CPR E / SE 492 BIWEEKLY STATUS REPORT 3

February 22nd - March 1

Senior Design Team 15

Debugger and Visualizer for a Shared Sense of Time
on Batteryless Sensor Networks

Client/Advisor
Dr. Henry Duwe

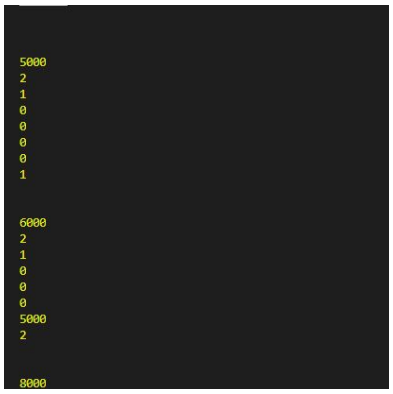
Team Members
Adam Ford - Report Manager
Allan Juarez - Scribe
Maksym Nakonechnyy - Design Lead
Anthony Rosenhamer - Facilitator
Quentin Urbanowicz - Test Engineer
Riley Thoma - Project Manager

Biweekly Summary

Over the past week, we have been incorporating more of our clients' feedback into the code we have written so far. This involved making changes to the trace file format for the simulator and working more on the interface for the client. We also have continued to develop data processing on the backend side to parse the trace files and generate the data needed to be visualized. On the frontend side, we have continued to develop the network visualization by switching the graphing library to a completely open source library as per our client's request.

Accomplishments from the Past Two Weeks

- Backend Team (Adam and Allan)
    - Setup trace file parsing framework and preliminary functionality
    - Batch inserted parsed objects to Mongo for future querying
    - Some contents of a trace file

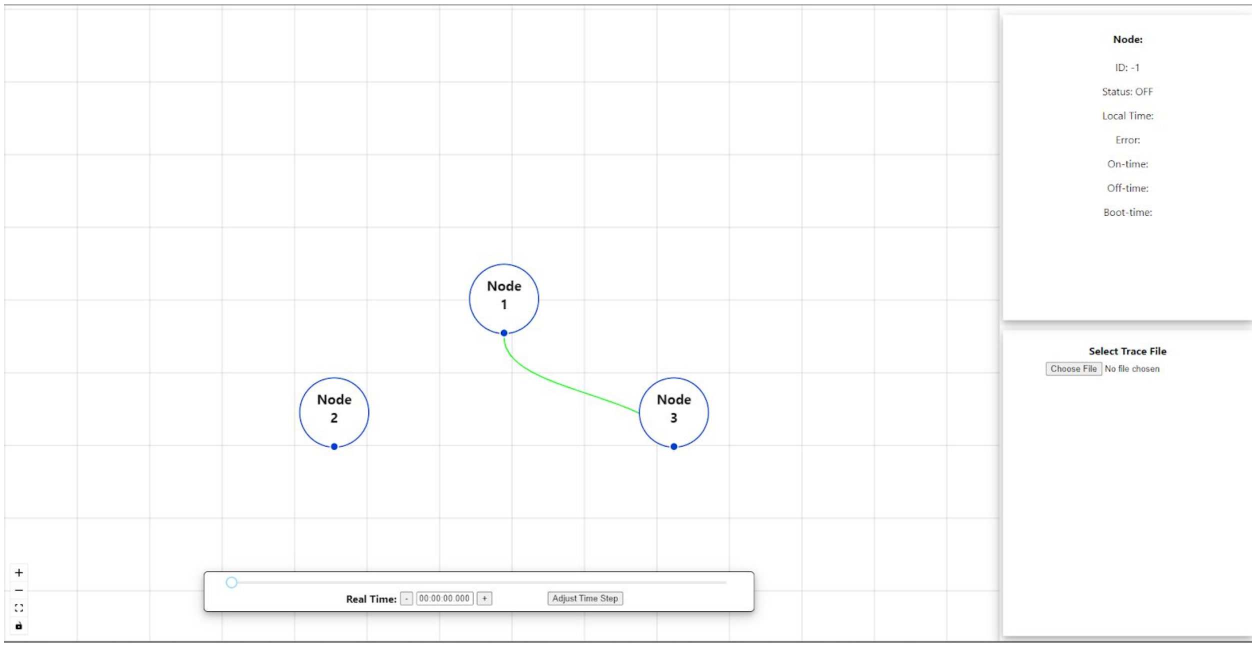


```
5000
2
1
0
0
0
0
1


6000
2
1
0
0
0
5000
2


8000
```

