer. This resistance was chosen to as it modulates the tone to sound non-threatening but like a warning. A brief survey of three rowers agreed this would be the best tone. These rowers also happened to be my irritated housemates.

3.7.2 Use

These components were integrated into the system. The buzzer and LED were wired in parallel to the same pin in the Pico, to allow the flashing and buzzing to occur simultaneously. If the system detected that the current GPS position of the node was within the distance specified by the `GPS_DISTANCE` value in `config.py`, using the Haversine formula I implemented.

3.7.3 Testing

The testing of the user interface first consisted of ensuring each piece of hardware worked. I ensured that pressing the button registered. I wrote a short program to flash the LED and buzz the buzzer when the button was pressed. I then made sure that the system generated packets when the button was pressed. This was done by generating messages with unique IDs based on the node address and the number of messages that have been generated by the node when the button was pressed and ensuring that these messages were received by nearby nodes. Once satisfied that this worked in conjunction with the GPS and MANET, I moved onto the evaluation.

3.8 Repository Overview

Matthew – will be typeset / formatted slightly more nicely, this is the 'bare bones'

```
(doc) LICENCE
(doc) README.md

(file) Development
    (file) Application
    (file) MACEpidemic
    (file) Misc
    Contains the code used in development of the system. MACEpidemic and
    Application contain the code generated when building the MANET and
    interface to it respectively. Misc contains a more varied set of files,
    generated when trying out hardware and other small components
    of the system.

(file) Evaluation
    Split into various folders depending on the evaluation being conducted.
    It contains the code used to run each subsection of the evaluation,
    results and python files for analysis of the results.

(file) OnDevice
    (file) lib
    (doc) boot.py
    This is the final product, the code that should be uploaded to the Raspberry
```

Pi Pico. The boot.py file contains the implementation of the system to help avoid collisions. The lib file contains adafruit_gps.py, a library I use, licenced under the MIT licence. rfm69.py is a modified version of adafruit_rfm69.py, which I have modified to include fixed length packets and fixed some bugs, so it is a combination of code I wrote and others' code.

Evaluation

The majority of the evaluation was conducted according to the plan in Appendix B [Appendix B]. It was split into a performance evaluation of the MANET and then a more qualitative evaluation of the whole system.

4.1 Evaluating the MANET

The evaluation of the MANET was designed to mirror the evaluation performed by Vahdat and Becker's paper, with the key difference that my project was implemented in hardware, while theirs was virtualised.

4.1.1 Latency

The latency of the system was as the time between a node being generated at node A and received at node B. Node B was then moved further away from node A, eventually moving to 500m away with an interim node, C. The distances between nodes were 0, 10, 100, 250 and 500 metres. 60 messages were generated by A at each distance. Figure 4.1 shows the setup, with radio connections between nodes marked in blue.

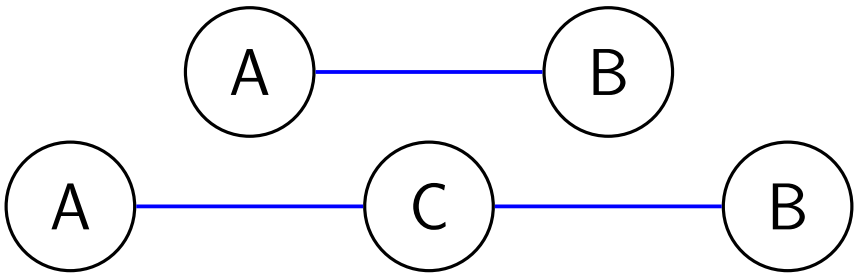


Figure 4.1: The network used for latency testing

Figure 4.2 shows the percentage of packets delivered over time. Figure 4.3 breaks down the data behind this, showing the times that each packet arrived at each distance. It is unsurprising to find that the charts for 0 and 10 metres differ little, with an average delivery time of 15.0 and 15.7 seconds respectively. The 100 metre interval follows a similar pattern. There is then a significant drop-off when the distance is increased to 250 and 500 metres, likely because more than one hop needed to be used to move messages from one node to another. There is a significant increase in the standard deviation, due to the increased variability from passing messages over multiple nodes.

