CHAPTER 08 Control Messages Visualization

8.1 Motivation

As the network size increases, the debugging process becomes a bit complex and tracking bugs causes can be quite difficult, therefore, to aid with the debugging process and to visualize the control messages and data transactions between the nodes, a GUI was built using JavaFX to demonstrate the flow of such messages between nodes, visualize how the stages the network go through to build a route between the source and the destination and demonstrate how the network behaves when a node is down.

8.2 Structure

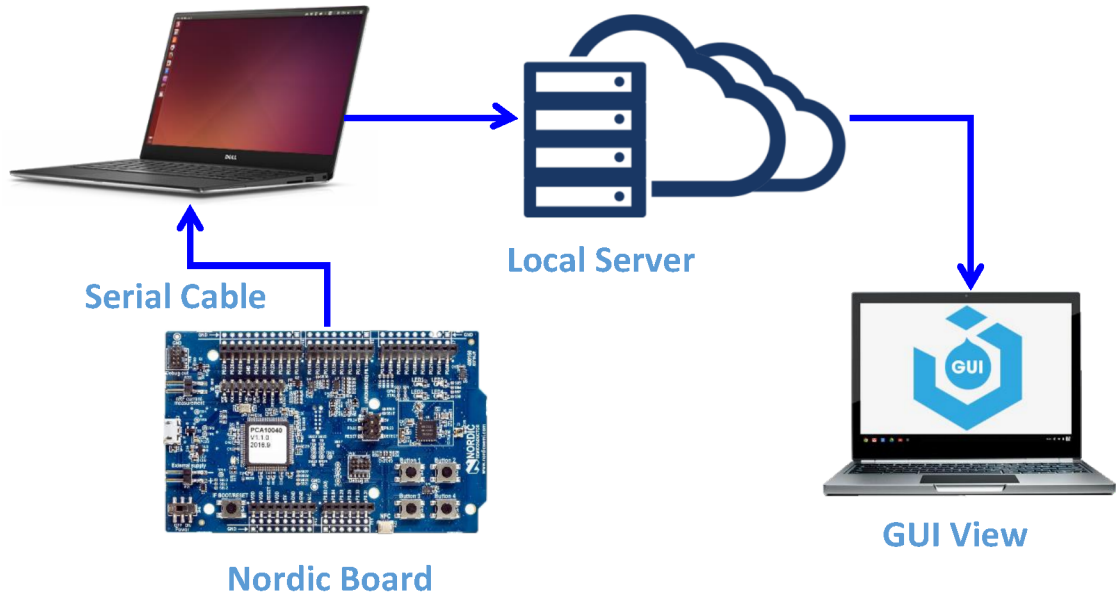

Figure 8- 1| Visualization Structure

The idea revolves around fetching the data from the Nordic boards serially then uploading it to a local server hosted on one of the laptops as shown in Figure (8-1). One laptop fetches the uploaded data from all the nodes in the network and undergoes various processes depending on the type of data such as:

1. Visualize the control messages and data transactions
2. Plot the throughput against time
3. Plot a pie chart of amount of send data versus overheads
4. Plot the traffic of each node
5. Plot a pie chart of amount of actual data versus the redundant (relayed) data
6. Organize the sent data from the sensors into a table.

All the aforementioned features are updated dynamically in run time whenever a node sends new data.

8.2.1 Local Server

The local server receives data from the laptops connected serially with the nodes, organizes them into one of the three tables and sends the fetched data to the laptop responsible for viewing the GUI. The local server implements a database of three tables. Each table is needed for a different purpose.

