An Advanced Networking Outreach Activity for Kids

Client/Advisor: Dr. Tom Daniels
Team: Grayson Cox, Austin Dvorak, Malcolm Johnson, Ryan Newell, Spencer Parry, Ross Thedens

Problem/Project Statement

- Wireless networks are now commonplace
- K12 students use wifi regularly
  - Aren't taught mechanics behind it
  - Opportunity to raise interest in computing fields
- How to teach children about wireless technology?
  - Connect intangible signals to observable phenomena
  - Visually demonstrate functions with GUI

- Portable wireless network nodes
  - Relay nodes
  - Camera nodes
  - Network master node
- User application
  - Show network statistics and sensor data
  - Network configuration

Functional Requirements

- Nodes form a wireless network
  - No setup required
  - Nodes can connect, disconnect, and reconnect without user input
  - No internet connection required
- GUI Web Application
  - Connect to Network Master Node from instructor computer
  - Display list of connected nodes
  - Stream video and sensor data from nodes

Non-Functional Requirements

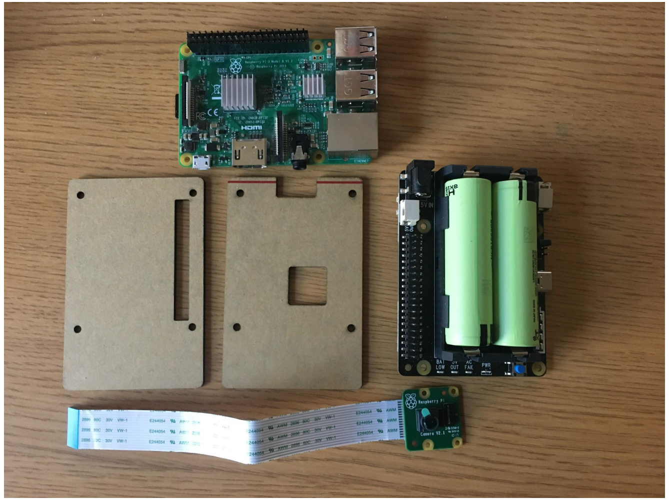
- Low latency network stream (<2 seconds)
- Durable nodes, enough to handle low (<5 feet) drops
- Interference tolerance without severely affecting performance of the network
- Robust and simple enough to use
- Appropriate for grade students and their instructors
- 30 foot line-of-sight connection range
- >2 hours of battery life on full charge and not require disassembly to charge

Literature Survey

- Sharma and Nekovee
  - OpenFlow, virtual nodes
  - GUI controller with video
- Biagioni (University of Hawaii)
  - Raspberry Pi 0W with onboard WiFi
  - AllNet protocol, simple data transfer

Resource Requirements

- Total cost around $960 (6 nodes)
- Should be used in an environment that will cause signal loss (such as a building with multiple rooms or a large field)
- A laptop to connect to the Master Network Node

