Project Mesh Network

DESIGN DOCUMENT

NetStruction

Team Number: SDMAY19-21
Client: Danfoss/Radek Kornicki
Advisor: Craig Rupp

Team:
Collin Vincent -- Dev Ops, Network Engineer
Cody Lakin -- Software Engineer, Data Acquisition
Gage Tenold -- Engagement Lead, Test Engineer
Colton Smith -- Project Manager, Hardware Integration
Will Paul -- System Architect, Hardware Integration
Ryker Tharp -- Documentation Review, Backend Developer

Team Email: sdmay19-21@iastate.edu
Team Website: http://sdmay19-21.sd.ece.iastate.edu

Revised: December 2018 / Version 2.0

Table of Contents

List of Tables and Figures

List of Symbols and Definitions

- CAN Bus: Connector Area Network Bus. A serial bus protocol for reading data from (typically) vehicles.
- OBD2: On Board Diagnostics (version) 2: A diagnostic connection and port standard to a vehicle's computer. Accesses data like speed, coolant temp, RPM, throttle position, etc.
- Pi/Pis: Singular and plural shorthand for Raspberry Pis.
- Ad-hoc network: A network where devices can connect momentarily to one another without the use of a router. [1][2]
- J1939: a formatting for CAN bus used to send heavy machinery data.[5]
- SQLite: A sql database that is intended to be used in applications and has no server client model.
- vCan: Virtual CAN. A piece of software for simulating CAN data.
- Central Hub: The heart of the Mesh Network, the access point for reading the collected data and where all data is ultimately directed towards.
- Electron: A software framework for creating desktop GUI applications. Applications that are built using this framework include: Slack, Discord, and WhatsApp.
- Mesh Network: A local network of connected nodes that connect dynamically to one another and transfer data back and forth.[3]
- Wifi: wireless radio wave base networking that is implemented based on IEEE standards.
- OSLR: Optimized State Link Routing, constructs a graph of the network to coordinate packet forwarding.
- SocketCAN: Drivers to interface PiCAN with the data.
- JSON: open standard file format for attribute-value pairs.

1 Introduction

1.1 Acknowledgement

The Mesh Network team would like to show appreciation to Danfoss for providing us with the expertise and hardware necessary for working on this project. We would also like to thank Craig Rupp for providing additional insight as our advisor and proposing a variety of useful technologies. As students we want to thank all involved for this opportunity to put our skills to use in a practical application, and to experience project development and management in a professional capacity.

1.2 Project Statement

Using a mesh network in a jobsite involving heavy equipment and/or a variety of vehicles is already a common practice. Collecting information and distributing it over a network allows managers to be better informed of the condition of their work which allows them to make better decisions regarding various operations [4]. Information such as vehicle fuel level, location, etc. can be monitored to optimize the workflow of the job. These networks are only applicable in areas that support internet connections. This projects goal is to give job sites that lack this internet connection and provide them with the same information collection so they can improve those locations as well.

With internet access, each vehicle can connect directly to a service to transmit its collected data to a point where it can be processed and redistributed to those users that can utilize that information. Our mesh network solution is going to have those vehicles connect to each other, rather than a global connection, and form a chain of communication connecting back to the users that require that information. In this way, work areas that don't have the conditions necessary for a data network can achieve the same benefits as those that do.

There is a market for companies that go out to under-developed countries or regions that don't have easily accessible internet/cell service. These companies are the target users of this project with our intention being to help them improve their operations through data collection and predictive services being performed locally. Users of this system will be able to install devices on their equipment that collect data and collate it into a hub where they can view that data in a meaningful format. Information such as fuel level for each vehicle can allow those users to optimize the paths that fuel delivery vehicles take which will decrease the slack time of each vehicle, decreasing the total time taken in a project. This is one of many aspects our users can take advantage of with this system.

1.3 Operating Environment

The operating environment will be the construction worksites that have limited access to network connections including cellular. Raspberry Pis will be used to facilitate the data collection and communication. These worksites can vary dramatically in size in terms of vehicles in use which has important design implications. Vehicles in use by the environment are varied in terms of operation data and importance. Work conditions within each environment can be quite different from other environments, resulting in different complications for each location.

