Names: Shkar Bassam, Kevin Wang
UB Email: mbassam@buffalo.edu, kwang47@buffalo.edu
UB Person Number: 50236189, 50313540
Class: CSE486

PHASE 4

Introduction:

The motivation of this phase is to create an election to the leader in RAFT in an environment where there are at least 5 nodes. In order to successfully have a leader election, we will need to implement Heartbeats, Timeouts, remote-procedure-calls (RPCs) to each node that we have. It is also important to create multiple asynchronous threads to build this phase and have it fully functioning. Different from other phases, in this phase we will see the comparisons play an important role in deciding who can and who cannot become the leader.

Implementation:

This phase will need to have a couple of functions implemented. First, we will need to create STORE and RETRIEVE functions. Store will be used to send out store requests to the RAFT cluster. We will also need to implement retrieve which can send a request to any of the nodes to retrieve all the committed entries at that particular node. Later on, we will need to implement safe log replication with consistency checks to make sure all the followers are on the same page when a candidate becomes a new leader, that the leader will send to all followers nextIndex[] its own index + 1. After that, the AppendEntry RPC is used to replicate the log's on the follower nodes. Similar to the previous phase, we will implement the append reply function so that when a follower receives an AppendEntry RPC, the follower can choose to accept or reject. Lastly, we will check for the leader election which is similar to phase 3, but this time the followers will have additional rules to grant a positive vote.

Validation:

For testing purposes, we will require a listener thread at the Controller end to receive messages from the leader. We will use our controller(hidden test-cases) to evaluate. As long as we follow the naming conventions in the JSON message and you are handling all the specified controller requests on the server-node's side there will not be any issues in testing. Additionally, we will need to check if a particular entry has been replicated on the majority of the followers and send that in the APPEND RPC to the followers which will apply that to their own logs for a final commit.

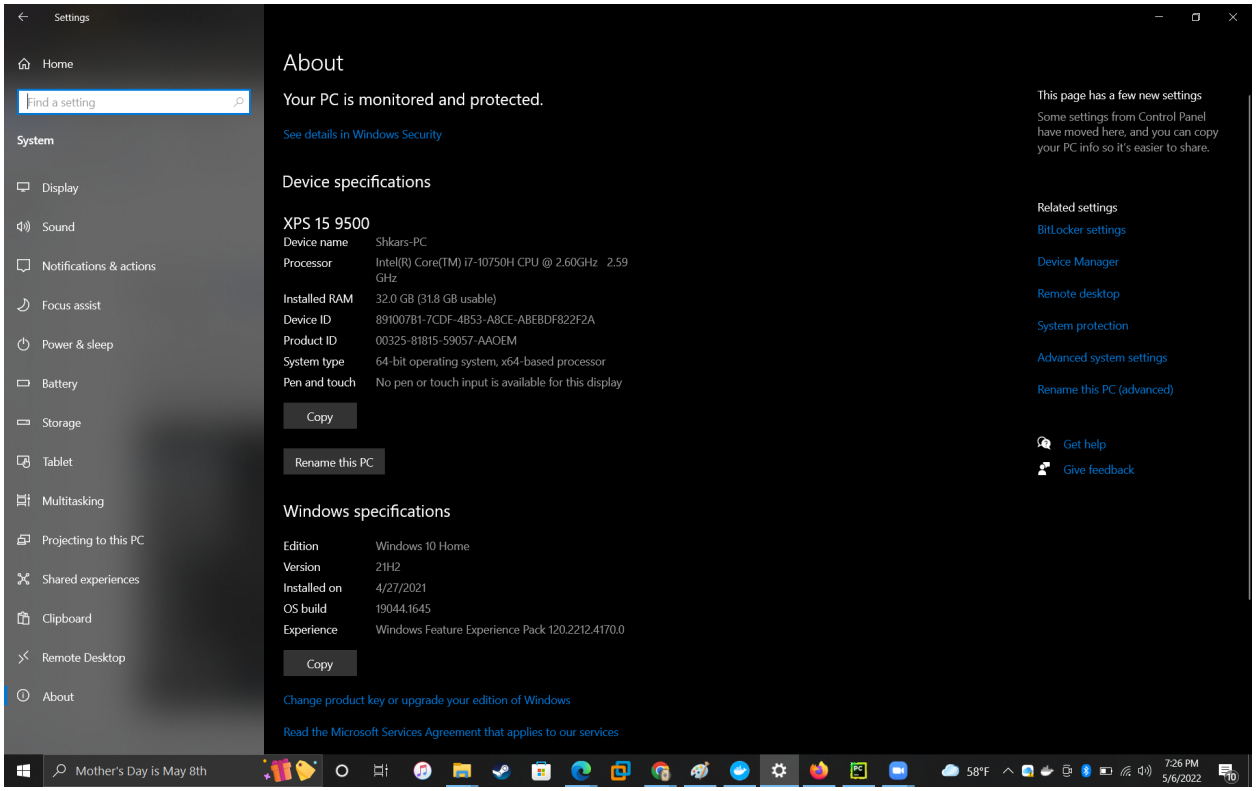
Names: Shkar Bassam, Kevin Wang
UB Email: mbassam@buffalo.edu, kwang47@buffalo.edu
UB Person Number: 50236189, 50313540
Class: CSE486

