# CS Capstone Winter Progress Report

February 16, 2018

# **Project ROUS**

Prepared by

## Group 46

## Rodents Of Unusual Size


Hayden Anderson _____ _____

*Signature* *Date*

Kyle Prouty _____ _____

*Signature* *Date*



**Abstract**

This document will give an overview of the project, the progress we have made, and each group members individual workload over winter break and the first five weeks of winter term. It will begin by describing the projects purpose and the goals we set out to achieve. The next sections will cover each individuals current progress, what they have left to do, the problems they encountered with solutions, and finally any other relevant information. Lastly, there is a group conclusion.

# CONTENTS


**1 Project Overview** 3

1.1 Purpose . . . 3

1.2 Framework Design . . . 3

1.3 Group Goals . . . 3

**2 Hayden** 3

2.1 Current Progress . . . 3

2.2 To Do . . . 4

2.3 Problems Encountered . . . 4

2.4 Solutions . . . 5

**3 Kyle** 6

3.1 Current Progress . . . 6

3.1.1 Winter Break . . . 6

3.1.2 Project Setup . . . 6

3.1.3 Work-flow . . . 6

3.1.4 Logic Implemented . . . 6

3.1.5 Hardware . . . 7

3.2 Whats Left . . . 7

3.2.1 Make Bidding Robust . . . 7

3.2.2 Finish Message Encryption . . . 7

3.2.3 Complex Objectives . . . 7

3.2.4 NodeJS Web Server . . . 7

3.2.5 Make Configuration Robust . . . 8

3.2.6 One-to-One Printer Relationship . . . 8

3.3 Problems Encountered . . . 8

3.3.1 Bidding . . . . . 8
3.3.2 Stateless Environment . . . . . 8
3.3.3 Solutions . . . . . 8
3.4 Other Relevant Information . . . . . 8
3.4.1 Client Demo . . . . . 8
3.4.2 Moving Forward . . . . . 9

**4 Conclusion** 9

4.1 Overall Summary . . . . . 9

# 1 Project Overview

## 1.1 Purpose

The framework being developed will create a network of nodes that will communicate by broadcasting simple messages to each other. Nodes will use network protocols to pass these messages in order to self organize on an objective. Users will be able to input print job objectives into our framework and view the results from a graphical user interface.

Overall the rationale for the entire framework is to have a collection of loosely coupled nodes that can self organize to accomplish a print job. This framework will be robust in the face of any nodes that become untrusted. Nodes in this framework will communicate with each other using simple messages over wireless network protocols.

## 1.2 Framework Design

The ROUS framework being implemented will be organized into three parts: software, hardware, and a graphical user interface. These separate pieces will work together to make up the overall framework. This system will have the functionality to be given a print job objective, organize on that objective, and then accomplish that print job objective.

In this framework all communication happens by broadcasting simple messages. A user interface will provide users, observers, and administrators the ability to interact with the framework. The framework will contain a network of nodes that communicate by broadcasting simple messages.

## 1.3 Group Goals

There are a lot of end goals we have for our system. For the system to work as we intend, it shall:

- Enable nodes to self organize on a structured objective
- Allow an objective to be input
- Allow a source of mistrust to be input
- Output readable configuration data
- Output readable system state data
- Output readable results of objective
- Automatically validate and connect to authorized nodes
- Allow nodes to share objectives with each other
- Framework state changes based on the input of a source of mistrust
- Have a user interface
- Offer a print service

# 2 Hayden

## 2.1 Current Progress

During this term, I focused on a few key elements for the project: the administrative side, the Graphical User Interface (GUI), and testing. Our client, Lonnie, requested that we use a Scrum approach for our software development workflow. Using Scrum, I have kept a backlog of tasks for Kyle and I to work on using a Slack plugin called Workstreams and an application on my phone called Scrum App. Using these applications has been tremendously helpful for our workflow and setting goals each week when we meet. By utilizing these apps, our team is able to stay on task, meet deadlines, and keep a close eye on any outstanding tasks or tasks that we are awaiting further information.

I have also made progress on the GUI. Originally, I started using basic HTML, CSS, and PHP. I quickly moved away from that and started working on the website using a Nodejs server and trying to figure out how to use just Nodejs for the front end. Eventually I came around to creating a react application running on top of a Nodejs server. The GUI contains a title bar at the top welcoming the user to the ROUS system. Above the welcome is an open source image of a beaver that rotates 360 degrees. This currently serves as an attention grabber in the middle and may eventually be pushed off to the side to take up less space or be removed entirely. The main function of the GUI are the buttons in the center that say ”LED ON” and ”LED OFF”. These buttons are intended to be used to work with the Raspberry Pi’s and the ROUS system to allow the user a visually pleasing way to insert objectives into the system.

