An Advanced Networking Outreach Activity for Kids

DESIGN DOCUMENT

sddec20-05

Dr. Tom Daniels

Grayson Cox | UI Developer | Agile Project Manager

Austin Dvorak | Network Systems Manager

Ryan Newell | Hardware Systems Admin

Spencer Parry | UI Developer

Ross Thedens | Communication System Manager | Meeting Secretary

Team Email: sddec20-05@iastate.edu

Team Website: http://sddec20-05.sd.ece.iastate.edu/

Revised: April 26, 2020 (V3)

Executive Summary

Development Standards & Practices Used

- Standards
  - Quality management (ISO 9001)
  - Developing information for users in an agile environment (ISO/IEC/IEEE 26515)
- Practices
  - Agile development
  - Code reviews
  - Weekly meetings
  - Rigorous testing for security and reliability

Summary of Requirements

- Support mesh and dynamic ad-hoc networking capabilities
- Ability to stream data between nodes
- Easy to use and set up
- Hardware should be mobile and durable
- Can be operated without the need for an Internet connection
- Range between nodes should support a 40 foot radius (when in eye-sight)
- Nodes will consist of a single-board computer, rechargeable battery, and camera/sensors

Applicable Courses from Iowa State University Curriculum

- CprE 489 - Computer Networking and Data Communications
- CprE 430 - Network Protocols and Security
- CprE 537 - Wireless Network Security
- CprE 230 - Cyber Security Fundamentals
- ComS 252 - Linux Operating System Essentials
- SE 319 - Construction of User Interfaces
- SE 329 - Software Project Management
- ComS 309 - Software Development Practices

New Skills/Knowledge acquired that was not taught in courses

- Web Development with Angular/Typescript
- Ad-Hoc/Mesh Networking and Router Configurations
- Raspberry Pi Configuration

Table of Contents

Figures and Tables

Figures

Tables

1. Introduction

1.1 Acknowledgement

We would like to acknowledge our client and advisor Dr. Tom Daniels for his guidance and support on this project. We would also like to acknowledge the help of ETG staff for helping us get our hardware deliveries, especially during the campus closure during the pandemic.

1.2 Problem and Project Statement

Wireless networking has become a fundamental part of many people's everyday lives. This is especially true for students across grades 4-12, who are beginning to spend hours each day on school-issued internet-connected devices. However, few students understand the technology that underpins these networks. The internet is often portrayed as an amorphous portal granting access to huge amounts of information. In reality, this information is stored on servers around the world and delivered to end users via routing protocols. By creating a teaching tool that demonstrates routing protocols, we can illustrate the challenges involved in creating a reliable wireless network, increase technology literacy, and stimulate students' interest in computer networking careers.

Our approach involves creating a teaching toolkit consisting of a set of portable wireless network nodes, a network monitor/configuration application, and a set of lesson plans. A grade school instructor with no computer networking training will be able to operate it. The nodes will automatically form an ad-hoc network with one another when powered on (and within range of one another). Network data will be provided by video cameras and various sensors on the nodes, such as temperature sensors. The data will travel the network to reach a master node which is directly connected to a network monitor application running on an instructor's desktop or laptop (referred to as "station" in this document). The monitor application will control which sensor/video streams are delivered to the master node and display the data received. The lessons will challenge students to transmit data across a long distance (beyond the range of a single node; from the main office to a classroom, for instance) to the network monitor application. As students add nodes into the network and position them to attain connectivity, they will observe the routing of the data (which nodes are reached) in the monitor application. They will use this information to figure out where data is unable to travel and rearrange the nodes accordingly.

Ultimately, our toolkit will act as a miniature mock-up of the real internet. Students will be able to appreciate how data from a web server hops from node to node over the internet before reaching their devices. Following the activities, the instructor will discuss grade level-appropriate topics related to networking from the lesson plan with the class. For younger students, this may involve the fact that wireless signals have a limited range, while for older students it may include a discussion of the routing decisions made by the ad-hoc protocol. Our project will provide a flexible way to teach computer networking to grade school students.

1.3 Operational Environment

A collection of networking nodes will compose the main aspect of the system. It is expected that these nodes will be used indoors, but they should be able to operate outdoors as well. They do not

need to be weatherproof, however, they will need to be durable enough to protect the internal components.

There will be a graphical user interface aspect to the system, which will allow users to view sensor and video data transmitted from the nodes. This can be accessed via any web browser on a separate computer, and no internet connection should be necessary as long as there is a connection between the computer and the master node.

1.4 Requirements

Nodes will need to be powered by rechargeable batteries so that they do not require a fixed power cord when in use. They will also connect to each other wirelessly. Both of these requirements ensure that nodes are mobile.

Each node should support ad-hoc networking capabilities. Multiple networks should be able to coincide and stay independent of each other. Nodes should use their respective networks and safely handle interference and other network errors. To demonstrate functionality, nodes will send data across the network. Camera feed, sensor data, and control signals will need to be transmitted through nodes to the master node. The graphical user interface will display this data and give a visualization of the network.

The packaging of each node will need to be durable enough to survive drops since they will be moved around frequently. Each node will consist of a single-board computer with a case and a rechargeable battery. Some nodes will contain sensors and cameras to serve as data sources for the network. Components will be bundled together in a package that makes it intuitive to charge and easy to use and set up.

Besides being wireless and easy to place, the software should also be user-friendly and intuitive to use. The instructor may not have much knowledge on computer systems so the monitoring application should be simple enough that most kids and adults can understand the interface. To aid in the ease of set up, nodes will not need to be connected to an existing network. The only network that they will use is the one they create themselves.

1.5 Intended Users and Uses

The project has three main types of users: students, instructors, and system administrators. The goal of the project is to educate students on how mesh networking works, therefore, the students would be the target audience. The students would not interact with the GUI as much as the instructors or teachers would, but the GUI should be user friendly enough for anyone in the three target audience groups to use. The students largely will be moving or placing the nodes in different places, and the instructors will use the GUI to interpret the data for the class.

The instructors would have knowledge on how to use the nodes and the GUI so they can teach their students general concepts about WiFi such as range and interference. They would know how to position the nodes to display on the GUI how different forms of interference can cause the video to buffer more or less than other forms of interference. The instructor would also use the GUI to change settings on the nodes, like grouping nodes into two separate groups.

The system administrators would have access to the file system and command line of the nodes to manage updates or to diagnose problems with the nodes. The administrators would only have to be

involved if there is a problem with one of the nodes, or a mandatory update needs to take place. Students and instructors would not have access to the system administrator settings; however, system administrators would have full system access, as well as access to the student and instructor GUI.

1.6 Assumptions and Limitations

Assumptions

- The user owns a station computer with a web browser.
- The user's environment does not abnormally impede WiFi connectivity
- The person running this activity is able to charge the nodes when batteries run low.

Limitations

- The project will be completed by the end of the fall 2020 semester.
- Nodes will need to be mobile.
- Internet connection may not be available.
- Each node will be powered by a single-board computer.
- Nodes will need to be stored in a compact space.
- Battery life should last at least two hours.

1.7 Expected End Product and Deliverables

The main deliverables will be a set of at least eight wireless, rechargeable nodes. Some of these will be "sensor" nodes with sensors (i.e. temperature sensors) and camera modules attached. Others will be "relay" nodes that simply carry traffic through the network. These two types of nodes are collectively referred to as "mesh" nodes. A single "network master" node will run the backend to the GUI and route data between the GUI and network nodes. Sensor data and network statistics (i.e route traces to particular nodes, packets dropped, etc.) will be transmitted through the network and displayed on the GUI in charts and graphs; camera frames will be displayed as a video stream. We anticipate delivery of functioning Relay Nodes by September 6 because of the simplicity of their design. The Sensor Nodes with sensor data and streaming capability will be delivered by October 17. The Network Master Node component is to be delivered by November 8.