8.2.1.1 Graph Table

Source Address	Message	Destination Address	Timestamp

Table 8- 1| Graph Table

1. Destination Address: Lines and curves are drawn by the information provided by the message-receiving node not the message-sending one. This field holds the unicast address of the node reporting the reception of a message

2. Message: Holds one of the following [RREQ / DRREQ / RREP / RERR / Data]

3. Source Address: Holds the address of the node originating the message

4. Timestamp: Holds the message timing in milliseconds

8.2.1.2 Metrics Table

Address	Command	Data	Timestamp

Table 8- 2| Metrics Table

1. Address: Unicast address of the node reporting a metric defined in the command column.

2. Command: Holds one of the following metrics [Throughput – PktActual (representing actual data sent) – PktOverhead (representing overheads of the control messages) – PktRed (representing the redundant related data)]

3. Data: Holds the value of the metric defined in the command column.

5. Timestamp: Holds the metric timing in milliseconds.

8.2.1.3 Sensors' Table

Address	Type	Value	Timestamp

Table 8- 3| Sensors' Table

1. Address: Unicast address of the node reporting the sensors' data defined in the type column.

2. Command: Holds one of the following sensor types [Temperature - Pressure]

3. Value: Holds the value of the sensor type defined in the type column.

5. Timestamp: Holds the metric timing in milliseconds.

8.2.2 Graphical User Interface

The GUI windows consists of three main tabs:

8.2.2.1 Live Graph tab

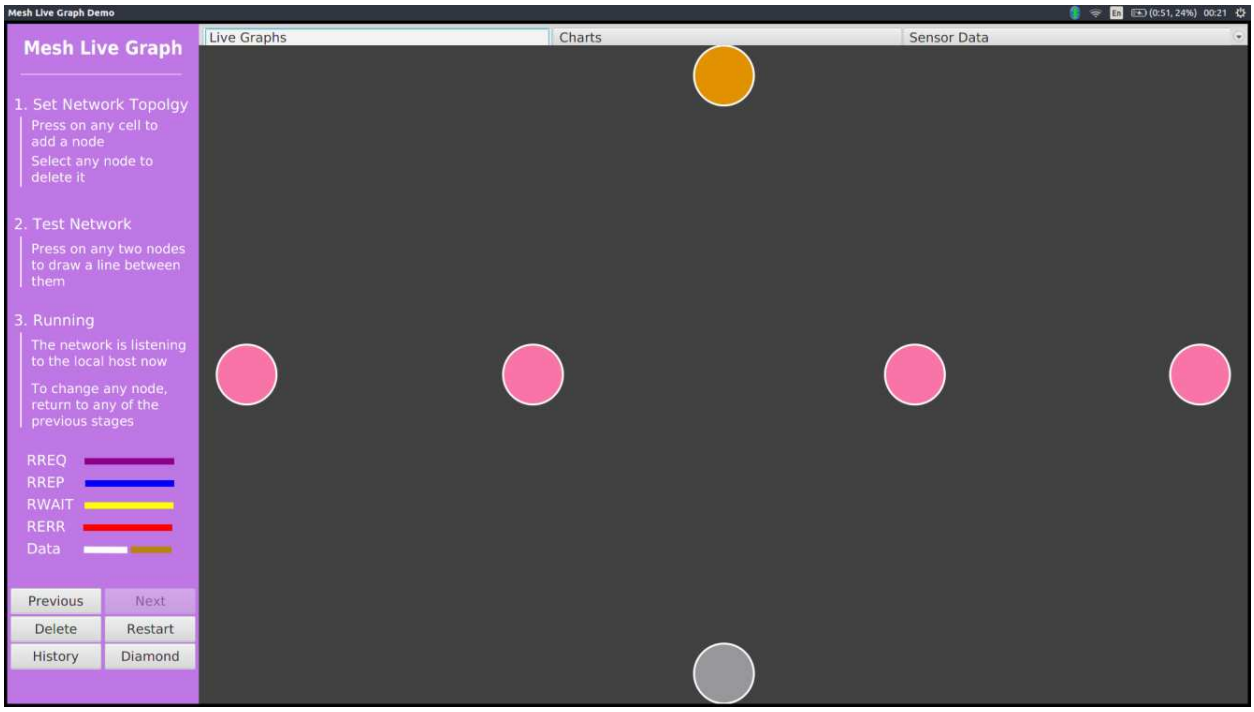

Figure 8- 2 | Live Graph Tab

Figure (8-4) shows how this tab looks like. It consists of three main stages:

1) Setting the network topology

- Here the physically constructed network by the Nordic boards is set to enable tracking of the control messages and data segments. The window is divided into cells representing the possible locations of the nodes. On pressing any empty cell, a new node will be added to the network and an information dialog will pop out asking for the unicast address of the node and its role in the network (i.e. Whether it is a central node, an intermediate node or a destination node). To delete a node, simply press on any of the pre-added nodes.
- To facilitate the network creation process, some pre-defined topologies are available which are:
  - Diamond Topology as shown in Figure (8-4).
  - Train Topology as shown in Figure (8-5).