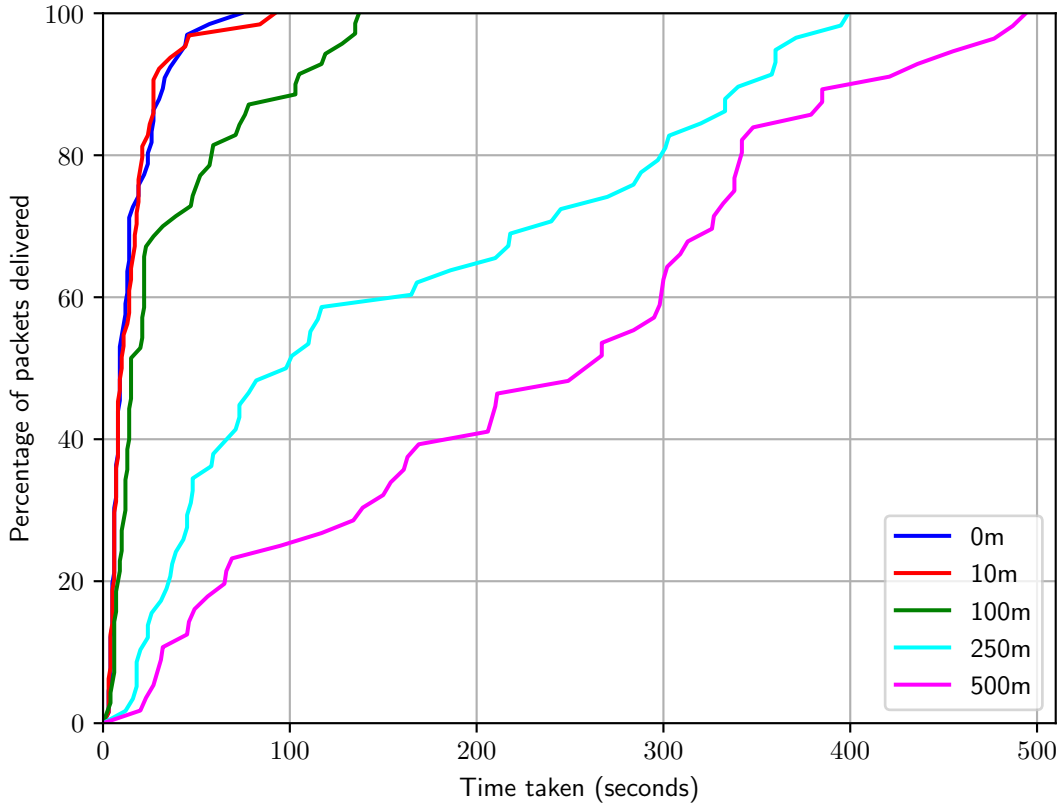


Figure 4.2: The percentage of packets delivered with time

Distance	Average Delivery (seconds)	Standard Deviation
0	15.045454545454545	13.795510708287654
10	15.703125	16.1986493336443
100	34.91428571428571	37.96605358066329
250	151.74137931034483	126.72064855622898
500	233.35714285714286	138.67941300653365

Table 4.1: Average delivery and standard deviation in seconds

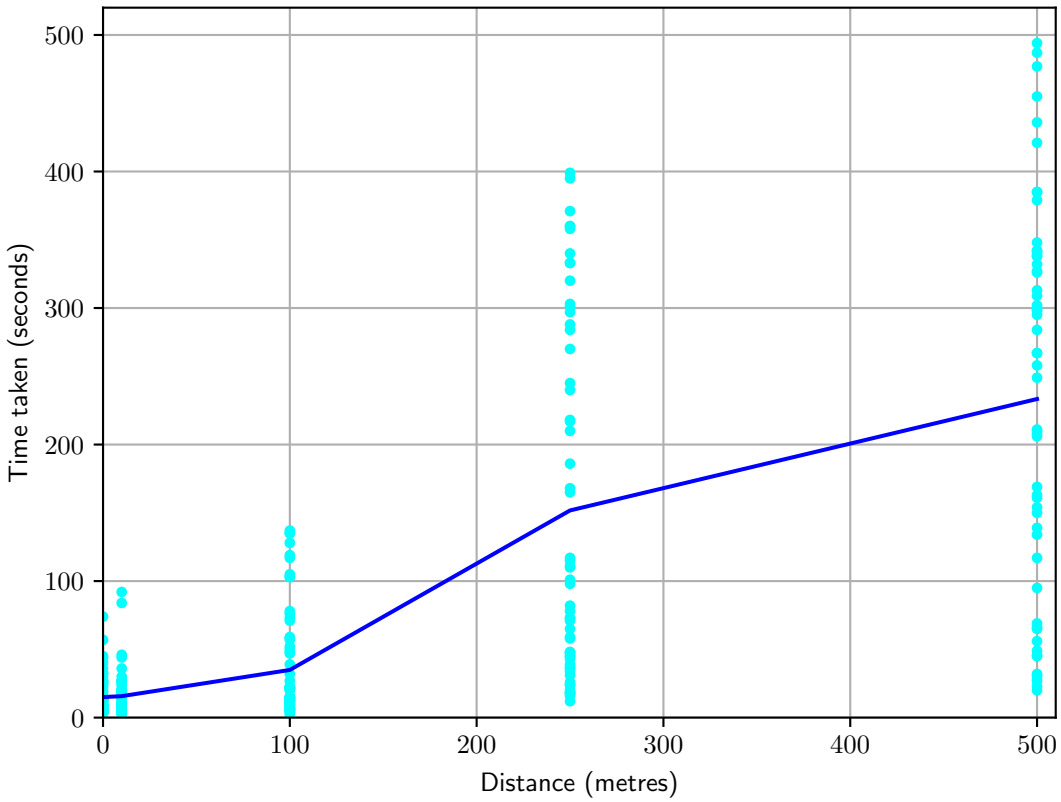


Figure 4.3: The time taken to deliver messages over distance intervals

4.1.2 Bandwidth

Testing the bandwidth of the system was done with the aim of working out how many messages the system can cope with. In reality, it is unlikely the MANET will need to process as many messages as in this test. Bandwidth testing was conducted between two nodes, with one node generating messages at a set rate and broadcasting them to the other node. Messages were generated at an increasing rate until the number of messages received by node B plateaued. Figure 4.4 shows the setup for this test.

As shown in Figure 4.5, the MANET has a relatively low bandwidth, with plateauing at around 0.35 messages transferred on average each second, equivalent to taking around 3 seconds to pass one message from node to node. To improve the bandwidth of the system, I could decrease the time between discovery ('hello') packets sent by each node.

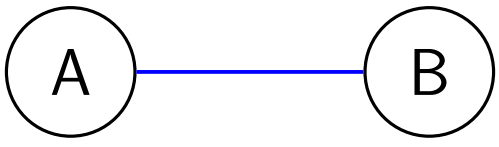


Figure 4.4: The network used for bandwidth testing

4.1.3 Percentage of Packets Delivered

Throughout testing, 100% of packets were delivered. This is best illustrated by the test with five nodes. To run this test, I recruited four volunteers to move around a large field, approximately 500 x 500 metre area. This allowed nodes to move in and out of range of each other. Each node had a