The user interface component will be in the form of a web-based application that can be accessed through any web browser. The UI will offer at least two displays: one large display for visualizing a specific stream of data and one display for controlling the nodes and the large display. The idea is that the instructor can control the system on a small screen and show the large display on a projector or television for the students to see. We anticipate delivery of a full UI by November 15.

Lastly, to assist educators in using this technology, we will prepare a set of lesson plans and activities which demonstrate the workings of wireless networking. This deliverable serves as a vehicle for demonstrating our project, and the delivery date is set for December 12.

2. Specifications and Analysis

2.1 Proposed Approach

Recalling the Summary of Requirements, there are some major objectives that this project needs to meet. The most limiting requirement is the budget. Due to wanting to keep each node at $65, limits the hardware. This means that our approach to what operating system, networking protocols, etc. we use changes due to the limited processing power of the device.

A single board computer was deemed the correct choice due to the low cost ~$35 per board, but also the higher processing power and I/O abilities over embedded boards, along with the high mobility. To increase this mobility, a battery pack would be included to allow the nodes to work without wall power. The next step was deciding how the networking would be handled. Certain distributions of Linux aimed at computer networking come with support for mesh networking protocols, so the decision has to be made about which one to use. These distributions should allow for automatic connecting and disconnecting of links to aid with ease of use. A web server would be set up on one of the links to allow for a cross-platform GUI that would show all of the required information coming from the network, along with easy configuration of the network and nodes.

2.2 Design Analysis

Deciding on the hardware of the project seemed to be relatively easy. Due to the widespread support and price of the board, Raspberry Pi's will be used as the nodes. This also makes figuring out which mesh-networking protocol and linux distribution easy. After research, it was found that IPFire and OpenWRT were the best distributions to use, as there were already examples of these distributions working on the Raspberry Pi. Selecting the ad-hoc networking protocol was the next step.

There are 3 main types of protocols: proactive, reactive, and hybrid. Proactive protocols create a list of connections and their routes at the expense of latency when adding and subtracting nodes, while Reactive floods the network with packets at the expense of latency when sending packets. Hybrid routing creates a table initially and floods only when it cannot find a path to the destination. Due to how much bandwidth flooding a network stream with video packets would require, the decision was made to go with a proactive protocol.

There are 5 main proactive routing protocols: OLSR, Babel, DSDV, DREAM, and BATMAN. BATMAN and OLSR are the two most prominent protocols and are both supported by OpenWRT and IPFIRE as well as Raspbian. After initial testing of the system on OpenWRT it was deemed infeasible and we switched to Raspbian, as we could not get mesh networking to work with raspberry pis on OpenWRT.

For the user application component, it was decided that Angular/Typescript would be used for the UI due to the ease of use and since the members working on the UI have had previous experience in Angular development. To handle our requirement for video streaming and possibly other forms of live data, we will be using WebRTC, which is an open-source framework that provides real-time

communication capabilities in a web browser. With WebRTC, we are able to provide the best video streaming quality with the lowest latency of all the strategies with which we experimented.

Lastly, we have decided that the backend to the user application would be in the form of a Java Spring Boot application running on the Network Master Node. This application will work with the mesh network directly and will provide an API for the user application to fetch information about the mesh network, such as the list of active nodes, information about each node, network statistics, and sensory data from Sensor Nodes. By choosing to have a Spring Boot application serving as the interface between the network and the user application, we are also able to take advantage of its web socket capabilities to serve as the signalling server for live media streams via WebRTC.

2.3 Development Process

Because of the lack of certainty in the design of this project, our team requires an iterative development method that allows for design decisions to take place at any point in the project lifecycle. And given the vagueness of our initial project description, we need an approach that will assist us in reaching our final design starting with a high-level understanding of the project. Therefore, we will conform to an Agile development process with a top-down design approach because it mitigates the risks associated with the inherent variability of this project.

The development lifecycle of our project will consist of a number of sprints, each lasting two weeks. Prior to each sprint, the team will hold a sprint planning meeting in which we choose tasks/features to implement during the coming sprint. Following each sprint, the team will meet for a sprint retrospective meeting, where we will discuss our progress and think critically about our performance.

We will be using the tools provided by GitLab for our artifacts. Individual units of work (i.e., tasks, features, or user stories) will be represented as Issues in GitLab, which can be created by and assigned to any team member. Each Issue is associated with a single Git branch, so Issues are loosely coupled, allowing team members to work independently.

To control the quality of our implementations, all code will go through peer reviews before going into production. These code reviews will be facilitated via merge requests in GitLab. Once a team member finishes work on an Issue, he will open a merge request and assign at least one other team member to review the work.

To track the progress of the team, Issues will appear on sprint boards. We will use different boards to denote the status of every Issue. The board in which an Issue resides will indicate whether it is *Open*, *InProgress*, *InReview*, or *Closed*.

2.4 Conceptual Sketch

Our conceptual sketch of the project is shown in Figure 1.