Lastly, I am creating tests for the project. I haven’t created a large number of tests for the project yet, however I have made one test to ensure that the printing service script works correctly. This is critical for testing early on to ensure we are meeting the project deliverables. Additionally, keeping the system modular has been one of my highest priorities to help with the final testing phase. By keeping the code modular, unit tests can focus on one block at a time ensuring that the individual parts work and then progress testing to include all parts working together.

### 2.2 To Do

While there has been a lot accomplished on the project, there is also a lot more of the project to go. Specifically, there is work that needs to be accomplished on the GUI in order to send signals to the ROUS system as well as present the data I receive from the system. Furthermore, the interface needs to be streamlined and be able to showcase all of the implemented services for the user to choose. I also need to make an administrator view that is more of the information presenter that will showcase the system and the nodes connected. The plan is to create a layout with the left side of the screen containing a general log of the system as a whole. On the right, will be a layout of each node connected to the system and their individual details. I also plan to implement an ”untrust” button that the administrator can press to state that the node indicated is no longer a trustworthy node. There will be a lot of dynamic positioning on the page to account for the vast difference in amount of nodes the system can contain at once. We have nine Raspberry Pi’s to utilize for our testing. The GUI will need to be capable of showing at least eight concurrent nodes successfully at a time and in a visually pleasing order.

Creating a space for both the user and the administrator to work seamlessly will require me to create a seamless dynamic movement system for all of the objects that will be on screen. Due to the dynamic nature of the ROUS project, many things are always changing. This means I need to do a plethora of manual unit testing to ensure the elements shown on screen do not overlap and present the correct data.

Moving forward, the administrative work consists of regularly reserving rooms, email correspondence, and keeping an updated backlog. For our weekly meetings, we meet with our TA, our client, and when we work on the project together. There are many restrictions when trying to reserve a room on campus so I need to make sure I check every avenue of project and study rooms available. Most of the time when emailing, the recipient is our client. The check-ins with our client, Lonnie, change from week to week depending on his needs and our meeting agenda. There are times when he comes to campus for an in person demo and other times that we meet over Skype to have a Scrum review. These meetings are necessary to maintain communication with the client and internally as a team. We are able to have decisions made, time for feedback, and have any and all questions answered.

My day to day tasks involve updating the project tasks, communication with the client, and communication within our team. I constantly need to keep up with the workflow and keep our backlog up to date. As our progress continues, there are more items to document in the backlog. I also need to add more tasks to our list to complete. I have the general tasks added, but when we encounter an error by the end of our sprint, I add it to the queue to be fixed in the next sprint.

### 2.3 Problems Encountered

Many of the problems I have encountered thus far have consisted of server issues, the application relaying messages with the host server, and my initial lack of knowledge with JavaScript, Nodejs, and React as a whole. Specifically, I was having problems with consistently updating the backlog and making sure the group was on the same page, and getting started with Nodejs and React.

The majority of my server issues stemmed from coding issues and inconsistencies with tutorials used to learn how the application and server components were supposed to interact with each other through code. My messages to the server were not making it through properly and time was wasted using just Nodejs and trying plenty of workarounds. For example, one week we would be working on just getting things connected, and the next we found a whole bunch of issues with the methods we used. Many of the decisions made early on in the project caused further problems which created new doubts whether creating new tasks was worth it when we may have to take a step back and try a different approach. Another problem I struggled with was working with React. React is much more object-oriented than I am used to programming for web interfaces. This created a huge learning curve that I am still working through. The change from programming to a more object-oriented method has been a challenge.

### 2.4 Solutions

I should have started with React and modularized my method of thinking. If I had focused on just the web interface, I would have saved a lot more time, effort, and stress. I tried to do everything at once and it got me nowhere because I was focused on the connection to the server. I have made it a priority to consistently keep up with the backlog to create a clear path forward to complete the project and meet the client deliverables. Finally, I solved my inexperience with React by going through some online tutorials and getting some help from peers that have experience programming in React.

# 3 KYLE

## 3.1 Current Progress

### *3.1.1 Winter Break*

Winter break was fun and productive, I learned a lot but in the end we abandoned all the work I did because we realized I was heading down the wrong direction. Over winter break I experimented and implemented the community of nodes, setting them up in an ad-hoc network. This required a lot of work that was unnecessary to the projects goal, mainly having to use low-level routing protocols that not all wifi cards support. We realized that ad-hoc networking was not giving us the benefits we thought it would and moved back to having the nodes just sitting on a router in infrastructure mode using regular dhcp protocols to determine IPs and do other setup stuff.