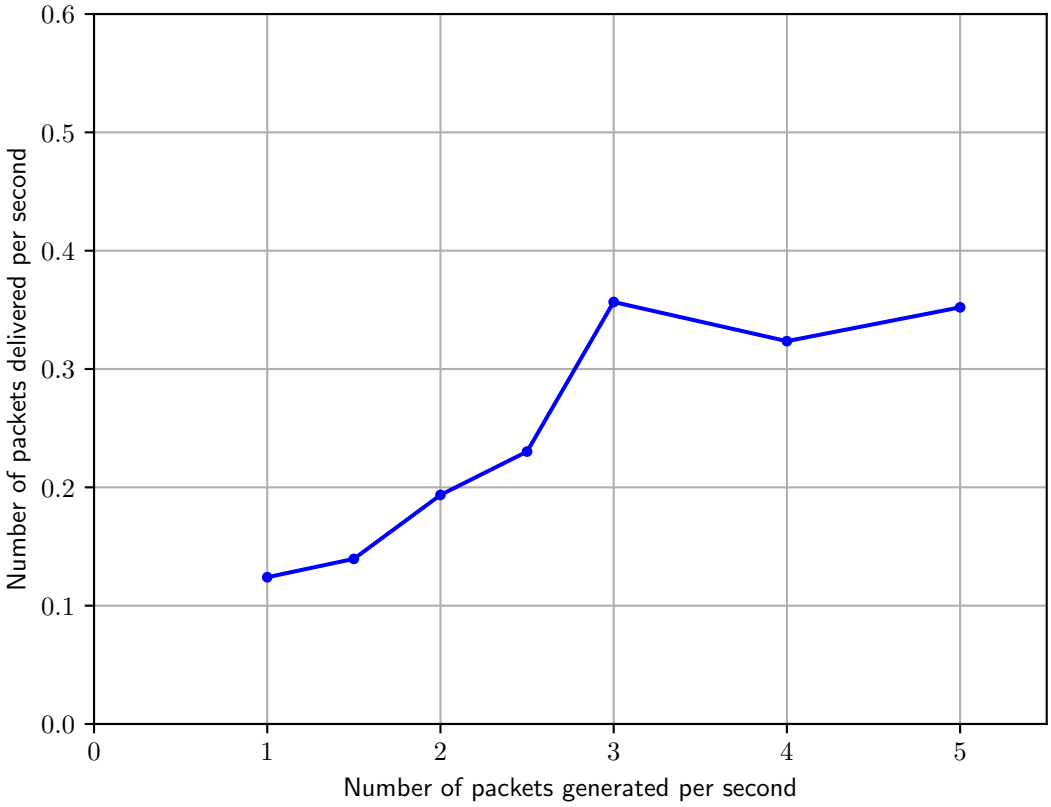


Figure 4.5: The bandwidth of a fully connected two node network with an increasing number of messages generated each second

0.1 probability of generating a message, with up to 30 messages being generated. The nodes moved around this test area for approximately 15 minutes. This is a reduced version of the evaluation performed in Vahdat and Becker's paper [10], where 50 nodes were simulated within a 1500 x 300 metre area.
While this test was run three times, only two of the results can be used due to a loose connection meaning a node did not work on the second run. In both these tests, 100% of packets were delivered.

4.1.4 Partition Testing

Partition testing was conducted with four nodes in two pairs. Each pair was set up to communicate with each other, then brought into range of the other pair. Figure 4.6 shows the starting (above) and finishing (below) states of the network.

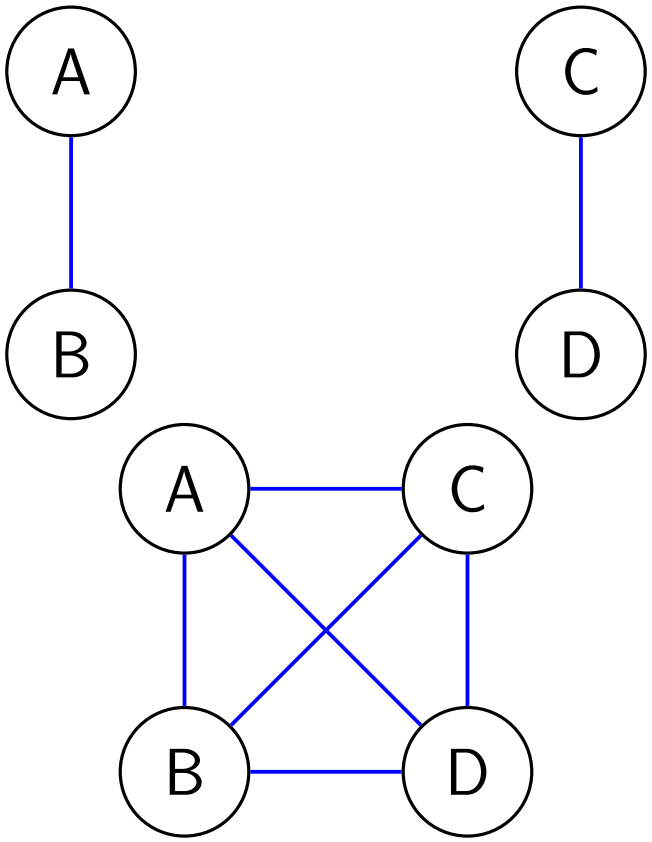


Figure 4.6: The network used for latency testing

4.1.4.1 Asymmetric

Asymmetric partition testing consisted of the {A, B} pair holding 30 messages before partition. After partition, the {C, D} pair had all messages. Over 2 partitions, the average time taken for all nodes in the network to hold the same messages was 106.5 seconds. Given the approximate speed of a rowing boat is 15km/h, and the radio range is 500 meters, this would allow two boats to be in range for 120 seconds, just enough time for the maximum number of messages to be transferred between two boats.

4.1.4.2 Symmetric

Symmetric evaluation of the network gave the two pairs of nodes different sets of messages. The average time for nodes to receive all 30 messages was 130 seconds, 23.5 seconds greater than an asymmetric partition.

Run	Time for Node C (seconds)	Time for Node D (seconds)
1	50	135
2	78	44

Table 4.2: Time taken for all messages to be received in asymmetric partitions

Node A	Node B	Node C	Node D
87	159	145	129

Table 4.3: Time taken in seconds for all messages to be received in symmetric partitions

4.2 Evaluating the System

4.2.1 On Water

The water evaluation was conducted last. I was concerned that there might be an accident and the project become water damaged or fall into the river, so all trials before this were conducted on land. The system did work on rowing boats, notifying rowers of other boats with the technology and of registered obstacles.
I examined the GPS tracking on the water as I was concerned that bridges or similar obstructions to the sky would interfere with the GPS tracking on each boat. This would be of particular concern as a bridge could be considered an obstacle. Figure 4.7 shows the GPS trace underneath a bridge. While the error caused by the bridge is minimal, it could be improved by applying a Gaussian or Kalman filter, as done in other GPS tracking for rowing boats. An issue this experiment inadvertently highlighted is that the trace does not appear to be accurate, showing the boat moving along the ground at some points. When comparing this to the GPS points taken in later experiments, this does not seem to be a systematic error, but rather a random error in the placement of the GPS in that outing. This suggests that an obstacle may be recorded in the wrong place due to an inaccurate GPS reading. To evaluate the system I went rowing with a volunteer, each of us with a device on our