1.4 Intended Users

This system is designed to provide a method for collecting and using information in an area with only the use of local networking. Therefore the intended users for this system are primarily companies/operations that are having to work in an environment where there is little to no internet or cellular service. An example of this would be in an underdeveloped country where a company may be creating roads to connect places of interest. In this scenario there wouldn't be reliable connection to the internet without the use of a satellite phone. To that end the network will connect the vehicles being operated by the company in this area through wifi and share the information they are collecting with each other and take this information back to a hub. There a manager(s) can utilize this data to track resource expenditure, equipment condition, and work being performed.

1.5 Assumptions and Limitations

Assumptions:

- After the members of this team have graduated, the project will be handed to Danfoss to be maintained and improved.
- End users will be able to understand the information collected and its implications.
- End users will have a device that will have software installed on it to operate as the central hub.
- The system will be modifiable to fit a wide array of applications/work.
- Each device installed on a vehicle will attempt to collect data 24/7.

Limitations:

- The project will use a raspberry pi device.
- Since collecting actual vehicle data is impractical, this data will be simulated.
- The system must function on a variety of devices.
- Only one month of data per vehicle will be stored on each device.

- For our purposes, a range of at most 100 meters will be tested

1.6 Expected End Product and Deliverables

There are three major deliverables to this project. At the start of the second semester a functional prototype that demonstrates the technologies we have chosen work together to achieve the goals of this project will be delivered. Towards the middle of the second semester a Proven Concept will be delivered, which should have all major components of the system functioning and working in tandem. Our final product delivered at the end of the second semester before we graduate will be a Hand-Off version of the system, so that Danfoss can take the project and modify/improve it for use in the field.

- Prototype - January, 2018
    - Prototype will demonstrate the use of SQLite to store and distribute the information.
    - Each raspberry pi will be able to connect to each other and transmit data.
    - A central hub will be able to connect to any raspberry pi and collect the data.
    - The central hub will have a preliminary means to display the data to an end user.
    - A demonstration should be available to show the client the results of this prototype.
- Proven Concept - April 1st, 2019
    - Each raspberry pi will simulate field data and connect to each other device when necessary[1][2].
    - The central hub will have an complete front end application that displays the data in readable and meaningful manner.
    - Predictive analysis is being used to take the data collected and further assist end users in utilizing the information to improve their operations.
    - A demonstration of the complete system's facilities should be available to prove the results to the client.
- Hand-Off Version - May 1st, 2019
    - After all required aspects of the project have been completed, the focus will be to completely document the system and make it easily understood and configurable by the teams at Danfoss. If possible, devices should be interchangeable and the network should be flexible in terms of size, data, and device type.
    - All documentation will be prepared for transfer.
    - All code will be finalized and documented well.
    - Modification and manuals for both software and hardware information will be prepared and available.