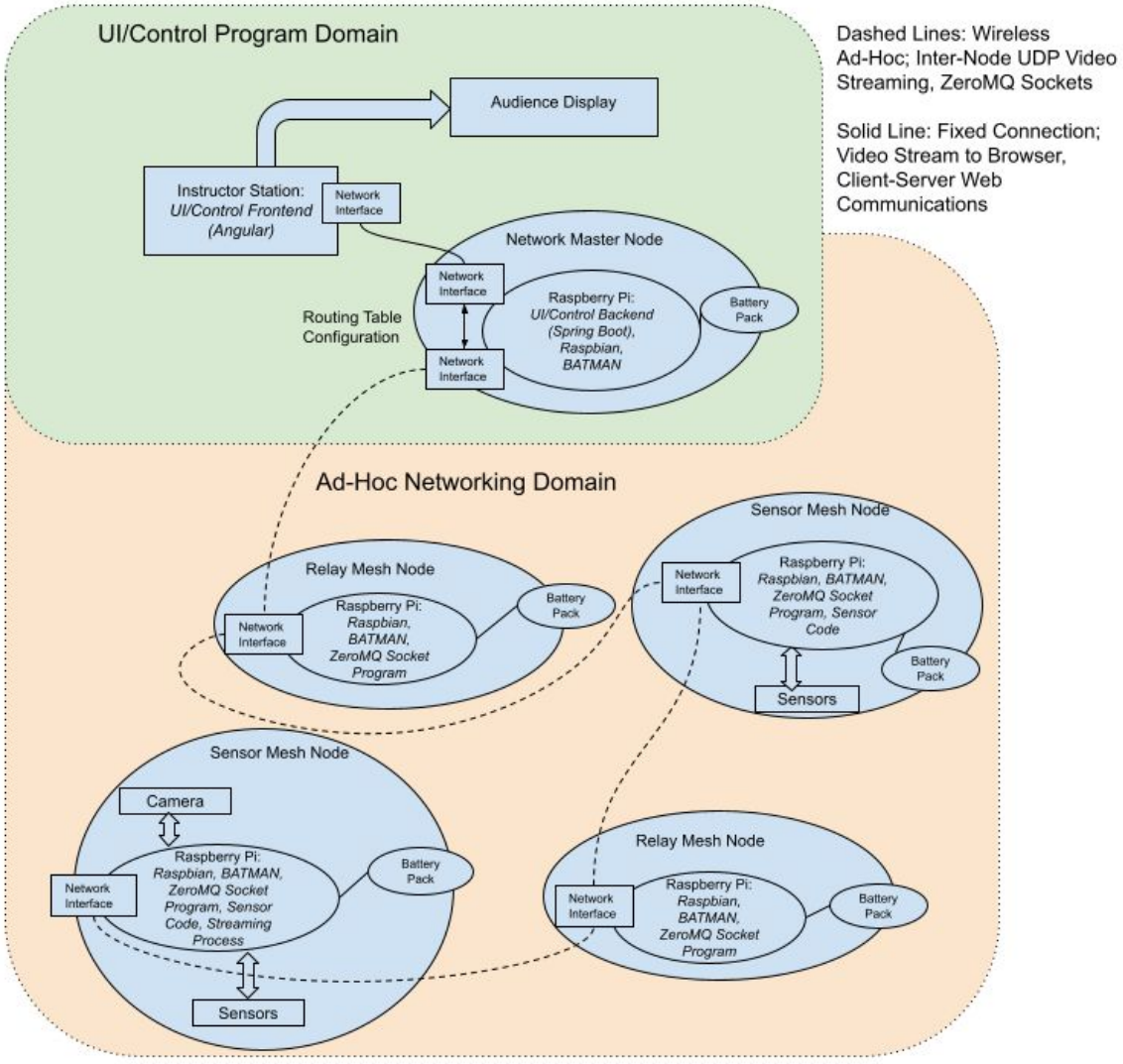


Figure 1: Conceptual Sketch

Our project consists of two major domains: a UI/Control Program domain and an Ad-Hoc Networking domain. In the UI/Control Program domain, the instructor's station runs the UI/Control frontend (an Angular web app), which allows users (students and the instructor) to initiate video streams, request sensor data and network statistics, and view incoming data in graphical displays. The data visualizations occur on the Audience Display, which may be the instructor station's screen or an external display (such as a projector).

A network interface on the instructor station is connected to another network interface on the Network Master Node (WiFi or Ethernet). A second (wireless) network interface on the Network

Master Node is connected to the ad-hoc network This allows the station constant access to the ad-hoc network without connecting to it directly. The Network Master Node will contain special routing table entries to ensure data is forwarded properly between its interfaces. It also runs the UI/Control Program backend (implemented in Spring Boot) and includes the Raspbian Linux distribution and BATMAN ad-hoc routing package (batman-adv) necessary for participation in the ad-hoc network.

In the Ad-Hoc Networking domain, each mesh node will consist of a Raspberry Pi board with a wireless network interface that supports ad-hoc networking. Each board will run the Raspbian distribution of Linux. The connections from node to node will be managed by the BATMAN ad-hoc networking protocol. On top of the Linux operating system, there will be other drivers and software to capture data from cameras and sensors connected to some of the nodes (via GPIO or ribbon cable). This data will be transmitted back to the master node and to the instructor's station at the behest of the UI/Control Program, which sends control signals to the nodes in the network to initiate transfers. The ZeroMQ networking library will be used to transfer all non-video data through the network. Video data will be transmitted using WebRTC via the raspivid utility and a compatible video streaming utility.

All nodes (mesh and Network Master) will be powered via rechargeable battery packs and stored in portable, durable, drop-proof casings. Each set of nodes distributed to instructors will be preprogrammed with a unique identifier so that all node sets form separate ad-hoc networks, even when used within range of one another.

Figure 2 shows the system modules involved in our project for each physical computing device involved. This includes the mesh nodes, the Network Master Node, and the instructor station.

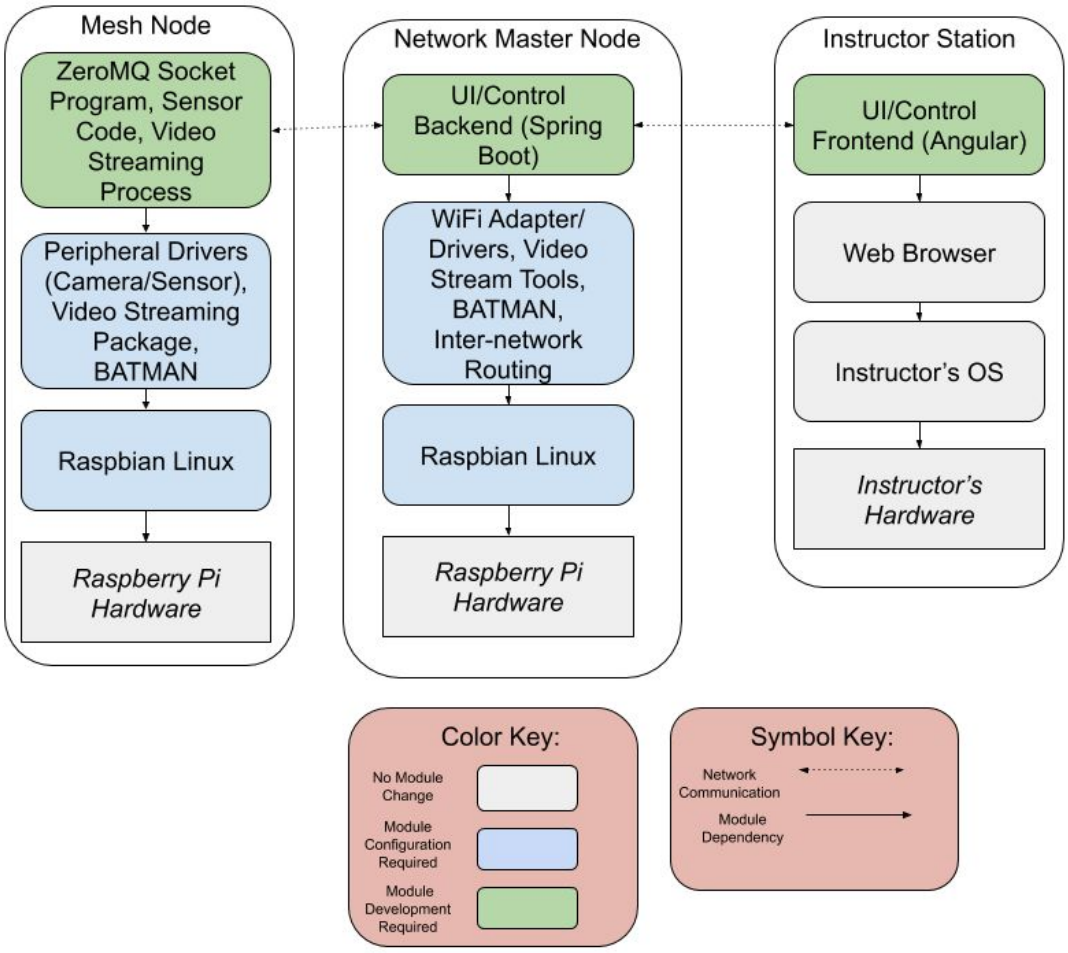


Figure 2: System Modules

The sharp-cornered blocks represent the base hardware layers of each device, while rounded-cornered blocks represent primarily software modules. As shown by the color key, the gray modules already exist and require no special configuration. The blue modules exist but will need to be configured/installed as part of the project. The green modules do not exist and will need to be developed as part of the project.

The arrows indicate the dependencies between layers. For mesh/Network Master Nodes, the top-level software will rely on device drivers for the peripherals and the BATMAN protocol. The drivers will be installed in Raspbian, which will be installed on the Raspberry Pi hardware. Network communication will be initiated in the top-level software, although the actual data transfer and routing is handled by the routing distribution and network interfaces. The UI/Control program backend will communicate over the network with the top-level software of the mesh nodes. Since the frontend runs in a browser, no special configuration is required on the instructor's station. Once the master node is connected and started, the instructor will simply have to launch a web browser and load the app's webpage.