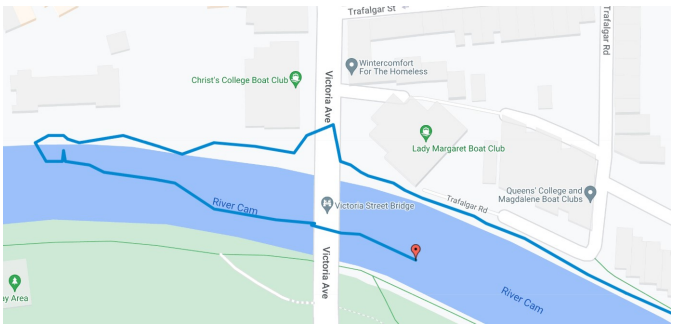


Figure 4.7: The GPS track taken by a node, shown on Google Maps [19]

boat. We ensured that the boats were notified when they were approaching each other. We found this notification to come at around 10 metres between boats. However, when the boats are rowing alongside one another, with no chance of collision, they continually alert the user to the other boat.

The nodes also transfer knowledge of obstacles between them. To test this, one boat went further up the river and registered a 'new obstacle' at a pre-agreed point (shown in Figure 4.8, opposite the Goldie Boat House) by pressing the button on the device. The first boat then returned to the second, which moved towards the potential obstacle. This gave a time delay between the boats approaching the obstacle, showing that messages are propagated through the network to warn other boats of the obstacles. The node warned the other boat of the obstacle.

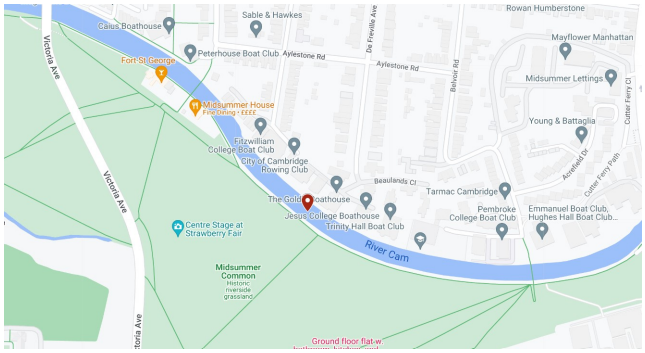


Figure 4.8: The registered obstacle as a red marker on Google Maps [19]

4.2.2 Qualitative Evaluation

For a qualitative evaluation, I surveyed the rower in the other single. While their overall impression was positive, the survey highlighted extra work that needed to be done to the user interface. They pointed out that the buzzer, even if the bug for notifying users about boats they are rowing alongside is fixed, might be a violation of rules about noise during certain times, so it could include another button to turn off the buzzer if needed.

There are some limitations to the system due to its design. The format of the packets has meant that there can be at most 255 nodes in the system, as there are two bytes reserved for the node address and address 0 is reserved for broadcast. The number of obstacles stored in memory is limited by the available memory on the chip, 1.9MB as 1MB is dedicated to the code running the node. Assuming each obstacle takes 10 bytes, 95000 obstacles can be stored on a device. This is effectively infinite. However, the message buffer is limited to 30. This limit on the number of messages a node can propagate is placed there to allow the unique ID for each message to fit into a single 64 byte packet, but means that the maximum number of obstacles a node can pass to a neighbour is 30.

Conclusion

5.1 Summary

In the introduction, I highlighted the issues around crashes in rowing boats and the need for a system to help prevent such crashes. This dissertation has presented such a system, propagating knowledge about obstacles through a network of rowing boats using the Epidemic routing protocol.

There have been other approaches to this problem. ROWCUS, a company that has "decided not to pursue" their system any further, used radar to attempt to solve this problem. This technological approach has the benefit that no person has to mark the location of an obstacle to be warned of it but is likely to miss some obstacles, particularly those submerged underneath the water, arguably the most dangerous obstacles as the users cannot see them.

5.2 Achievements

Three success criteria were laid out as part of the preparation. This project has met and exceeded the success criteria, implementing high and low priority messaging within Epidemic routing, thoroughly evaluating the network and conducting a review of the application layer.

As part of this project, I have contributed to open source libraries

What have you done? In this project I have

The evaluation shows that. This project is weak in. Low bandwidth

5.3 Reflections

The preparation chapter lays out the choices I made with respect to the software development methodology, tools and programming language. These choices were all found to be effective in designing and building the system. It was particularly helpful to have a plan that split the project into stages with defined deliverables at each stage.

While the research phase is designed to cover broad parts of the area in order to find the best solution to a problem, I spent too much time researching various potential routing protocols. With the benefit of hindsight, it would have been beneficial to settle on the Epidemic routing protocol earlier in the process.

There were both upsides and downsides to implementing the project in hardware. It taught me huge amounts about serial communication protocols and other low level implementation details. It also allowed for a more holistic approach to the evaluation, using the system on rowing boats. However, it is undeniable that the low level details made some aspects of the project more difficult, particularly when it came to debugging a problem. For instance, I soldered an antenna to a radio module in such a way that caused errors in the received packets, a problem I did not diagnose until I swapped the module for another one.

5.4 Future Work

There is future work that could be conducted in relation to both the networking and application components of the project.

Some of the networking extensions were drawn from my research into networking protocols other than Epidemic. For instance, as in GPSR, the GPS location could be used, either by only contacting nearby nodes to increase the probability that an anti-entropy session is successful. The location could also be used to prioritise sending messages about new obstacles to nodes that are near these obstacles. A metric for transmission quality, in radio communication the received strength signal indicator (RSSI), to influence whether an anti-entropy session was initiated.

The application component could be further improved by taking the direction of the rowing boat into account when notifying the user of an obstacle. The GPS could be further improved by applying a Gaussian or Kalman filter to reduce the noise in the location, particularly under bridges. Additional improvements could be made from the suggestions of users, such as a switch to mute the buzzer.

5.5 Other applications

Such a system could be utilised for other applications. The most obvious is collision avoidance for other water sports, such as canoeing and kayaking. The MANET within the system could easily be ported to use in autonomous vehicles. There are more avant-garde applications for this system, for instance within Bumps to decide if boats have collided, removing human bias.

