Project 2

Please add the following statement at the beginning of your report. I have neither given nor received unauthorized assistance on this work.

Sign: Shefali Mittal Date: 6th August 2023

Implement a fault-tolerant 2-phase distributed commit(2PC) protocol and use controlled and randomly injected failures to study how the 2PC protocol handles node crashes.

For this I have created, 2 main classes to handle 2PC protocol, one for coordinator in coordinator.py file. The other one is for nodes in node.py. To start 2 nodes I have used main.py file to create 2 SimpleXMLRPCServer on nodes on 5001 and 5002 ports. The coordinator will connect to these 2 nodes and complete the 2phase distributed commit scenarios.

The following scenarios are covered in my coordinator file:
Part 1:
If the coordinator fails before sending the "prepare" message, nodes will not receive the "prepare" message until the time-out and will abort. So, they will respond "no" to the "prepare" message after the coordinator comes back up and sends the "prepare" message.

Part 2:
If the transaction coordinator does not receive "yes" from a node, it will abort the transaction.

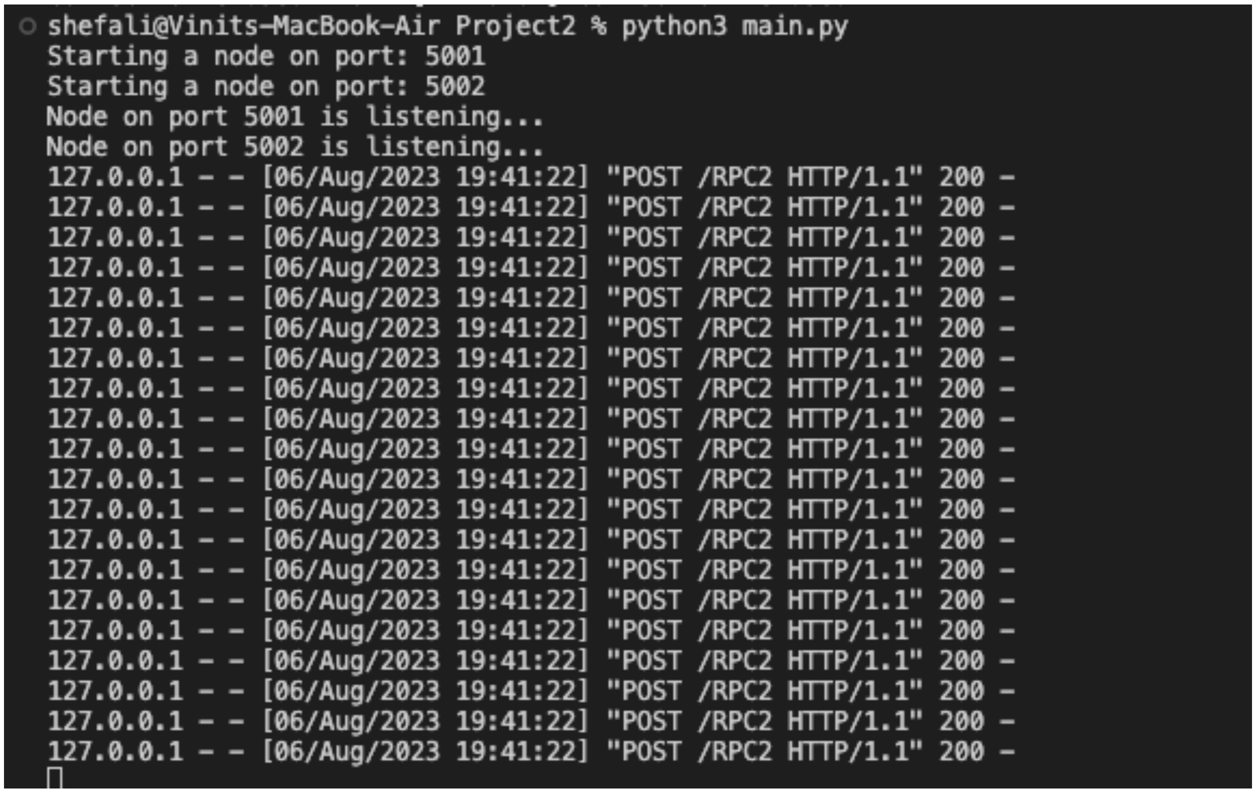
Part 3:
TC needs to store the transaction information on disk before sending the "commit" message to the nodes. If the TC fails after sending one "commit" message to the nodes, it can't abort. When it comes back up it will send the "commit" message to the nodes that it didn't send the "commit" message to.

