CPRE 491 WEEKLY REPORT 09

Project Molecule

01 – 07 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

This week we examined the User Interface particle in depth to refine system architecture and determine our development path. This work informed how particles communicate with each other and what permissions different kinds of particles have. Using the architecture, we divided development tasks for the coming week. We also divided work on the design document and worked on it throughout the week.

3 PAST WEEK ACCOMPLISHMENTS

All Members:

- Discussed User Interface and system architecture

Ryan Wade:

- Administration
- Worked on System Architecture (User Interface and Particles)
- Worked on Design Doc

Nathan Volkert:

- Started work on particle API
- Worked on Design Doc

Daniel Griffen:

- Researched encryption solutions
- Worked on Design Doc

Alex Berns:

- Worked on dynamic UI state and events
- Worked on Design Doc

4 INDIVIDUAL CONTRIBUTIONS

NAME	Hours this week	Hours cumulative
Ryan Wade	8	49
Nathan Volkert	8	42
Daniel Griffen	8	46
Alex Berns	7	43

5 COMMENTS AND EXTENDED DISCUSSION

5.1 USER INTERFACE

We determined the User Interface (UI Engine) was a Neutron (System Resource Wrapper) which has permission to host a web server and send messages to event listeners. Electrons register UI panes with the UI Neutron. This pane state is synchronized throughout the system. Only one UI Neutron may be running on a node at a time. UI Neutrons running on different nodes may be viewing different panes, but if they display the same pane, their state must be the same.

Below is a flowchart showing how UI messages propagate throughout the system:

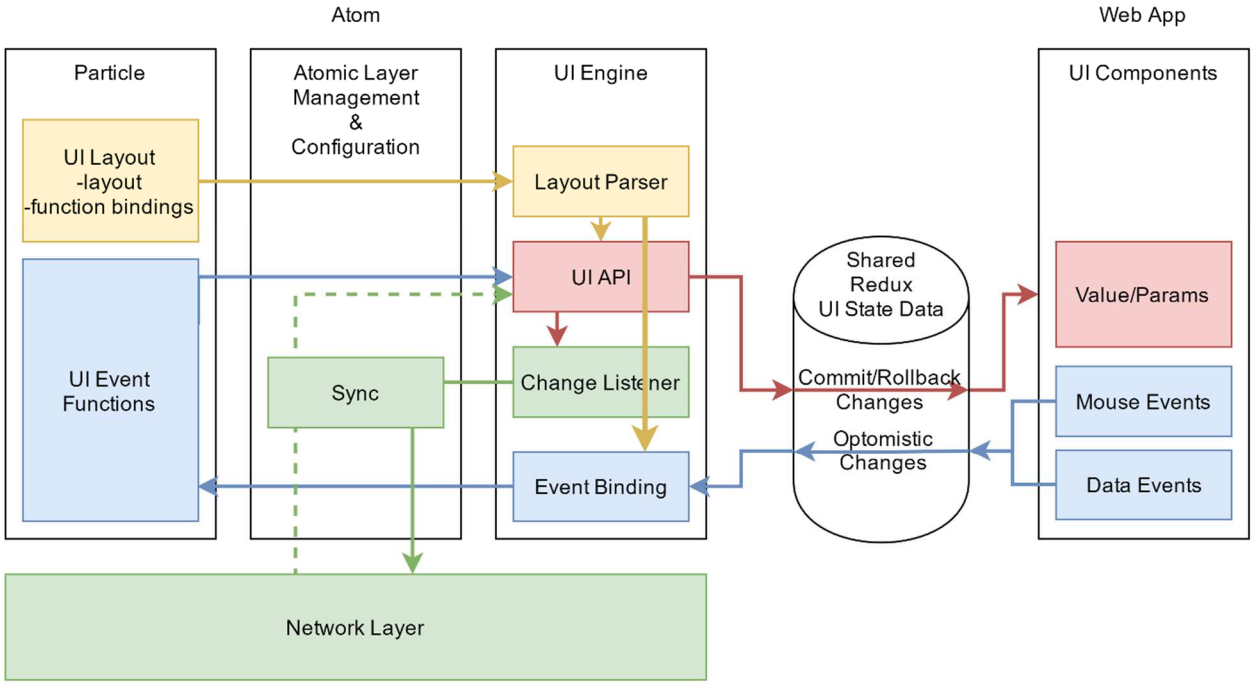

Figure 1: UI & System Architecture

Yellow: Electrons register Panes and Message Listeners (Event Bindings) with the UI Neutron.

Red: UI Neutron receives messages and updates the Web Apps state and user interface. Any changes to the state also fire the synchronization Change Listener

Green: When the UI Neutrons state changes (red), the data is synchronized between all of the nodes. This will cause UI Neutrons on other nodes to receive similar UI Updates (green dotted line to red)

Blue: Web App UI action occurs (onClick, onDataChange) and optimistically changes the user interface. The UI Neutron sends a message to the listeners (Event bindings) registered from the Electrons (yellow). The Electron receiving this event processes the changes and then commits them to the UI Neutron's UI API (red)

5.2 Bonding Layer

https://gitlab.com/may1739-molecule/molecule-core

Bonding layer did not undergo many changes this week as most time was spent researching methods to encrypt the system. An early draft of the encryption design can be found in the design doc. When a connection is made between bonding layers the first thing that is done is the handshake. The handshake takes care of verifying the identity of the two nodes and determining the key used for encryption. After that messages will by encrypted by a stream cipher.

6 Plan for coming week

Using the UI & System Architecture diagram, we split up development goals for the coming week:

Alex: (red & yellow) Work on loading config and setting up redux state

Ryan: (blue and red arrow) web binding and rust to node communication

Daniel: (green) networking at the Bonding Layer and synchronization at the Atomic Layer

Nathan: Particle Box, Particle API

7 Summary of weekly advisor meeting

First we discussed the UI & system architecture (figure 1) that Ryan created and got initial feedback. In summary, we need to add textual description of the diagram (which is now included in this report). He also asked for clarification for ambiguous lines and processes in the diagram.

Given the diagram, we also revisited the use of a state machine for formal analysis of our design. This will help ensure our system does not get into an undesired state where permissions are averted. Arun pointed us back to our cpre 281 notes on how to effectively create a state machine diagram.

Daniel demoed the rust bonding layer prototype and described how it would be used for networking & system discovery. We used this to discuss the fault tolerance for the system and what happens when a node disconnects suddenly.

Lastly we talked about plans for the next week, as seen above.