Bibliography

[1] S. Ramskill. (2023) Tweet by @lliksmar. [Online]. Available: https://twitter.com/lliksmar/status/1637147870682906624?cxt=HHwWgIC8zY_UqLgtAAAA

[2] J. Jubin and J. Tornow, "The darpa packet radio network protocols," 1987.

[3] D. Beyer, "Accomplishments of the darpa suran program," 1990.

[4] P. Nucholas and K. Hoffman, "Computational challenges of dynamic channel assignment for military manet," 2015.

[5] R. Davidescu and N. Eugen, "Ad hoc networks for the autonomous car," 2017.

[6] Y. Jin, X. Tan, W. Feng, J. Lv, A. Tuerxun, and K. Wang, "Manet for disaster relief based on ndn," 2018.

[7] R. Pi, *RP2040 Datasheet A microcontroller by Raspberry Pi*, 2022.

[8] A. Moore, "Computer networking slide set for ib computer science," 2022.

[9] M. Taylor, "Equipment on the cambridge women's boat," 2013.

[10] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," 2000.

[11] ROWCUS. (2022) Rowcus | rearview radar for collision avoidance. [Online]. Available: https://www.rowcus.com/

[12] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. (2001) Agile manifesto. [Online]. Available: https://agilemanifesto.org

[13] ——. (2001) Agile principles. [Online]. Available: https://agilemanifesto.org/principles.html

[14]

[15]

[16]

[17]

[18]

[19]

Appendices

A – Guide to Building a Node

B – Evaluation Plan

Part II Project - Plan for Evaluation

March 2023

Overview

The evaluation for the mobile ad-hoc network (MANET) will be split into two parts – first the evaluation for the pure MANET and a whole system evaluation. The evaluation of the MANET will examine the implementation of Epidemic, the delay tolerant networking protocol will be checked for correctness and performance. The system evaluation will look at the MANET in the context of the use case, on the water.

All data will be logged on the node in a CSV file with columns
`Node Address, Time, Time Since Node Startup, Event Type, Event Information`
where the Event Information varies with the event being logged and may contain the GPS location and message keys. This data will then be analysed on my device using Python code.

Evaluating the MANET

These tests be conducted in a field, as this will give an outdoor environment similar to the use case, but I will be able to better control the conditions the node is in. They will use four nodes, as this is the maximum number of nodes we can accurately calculate time for. Two nodes will use GPS to find the current time and two will use the serial connection to laptops to calculate the time. These tests will be conducted first on a small scale, with short tests using a small number of nodes, to ensure the tests can be run. After this has been confirmed, the tests will be run for a longer time with the maximum number of nodes.

The tests can then be compared to the evaluation in the initial epidemic paper [3], which simulated the nodes with 50 mobile nodes in a 1500 x 300 m space using the Monarch extensions to the ns-2 packet-level simulator rather than hardware as I am doing. Evaluation metrics examined included message delivery latency, delivery rate, the average and maximum number of hops a message took to get to a node.

The first test will be the percentage of packets delivered in the four node network. This will be time limited (i.e. if a message is not received in x minutes, it is considered undelivered). Nodes will randomly generate a new message every second, for a total of 25 messages, mirroring the structure of testing used in Vahdat and Becker's paper [3]. The nodes will be in a box of area $20m^2$ with the range of the radios reduced to approximately $5m$ to simulate the environment in which the system will be used. The nodes will move constantly.

Next, the transfer delay will be measured. This will be the average time taken for each message to be delivered, and will use the same setup as percentage of packets delivered testing.

Finally, the time taken to propagate messages after partition will be measured. This will include one-sided, asymmetric partitions, where only one set of nodes has messages the other set has not seen. It will also include symmetric partitions, where both sets of nodes have messages the other set has not seen. The number of unseen messages will be increased and each test will be run five times.

Figure 1: The setup and axes for delivery rate and transfer delay

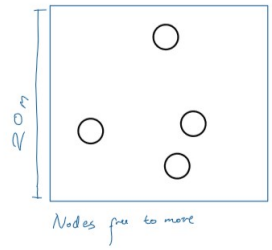
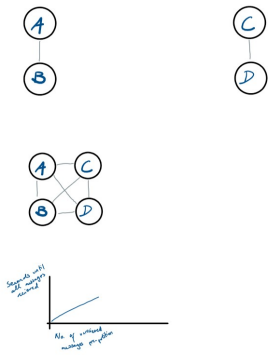


Figure 2: The setup and axes for partition testing



Evaluating the System

This will be performed on the water, the environment the MANET will be used in, after a 'dry run' on land to ensure there are no obvious flaws with the system. To examine the use of the system and how long it takes messages to arrive at other nodes, a well known obstacle has been selected. This is the red buoy at 51.48236626931181, -0.22641424521527762, shown below [1]. Using the time given from the GPS chips, we can then map the locations and time since the obstacle was added that other nodes receive messages about the buoy. This experiment will be run multiple times to gather a sufficient volume of data about the propagation of new obstacles.

Additionally, this will allow me to look at the behaviour of users when adding a new obstacle then tweak the MANET to fit. While there is a ground truth about the location of an obstacle, it is likely that most users will not be directly above the obstacle when they log it.

Figure 3: The location of the ‘red buoy’ obstacle [1]

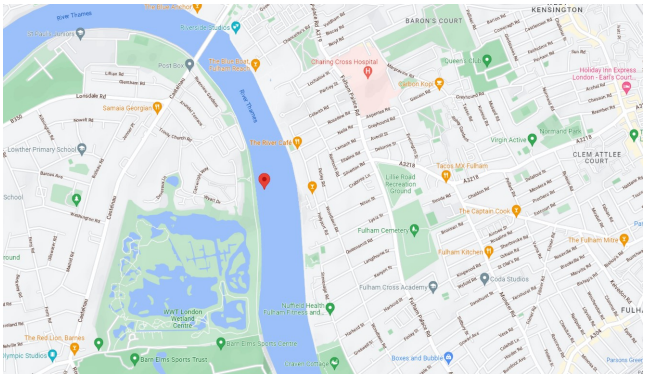


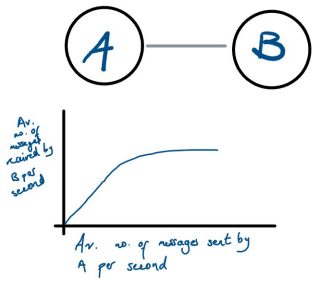
Figure 4: An image of the ‘red buoy’ obstacle [2]

Extensions

If there is sufficient time, further evaluation can be performed on the MANET. This will be structured in a similar way to the tests in the 'Evaluating the MANET'. Bandwidth will be tested in a fully connected network, where the number of messages per second transferred between two nodes may be found by generating random packets at set intervals at one node (node A shown below), and seeing how many are passed to another node (node B below). This test could then be performed with both nodes generating and receiving messages, to examine bandwidth on a two way connection.