2 Specifications and Analysis

2.1 Proposed Design

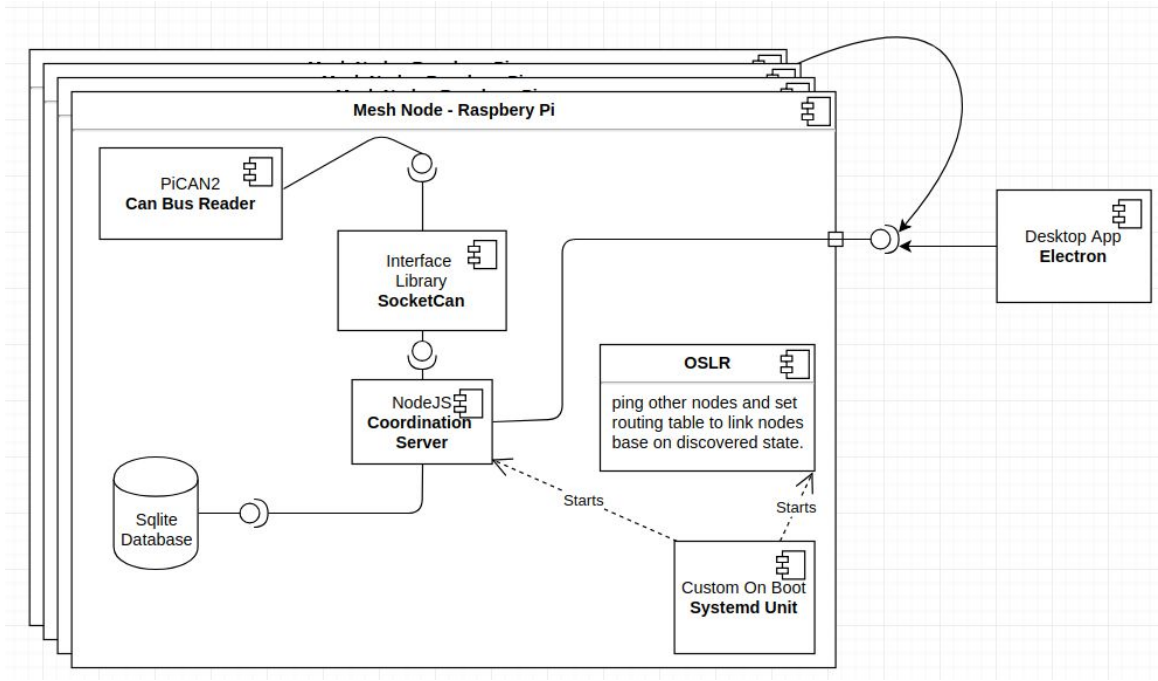

Figure 1: Component Diagram

Our Design centers around the Pis. We are using PiCAN2s to interface with the construction equipment. We then use the SocketCan library to making interfacing with the PiCAN easier. We are using sqlite to store all the data from the equipment. We will have a nodejs process that will read data from SocketCan and write it to the database, as well as function as an http server listening for api request from the network. The Manager will be running an electron app which will make request to the pis for the data. We are using Optimized State Link Routing protocol to allow for routing packets that require hops through our ad-hoc network[2]. This works by broadcasting a “hello” packet to all devices on the network and record what devices it has direct access to. Then all of the devices start broadcasting their personal view of the network graph. The devices forward other devices broadcast so that all devices connected eventually construct a complete network graph. It then calculates optimized packet routes between all nodes and adjust the linux routing table to have records of required hops.

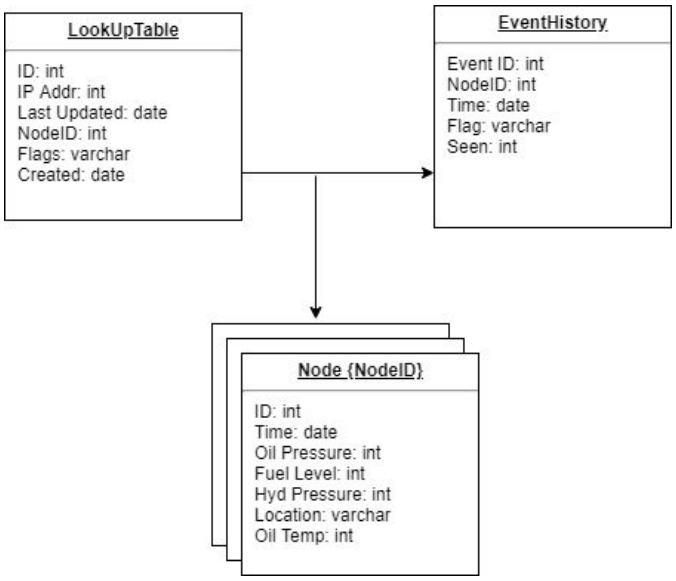

Figure 2: Database Schema

The database is designed to handle variable number of nodes and multiple sources of competing information. This is accomplished in the three section design, where there will be a lookup table containing information on the nodes, an event table recording incidents within the network(such as data spikes, inconsistencies, or a missing connection), and a set of tables that will contain sensor data for each node within the database. Database distribution across the network is achieved by ensuring each nodes contains the most up-to-date information it can access for each other node. By timestamping the sensor data that is processed, the system can transfer only the new data to other points in the network. The look up table contains connection information and organizes the set of Node tables. When a connection is established, this table will determine which node is connecting and where to store the new information, as well as indicating to the connecting node what new information is required. Event History serves two purposes, first to flag events within the system that are undesirable then alert end-users to their occurence, and second the records can be used to record network connection tendencies that can be used to improve the performance of the network.