  The aforementioned topologies accepts adding or deleting the pre-defined nodes as well. Alternating between the topologies can be done by pressing the custom button shown in the left bottom of Figure (8-4).

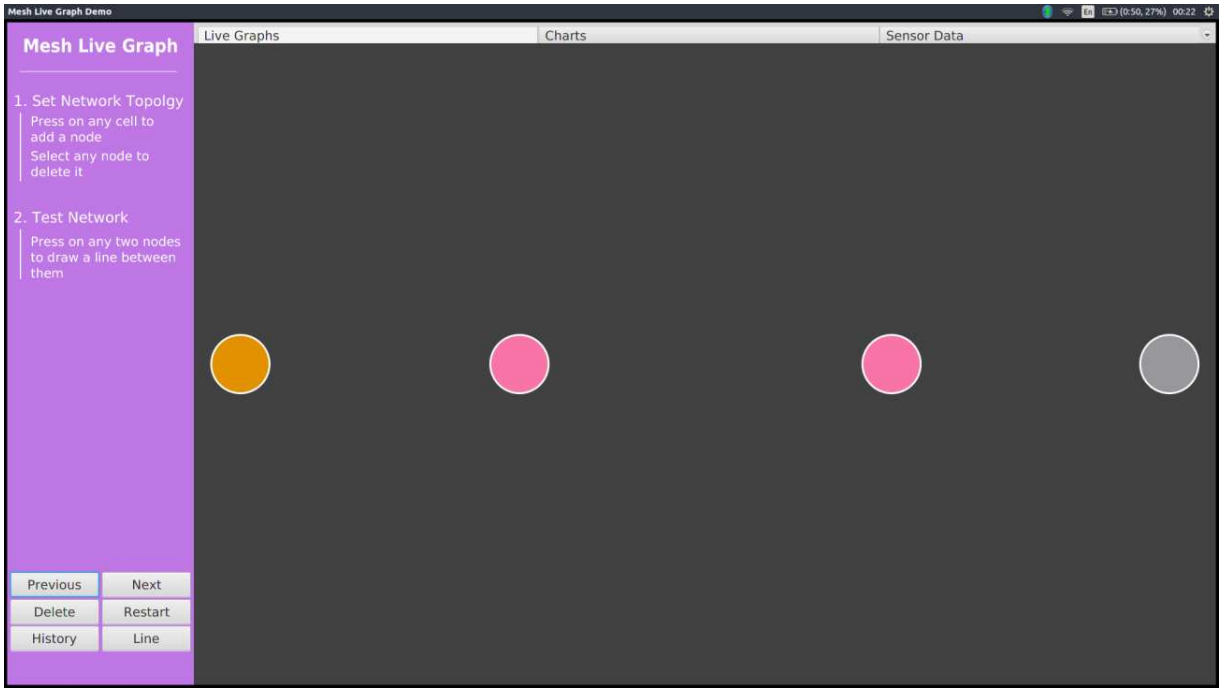

Figure 8- 3|Train Topology in GUI

2) Testing the network

- This phase is not mandatory and can be skipped. Pressing on any two successive nodes will draw three control messages which are RREQ, RREP and Data. This phase is only make sure that the drawing process is correct and the nodes are adjusted in the correct places.