Further extensions to the system evaluation will include a questionnaire for those who use the

Figure 5: The setup and expected graph for testing bandwidth



device, covering a range of users, including both coxes in coxed boats and rowers in coxless boats.

References

[1] Google Maps. 51.4823, -0.2264. [Online] Available at: https://www.google.co.uk/maps/place/51%C2%B028'56.5%22N+0%C2%B013'35.1%22W/@51.4820341,-0.2295313,15.5z/data=!4m4!3m3!8m2!3d51.4823663!4d-0.2264142!5m1!1e4 [Accessed March 2023]

[2] Riddell-Webster, A. Red buoy. Taken March 2023.

[3] Vahdat, A, Becker, D. Epidemic Routing for Partially-Connected Ad Hoc Networks. Published 2000. Duke University.

C – Progress Report

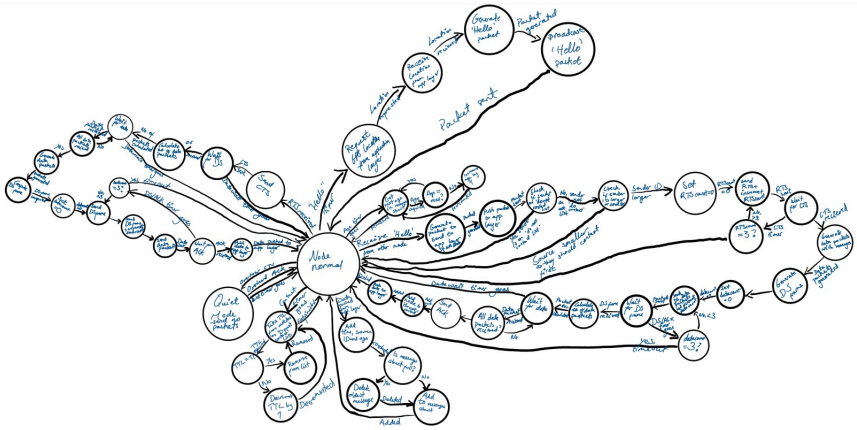
Part II Project - Progress Report

April 20, 2023

Name: Alex Riddell-Webster
College: Murray Edwards
Email Address: ahr38@cam.ac.uk
Director of Studies: Luana Bulat
Supervisor: Matthew Ireland
UTO: Professor Jon Crowcroft
Overseers: Ferenc Huszar and Andreas Vlachos
Title: A MANET to Facilitate Collision Avoidance in Rowing Boats

My project, a mobile ad-hoc network (MANET) to facilitate collision avoidance in rowing boats, attempts to reduce the frequency of rowing boat collisions. The technical core of the project is the Epidemic routing protocol [1], modified to include medium access control based on the Multiple Access with Collision Avoidance for Wireless (MACAW) protocol [2]. Both medium access control and Epidemic are implemented in the by the same state machine, to simplify the implementation and prevent work being repeated – a waste of the limited resources on the Raspberry Pi Pico [3]. The initial state machine is shown in Figure 1, although it has been changed during implementation. Most notably, data send (DS) packets have been removed and the information being put into the data packets to reduce the number of packets sent.

Figure 1: Network state machine

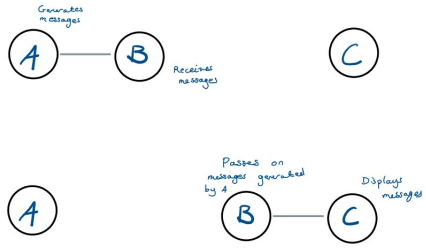


Construction of the MANET is going well. It has been constructed in hardware, working with the Raspberry Pi Pico, an ARM-based microcontroller without an operating system [3] and RFM69 radio. I have finished the networking machine so am currently tweaking and evaluating the network state machine while finishing the application machine. As the network has a physical implementation, I intend to test the network on rowing boats, the environment it would be used.

To ensure the network was delay tolerant, I ran a test with three nodes, A, B and C. At the start, nodes A and B were in range of each other and node C was out of range. I set node A up to generate random messages every 40 seconds, with a time to live (TTL) greater than 2 so the messages would survive for two 'hops' across the network. I then moved node B out of range of

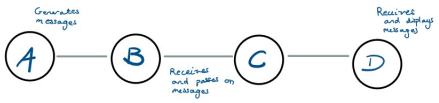
node A and into range of node C, which then displayed any messages it received so I could check that they were the same as those generated by A, with a reduced TTL. Figure 2 shows the setup.

Figure 2: Using three nodes to check the network is delay tolerant



Another test involved four connected nodes, A, B, C, and D, where the transmit power of each node was significantly reduced, so each node had at most two connections. Node A generated messages, and I checked to see if D received them. I will use a similar set up in the future to test the percentage delivery and latency of packets. The setup is shown in Figure 3.

Figure 3: Using four nodes to check the network can transfer packets over several links



The most significant obstacles have been in radio communication. The FIFO buffer on the RFM69 was occasionally being overwritten as the controlling library was not clearing the FIFO. I modified the library to clear the buffer and allow the sending of fixed length packets. Changing to fixed length packets (64 bytes, the maximum length of the FIFO) allowed for more reliable communication. Additionally, CircuitPython (the language in which AdaFruit's libraries are written) does not support interrupts. To work around this, I poll to see if a condition is met when a corresponding timer elapses.

Given the work completed so far, I am two weeks behind the timetable laid out in October. As I am working on the application machine and evaluation of the network in parallel, the project will likely be back on timetable by mid-February.