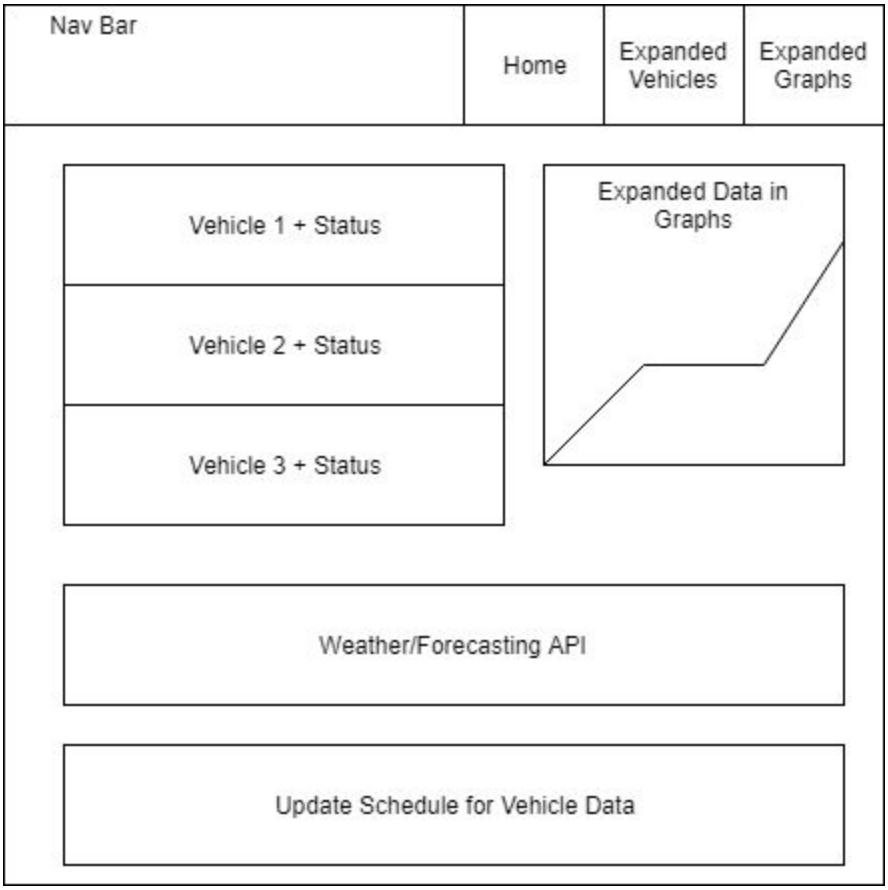

Figure 2: UI Mock Diagram

The front end of our application is built using electron. Data transmitted from the nodes around the network will be routed here and configured to be easily understood by a foreman on a work site. The data will be stored in JSON format and used to update the application in real time. The main features of the app will include: an updating table of all of the vehicles on the network and their status, data visualization features that allow you to look at the readouts of specific vehicles, a weather API that lets the current forecast be viewed (this was an addition the client asked us to put in for convenience sake), and an update schedule for when each vehicle should have a new update. The application will also feature push notifications for when certain flags are met to notify the crew when vehicles are at risk.

The main requirements of the project are:

- ability to support a large number of nodes within a network
- ability to add and drop connections rapidly
- support a network of devices with no internet connection
- nodes must collect at minimum five types of data concurrently
- data must be shared between all nodes within the network

2.2 Design Analysis

The diagram of our system in Figure 1 accounts for all of the functional requirements that our system will need as of writing this, though there is potential for expansion later on. In particular, our system is designed with reliability, data integrity, and scalability in mind. Since our system is a network of mobile nodes that will connect and disconnect from the network as time progresses, it needs to handle these changes in network configuration adaptively. Since our system is designed to utilize any node path back to the central hub, this requirement is handled well. With data integrity we have each node collecting the information of each other node with timestamps in order to send it back to the central hub. This satisfies data integrity since our system will be able to handle discrepancies because of the multiple storage locations, and the currency of the data will be handled by the time-stamping. Scalability is another major point in the system, since the operations of end users can have a considerable number of vehicles in use. To handle this our design allocates storage for each node separately and will use timestamps to prevent nodes from updating the same information twice. This prevents nodes from placing a large processing burden on other nodes, allowing the system to grow quite large in size.

There are some design concerns and unknowns concerning our design. The two major ones are availability and security. For availability the concern is due to the lack of internet connection. Since our nodes are attached to moving vehicles, the distance between pieces of the network isn't static. Depending on the strength of the signal we use to connect these nodes, machines may be out of network range which means they won't be sending the information they collect to other nodes. We have plans to test a pylon system to extend the range of the network, but that isn't part of the design yet. Security is the other major concern of our current design. When designing the system initially security wasn't a major concern. Other than adhering to IEEE standards for communication [1] and storage the design as it stands is lacking in the security department.