3. Statement of Work

3.1 Previous Work And Literature

Several alternatives already exist for ad-hoc networking instruction. These primarily consist of computer programs designed to simulate various network types. The ns-3 project [1], supported by the University of Washington NS-3 Consortium, provides a simulated networking environment. Events can be scheduled in ns-3 to cause signals to be transmitted from various nodes at various times. The ns-3 simulator is also capable of interfacing with both virtual and real devices. The mesh package in ns-3 can be used to provide a MAC-layer ad-hoc routing capability for network simulations. Ns-3 is an appropriate tool for graduate and possibly undergraduate study of networking, but it does not effectively meet the needs of a grade-school user base. Ns-3 is primarily configured via the command line, which would be difficult and frustrating for an instructor or grade-school student to use. Students may also have trouble visualizing a network of simulated, virtual nodes; using all physical nodes would provide students with a tangible networking experience that feels more relevant to them.

While our project consists of fully-physical nodes, it shares some points in common with previous academic experiments both physical and virtual. E. Biagioni [2] from the University of Hawai'i at Mānoa created an ad-hoc wireless network using four Raspberry Pi Zero W computers and a station. The network was achieved using WiFi and not Bluetooth, similar to our planned approach. We are wary of Bluetooth, as it may be difficult to obtain the bandwidth needed to stream video through the network. The nodes were placed in rooms around a university building, and a utility similar to traceroute was used to test the network strength. However, the goal in this instance was to evaluate a specific ad-hoc networking protocol, AllNet. In order to keep students' attention and provide a demonstration that feels relevant to students, we need to send live data through the network, such as a video feed.

Another experiment by Sharma and Nekovee [3] is similar to our proposed approach. Unlike our approach, they used virtual nodes. However, their approach does involve sending video over the network. It also centers on the automatic configuration of new nodes as they are added into the network (i.e. assigning IP addresses). This is critical, as we want to avoid having instructors enter arcane commands into a terminal to set up the nodes properly. It also contains a controller with a GUI application, which is very similar to our UI/Control program concept. In completing our project, we hope to combine the aspects of these two prior projects that will benefit students the most and lead to the simplest kit for instructors to set up.

3.2 Technology Considerations

Raspberry Pis are great for this application due to the amount of support they have. If we were to go with a different board we may not be able to find a suitable linux distro that supports mesh networking. They are also cheap enough for our budget and have a wide array of accessories designed specifically for the device. The trade off with this is that because the boards are so cheap, they have low power cpus and low amounts of ram, along with cheaper wifi modules. The range of these modules is not very high and we may have to buy better modules, but, since the boards are so cheap, it will not make the final design infeasible.

For the software, there were a variety of routes we could go with. There was a possibility of using Bluetooth mesh networks to design the system, however we found that Bluetooth LE might make that infeasible for the timeframe and budget given. Our best option would be to use either BATMAN or OLSR for our mesh networking protocol over some linux distribution, and were able to narrow it down to either OpenWRT or Raspbian, starting testing first with OpenWRT, due to its inherent Raspberry Pi support, and small size. Both BATMAN and OLSR possess the same performance traits, so it will just be down to picking one and going with it for the rest of the project.

3.3 Task Decomposition

- Nodes
    - Support Ad-Hoc Networking
        - Configure Raspbian Linux
        - Configure BATMAN
    - Battery and System Monitoring
        - Rechargeable Battery Installation
        - CPU, RAM Usage Reporting
    - Mesh Node
        - ZeroMQ Socket Program
        - Case installation
    - Sensor Nodes
        - Stream Camera Feed
        - Obtain/Interpret Sensor Data
        - Case installation (Camera included)
- Network Master Node
    - Host UI Component
    - Host Backend Component
    - Manage Node Connections on Startup
    - Routing Table Configuration (to forward data between ad-hoc network and instructor station)
    - Case installation
- User Interface
    - Audience Display
    - Node Selection Page
        - List of Active Nodes in Network
        - Topological Network Graph
    - Node Detail Page
        - Node Information Display
        - Node Information Configuration
        - Data Stream Visualizations
            - Video and Sensor Data (for Sensor Nodes)
            - Network Statistics
        - System Monitoring Indicators
    - Data Model
        - Nodes
        - Stream

        - Network Layer (i.e., Services for communicating with backend)
        - Graphic Design
            - Artwork
                - Application Icon
                - Mesh Node Icon
                - Sensor Node Icons
                    - Camera Node Icon
                    - Other Sensor Icons
- Backend
    - Node Status Microservice (node connect/disconnect information)
    - Network Statistics Microservice (packets sent/received/lost, etc.)
    - Camera Stream Microservice (for selection of stream when multiple streams can be displayed)
    - Sensor Data Microservice
    - Network Configuration Microservice (edit node properties such as hostname, etc.)
- User Documentation
    - User Manual
    - Lesson Plans

3.4 Possible Risks And Risk Management

One of the primary risks of the project is the transition to online instruction (as of March 23, 2020). While our project can be completely done outside of the labs it will be difficult to do group work such as bi-weekly client meetings. We can use Zoom for the meetings, however if someone wants to test some things on the hardware we have to pass the hardware along rather than get to meet in a team. To ensure that everyone is on track with what they need to do, we will have weekly 'touch-base' meetings where we check in each member of the team through Zoom.

Another risk would be that the hardware that we use is incompatible with the software/unable to perform the design requirements. In this case, we have found multiple possible technologies (i.e. BATMAN and OSLR) to use and different hardware components that are advertised to work, so we would just try each one until one of them works as needed.

3.5 Project Proposed Milestones and Evaluation Criteria

Our first milestone, the completion of one or more mesh nodes, will be evaluated by our ability to send and receive data in a network of mesh nodes. The mesh nodes should also be able to report network statistics. This does not include any other components of the system, but ad-hoc networking between the nodes is required.

Our second milestone will be the completion of a functional camera node and video streaming functionality. We expect to be able to connect a camera node to the network via the Network Master Node and then view live camera feed in the user interface.

Our third milestone will be the completion of the battery installation. Each network node should be connected to a rechargeable battery. The battery will need to be securely fastened to the node to prevent its loss. This milestone will also include the implementation of CPU and RAM usage monitoring, which will be reported back to the UI.

The fourth milestone will be marked by the completion of the Network Master Node software including a backend with microservices supporting sensor, camera, network statistics, and node status data request and retrieval from the network. It will also include a partial implementation of the user interface. Our Network Master Node will be able to connect to the mesh nodes to the network, and the user interface should be able to query that information and show all active nodes on a display. The UI will also allow selection of camera streams, network statistics, and sensor data for viewing.

Our fifth milestone will include a finished UI with a dynamic list of all active nodes and a topological network graph showing the connections of the ad-hoc network. At the same time, the UI and Backend will be optimized for the Raspberry Pi to ensure the Network Master Node can handle its processing load effectively and the network is not overrun with unnecessary packets.

Lastly, the sixth milestone will be the completion of our project, accompanied by a set of lesson plans and a user manual. Additionally, we may add one or more new types of Sensor Nodes (with additional sensor types) that can be connected to the network.

3.6 Project Tracking Procedures

To track our progress throughout the lifetime of the project, we will use the project management tools offered in GitLab. We have set up Issue boards to denote certain Issues as Open, Blocked, InProgress, InReview, or Closed. These boards will be used to track the progress of every iteration (represented with a Milestone in GitLab) of the project. For every Milestone, GitLab offers a burndown chart showing the amount of work completed over time. These burndown charts will also show progress for the entirety of the project as we complete our deliverables.