3) Listening to the local server

- In this stage, a background task is executed that fetches the uploaded commands by the laptops connected serially to the Nordic boards from the local server periodically and delivers it to the UI thread. The UI thread draws the lines and curves between the nodes depending on the type of control message then removes the record from the database to avoid duplicate drawing.
- Each of the fetched commands is time-stamped so as to correctly draw the control messages in the correct order. These commands are:

1. [GUI] Source Address - RREQ - Destination Address - Timestamp
2. [GUI] Source Address - RREP - Destination Address - Timestamp
3. [GUI] Source Address - RERR - Destination Address - Timestamp
4. [GUI] Source Address - RWAIT - Destination Address - Timestamp
5. [GUI] Source Address - Data - Destination Address – Timestamp

- Figure (8-6) shows a sample of the drawn lines and curves.
- To view the information associated with any node, the left mouse button shall be clicked. Figure (8-7) shows the pop out dialog for the information.
- Whenever any node is down, its color changes into red and the information dialog changes the status into down as seen in Figure (8-8).
- There are various control options represented by the buttons located on the bottom left as seen in Figure (8-5):

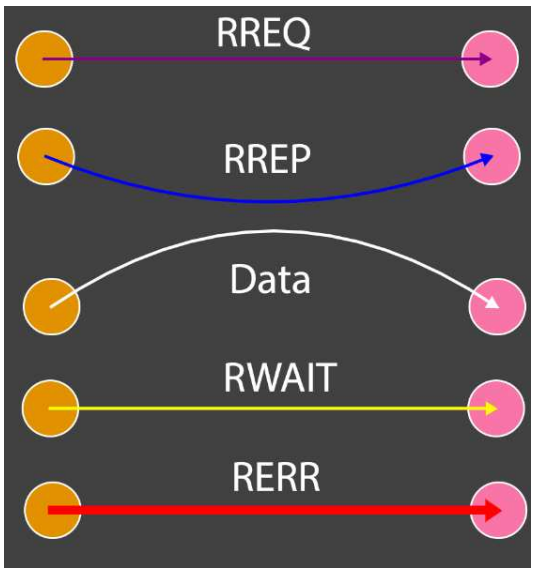

Figure 8- 4 | Control Message in the GUI

1. "Next" and "Previous" buttons: The user can alternate back and forth between the previously mentioned stages by pressing them.
2. "Delete" button: Deletes all the records from the local server to start over the listening process.
3. "Restart" button: Deletes all the drawn lines and curves from the graph to start over.
4. "History" button: Whenever the background task fetches a command to draw, the record is deleted to avoid duplicate drawings. This buttons allows to view the history of all the drawn control messages. Figure (8-7) shows the pop out dialog that views such data.

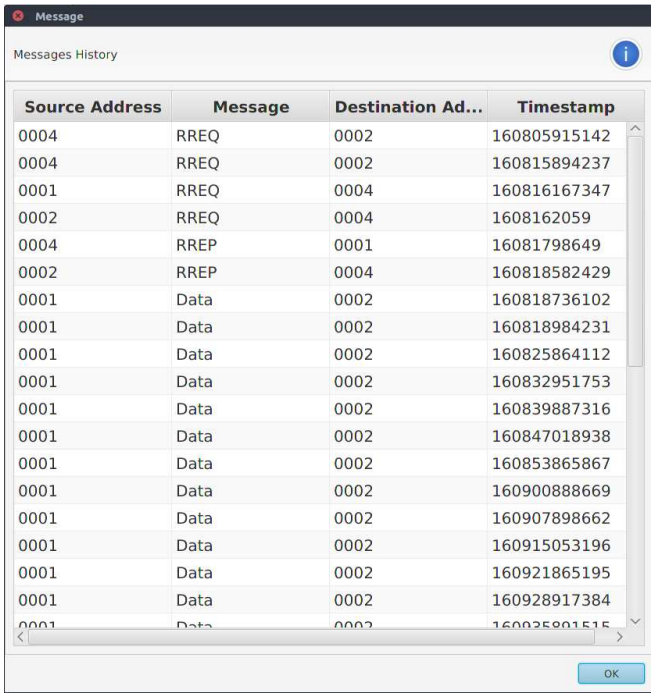

Figure 8- 7| Control Messages History Dialog

Figure 8- 5| Node Information

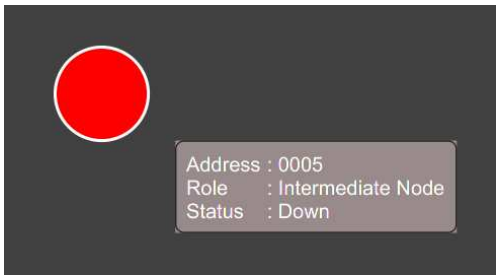

Figure 8- 6| Down Node Information

8.2.2.2 Charts Tab

- This tab is associated with visualization of various metrics like the throughput, the traffic of each node, overhead percentage and redundant (relayed) data percentage as well. Figure (8-8) shows the previously mentioned metrics.