### *3.1.2 Project Setup*

When setting up the project I researched what a normal project setup and work-flow looked like for organizing a python package. I thought about how code would be organized and how to make navigating the code and project directory straight forward. By doing this initial setup I hope that it will make maintaining and understanding the code easier.

### *3.1.3 Work-flow*

The problem with working on a community of nodes where every node in the community is running the same code, is that every code change needs to be updated on every node. To make this process easier on myself I implemented a nice work-flow. I have an authorized ssh key on every node so that I don't have to enter a password when coping files or connecting to the nodes via ssh. This allowed me to create a script that automatically copies my updated changes to each of the nodes with just a one click script. This makes deploying changes quick and easy and speeds up development.

### *3.1.4 Logic Implemented*

To start the most basic logic implemented was the ability for each node to be able to listen and receive all messages communicated over UDP via a multi-cast address. Along with the functionality to listen and receive messages, I implemented the ability for nodes to send messages to a multicast address. This basic functionality was at the heart of the entire framework. This gives nodes the ability to be stateless. They do not need to know about other nodes, they can just listen for messages that other nodes send, figure out for themselves what to do with those messages and then either responded, or perform some action to obtain a result.

The next major piece of functionality implemented was the ability to filter messages. Since every node that is listening on the multicast address will see every message sent on that multicast address, meaning that, as a node, I will see my own messages along with messages from any node regardless of trust. To fix this issue a filtering system via a blacklist was implemented. The main reason for the blacklist is to implement a system of trust. With the blacklist a node can untrust another node. Since every node sees every message, a node would still receive all messages, even from an untrusted node, but now, that node can just tell itself to ignore those messages. This system also helps the node ignore itself so that it doesn't see it's own messages.

After a message is filtered to check for the trust of the sender, the node checks the message to see if it can perform the service that the message contains. If the node does have that service, it then enters the bidding phase.

The next big feature I implemented was the bidding system. This is one of the biggest features in the system and it demonstrates giving the nodes the ability to socially collaborate on an objective. When a node has determined that it can perform a service received in a message, it then enters the bidding phase. In this phase the node, rolls a bid, then spawns a thread to send that bid so that it can instantly listen for bids from other nodes. Since the entire system is built to be stateless, I have no way to know how many bids I will get. This adds complication into the issue. Since I don't know how many bids I will get, my option was to wait for a time to live amount of time. Once the time to live is over, I

look at the bids I receive, and I look at my own bid, then from that information I can decide if I was the winner. If I was the winning node, I perform the objective and confirm it with the user. If I lost the bid, I do nothing.

Other then these major features there is a lot of little logic like configuration files, helper functions, utility functions, and build files. We are currently demonstrating functionality using LED's. There is a lot of logic that goes into making them turn on. The RPi LED code needs to run in its own thread, so there is a lot of code to properly keep track of threads and manage the threads created, so I can destroy them when needed. The LEDs act as printer to simulate that functionality, since the printer will work in a one-to-one relationship, the LEDs currently work in that same one-to-one relationship.

#### *3.1.5 Hardware*

I just wanted to touch on the hardware. We are implementing this community of nodes on three different models of raspberry pis, a pi two, two, and zero w. For expo, since wifi will be an issue, we will remove the pi zero w and replace them with pi twos or threes so we have the ability to hard wire them with ethernet cables to the router. Other then the pis themselves, the only other hardware is the router that they all connect to and communicate through.

Using the RPi's GPIO pins we are using LEDs to demonstrate printer functionality. That requires the LED's, wires, breadboard, and some small resistors.

### 3.2 Whats Left

#### *3.2.1 Make Bidding Robust*

The bidding works and helps us to keep moving forward with the project. It works and fixing it, to work 100 percent, would probably require me to write a thesis. So I plan to keep testing and refining the code so that at expo we will have a product that performs to the best that time allows. As far is our clients concerned, he thinks that there is no way for it to be perfect and some level of un-robustness is expected.

#### *3.2.2 Finish Message Encryption*

I have started a system of message encryption. This will give the system the ability to only see specific messages. It also removes the ability for outside nodes to see the messages. With this encryption I am implementing a system where different actors are given different keys. As an example if you are a node you would get the key to send node message but would not have the key to send administrator messages. This would give us the functionality to allow the allow administrators to send un-trust node messages. If both admin and node could send these messages then the trust would be meaningless. Only admins have right to un-trust nodes now while nodes don't have those rights, they just know what nodes they don't trust. This is the begging of grouping users and messages to expand functionality.