The Pros of our current design are-

- Raspberry Pis have plenty of supporting material online for working with CAN bus data.
- Making a custom Sql based solution allows for us to customize it to our exact specifications, handling data integrity and scalability.
- Utilizing the OSLR routing protocol allows to achieve the exact type of connection we want our network of nodes.
- Wifi is the most common type of connection type for mesh networks and has plenty of supplemental materials online that we can use.
- Because of data storage and transfer methods in the design the network can be scaled to large sizes.

The Cons of our current design are-

- We are still working with simulated CAN data for the time being, this could handicap us in the future if we can't make the switch to using real live data
- Electron may be a little overqualified for this design and could make the final web application more bloated than it needs to be
- By making our own database solution, we may have a lack of possible resources to look at for assistance when building it
- The expansive geographic nature of the network makes satisfying availability a concern.
- Security is currently low on the list of concerns, which may prove problematic later.

3 Testing and Implementation

3.1 Interface Specifications

Since the base of this project is that the network itself needs to be in working order we are going to use manual tests to ensure they connect to each other over the network. This will be done by simply monitoring the hardware to verify that when it is able to connect to the network it does so. Testing the database will be slightly easier as automated test cases can be used to ensure everything works as planned.
An important technical interface important to our project is CAN bus. It is a bus standard designed for vehicles. We must test the proper way to parse and store/send the data coming in from the CAN bus, as well as make sure the drivers and connection to the bus module are functional.

3.2 Hardware and software

We will be be using an OBD2 Emulator to test, assuming we can acquire one. This will give us an ability to input the appropriate live data into the system. In addition, or if we cannot get an OBD2 emulator, we can test the system using our own vehicles, by plugging in 3-6 of the Raspberry Pis into our cars and using the actual data from the cars as we drive.

Test cases developed for SQLite will be written in a mixture of NodeJS and Python. Nodejs will be used to generate the test data and simulate machines moving around and different data types coming in. The Python will provide the scaffolding for the actual test cases. It will contain what we should expect the system to do based on the defined test cases. Having automated test cases based on NodeJS and Python will help with project development because using them is a good form of integration testing. With each new feature developed we will ensure systems that were working before remain working as we integrate new features.

3.3 Functional Testing

Using a series of unit, integration, system, and acceptance testing we will verify that the functional requirements of the system are being met. Each test will be performed when the components reach a required stage a completion to ensure they can accomplish the tasks needed of them. After new tests are introduced, we will test previous tests to prevent any regression of the system. The major tests planned are as follows:

- To test the nodes collection ability, we are going to have them simultaneously take in five streams of data, simulating oil pressure, oil temperature, fuel level, hydraulic press, and gps location. This is a unit test that will be performed using vCAN to feed simulated data provided by Danfoss to the raspberry pis. We will be observing the correctness of data stored to the nodes and the speed at which it is processed.
    - Success condition: Each of the five sensor streams are interpreted concurrently and match the simulated Data.
    - Failure condition: All or some of the data from the streams are incorrect or missing.
- To test the nodes ability to connect to each other and the hub, we plan to have at minimum three devices running, and turn their connections on and off to test the system's ability to manage the connections between each device. This is an integration test that will be performed using raspberry pis. We will be turning off their wifi connection to simulate an out of range node then turning it back on to observe the connection speed and reliability.
    - Success condition: Each node is able to connect and disconnect from the network as expected.
    - Failure condition: Any node fails to connect initially, or fails to reconnect to the network.
- To test the central hub's ability to collate the information we will have at minimum three devices connected in sequence collecting and transferring data which we will make tests to verify that data is the same throughout each device. This is a integration test to ensure the network can transfer information distributively. Unit testing comparing the values between each system after the test is performed will check for accuracy of data.
    - Success condition: All three nodes data is successfully transferred to the central hub, and is accurate.
    - Failure condition: Some or all of the data is incorrect or missing on the central hub.
- To test the system's ability to dynamically add nodes we will have a single device connected to the central hub and then start activating and deactivating nodes in sequences that will be specified at a later time. This is a system test where nodes should be able to be added on the fly to the system. Testing this includes verifying the connection data of each node is stored and checking that no prior nodes are affected adversely.
    - Success condition: Nodes can be added to the system, and their connection information is stored correctly.
    - Failure condition: Nodes fail to be added to the system, and/or their connection information is stored incorrectly or causes other nodes to be adversely affected.
- All of these tests will be performed without the devices maintaining an internet connection which covers the requirements of low to no internet within the system. This is the main acceptance test attached to every component of the system.

3.4 Non-Functional Testing

Performance
Performance, in this case speed, is important to our project. Measurements from machines must reach the front-end within an amount of time that makes them useful. Performance will also be important in regards to keeping all data consistent, without bogging down the network.