3.7 Expected Results and Validation

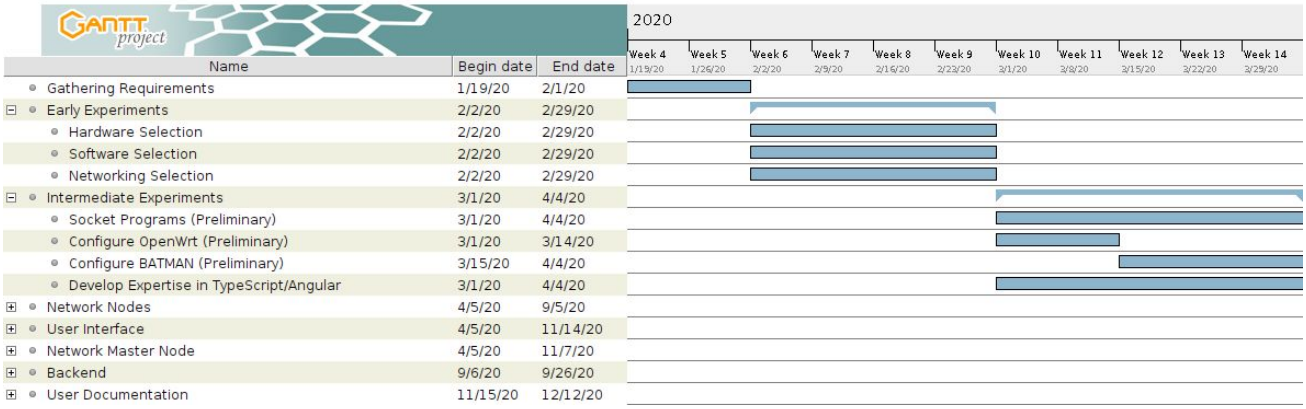
The determination of our desired outcome comes down to these three main goals: functional lesson plans, an easy-to-use user interface, and lack of interference from other networks. The project should include lesson plans that will teach students how mesh networking works, and those should all work seamlessly with the provided equipment. The user interface needs to be designed in a way that an instructor or a student can operate it with minimal knowledge on how it all works. Finally, in order for the first two goals to be fulfilled, the system should not encounter any outside interference from other types of networks.

We have a number of test cases that we are going to use to confirm the results work at our expectations. We intend on testing the system in multiple environments, such as inside one of the university buildings where there would be other networks that could potentially cause interference, or outside where we could test maximum range of the product. We also plan on having a test user from outside of the project test the user interface, making sure that an outside individual can operate the lesson plans, user interface, and system with ease.

4. Project Timeline, Estimated Resources, and Challenges

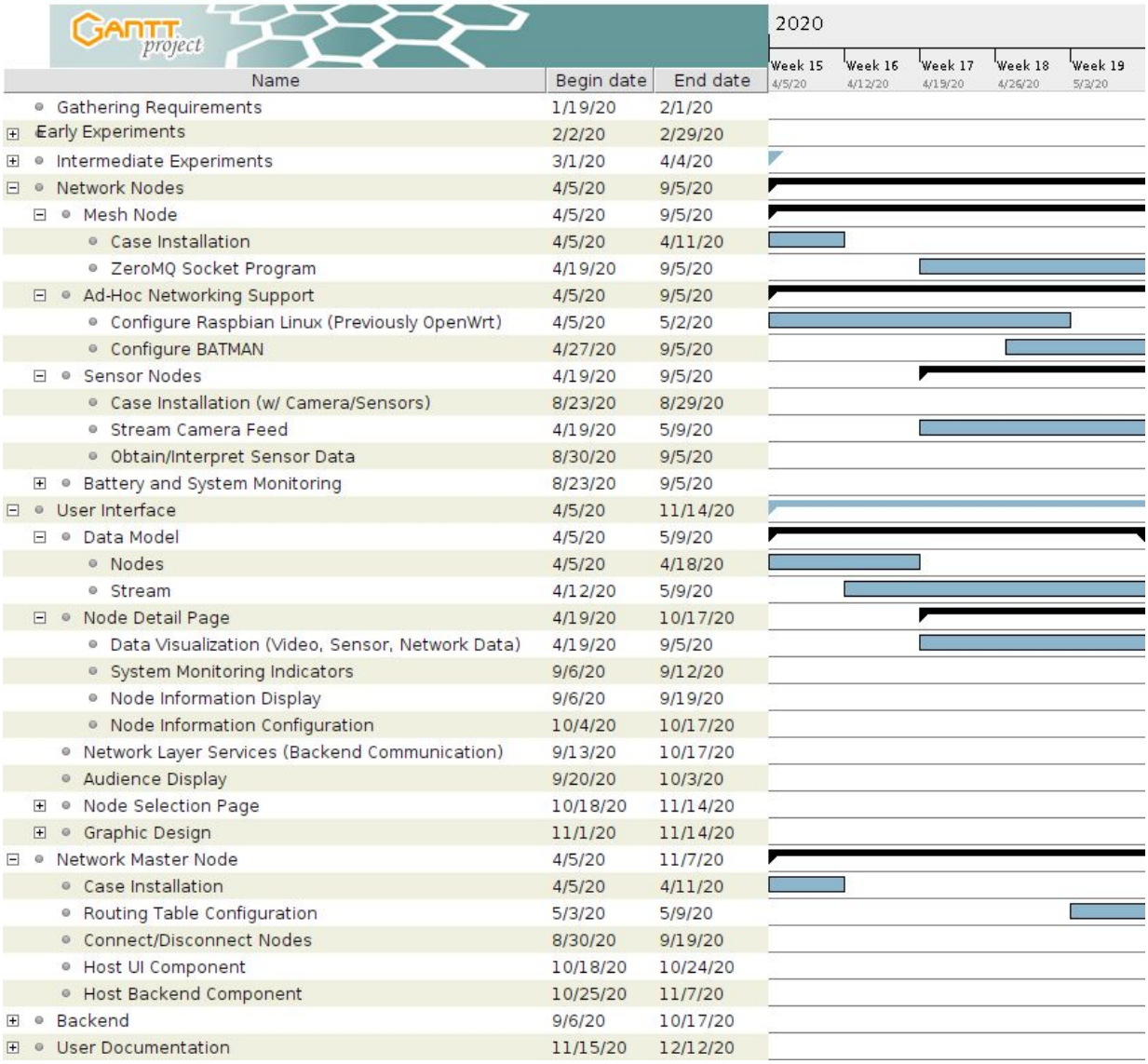
4.1 Project Timeline

Figures 3, 4, 5, and 6 below show the entire Gantt chart for our two-semester senior design project. The figure captions state the time period each image represents. Tasks with subtasks that are not present in a given time period have been collapsed. Figure 3 shows the exploratory steps we are performing prior to the receipt of our official hardware for the project.