The remaining work is first to finish the application machine and evaluate the MANET. Evaluation metrics will include the percentage of received packets, transfer delay and variance, and the time taken to propagate messages in a previously segmented network. Finally, I will pull the application and network machines together, running them on the two cores in the Pico, with concurrency control over key data structures and the GPS chip. A concern here is the Adafruit Blinka libraries allowing interoperability between CircuitPython and MicroPython [4], given the errors and incompleteness found in other libraries.

References

[1] Epidemic Routing for Partially-Connected Ad Hoc Networks. Vahdat, A, Becker, D. Duke University. 2000.

[2] MACAW: A Media Access Protocol for Wireless LAN's. Bharghavan, V, Demers, A, Shenker, S, Zhang, L. ACM SIGCOMM Conference. 1994.

[3] RP2040 Datasheet A microcontroller by Raspberry Pi. Raspberry Pi Ltd. 2022.

[4] GitHub - adafuit/Adafruit_Blinka: Add CircuitPython hardware API and libraries to MicroPython & CPython devices. https://github.com/adafruit/Adafruit_Blinka. Adafruit Industries. GitHub. Accessed January 2023.

D – Project Proposal

Part II Project – Project Proposal

October 14, 2022

Preliminary Information

Name: Alex Riddell-Webster
College: Murray Edwards
Email Address: ahr38@cam.ac.uk
Director of Studies: Luana Bulat
Supervisor: Matthew Ireland
UTO: Professor Jon Crowcroft
Overseers: Ferenc Huszar and Andreas Vlachos
Title: A MANET to Facilitate Collision Avoidance in Rowing Boats

Introduction

My project will implement routing in a Mobile Ad-Hoc Network (MANET).

Routing protocols find a path from a source to one or more destinations destination within the network. Different routing protocols optimise different parameters, and are better suited for different network topologies and applications [1].

A MANET is characterised by wireless nodes, a frequently changing network topology and no reliance on pre-existing infrastructure. They are decentralised and therefore have no single point of failure [2]. MANETs have a large range of uses, from the military [3] to facilitate communication, to autonomous vehicles [4] or disaster relief scenarios [5] to gather and move data across locations where previously existing infrastructure has been destroyed.

My project wishes to use a MANET to share a set of locations throughout a set of rowing boats, in order to facilitate collision avoidance. Collisions in rowing can cause damage to both rowers and their equipment. This project's motivation is to avoid rowing boats colliding with each other and with other obstacles. A radar and AI based obstacle detection system exists [6]. However, to the best of my knowledge, collision avoidance has not been attempted by networking boats together. My project will represent each boat as a node in a network. Each node will store a set of locations the user should be warned about; passing location data throughout the network will be the technical core of the project.

My project will be implemented in hardware, in the real world. In general, networking protocols can be implemented in simulation or in hardware. Depending on the nature of the simulation environment, it might not be possible to use exactly the same code in the simulator as in the real hardware. Due to the time constraints on a Part II Project, I intend to implement my project only in hardware.

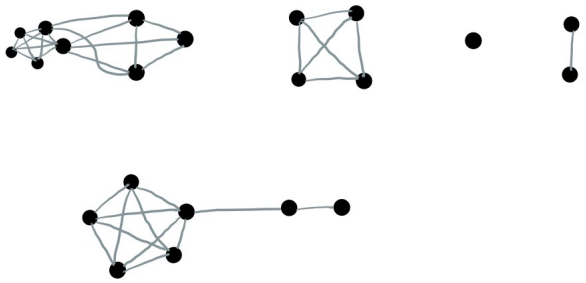
As stated in the cover sheet, Human Participants will be used to help test and evaluate the project. This will comprise a few volunteers to row boats, allowing the network to be run on the water. These volunteers will all be members of Cambridge University Boat Club, able to safely row a boat and navigate the river where the network is being tested.

Structure

The first part of the project will be dedicated to research, looking at the Epidemic routing protocol. Epidemic was chosen as it is a delay tolerant routing protocol, and best fits the likely topology of networks generated by rowing boats. There is a high chance of partitions in these networks. However, the nodes in the networks will be highly mobile, meaning that data can still be transferred through the network through exploiting this mobility. A potential topology for the network, generated from looking at satellite images of a 5.5km stretch of the river Thames [7], is shown in Figures 1 and 2, both on a map and as an abstracted topology.

Figure 1: Rowing boats, potential obstacles and assumed connections marked on a Google Maps map of the river Thames

Figure 2: The network of rowing boats in an abstracted form



The first research phase will also decide on any changes that need to be made to the routing protocol to make it better suit the application. Finally, the first research phase will refine the evaluation metrics needed for the project.

The second research phase will look at microcontrollers, particularly the Raspberry Pi Pico. It will also consider multi-threading on the Pico and in CircuitPython and how this can be exploited to most effectively implement my Part II project.

After the research phases, I will implement point to point communication between two neighbouring (in radio connection range) nodes in hardware.

The next part of the project will implement broadcast and controlled flooding routing between

at least three nodes. A packet format will be defined as part of this, as flooding will form the start of the Epidemic routing protocol.

After a flooding protocol has been implemented, I will build on it to implement the Epidemic routing protocol. The code will be written in a two week block, then tested on hardware in a third week.

The application layer will then be implemented. This will ensure that the information passed through the network can be used. The application layer should warn the user when they are approaching an obstacle, and allow the user to add obstacles, which are then passed to the routing layer to be propagated through the network. My project aims to keep the routing and application layers separate for ease of construction, testing and evaluation.

The hardware will then be tested, tweaked and evaluated. Correctness will be evaluated first on land, likely in a field, where analysis of the network is easier and larger numbers of metrics can be examined than in the use environment. Performance will be evaluated in the use environment - on rowing boats on the water. I intend to evaluate the routing and application layers separately. Evaluation of the network will likely consider connected and partitioned instances of the network separately. Evaluation is likely to look at the time taken for the network state to be flooded through the nodes, and routing tables then updated, as well as a packet loss ratio [8]. Further evaluation would conduct some case studies, tracking a packet through the network and ensuring no unnecessary latency is added. The power consumption of each node could also be measured, giving a proxy measure for the traffic passing through each node. Due to time constraints, I consider these further evaluations to be extensions.