Our performance tests will record time taken by each key operation over a large number of test runs. Each component will be tested with a variety of conditions. The CAN bus to Database component will be tested collecting between one and six different measurements from machines at intervals ranging from every second to every ten seconds. The Database to Front-end component will be tested with a variable number of devices collecting data and a variable number of intermediary devices between the front-end device and the collecting devices. The Data Analytics Computations can be tested for each case of the previous component tests.

Timed Tests:
CAN bus to Database, 1-6 measurements, 1-10 second intervals
Device to Device, 0-4 intermediary devices, all collecting
Database to Front-end, 0-5 intermediary devices, 1-6 devices collecting
Data Analytics Computations, each kind of computation

Usability
Usability was a quality that was stressed by our client. This product will be used by workers who are not computer experts, and may not necessarily be skilled with computers. Only basic computer skills should be assumed for front-end use, such as opening the program, navigating through various pages, and using operations that are self-apparent on those pages.

Front-end use cases should be straightforward and intuitive, achievable without external instruction. Front-end errors should be handled without action from the user, allowing the user to continue the use case. Setting up new devices should involve nothing more than cloning an image to an SD card, setting the IP, and then plugging it in. The previous two steps could likely be performed by one user for a whole operation.

Usability Tests:
Each use case will be completed by our members to test functionality
Each use case will be completed by uninvolved individuals to test ease-of-use
Each use case will be completed by the client to gauge satisfaction

Scalability
A certain amount of scalability is needed because applications of our project could involve many machines on a worksite. The product needs to scale well up to a large amount of devices, perhaps a hundred or more. Our resources allow us to test with six devices, but it is important that having many database entries and network nodes will not negatively impact functionality or performance.

With our resources, it will not be possible to test with hundred of devices. However, we can reconfigure the devices many times over to simulate having many nodes in the network, at least in terms of the data that must be stored and transferred. It will not allow us to test scalability of many devices collecting data at once.

Scalability Test:
Reconfigure devices and collect some data until we have up to 100 devices registered, then test all functionality and perform timed tests

3.5 Process

Our process starts with clarifying all the details of the project with the client. These details include the scope, requirements, schedule, and end goals of the project. Once this has been achieved we move to the brainstorming step. Here we take the details from step 1 and think of use cases, and solutions to those use cases. If at anypoint there is questions we can revisit step one to consult with the client. After the brainstorming step we move on to the research step. Here we can look into how feasible it is to implement any of the solutions that we brainstormed or the technologies we can use to implement the solutions. The next step is the design/creation step. This step uses the ideas generated in step 2 and the information learned in step 3 to produce a general system design. Implementation of the design then begins and if issues arise we can take steps back to revisit problem areas. Once the design and creation step is finished we move onto the testing phase. Here we design and implement test to verify that our application is bug free and meets functional requirements. Once the project is finished we revisit the process of the project and reflect on how it went. This is used to see what when wrong/right and how things could change in the future to ensure that similar problems do not reoccur. This is used to help future estimates for project completion as well. Finally the project is delivered to the client and the process is complete.