#### *3.2.3 Complex Objectives*

One of the goals of our system is to be able to handle complex objectives. That means that I can send a complex objective to a node and he can delegate out all of the tasks inside the complex objective. So far I have a semi working logic to accomplish this goal. I need to finish testing my solution and make sure it is robust.

#### *3.2.4 NodeJS Web Server*

Overall Hayden is handling the front end user interfaces. Hayden is doing a lot of the GUI interface work, while I am developing the background server. This server will not have much functionality other then; hosting a web server that runs Hayden's react app, allowing Hayden to make get and post requests to get data and send in data, and to listen on the multicast address for information messages from the nodes. This multicast listener will only listen for specific messages that contain data that Haydens react app will then use to display it to the user in a nice and clean browser interface.

#### 3.2.5 Make Configuration Robust

In order to add one our one-to-one printer setup, I am going to expand out current configuration setup to make adding outside services easier and cleaner. The goal of this will be to allow these outside services to be added easily so that added functionality and services is painless and straightforward.

#### 3.2.6 One-to-One Printer Relationship

I think that once I get a more solid configuration setup that finishing the one-to-one printer relationship will be a breeze. The system was currently designed to toggle the LEDs in a one-to-one relationship way. This will be the same sort of way that a printer will be implemented so adding this feature should go quick.

### 3.3 Problems Encountered

#### 3.3.1 Bidding

Overall the biggest issue I have had is when it comes to bidding. The bidding has a lot of issue to think about. What happens if I don't receive all the bids so no one wins? What happens if I no bids come in because the timing is off? There are a lot of race conditions and unforeseen issues that come up when the system is stateless and has no sense of time. I lot of time went into testing and re-testing different features such as the time to live and the amount of nodes. Moving forward, bidding works, and I will continue to look for, and find solutions for bugs and edge cases that pop up.

#### 3.3.2 Stateless Environment

The fact that every node acts in a stateless way where it only relies on information it has or can ask for brings up a lot of issues to think about. Nodes only have the ability to send and receive messages, and then self determine what to do with those messages. They interact with their community by bidding on the service they received with other nodes in the system. But they don't know about the other nodes, they can only ask questions and receive responses. This adds a fair amount of thought to be added to how each interaction will work.

#### 3.3.3 Solutions

The main solution to some of the problems with bidding I think is going to be to confirm with the user when a node has one a bid. This will remove the cases where all the nodes lose and forces them to rebid. It also ensures that the objective only has to get sent once and the user gets a confirmation that a node will be caring out the objective.

Solving the stateless environment issue just requires a lot of thought about how the functionality will be implemented. Since there is no concept of time or forced sequence, then a lot of thought and experiments have to be made. Testing, testing, and re-testing to find the random bugs that pop up.

### 3.4 Other Relevant Information

#### 3.4.1 Client Demo

Next week we are hosting our client on campus and giving him a demo of the current functionality. This forces us both to polish up our alpha stage system and make sure it is functional for a demo. It also forces us to think through issues that we want to discuss that could direct the project down a better path. Along with that comes an outside viewer that can notice and suggest things that we have not thought of. This will also give us a good indication that the progress we have made so far is good, and that we are in a good place moving into the end of the term.

#### *3.4.2 Moving Forward*

Moving forward we are in a good place. We have a solid plan for what we will have done by the end of the term. We communicate weekly with our client and have a good relationship with him. Hayden and I communicate well and I feel we work well together. We continue to improve our poster and I set up a poster review with another team.

## 4 CONCLUSION

### 4.1 Overall Summary

Overall we have made a lot of good progress towards our end goal this term. Our system allows nodes to enter and exit the network freely. It allows nodes to communicate with each other by being able to listen for messages and then response, and to also send messages themselves. Node can filter messages from un-trusted nodes, check if they offer the service, and if they do, bid on the chance to complete that service objective. Bidding adds our social collaboration by demonstrating that a community of nodes can work together to accomplish a goal. With just that logic the foundation for all of the rest of the functionality is setup.

Moving forward, we both have a list of goals to complete before the end of this term. We plan on having a complete beta by the end of the term that demonstrates the goals set out in our requirements document. In the end, this term has been productive, we are keeping up on the work-flow and helping each other with ideas, moving forward we are in a good place to finish the term out with a robust system.