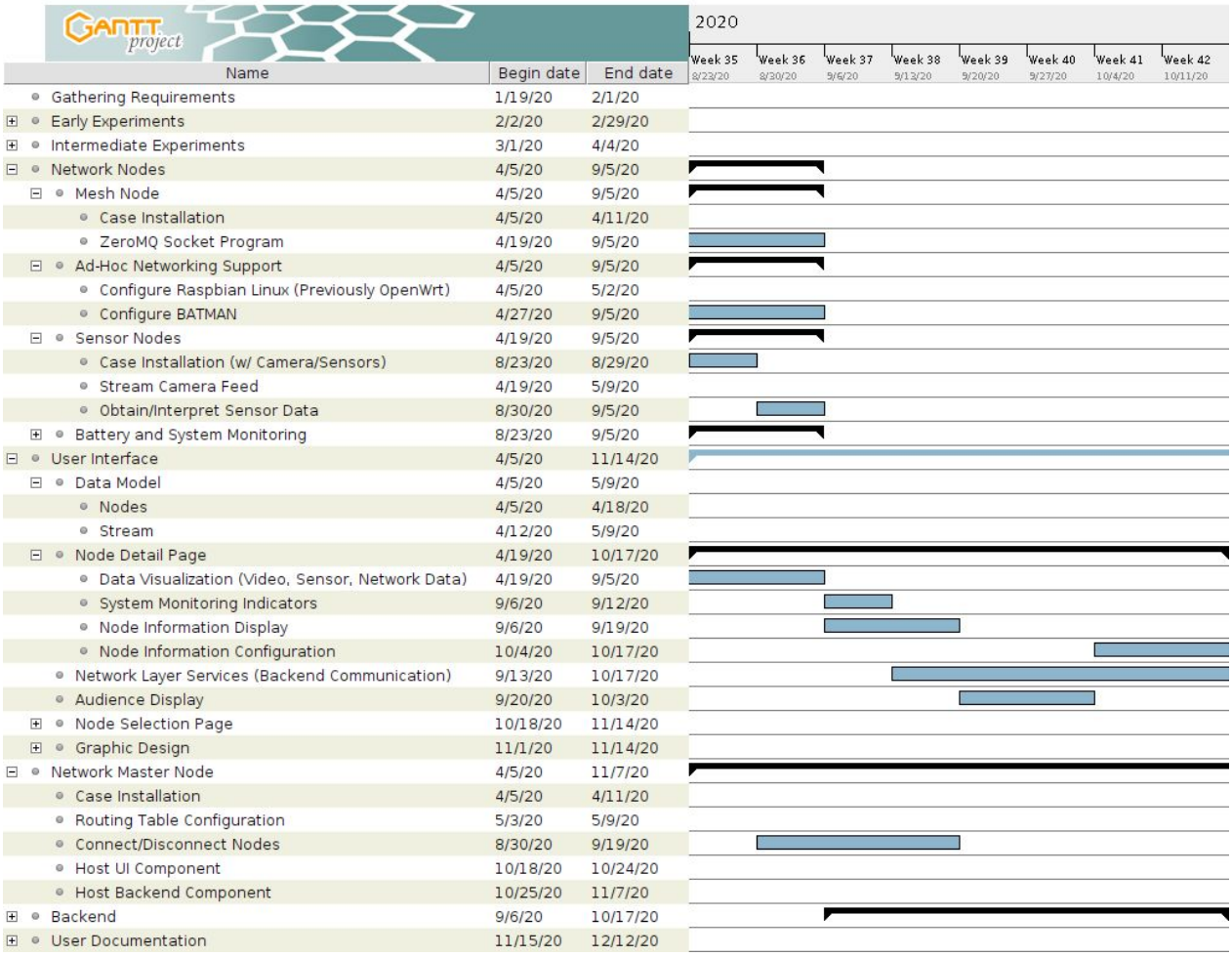
Name	Begin date	End date
Gathering Requirements	1/19/20	2/1/20
Early Experiments	2/2/20	2/29/20
Hardware Selection	2/2/20	2/29/20
Software Selection	2/2/20	2/29/20
Networking Selection	2/2/20	2/29/20
Intermediate Experiments	3/1/20	4/4/20
Socket Programs (Preliminary)	3/1/20	4/4/20
Configure OpenWrt (Preliminary)	3/1/20	3/14/20
Configure BATMAN (Preliminary)	3/15/20	4/4/20
Develop Expertise in TypeScript/Angular	3/1/20	4/4/20
Network Nodes	4/5/20	9/5/20
User Interface	4/5/20	11/14/20
Network Master Node	4/5/20	11/7/20
Backend	9/6/20	9/26/20
User Documentation	11/15/20	12/12/20

Figure 3: Gantt Chart CprE 491 (01/19 - 04/04)



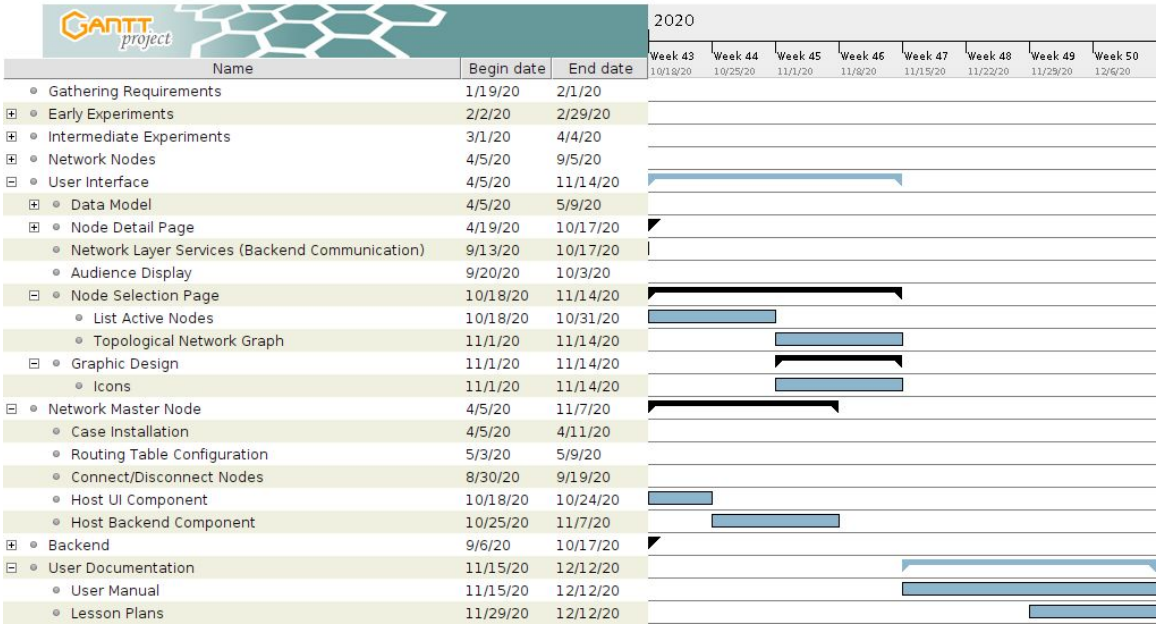
Name	Begin date	End date
Gathering Requirements	1/19/20	2/1/20
Early Experiments	2/2/20	2/29/20
Intermediate Experiments	3/1/20	4/4/20
Network Nodes	4/5/20	9/5/20
Mesh Node	4/5/20	9/5/20
Case Installation	4/5/20	4/11/20
ZeroMQ Socket Program	4/19/20	9/5/20
Ad-Hoc Networking Support	4/5/20	9/5/20
Configure Raspbian Linux (Previously OpenWrt)	4/5/20	5/2/20
Configure BATMAN	4/27/20	9/5/20
Sensor Nodes	4/19/20	9/5/20
Case Installation (w/ Camera/Sensors)	8/23/20	8/29/20
Stream Camera Feed	4/19/20	5/9/20
Obtain/Interpret Sensor Data	8/30/20	9/5/20
Battery and System Monitoring	8/23/20	9/5/20
User Interface	4/5/20	11/14/20
Data Model	4/5/20	5/9/20
Nodes	4/5/20	4/18/20
Stream	4/12/20	5/9/20
Node Detail Page	4/19/20	10/17/20
Data Visualization (Video, Sensor, Network Data)	4/19/20	9/5/20
System Monitoring Indicators	9/6/20	9/12/20
Node Information Display	9/6/20	9/19/20
Node Information Configuration	10/4/20	10/17/20
Network Layer Services (Backend Communication)	9/13/20	10/17/20
Audience Display	9/20/20	10/3/20
Node Selection Page	10/18/20	11/14/20
Graphic Design	11/1/20	11/14/20
Network Master Node	4/5/20	11/7/20
Case Installation	4/5/20	4/11/20
Routing Table Configuration	5/3/20	5/9/20
Connect/Disconnect Nodes	8/30/20	9/19/20
Host UI Component	10/18/20	10/24/20
Host Backend Component	10/25/20	11/7/20
Backend	9/6/20	10/17/20
User Documentation	11/15/20	12/12/20