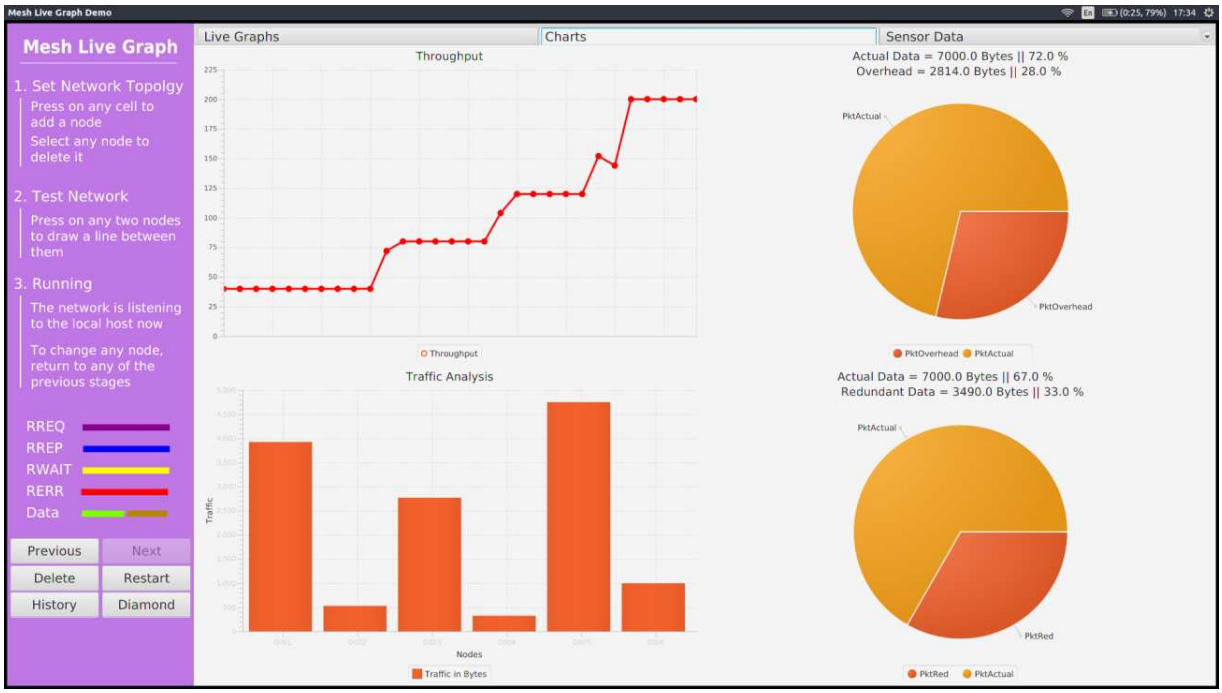

Figure 8- 8| Charts Tab

1. In this tab, two background tasks are executed to fetch the data from the second table resembling the metrics.

1. The first background task is associated with drawing the live line graph of the throughput:

   - As mentioned in section (9., the application publishes each sensor's data every two seconds. The throughput calculation procedure is to collect the received data in an interval of ten seconds and divide it by ten seconds then the value is uploaded to the local server by the following command: **[GUI] Address - Throughput - Value – Timestamp**.

2. The second background task is associated with drawing the nodes' traffic and visualize the overhead percentage and the redundant data percentage:

   - Whenever a node sends any control message like (RREQ, RREP, RWAIT, RERR), it uploads the following command to the local server: **[GUI] Address - PktOverhead- Value - Timestamp** where value is the size of the control message in bytes. All control messages are considered overheads on the sent data.