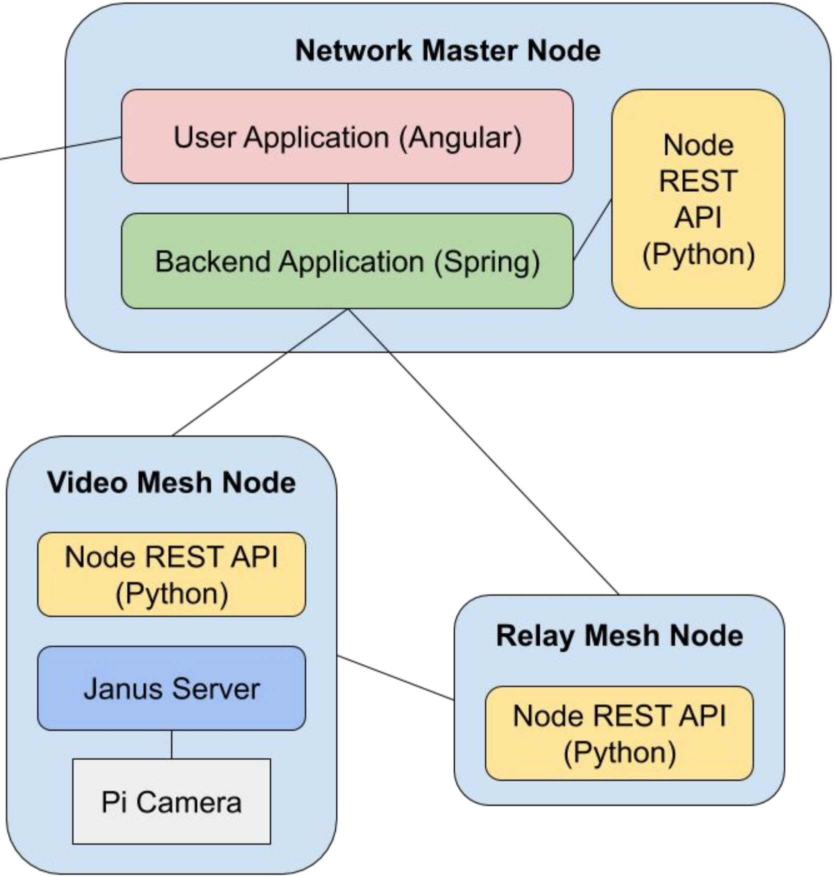


Risk Analysis & Mitigation

- Moving to virtual instruction and meetings
  - Meetings were more difficult when you cannot meet in person
  - Using tools such as Zoom and Google Drive helped mitigate this risk
- Performance limitations of Raspberry Pi
  - Using H.264 codec helped with the performance issues
- Compatibility issues with various browsers
  - WebRTC is well supported and provided low latency
  - Janus Web Server used due to minimal issues with Firefox

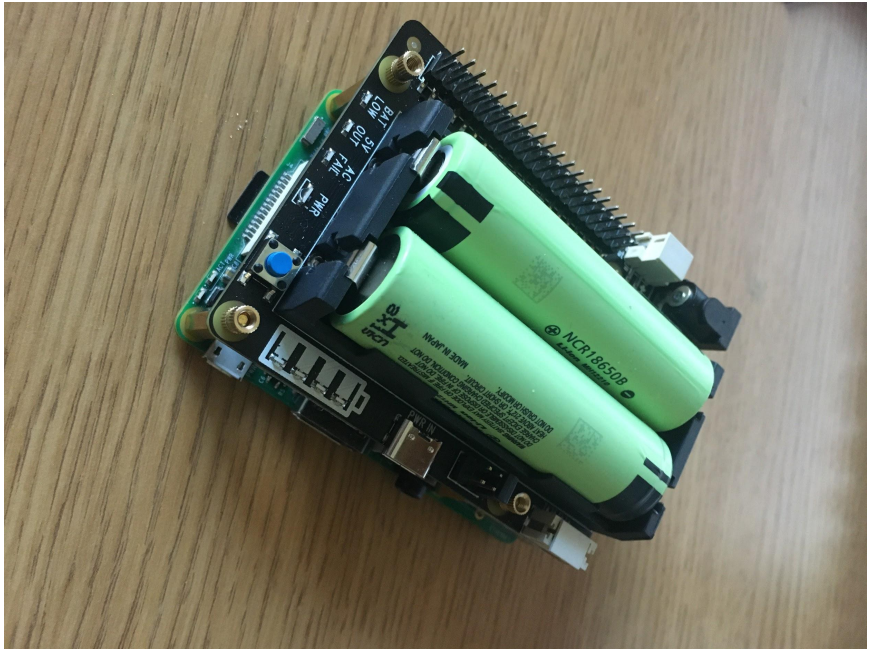
System Analysis

- Network Master Node
  - Hosts User Application
  - Controls Mesh Network
- Relay Mesh Node
  - Forwards packets among the network
- Video Mesh Node
  - Captures live video with Pi camera
  - Serves video using WebRTC



Hardware Design

- Node Base
  - Raspberry Pi 4 Model B
  - UPS Raspberry Pi Hat w/ Batteries
  - Case
- Video Node
  - Includes a camera and camera mount
- Case
  - Customizable to fit the Raspberry Pi and UPS in same case
  - Camera mount is easily attached
- UPS Hat
  - Allows nodes to operate wirelessly
  - Batteries can be charged without taking them out of the nodes



Hardware Testing

- Battery Lifespan
  - Power lasted 3+ hours
  - More than enough power for our requirements
- Durability of case & node
  - Unable to test because of timing conflicts
  - Replace with enclosed case that fits the UPS shield in the future
- System Startup
  - Checks that each node boots up and can connect to the mesh network
  - Each node has been confirmed to be working as intended

