Trav Feller
Lab 8

1. This lab uses a vector clock (Links to an external site.) algorithm to create unique message IDs based on a sequence number. Could we replace the sequence number with a timestamp? What are the advantages and disadvantages of such an approach?
    1. We could, but it would mean that it would not be easily known if each node has all the current messages up until the current timestamp because the sequence lets us know if we miss a node. Nevertheless, a timestamp would allow us to keep track of relative messages, we could filter for messages on a certain day/time.
2. Are the temperature messages in order? Why or why not? If not, what could you do to fix this?
    1. The messages are in order on my system because I added them to the map with the key value equal to their sequence number so that they would all be in "alphabetical" order.
3. How did you avoid looping (sending messages back to someone who already has it)? Why was the unique ID helpful?
    1. I had a helper function in my send_rumor rule that would check to see if I had any rumors that I needed to send to my current peers. It would return a nested map of all the readings that the peer has missed and if the map was empty, it wouldn't send anything. As a double redundancy, I also only updates messages on my receiving node if they were not already in my ledger of received messages.
4. The propagation algorithm sleeps for `n` seconds between each iteration. What are the trade-offs between a low and high value for `n`.
    1. I noticed that as I ran my engine with a sleep of 1 second and added more than 5 nodes, My pico started to slow down because of all the commotion that was happening, but when I ran it for even 2 seconds or slower, it improved the speed and flow of the application, while still having fairly responsive times for propagating messages. These are a few trade-offs between the two.
5. Did new messages eventually end on all the nodes that were connected? Were the messages displayed in the same order on each node? Why or why not?
    1. Yes! All messages did end up at each node successfully. They did show in the same order because I stored my messages in map with the sequence number listed as the key value so they auto-ordered in sequential order.
6. Why does temporarily disconnecting a node from the network not result in permanent gaps in the messages seen at that node?
    1. Because each node is taken care of by its peer nodes, when the peer nodes get a response that the neighbor is online and sees that the seen message shows

outdated info, it will send ALL data that the node has missed to catch up, not just the latest message.

7. Describe, in a paragraph or two, how you could use the basic scheme implemented here to add failure detection to the system using a reachability table.
    1. If we were to instantiate a reachability table, where you could see how each node is connected, then in the case of a failure, a node could potentially look up its peer's peers to ask them for the current info. This would mean that we would have to implement more rules to allow nodes to accept requests from other nodes in the graph that are not directly related, but it would allow for nodes to verify information sent and received.

       For example, if node ee was only connected to node dd and only through node dd was ee able to get information and messages from the rest of the graph, when node dd went down and stopped sending information, node ee could use a reachability table to look up what nodes feed into dd and request to be reconnected with one of them instead of dd so that node ee could catch up. There would be no issues with connecting to any other node because the data in the whole graph should all be synced in the end. You would just have to deal with successfully deleting peers and subscriptions.