   - Whenever a node publishes any data, it uploads the following command to the local server: **[GUI] Address- PktActual- Value - Timestamp** where value is the size of data in bytes.

- Whenever a node relays any data, it uploads the following command to the local server: **[GUI] Address - PktRed – Value - Timestamp** where value is the size of the relayed data in bytes. All relayed data are considered redundant data.
- Regarding the traffic live graph, all three of the fetched commands are drawn in respective of the address uploading them.
- Regarding the data versus the overhead pie chart, only the "PktOverhead" and the "PktActual" commands are drawn.
- Regarding the actual data versus the redundant data pie chart, only the "PktRed" and the "PktActual" commands are drawn.

8.2.2.3 Sensors' Data Tab

- This tab is associated with viewing the sent sensors' data and organizing them in a tabled manner. Figure (8-9) shows the data received in an experiment.
- The background task fetches the following commands:
  - **[GUI] Address - Pressure - Value - Timestamp**
  - **[GUI] Address - Temperature – Value – Timestamp**

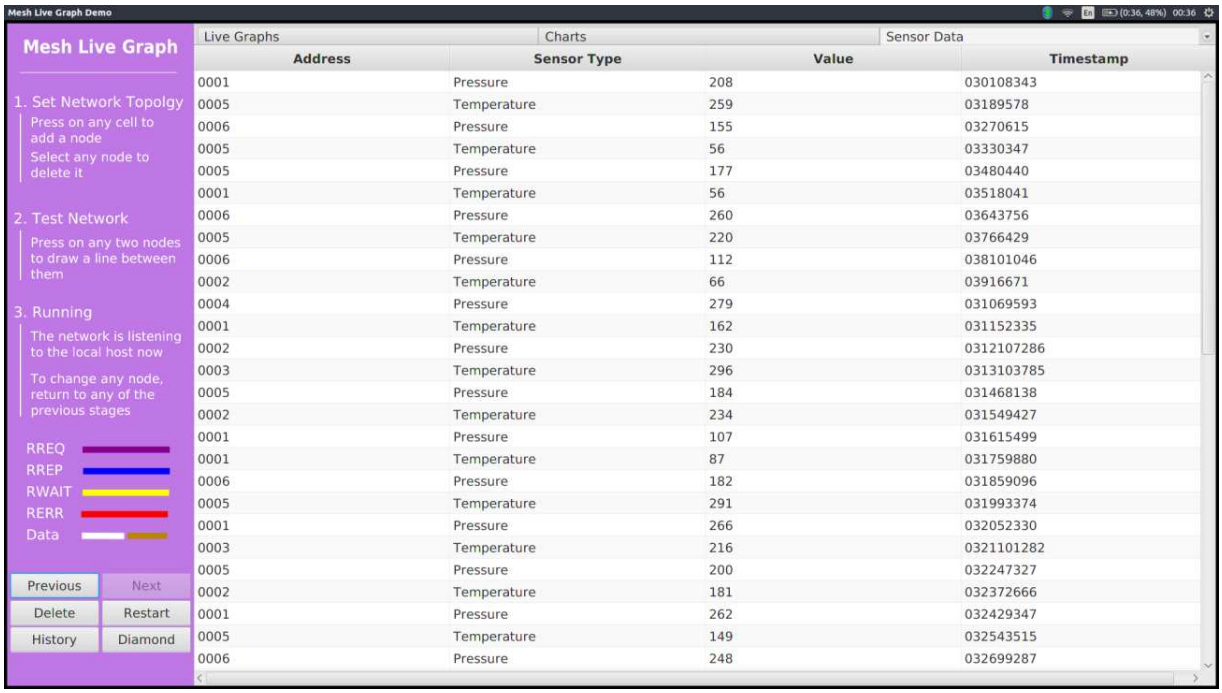

Figure 8- 9| Sensors' Data Tab

8.2.3 Serial Port Tracker

Each node is connected to a laptop so as to fetch the data from the serial port it is connected to. A Java code is built to listen to the connected port. All the commands mentioned in section 3.2.2 are read from the serial port and uploaded to the local server.