Mesh Network Design

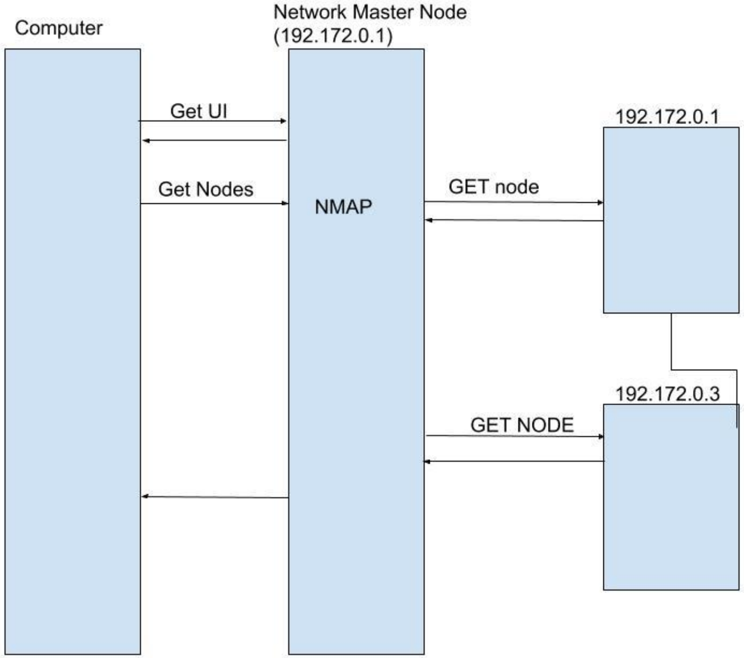
- Built on BATMAN-adv
- Wrote startup script to pull information from a configuration file
    - Allows the UI to call a Flask api that modifies the configuration file
    - Can change network name, ip address node name and type

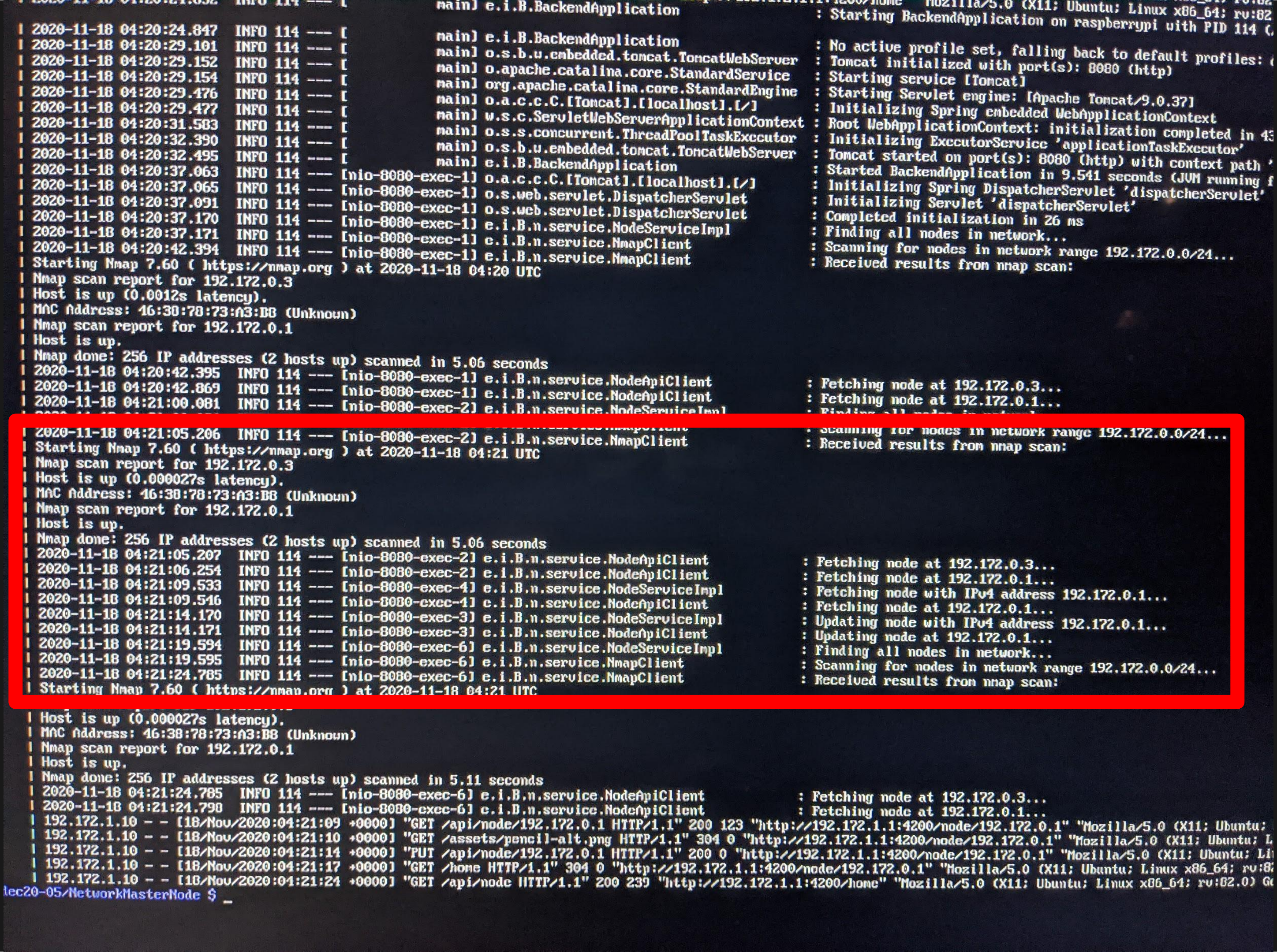
- Wireless Access point that a user's computer can connect to
    - HOSTAPD

Mesh Network Testing

- Basic Pings to verify functionality

- Future Plans
  - More rigorous testing
  - Check bandwidth, speed, interference, latency.
  - Connect Mesh interface to Wireless AP to allow for ca
    streaming




: Starting BackendApplication on raspberrypi with PID 114
2020-11-18 04:20:24.847 INFO 114 --- [main] e.i.B.BackendApplication : No active profile set, falling back to default profiles:
2020-11-18 04:20:29.101 INFO 114 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-11-18 04:20:29.152 INFO 114 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-11-18 04:20:29.154 INFO 114 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.37]
2020-11-18 04:20:29.476 INFO 114 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-11-18 04:20:29.477 INFO 114 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
2020-11-18 04:20:31.583 INFO 114 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-11-18 04:20:32.390 INFO 114 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path
2020-11-18 04:20:32.495 INFO 114 --- [main] e.i.B.BackendApplication : Started BackendApplication in 9.541 seconds (JVM running
2020-11-18 04:20:37.063 INFO 114 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-11-18 04:20:37.065 INFO 114 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-11-18 04:20:37.091 INFO 114 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 26 ms
2020-11-18 04:20:37.170 INFO 114 --- [nio-8080-exec-1] e.i.B.n.service.NodeServiceImpl : Finding all nodes in network...
2020-11-18 04:20:37.171 INFO 114 --- [nio-8080-exec-1] e.i.B.n.service.NmapClient : Scanning for nodes in network range 192.172.0.0/24...
2020-11-18 04:20:42.394 INFO 114 --- [nio-8080-exec-1] e.i.B.n.service.NmapClient : Received results from nmap scan:
Starting Nmap 7.60 (https://nmap.org) at 2020-11-18 04:20 UTC
Nmap scan report for 192.172.0.3
Host is up (0.0012s latency).
MAC Address: 46:30:78:73:A3:BB (Unknown)
Nmap scan report for 192.172.0.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 5.06 seconds
2020-11-18 04:20:42.395 INFO 114 --- [nio-8080-exec-1] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.3...
2020-11-18 04:20:42.869 INFO 114 --- [nio-8080-exec-1] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.1...
2020-11-18 04:21:00.081 INFO 114 --- [nio-8080-exec-2] e.i.B.n.service.NodeServiceImpl
2020-11-18 04:21:05.206 INFO 114 --- [nio-8080-exec-2] e.i.B.n.service.NmapClient : Received results from nmap scan:
Starting Nmap 7.60 (https://nmap.org) at 2020-11-18 04:21 UTC
Nmap scan report for 192.172.0.3
Host is up (0.000027s latency).
MAC Address: 46:30:78:73:A3:BB (Unknown)
Nmap scan report for 192.172.0.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 5.06 seconds
2020-11-18 04:21:05.207 INFO 114 --- [nio-8080-exec-2] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.3...
2020-11-18 04:21:06.254 INFO 114 --- [nio-8080-exec-2] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.1...
2020-11-18 04:21:09.533 INFO 114 --- [nio-8080-exec-4] e.i.B.n.service.NodeServiceImpl : Fetching node with IPv4 address 192.172.0.1...
2020-11-18 04:21:09.546 INFO 114 --- [nio-8080-exec-4] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.1...
2020-11-18 04:21:14.170 INFO 114 --- [nio-8080-exec-3] e.i.B.n.service.NodeServiceImpl : Updating node with IPv4 address 192.172.0.1...
2020-11-18 04:21:14.171 INFO 114 --- [nio-8080-exec-3] e.i.B.n.service.NodeApiClient : Updating node at 192.172.0.1...
2020-11-18 04:21:19.594 INFO 114 --- [nio-8080-exec-6] e.i.B.n.service.NodeServiceImpl : Finding all nodes in network...
2020-11-18 04:21:19.595 INFO 114 --- [nio-8080-exec-6] e.i.B.n.service.NmapClient : Scanning for nodes in network range 192.172.0.0/24...
2020-11-18 04:21:24.785 INFO 114 --- [nio-8080-exec-6] e.i.B.n.service.NmapClient : Received results from nmap scan:
Starting Nmap 7.60 (https://nmap.org) at 2020-11-18 04:21 UTC
Host is up (0.000027s latency).
MAC Address: 46:38:78:73:A3:BB (Unknown)
Nmap scan report for 192.172.0.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 5.11 seconds
2020-11-18 04:21:24.785 INFO 114 --- [nio-8080-exec-6] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.3...
2020-11-18 04:21:24.790 INFO 114 --- [nio-8080-exec-6] e.i.B.n.service.NodeApiClient : Fetching node at 192.172.0.1...
192.172.1.10 - - [18/Nov/2020:04:21:09 +0000] "GET /api/node/192.172.0.1 HTTP/1.1" 200 123 "http://192.172.1.1:4200/node/192.172.0.1" "Mozilla/5.0 (X11; Ubuntu;
192.172.1.10 - - [18/Nov/2020:04:21:10 +0000] "GET /assets/pencil-alt.png HTTP/1.1" 304 0 "http://192.172.1.1:4200/node/192.172.0.1" "Mozilla/5.0 (X11; Ubuntu;
192.172.1.10 - - [18/Nov/2020:04:21:14 +0000] "PUT /api/node/192.172.0.1 HTTP/1.1" 200 0 "http://192.172.1.1:4200/node/192.172.0.1" "Mozilla/5.0 (X11; Ubuntu;
192.172.1.10 - - [18/Nov/2020:04:21:17 +0000] "GET /home HTTP/1.1" 304 0 "http://192.172.1.1:4200/node/192.172.0.1" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
192.172.1.10 - - [18/Nov/2020:04:21:24 +0000] "GET /api/node HTTP/1.1" 200 239 "http://192.172.1.1:4200/home" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:82.0)
lec20-05/NetworkMasterNode $ _

User Application

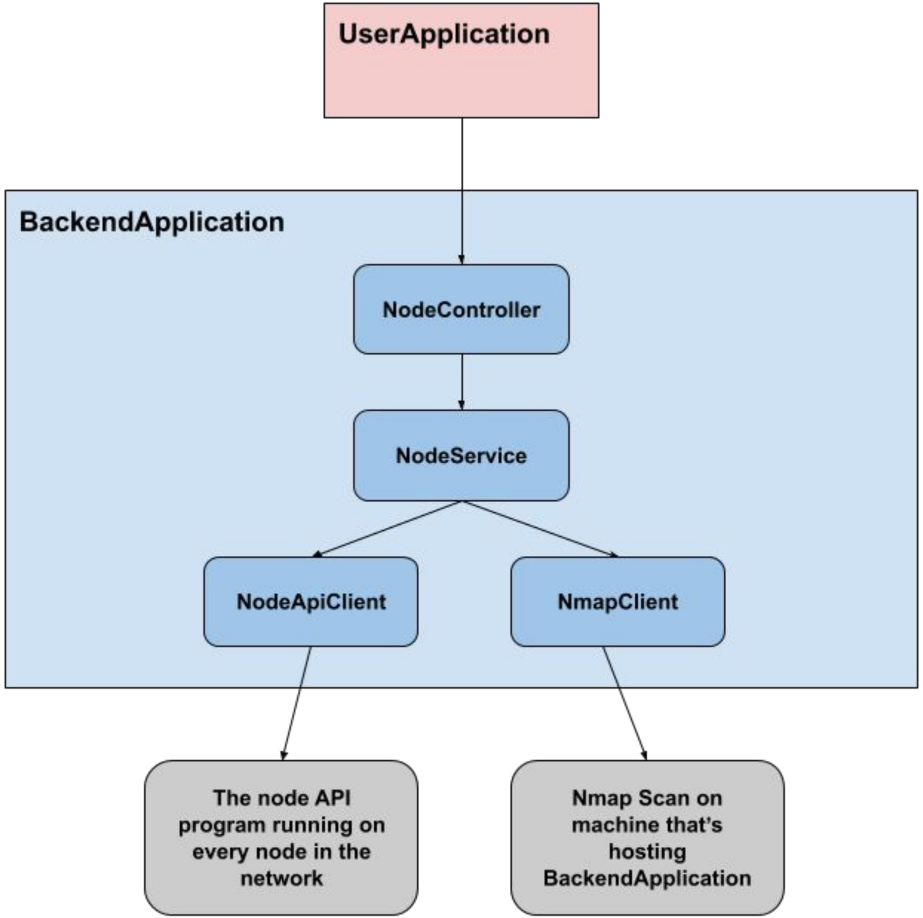
- Web-based application
- Visualizes live and static data from network
  - Network topology
  - Network statistics
  - Video streams
- Technologies used:
  - Angular
  - Docker
  - WebRTC

User Application Testing

- Test Plans
  - User Application can receive data from the Backend Application
  - Docker must build both the User Application and the Backend Application with one command
- Test Results
  - User Application received both mocked and real data from the Backend Application
  - Using Docker Compose, we were able to build the User and Backend Application with one command
- Conclusions
  - Keep in mind the performance constraints of a Raspberry Pi
  - More dependencies mean more time building and take up more space

Backend Application

- Simple REST service for accessing and controlling the Mesh Network
- Hosted on the Network Master Node
- Functions implemented:
  - Get all nodes in network
  - Get static node properties by IP address
  - Update a node's properties
- Technologies used:
  - Spring Boot
  - Nmap
  - Docker



Backend Application Test Results

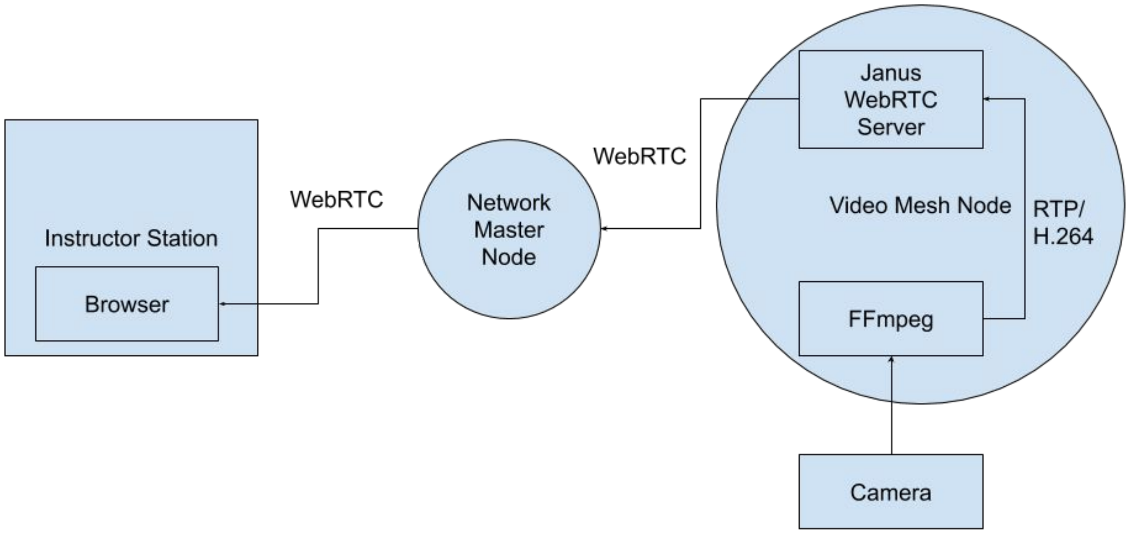
- Serving data to User Application
  - Total success
- Sending requests to Node REST API
  - Successful but sometimes unreliable
  - Hindered by issues connecting to Node REST API
- Discovering nodes in the Mesh Network
  - Successful but very slow because of Nmap
  - Sometimes unable to fetch node data because of issues with Node REST API
- Unit Tests
  - 100% code coverage (excluding model classes)
  - 100% test acceptance

Backend Application Retrospective

- Takeaways
  - Nmap wasn’t the best idea.
  - Exposure to Docker and docker-compose
  - Experience working with engineers of different disciplines
- Suggestions for future groups
  - Use something other than Nmap to perform network scans.

Video Streaming Design

- Stream Pi camera feed to browser page
- Two Components
  - FFmpeg
    - H.264 video encoding
    - RTP forwarding
  - Janus WebRTC server
    - Stream to Browser
    - JavaScript API
- Deployed with Docker



Video Streaming Test Results

- Test video to browser page
    - Successful test
    - Set bitrate, resolution, framerate

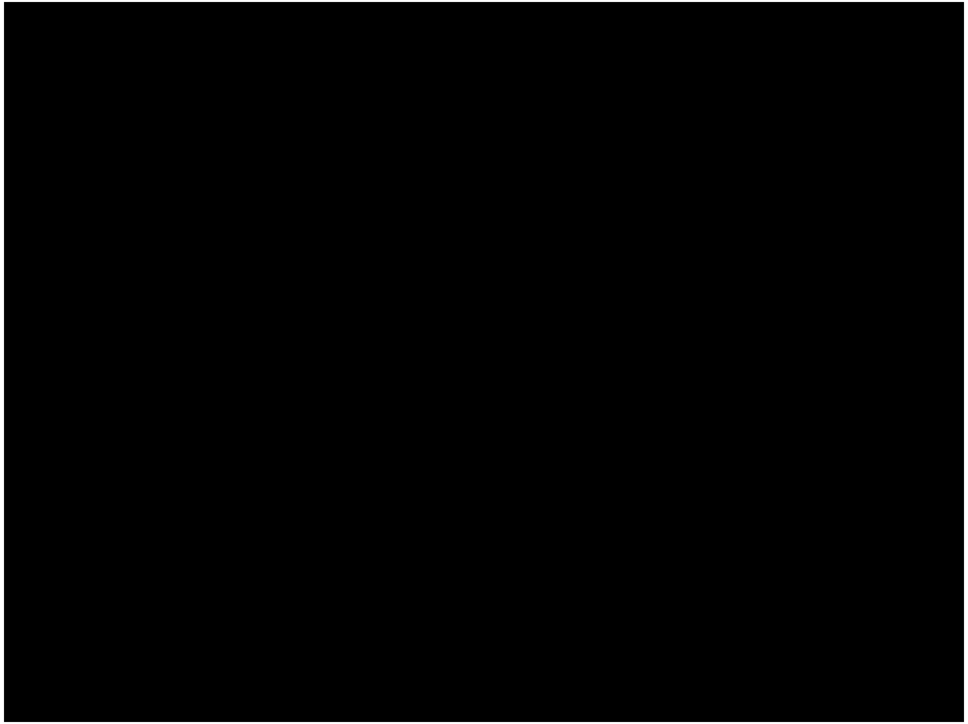
- Test video through mesh network
    - Unsuccessful test
    - WebRTC communication interrupted
    - Possible network bridge issue

Network bitrate	Video bitrate	Video framerate	Video resolution
200-2500 kbps	~500 kbps	15 FPS	480x320

Video Streaming Retrospective

- Lessons
  - Start integration early
  - Ask critical questions

- Future Work
  - Fix bridge issue



Thank you

References (from Literature Survey)

- 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

- 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