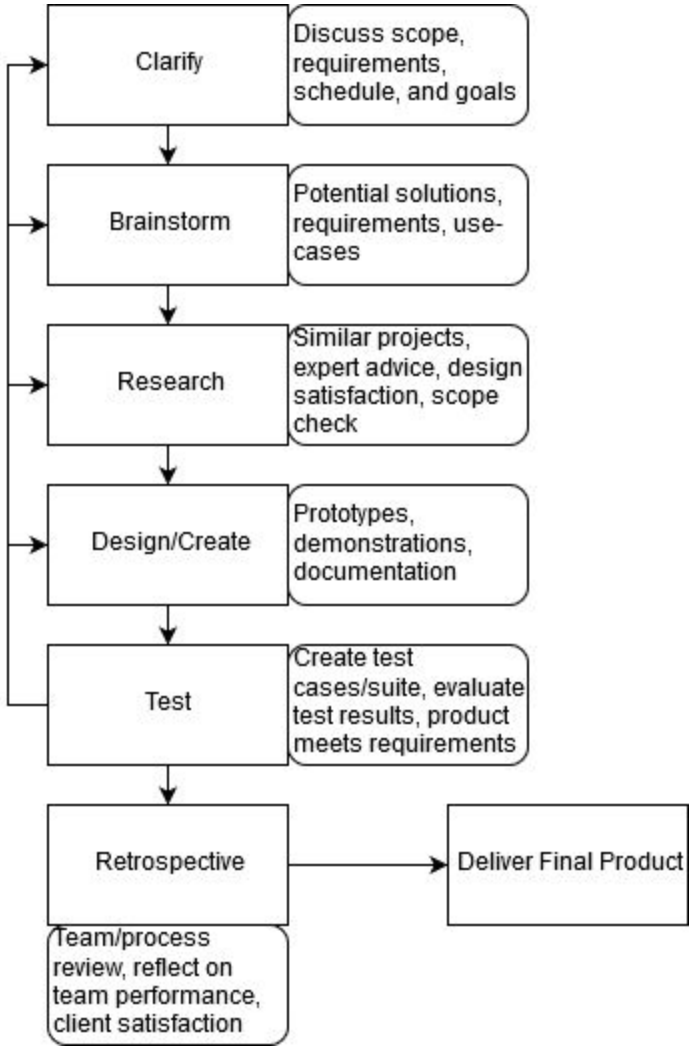

Figure 4: Design Process Diagram

Our design process consists of a modified waterfall method. By starting our initial phase with a system that involves all of the core pillars of systems design, followed by a testing phase that lets our team iterate back through the original designing process, we allow ourselves to continually build our design up. This process will be followed up by a retrospective and review of each major component as it is finished.

3.6 Results

Our project is currently in its infancy, meaning many tests haven't been performed yet. However, some individual components have been tested, as follows:

Modeling and Simulation:

- VCan is being used to simulate virtual CAN data to make sure we are correctly reading our telemetry data and translating it in a form we can understand. CAN

messages have a set number of bits. Each bit can be interpreted differently, depending on the protocol used. Our application of CAN will use J1939 for message format. So, our vcan module will simulate realistic messages in the J1939 format. We could also set up a connection PiCAN to PiCAN to simulate a more realistic situation. One Pi & PiCan can emulate a construction vehicle, sending realistic construction data over