Also, we will have to validate that the STORE req from the controller is not a trigger for the AppendEntryRPC. The AppendEntry RPC is triggered at regular intervals as a heartbeat (which is why it is also functioning as a heartbeat). The STORE req appends an entry to the leader's log and this new entry gets sent along to the followers in the subsequent heartbeat/AppendEntryRPC

Difference between the basic log replication in phase 2 and phase 4:

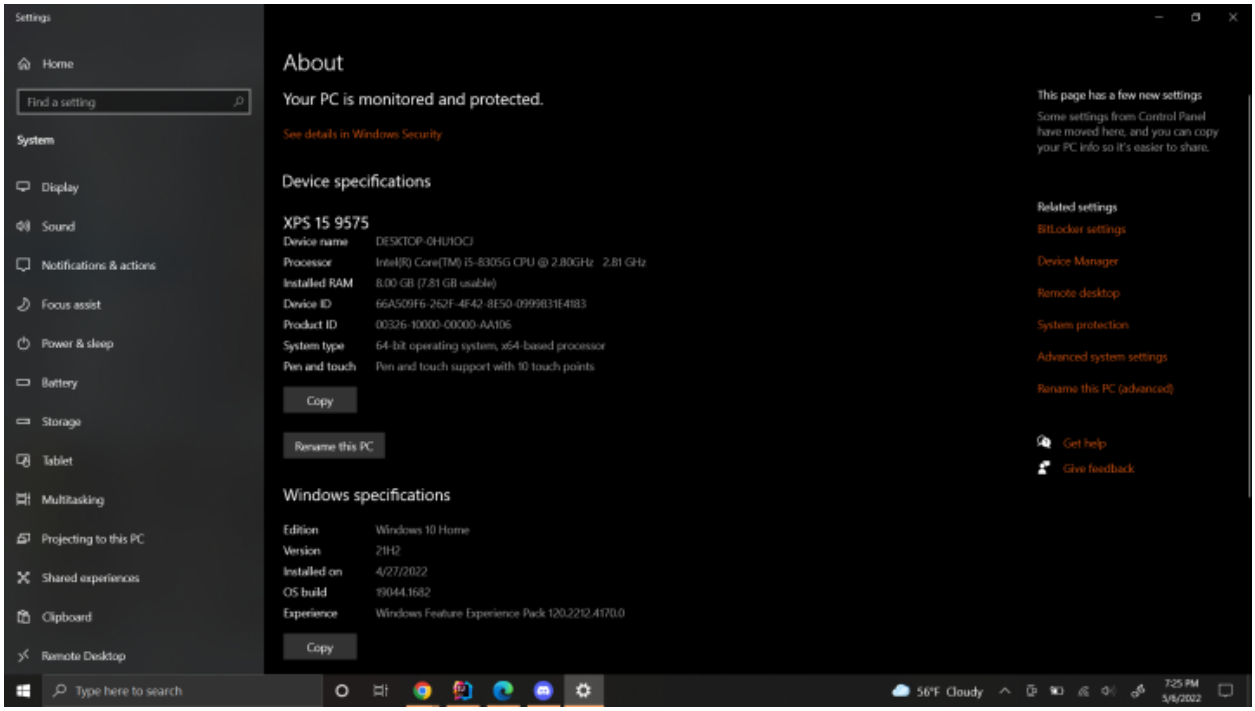
In phase 2, we build a simple mechanism of forwarding the client's request from the leader node to all the followers and ensuring that the request is executed on all the nodes by using CRUD. in this phase, we will be using RAFT to communicate with all the followers and ensuring the request is executed on all the nodes.

Shkar's Laptop Information:



Names: Shkar Bassam, Kevin Wang
UB Email: mbassam@buffalo.edu, kwang47@buffalo.edu
UB Person Number: 50236189, 50313540
Class: CSE486

Kevin's Laptop Information:



Names: Shkar Bassam, Kevin Wang
UB Email: mbassam@buffalo.edu, kwang47@buffalo.edu
UB Person Number: 50236189, 50313540
Class: CSE486

Diagram: