EE / CprE / SE 492 – sddec20-05

An Advanced Networking Outreach Activity for Kids

Bi-Weekly Report #1
08/17/2020 – 08/28/2020
Client & Faculty Advisor: Dr. Tom Daniels

Team Members

Grayson Cox | UI Developer | Agile Project Manager
Austin Dvorak | Network Systems Manager
Ryan Newell | Hardware Systems Admin
Spencer Parry | UI Developer
Ross Thedens | Communication Systems Manager | Meeting Secretary

Reporting Period Summary

This was our first reporting period for the semester, so we spent some time planning meetings and discussing our communication strategy to deal with restrictions caused by the pandemic. We have decided that meeting remotely would work best because some of us are working already and commuting to campus would be too inconvenient. We anticipate having in-person meetings on select occasions such as the weeks in which we have to submit iterations of our design document.

Reporting Period Accomplishments

- Got mesh network set up and all nodes connected
  - Switched over to generic raspbian image while still using batman-adv for packet handling
  - Optimized configuration to fit onto a single, 20 line bash file
- Got low-latency WebRTC demo stream working using Raspberry Pi Camera, Janus WebRTC Server, and FFmpeg
  - Installation process and setup for Janus
    - Several packages needed to be build from source due to version requirements
    - Learned what configuration files need to be edited to create a video stream
  - FFmpeg found to be superior to VLC and GStreamer for H.264 streaming from the Pi

        - Tests were done using the Janus WebRTC streaming video, streaming video from the camera module on the Pi to a web browser (over localhost, LAN, and WLAN interfaces)
        - VLC and GStreamer caused the stream to freeze
            - Stream appeared okay for ~1 second, then only a few frames over the next ~5 seconds, then a complete freeze
            - Tried reducing framerate and resolution, down to 1fps and smaller than 100x100
            - Explored possibility of different browsers performing differently (Chrome vs. Firefox)
            - Tried adjusting various network buffers without success
        - FFmpeg worked reliably, with a consistently high framerate
        - All options were tested without using the closed-source UV4L utility often mentioned in WebRTC tutorials for Raspberry Pi
            - We are on track to complete the project without having to involve any closed-source/proprietary software
    - Experimented with and studied Janus WebRTC API.
        - Found appropriate JavaScript library for video streaming client. Still in the process of adapting it to work in an Angular application.

Pending Issues

- Decide how network configurations will be handled
- No easy way to install dependencies for Janus/WebRTC streaming on Pis yet
    - Planning to solve this problem with Docker

Individual Contributions

Team Member	Contribution	Reporting Period Hours	Total Hours
Grayson Cox	• Coordinate schedules to choose meeting times. • Plan out repository structure for user application. • Studied JanusGateway API. • Work on video streaming portion of user application.	12	12
Austin Dvorak	• Got mesh network setup and working	12	12

	● Created bash file with required commands		
Ryan Newell	● Organized meeting schedule with team ● Researched methods for easy configuration of devices ● Experimented with methods for easy imaging	12	12
Spencer Parry	● Worked with team to coordinate schedules ● Researched UI enhancements that could be to our benefit	12	12
Ross Thedens	● Tracked down and installed Janus dependencies ● Tested/Evaluated VLC, GStreamer, and FFmpeg with Raspberry Pi & camera	12	12

Plans for Next Period

- Have working prototype of video streaming portion of user application.
  - Be able to display a video stream from the Janus server using the Janus Gateway API for client video streaming.
- Add configuration support to network bash configuration file
- Start work on visualization of network through batman-adv/alfred
- Prepare Docker container for easy installation of Janus and related dependencies on all nodes
- Develop camera messaging (turn on/off stream, support stream multiplexing) via ZeroMQ
- Begin developing interface for UI to access network statistics
  - Determine what API is available and what interfaces will have to be implemented to make statistics easily accessible to UI
  - Further develop ZeroMQ messaging system to access data from mesh nodes
- Research and procure temperature sensors
- Research viable battery configuration that fits requirements