Figure 4: Gantt Chart CprE 491 (04/05 - 05/09)



Name	Begin date	End date
Gathering Requirements	1/19/20	2/1/20
Early Experiments	2/2/20	2/29/20
Intermediate Experiments	3/1/20	4/4/20
Network Nodes	4/5/20	9/5/20
Mesh Node	4/5/20	9/5/20
Case Installation	4/5/20	4/11/20
ZeroMQ Socket Program	4/19/20	9/5/20
Ad-Hoc Networking Support	4/5/20	9/5/20
Configure Raspbian Linux (Previously OpenWrt)	4/5/20	5/2/20
Configure BATMAN	4/27/20	9/5/20
Sensor Nodes	4/19/20	9/5/20
Case Installation (w/ Camera/Sensors)	8/23/20	8/29/20
Stream Camera Feed	4/19/20	5/9/20
Obtain/Interpret Sensor Data	8/30/20	9/5/20
Battery and System Monitoring	8/23/20	9/5/20
User Interface	4/5/20	11/14/20
Data Model	4/5/20	5/9/20
Nodes	4/5/20	4/18/20
Stream	4/12/20	5/9/20
Node Detail Page	4/19/20	10/17/20
Data Visualization (Video, Sensor, Network Data)	4/19/20	9/5/20
System Monitoring Indicators	9/6/20	9/12/20
Node Information Display	9/6/20	9/19/20
Node Information Configuration	10/4/20	10/17/20
Network Layer Services (Backend Communication)	9/13/20	10/17/20
Audience Display	9/20/20	10/3/20
Node Selection Page	10/18/20	11/14/20
Graphic Design	11/1/20	11/14/20
Network Master Node	4/5/20	11/7/20
Case Installation	4/5/20	4/11/20
Routing Table Configuration	5/3/20	5/9/20
Connect/Disconnect Nodes	8/30/20	9/19/20
Host UI Component	10/18/20	10/24/20
Host Backend Component	10/25/20	11/7/20
Backend	9/6/20	10/17/20
User Documentation	11/15/20	12/12/20

Figure 5: Gantt Chart CprE 492 (08/23 - 10/17)



Name	Begin date	End date
Gathering Requirements	1/19/20	2/1/20
Early Experiments	2/2/20	2/29/20
Intermediate Experiments	3/1/20	4/4/20
Network Nodes	4/5/20	9/5/20
User Interface	4/5/20	11/14/20
Data Model	4/5/20	5/9/20
Node Detail Page	4/19/20	10/17/20
Network Layer Services (Backend Communication)	9/13/20	10/17/20
Audience Display	9/20/20	10/3/20
Node Selection Page	10/18/20	11/14/20
List Active Nodes	10/18/20	10/31/20
Topological Network Graph	11/1/20	11/14/20
Graphic Design	11/1/20	11/14/20
Icons	11/1/20	11/14/20
Network Master Node	4/5/20	11/7/20
Case Installation	4/5/20	4/11/20
Routing Table Configuration	5/3/20	5/9/20
Connect/Disconnect Nodes	8/30/20	9/19/20
Host UI Component	10/18/20	10/24/20
Host Backend Component	10/25/20	11/7/20
Backend	9/6/20	10/17/20
User Documentation	11/15/20	12/12/20
User Manual	11/15/20	12/12/20
Lesson Plans	11/29/20	12/12/20

Figure 6: Gantt Chart CprE 492 (10/18 - 12/12)

Our Gantt chart reflects several primary goals. Most importantly, we are pushing to complete the initial ad-hoc networking prototype by the end of CprE 491 and finalize it in the first two weeks of CprE 492 (see Figure 4, Figure 5). This requires the proper configuration of Raspbian and BATMAN, as well as the streaming and forwarding of data in the network. Prioritizing this goal will help us take our biggest risks first, when there is less pressure. Another large risk is the video stream prototyping. There are many possible configurations for video codecs (H.264, H.265, etc.) and browser-based streaming techniques (WebRTC, etc.), but a lot of testing was required to determine that WebRTC is our strongest option. Once we get the network prototype and streaming working, we anticipate that later tasks will be more straightforward.

We are also being cautious with the development of our user interface and backend (see Figure 5). These tasks will require a lot of coordination, especially when different team members are working on each part. This will also require some work on the network node software, i.e. to obtain the data requested or set the desired node configuration. We have distributed the UI (frontend) and backend work from early September to mid October. In some cases, the frontend development supersedes the backend development (i.e. the main audience display will be completed before the sensor data microservice is created). In these scenarios, both frontend and backend developers will collaborate to determine an interface beforehand, so that the frontend developers can test their UI with mock data following the interface pattern. The risk of having to change both UI and backend due to implementation issues on the backend is small, as these microservices will be very similar in their implementation (relying on ZeroMQ to communicate with nodes).

The end of our timeline (Figure 6) consists of UI improvements and production of two items of documentation: lesson plans and a user manual. Putting these tasks last helps us avoid the

possibility of a last-minute technical flaw upending the entire project. These tasks also serve as an assessment of our project goals. By writing our user manual, we will be able to assess whether our project is easy enough for an instructor to use. By writing the lesson plans, we will be able to assess whether our project has real teaching value. It will be helpful to keep these goals in mind throughout CprE 491 and CprE 492.

4.2 Feasibility Assessment

Mesh networking technology seems to be abundant for the Raspberry Pi 4. Once we are able to find a software configuration that works (i.e. Raspbian and BATMAN, Raspbian and OSLR, etc.), we will be able to proceed with less difficulty. The Raspberry Pi 4 hardware is powerful enough for networking and streaming video and will be able to fulfill what is required of it. Similar to mesh networking, streaming video is a matter of finding the right combination of codecs/software to achieve acceptable latency. The task of exhausting the possible options is relatively simple; we will be able to stretch into the summer if absolutely necessary to complete these tasks.

Frontend and backend implementation are similar to the mesh networking in their feasibility, but easier as multiple team members have prior experience in web development. Other requirements, such as the two hour battery life, will be simple to meet, as USB power banks already exist with the necessary capacity. Generally, our design is feasible as long as we try to keep to our schedule; the relative thinness of our final weeks will help if some tasks take longer than anticipated.

4.3 Personnel Effort Requirements

Table 1 shows our estimates of hours required to complete the major tasks of the project.

Task	ETC (hrs)	Task	ETC (hrs)	Task	ETC (hrs)
Configure Raspbian	10	Miscellaneous Data Reporting (CPU, RAM, Sensor, etc.)	18	Assemble Battery Pack	2
Configure BATMAN	20	Program to Stream Video	6	Assemble Nodes	2
Build UI for Master Node	40	Build Backend Component for Master Node	8	Build microservices for network control/data	10
Route video from mesh network to UI	10	Create lesson plans	8	Create user manual	8

Table 1: Task Estimated Time to Completion Table

4.4 Other Resource Requirements

This project will only need materials and parts to build each node. Each node consists of a Raspberry Pi 4, an SD card, a rechargeable battery pack (including the charger), and a case. In

addition to that, the Sensor Nodes will also have an additional sensor package; some will also have the Raspberry Pi Camera Module attached. To accommodate the camera, the Sensor Nodes will reside in a case that includes a spot to hold the camera.

4.5 Financial Requirements

The budget for the project has been estimated at around $500.00. We plan on developing a system of eight nodes, at a price point of $60.00-$65.00 per node. This would put the expected cost of the system at $480.00-$520.00, meeting our budget. The cost of each node will vary depending on what type of accessories, if any, that they will have. Some may include a camera module or other sensor, which could make the cost of the node higher.

5. Testing and Implementation

5.1 Interface Specifications

Any testing that requires a UI component will be conducted using the actual implementation of the user application. As more hardware features are implemented in the project, the user application will be updated to accompany said features. For example, when the nodes in the network are able to report their CPU usage, the UI will be updated to fetch and display CPU usage for selected nodes.

