CPRE 491 WEEKLY REPORT 11

Project Molecule

15 – 28 November 2016

May1739
may1739@iastate.edu

Dr. Arun Somani

Ryan Wade – Team Leader

Nathan Volkert – Communications Lead

Daniel Griffen – Key Concept Holder

Alex Berns – Webmaster & Scribe

1 CONTENTS

2 WEEKLY SUMMARY

Team members worked towards getting the project demo ready. Progress was made on interfaces between the different layers via a standard message format. We discussed which components were necessary for a demo, we decided that we needed a working UI, bonding and atomic layer. For the purposes of the demo the particle layer will just be a very thin shell with almost no functionality.

3 PAST WEEK ACCOMPLISHMENTS

All Members:

- Worked on documentation
- Weekly group meeting + advisor meeting
- Discussed Presentation and work needed before then

Ryan Wade:

- Message Format
- Dynamic UI Builder
- Worked on Documentation

Nathan Volkert:

- Worked more on the Particle API
- Created File Particle

Daniel Griffen:

- Created molecule-common repository, contains types relevant to all molecule components
- Tested cross-compilation and static binary building

Alex Berns:

- Redux Server

4 INDIVIDUAL CONTRIBUTIONS

NAME	Hours this week	Hours cumulative
Ryan Wade	12	79
Nathan Volkert	12	62
Daniel Griffen	11	64
Alex Berns	12	63

5 COMMENTS AND EXTENDED DISCUSSION

5.1 USER INTERFACE

A first version of the redux client server system was created. With this, when a change is requested to be made to a redux store on one view, it will send that action to the server to inform all other active participants to update their data. Actions are handled by a queue to help reduce conflicting actions from occurring. New clients are sent the action queue to recreate the current state, and still receive any new actions that are performed. The next set of steps is to create the save states for the UI to role back to incase of conflicts.

A dynamic UI builder was also created. This takes state data from redux to generate the UI at runtime. It also binds UI actions to redux actions.

5.2 PARTICLE LAYER

The first version of the particle layer has been mostly constructed. With this it sets up a common interface for the particle layer. This sets things up with some basic usages.

New creates a new particle instance to be called and listened to by the bonding layer.

There are get commands as well for the bonding layer to call as well to get the ID and available commands each specific particle has access to. This is to ensure only a particle and the socket to the bonding layer can learn/know the functionality so rogue nodes will not have power over the whole system. The user must approve functionality on particle installation.

Lastly the Particle layer communicates to the bonding layer via messages (explained below). This updates the bonding layer on any events that happen, successful completion of tasks and the like.

5.3 BONDING LAYER

Tested cross compiling for Raspberry Pis and tested creation of static binaries. Also worked on creating the common message API structures. The message API will be used throughout the program to communicate to different parts of the system. Messages have the following format:

Message

Message Header (Fixed Length)

Global ID: low collision error, pseudo random
Destination = <Particle,Instance> Signature, Broadcast, or Generic
Source = Signature
Action = String
Response Actions = count or null
Data Type = Stream (pipe name) or Packet

Message Response Actions

Message Packet or Stream

6 PLAN FOR COMING WEEK

This coming we will meet again as a team and with Dr. Somani. In this time we plan to finalize our plans for the fall demo. We also will begin linking the parts we all have been working on together and physically test out what we have so far. Lastly we will be preparing, practicing, and then presenting next week Wednesday.

7 SUMMARY OF WEEKLY ADVISOR MEETING

The one portion of this week's meeting had us looking over the various diagrams we have created and reviewing them with Dr. Somani.

The rest was devoted to discussing more about fault tolerance. We debated the implementation for voting consensus we have planned and how to avoid deadlock and livelock. Dr. Somani recommended looking into checkpointing for recovering from system failures. We were also pointed into looking at graceful degradation, the use of past behavior to predict trust, as a method to resolve deadlock. If one node has a history of going against consensus, it should be flagged. If a deadlock is reached, and the system can find no other method of resolution, remove the suspicious node from voting.