Success criteria

There are three success criteria I will hold for my project:

1. The Epidemic routing protocol is implemented on the network
2. An application layer to demonstrate the utility of the network has been implemented
3. An evaluation of the performance of the network has been carried out

Extensions

The success of my project will be defined by completion of the core criteria listed above. If there is time, I have set further challenges:

1. Case studies on the path and timing of individual packets are performed
2. The network is further evaluated by examining the power consumption of individual nodes as a proxy metric for traffic passing through a node
3. The User Interface of the device is evaluated
4. The application layer is further enhanced, using heuristics and extra data such as angle of attack from GPS and combining sensor data
5. The routing layer is further enhanced by passing additional data, such as location awareness, to the routing protocol

Plan of Work

Start of Block	End of Block	Block Length	Notes	Work to be Done	Milestones
14/10/2022	21/10/2022	7		Research - how to implement Epidemic protocol, evaluation methods used for ad-hoc networks	Develop a greater understanding of the Epidemic protocol and a plan for implementing it, create an evaluation plan
21/10/2022	28/10/2022	7		Learning how to use the microcontroller and boards	Ensure all the necessary hardware is available, develop a greater understanding of the Raspberry Pi Pico and CircuitPython
28/10/2022	04/11/2022	7		Start to work with the hardware - implement point to point communication between two nodes	Two nodes can send point to point messages
04/11/2022	18/11/2022	14	07/11 - Robotics Assignment 1	Implement controlled flood routing	Messages are flooded between at least three nodes
18/11/2022	02/12/2022	14	18/11 - 4s head; 28/11 - Robotics Assignment 2	Implement Epidemic routing protocol	Routing state information is shared between at least two nodes, Epidemic is implemented
02/12/2022	10/12/2022	8		Test, tweak and debug Epidemic implementation on hardware	Epidemic is implemented on hardware
10/12/2022	26/12/2022	16	14/11 - Trial 8s; Christmas	Time off	-
26/12/2022	02/01/2023	7	01/01 -> 11/01 - Camp	Implementing application layer - read location data from GPS and warn user when approaching known obstacle	The device warns the user when they are approaching a known obstacle
02/01/2023	16/01/2023	14	01/01 -> 11/01 - Camp	Implementing application layer - transfer data between the application and routing layers	The application layer is implemented on hardware
16/01/2023	23/01/2023	7		Tweaking the hardware, testing point to point links on land	The hardware runs on land, finish proof of concept
23/01/2023	30/01/2023	7		Water testing and tweaking	The hardware is implemented and run in the application environment (water)
30/01/2023	03/02/2023	4	03/02 - Cybercrime 1	Write progress report and presentation	Progress report and presentation
03/02/2023			**Progress report and presentation**		
03/02/2023	14/02/2023	11		Evaluation and tweaking on land	The hardware is evaluated for correctness on land
14/02/2023	21/02/2023	7	17/02 - Cybercrime 2	Evaluation and tweaking on water	The hardware is evaluated for performance on water
21/02/2023	07/03/2023	14	03/03 - Cybercrime 3	Dissertation - plan and bullet point what will be said	Dissertation bullet point form (first draft)
07/03/2023	21/03/2023	14	17/03 - Cybercrime 4	Dissertation - write out preparation and implementation	Dissertation has implementation and preparation written out
21/03/2023	04/04/2023	14	26/03 - Boat Race	Time off	-
04/04/2023	18/04/2023	14		Dissertation - write introduction, conclusion, evaluation	Dissertation fully written, sent to supervisor to proofread (second draft)
18/04/2023	02/05/2023	14		Dissertation - Take on criticism, add references and appendices	Dissertation - final draft
02/05/2023	12/05/2023	10		Contingency	-
12/05/2023			**Final deadline**		

Starting Point

I have a little experience with networking and routing protocols. My experience is limited to the Part IB networking module, although it is being expanded by the Part II Principles of Communications module and my research. I will need to add to my knowledge of networking and routing protocols.

I have previous experience using Raspberry Pi single-board computers with AdaFruit boards. I have no previous experience with microcontrollers. I will need to improve my knowledge of microcontrollers to complete this project, something I have set aside time for in my Plan of Work.

Resource Declaration

I plan to use my laptop to implement, evaluate and write up the project. It has a comprehensive system of backups through OneDrive and disk images. A backup of the project will exist with Git version control, hosted on GitHub. My own hardware, including Raspberry Pi Picos, breadboards and AdaFruit radio and GPS modules will be used to develop and implement the project.

Libraries to interface with the AdaFruit boards are written by AdaFruit in Circuit Python, and in my experience tend to be robust, although they occasionally contain bugs. If necessary, I can fork the code and implement bug fixes.

My project will partially rely on the correctness of routing protocols, work that others have already published. [9]

As the project has a real-world implementation, I have permission from Cambridge University Boat Club to test devices on their boats.

References

[1] 2022 Part II Lecture Notes on Principles of Communications. Crowcroft, J. 2022.

[2] Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Corson, S. and Macker, J. Network Working Group. 1999.

[3] MILCOM'09: Proceedings of the 28th IEEE conference on Military Communications. Richardson, K, Jimenex, C, Stephens D. 2009.

[4] AD HOC Networks for the Autonomous Car. Davidescu, R, Negrus, E. IOP Conference Series Materials Science and Engineering. 2017.

[5] MANET for Disaster Relief based on NDN. Jin, Y, et all. IEEE. 2018.

[6] DEVICE | rowcus. https://www.rowcus.com/device. ROWCUS. Retrieved 07/10/2022.

[7] Hammersmith Bridge - Google Maps. https://www.google.com/maps/place/Hammersmith+Bridge,+London/@51.4883478,-0.2302753,17z/data=!3m1!4b1!4m5!3m4!1s0x48760fb3f5c78f85:0x932267a238m2!3d51.4883478!4d-0.2302753. Google. Retrieved 10/10/2022.

[8] SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS AND NEXT-GENERATION NETWORKS Internet protocol aspects – Quality of service and network performance. International Telecommunication Union. 2011.

[9] Epidemic Routing for Partially-Connected Ad Hoc Networks. Vahdat, A, Becker, D. Duke University. 2000.