Additionally, connections among nodes in the network that function independently of the user application can be tested using third-party tools. For example, video streams between Sensor Nodes and the Network Master Node can be opened and viewed using VLC.

5.2 Hardware and software

Three Raspberry Pi 4 Complete Starter Kits were obtained from CanaKit for the testing of the mesh network as well as a Raspberry Pi camera module. 3 Raspberry Pis were purchased as to allow for a more complete network rather than simply 2 connections. The camera module was purchased to test the streaming functionality .

The software used for testing the mesh network is BATMAN. Spring Boot is used to send data between the frontend and backend, ZeroMQ can send shell commands from an html page, and raspivid will be used to control the camera. We will be using either RTSP or UV4L for streaming the camera data over a WebRTC connection. Angular will be used for the frontend GUI.

5.3 Functional Testing

Our functional testing will include unit, integration, system, and acceptance testing. Unit tests will involve testing specific functions in the Ad-Hoc Networking domain and UI/Control Program domain. Integration testing will be focused on the Network Master Node and the functional units that interconnect the Ad-Hoc Networking and UI/Control Program domains. System and acceptance testing will focus on testing the whole system's functionality. Below are some tests that we have performed and/or intend to perform as our implementation proceeds.

Unit Tests:

- Sensor Node camera streaming command produces an uninterrupted feed (without reporting buffering to the console)
- Functions on Mesh Nodes to retrieve sensor data, network statistics produce valid outputs
    - This will include power consumption and temperature/other sensor data, packets received and sent by each Pi, etc.
- Backend functions that handle UI inputs produce correct messages to network and/or correct responses to user
- Backend functions that handle incoming data from the network return valid results
- BATMAN configuration of all nodes is "correct"
    - A correct configuration will consist of the settings we originally programmed into the node

Integration Tests:

- Spring Boot backend receives commands from GUI and forwards them to the Mesh Nodes correctly
- All connected Mesh Nodes can send back sensor data and network statistics to Network Master Node
- Video streams are properly forwarded to the backend and appear on the display
- A single Mesh Node, when booted, pairs correctly with its Network Master Node; mesh network connectivity can be verified between the two (using a simple utility such as ping)

System Tests:

- Ensure all Mesh Nodes connect automatically to their associated Network Master Node when booted and within range
- Ensure all nodes indicated as available in the GUI are reachable from the Network Master Node (using ping)
- Ensure all Mesh Nodes reconnect to Network Master Node when brought into range after being isolated and disconnected
- Ensure commands executed to obtain network statistics and sensor data result in accurate updates from the Mesh Nodes to the GUI
- Ensure video feeds from all Sensor Nodes with cameras can be initiated and viewed from the UI/Control Program
- Ensure all sensor data from Sensor Nodes is visible in the UI/Control Program graphs and charts
- The automatic network connections should occur if all nodes are powered off and powered on again later (powering off the nodes should not leave them in a bad state that prevents the network from starting properly the next time)

Acceptance Tests:

- The instructor must not be required to use any command-line tools to begin an experiment
- Video must have a high enough frame rate and resolution so that students are able to see one another on the feed and appreciate the real-time changes in the stream when the surroundings on camera are manipulated

- The instructor must only need to connect to the Network Master Node's network and open the web app from their computer station to begin

5.4 Non-Functional Testing

Our non-functional testing will assess the project on its performance, security, usability, and compatibility. To test performance, we plan on verifying not only the capability of the Raspberry Pis within our network, but also the network performance when interference from outside networks is present. For example, the performance of our master network node will be tested by confirming that it can support the maximum number of nodes determined for a network configuration, and the network itself will be tested by confirming that outside Wi-Fi, Bluetooth, or other radio signals will not cause a disrupting amount of interference.

To test security, we plan on running vulnerability scans and performing penetration tests to confirm that our network cannot be attacked by outside malicious sources. To confirm the usability of the product, we plan on performing user group testing, by having members of the target group use our software. By doing this, we can confirm that the user facing side of the project is not too difficult to operate. Compatibility of the project will be tested via cross-browser testing. This will determine if the user can open the front-facing graphical user interface from any browser of their choosing.

5.5 Process

For the mesh network, BATMAN's configuration wiki was used to configure the mesh network. Each node was configured with the same mesh SSID, channel, frequency, and interface types to ensure that they would connect to each other. A bridge interface was created that would connect the wlan to a BATMAN interface so that BATMAN could handle the transmission of packets. OLSR was also tested by following the configuration wikis on OLSR's website

For the frontend, we have done a WebRTC test to show an RTC video stream on a simple web page. We have done tests to stream video from the Raspberry Pi camera module to a VLC player instance via RTSP.

For the backend, we have prototyped an initial Spring Boot application featuring a simple web page where you enter a command and a node identifier where when you run a command the specified node will run the shell command.

5.6 Results

After trying multiple different configurations for the mesh network for both BATMAN and OLSR on OpenWRT, we could not get the network to work. We decided to move over to Raspbian as there seemed to be more support for it on Raspberry Pis that was written for the most modern version of BATMAN-adv. After updating the nodes, we were able to see some limited success with the network, as now the wireless interface can communicate with BATMAN, however an external wireless adapter was required. We could not test inter-node connecting as we did not have more than one wireless dongle.

Our WebRTC demo yielded reasonable latencies and has been deemed acceptable for use in our project. RTSP streaming attempts to VLC player showed relatively high latency; RTSP may be difficult to convert to a web-appropriate streaming format for our frontend.

Communication between the Network Master Node and mesh nodes via ZeroMQ in Spring Boot (Network Master Node side) and Python (mesh node side) has been prototyped and tested. The shell commands entered in the simple web UI are sent to the corresponding mesh node (specified by an identifier string), where they are executed and the stdout/stderr result strings are transmitted back. The Spring Boot server prints these to the console. The Network Master to mesh communication is done via a publish/subscribe ZeroMQ socket, and the mesh to Network Master communication is done via a push/pull ZeroMQ socket. Due to the publish/subscribe architecture, all mesh nodes receive all command messages from the Network Master and must parse the identifier string to determine if they should execute them. This is easily scalable, as the network master node only needs one publisher socket to communicate with all mesh nodes and one pull socket to receive from all mesh nodes. However, it may be beneficial to establish true 1-to-1 communication with each node, which will require a different socket configuration.

6. Closing Material

6.1 Conclusion

At this point in our project, we have outlined the problem, proposed a solution, and outlined the requirements of our solution. We have also obtained initial hardware to begin researching and testing the implementation of our design. The project has been broken down into different components and we have explored and settled on a design that best fits our needs.

By the end of the Fall 2020 semester, we will have a set of nodes that communicate via an ad-hoc network and provide an interface so that users can view data generated by each node. We will also have a set of lesson plans that help encourage users to gain an understanding and solve problems that a network may encounter.

6.2 References

[1] *NS-3 | A Discrete-Event Network Simulator for Internet Systems*, 2020. Accessed on: Mar. 29, 2020. [Online]. Available: https://www.nsnam.org/

[2] E. Biagioni, "A Network Testbed for Ad-Hoc Communications using Raspberry Pi and 802.11," in *Proc. of the 52nd Hawaii International Conference on System Sciences*. Accessed on: Mar. 29, 2020. [Online]. Available: http://hdl.handle.net/10125/60191

[3] S. Sharma and M. Nekovee, "Demo Abstract: A demonstration of automatic configuration of OpenFlow in wireless ad hoc networks," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Accessed on: Mar. 29, 2020. [Online]. doi: 10.1109/INFCOMW.2019.8845307

6.3 Appendices

None used.