- Frontend Team (Maksym and Riley)
    - Switched the graphing library from GoJS to ReactFlow and redid the demo code again.
    - Switched out real time clock for a React component that will display time and allow it to be editable.
    - Current state of the frontend:



Node:

ID: -1

Status: OFF

Local Time:

Error:

On-time:

Off-time:

Boot-time:

- Simulator Team (Anthony and Quentin)
    - Fixed a bug with the timing handling so that communicating nodes shared the proper local times
    - Modified the trace file format according to the clients' recommendations
    - Began work on logic for loading network topology and initial state from a configuration file
    - Continued work to abstract node behavior logic for easier modification

Trace File Format

- The first byte indicates the trace file version to indicate how parsing should be handled
- The following bytes represent events occurring in the node network with variable lengths that depend on the event type

Event Types:

Share Time Request Event (35 bytes)						
Real time	Event type	Sender ID	Dest. ID	Sender Time	Dest. Time	Updated Time
8 bytes	1 byte	1 byte	1 byte	8 bytes	8 bytes	8 bytes
Share Time Response Event (19 bytes)						
Real time	Event type	Sender ID	Dest. ID	Updated Time		
8 bytes	1 byte	1 byte	1 byte	8 bytes		
Node State Event (19 bytes)						
Real time	Event type	Node ID	Updated Time			
8 bytes	1 byte	1 byte	8 bytes			

Pending Issues

The frontend team ran into a problem with the graphing library, GoJS, they were using. This library was a free version of proprietary software that could not be used for further distribution, which is what Dr. Duwe might want to do in the future. We then had to research an alternative and completely rewrite our current version of the frontend to use that alternative.

Individual Contributions

Name	Individual Contributions	New Hours (last week)	Total Hours
Adam Ford	Setup Trace File parsing framework, and inserted the parsed objects to MongoDB	6	31
Allan Juarez	Setup parsing method to read what is in the trace file and printing it out	6	31.5
Maksym Nakonechnyy	Rewrote the frontend app to use ReactFlow, a replacement for GoJS.	7	33
Anthony Rosenhamer	Updated the trace file format to include version data and flexible formats dependent on the event type Fixed local time handling to actively update node times while in the ON state	7	32
Quentin Urbanowicz	Began work on logic for constructing the simulated network's topology dynamically and setting its initial state with data from a configuration file. Continued abstracting and further modularizing node behavior logic for easier modification and implementation of custom behavior. Added documentation and performed various refactoring.	6	30
Riley Thoma	Finally found a suitable timer to use for the Real Time display and implemented that as far as I could. Played with styling of panels to work out if the UI our client wants would work well or not.	6	30

Plans for the Next Two Weeks

- Adam Ford - backend development
    - Rework Trace File parsing for new data format
    - Pseudo-code algorithm for tree production
    - Start writing tree production implementation
- Allan Juarez - backend development
    - Set up logic to parse through new trace data format
    - Storing events into mongo database
- Maksym Nakonechnyy - frontend development
  Note: my plans have not changed since the last report because I spent this week working on our issue with the library.
    - Finish implementing configuration file support.
    - Start implementing a node communication history panel.
    - Look into libraries to display network statistics.
- Anthony Rosenhamer - simulator development
    - Continue to modify the trace file format to include configuration details
    - Add more logic for handling the communication interface to allow nodes to communicate differently
- Quentin Urbanowicz - simulator development
    - Finish modularizing node behavior logic
    - Continue implementing dynamic construction and initialization of the simulated network of nodes using data from a configuration file
    - Continue researching and begin implementing a system for introducing pseudorandom variability into node behavior
- Riley Thoma - frontend development
    - Add node coordinate information to node information panel
    - Finish hooking up the Real Time timer component with the backend request
    - Modify the UI to fit our client’s needs for new graphs and diagrams to be displayed

Summary of Advisor Meetings

February 26th:

We discussed our planned Trace File format to ensure that it will cover all necessary information in a simulation. We received feedback to increase all time fields to be 8 bytes and consider having variable event types, rather than a single catch-all.

Additionally, we discussed some more frontend specifics, including the communication lines between nodes and a more specific layout of the application. This will aid the frontend team in finalizing libraries. We also covered the specifics of the graphs that would be displayed on the frontend in addition to the network structure visualization.