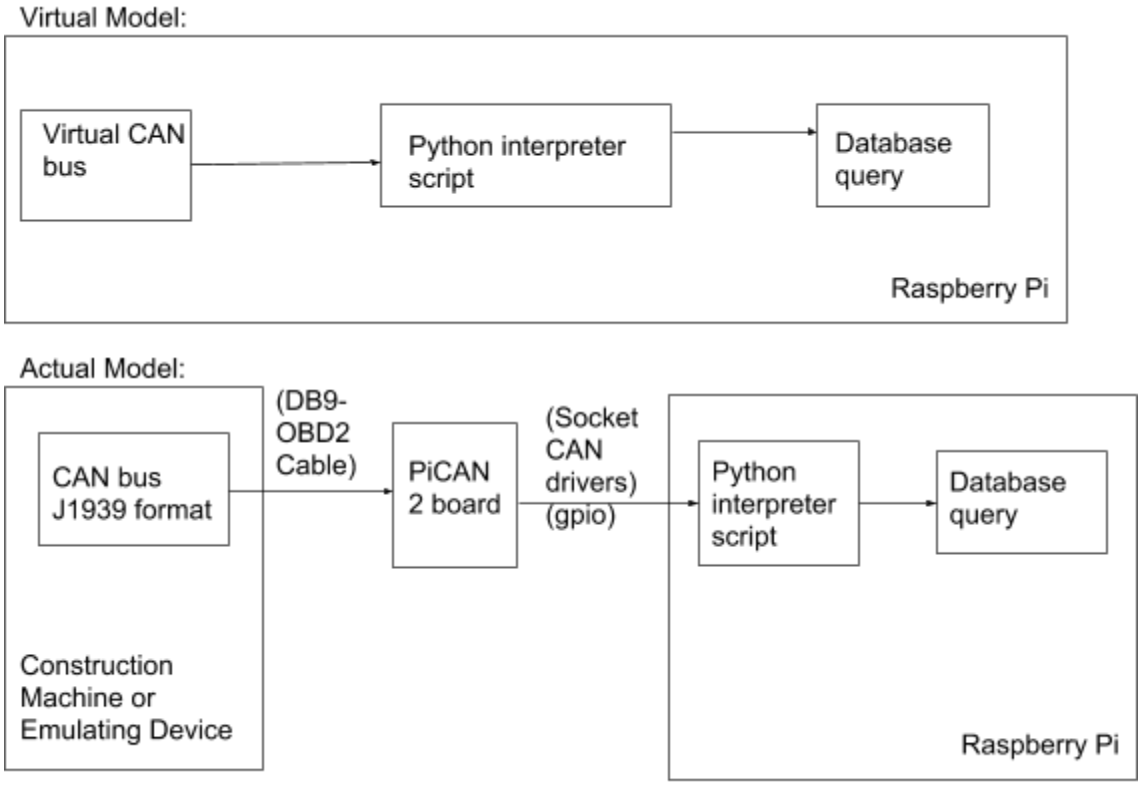

Figure 5: Data Acquisition Model

- To simulate our connected network of nodes at this stage, we have configured three of our Pis with the required libraries to run OSLR and communicate with each other. So far we have been using iptables to simulate distant be nodes to test using OSLR to coordinate the required packet hops to connect devices that are not able to directly communicate.
- Manual testing of the API can be handled by using the application Postman. Automatic testing of the API can be handled by Tavern. The test will be structured as such that testing data is sent in a request and there is an expected result. Automated testing will be beneficial, because as we add more functionality to the API regression testing will be handled by previously written test.
- SQLite has been tested by creating an instance on a machine, applying our database schema, and then testing each of the four CRUD(Create, Read, Update, Delete) operations on the database. We simulated adding nodes to the network which introduced a new table and then stored data to it. The next step is

to transfer data across multiple sqlite instances, but other components must be finished first.

I**mplementation Issues and Challenges**:

- Implementation of the networking and data acquisition portions will likely involve some trial and error. As such, it will be important to be able to test our data in a way that reflects realistic use. This will add difficulty because our resources limit us to a small number of tracking devices. Additionally, we do not yet have access to emulators for CAN or sensor data.
- Setting up each device as a router that recognizes each other device individually will present a challenge
- One of the Challenges will be getting each of the Pi's to allow network connections as an access point. As well as successfully identify other Pi's and pick one to connect to that will optimize the network connectivity. This will require a lot of system level network configuration that will be new to our team.

4 Closing Material

4.1 Conclusion

Our Mesh Network design consists of three main components. We have our system of Raspberry Pis that collect CAN data and transmit said data back and forth throughout the network through wi-fi connectivity utilizing OSLR as our routing protocol [2]. We have our database solution, built in Sqlite, that keeps a log of the data collected from every node in the network, and works to keep a log of all collected data on every node. Lastly we have a desktop application built in Electron that collects all the data in one hub for review and visualization capabilities. These components each contribute to satisfying the main functional and nonfunctional requirement of the project.

Since our project is a proof of concept project using specific hardware we will be using a lot of manual testing. Our testing consists of basic unit testing to ensure accuracy of information and speed of data transfer, integration and system testing to ensure integrity, reliability, scalability, and other requirements are met, and acceptance testing to determine if the project actually meets the client's needs. We can automate some testing around the OBD2, and with the electron application. Scripts can be developed to do some of the network testing but because thoroughly testing the network setup requires changing routing, firewall, and wireless card configurations on multiple Raspberry Pis a lot of the network testing will have to be done manually so that the correct tests are running with the correct network state. Through this testing approach we will be able to check how our design is able to satisfy its requirements, and to evaluate any changes to the design that arise as development progresses.

4.2 References

[1]Toh, Chai-Keong, and E. M. Royer. "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks." IEEE Journals & Magazine, IEEE, Apr. 1999
URL: http://ieeexplore.ieee.org/abstract/document/760423.

[2] Yan, Gongjun. "Vehicle-to-Vehicle Connectivity Analysis for Vehicular Ad-Hoc Networks." ScienceDirect, Academic Press, 10 Dec. 2016.
URL: www.sciencedirect.com/science/article/pii/S1570870516303274#sec0002

[3] Linear Technology, "SmartMesh Wireless Sensor Networking for Industrial IoT", 2016.

URL:http://www.avnet-israel.co.il/wp-content/uploads/2016/12/SmartMesh-Wireless-Sensor-Networking-for-Industrial-IoT.pdf

[4]Malesci, Umberto. “Mining Automation and Connected Mines.” Fluidmesh, 9 Nov. 2015.
URL: www.fluidmesh.com/mining-automation-connected-mines/

[5] SAE Committee “Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document J1939_201808.” SAE International, SAE, 20 Aug. 2018
URL: www.sae.org/standards/content/j1939_201808/.