Part 4:
A node needs to store the transaction information before replying "yes" to the TC. If it fails (time-out) after replying "yes"; after it comes back up, it will fetch the commit information from the TC for that particular transaction.

I also covered a scenario when everything goes right along with recovery scenario. I used logs throughout the test cases to keep track all the transactions for coordinator and nodes.

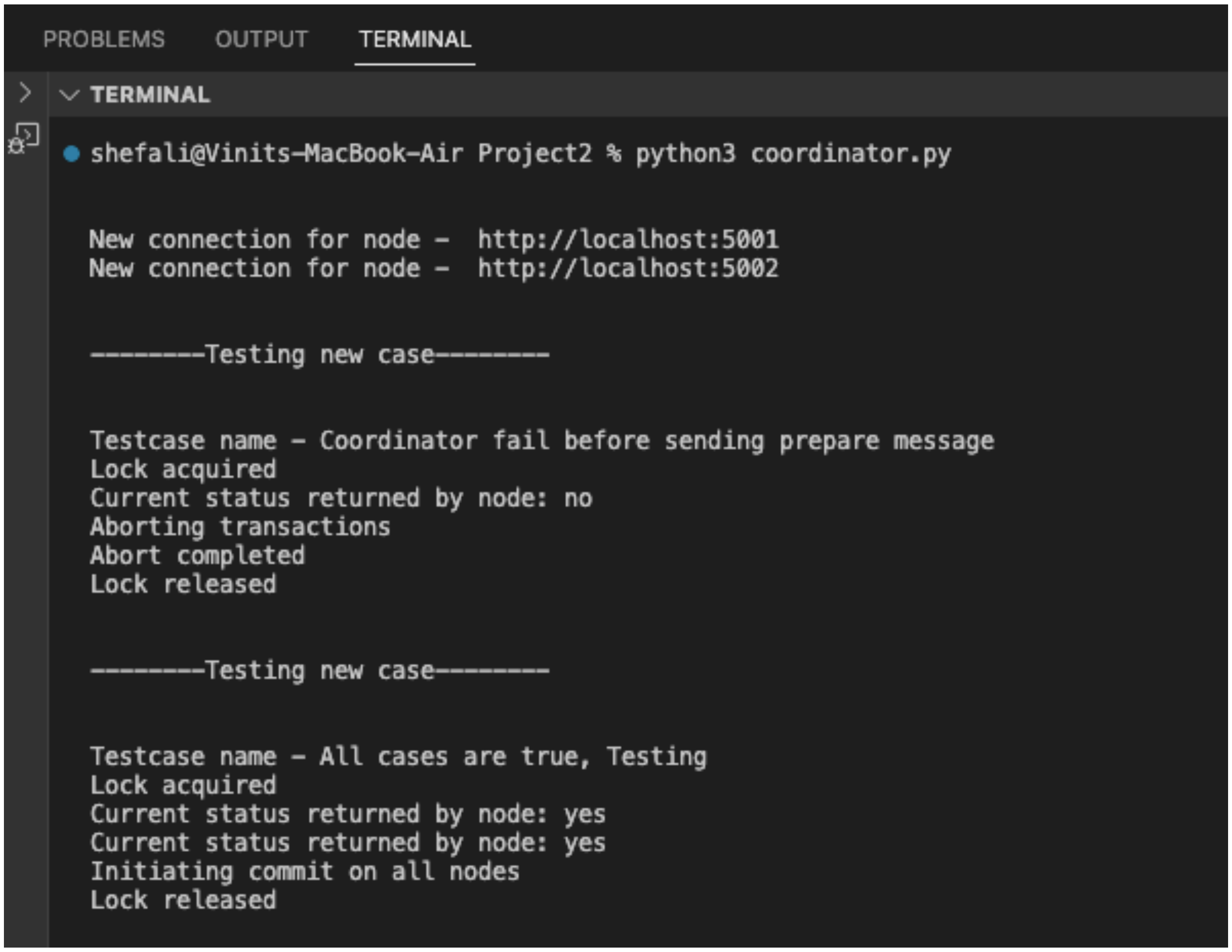
Below are the screenshots:

First run main.py:

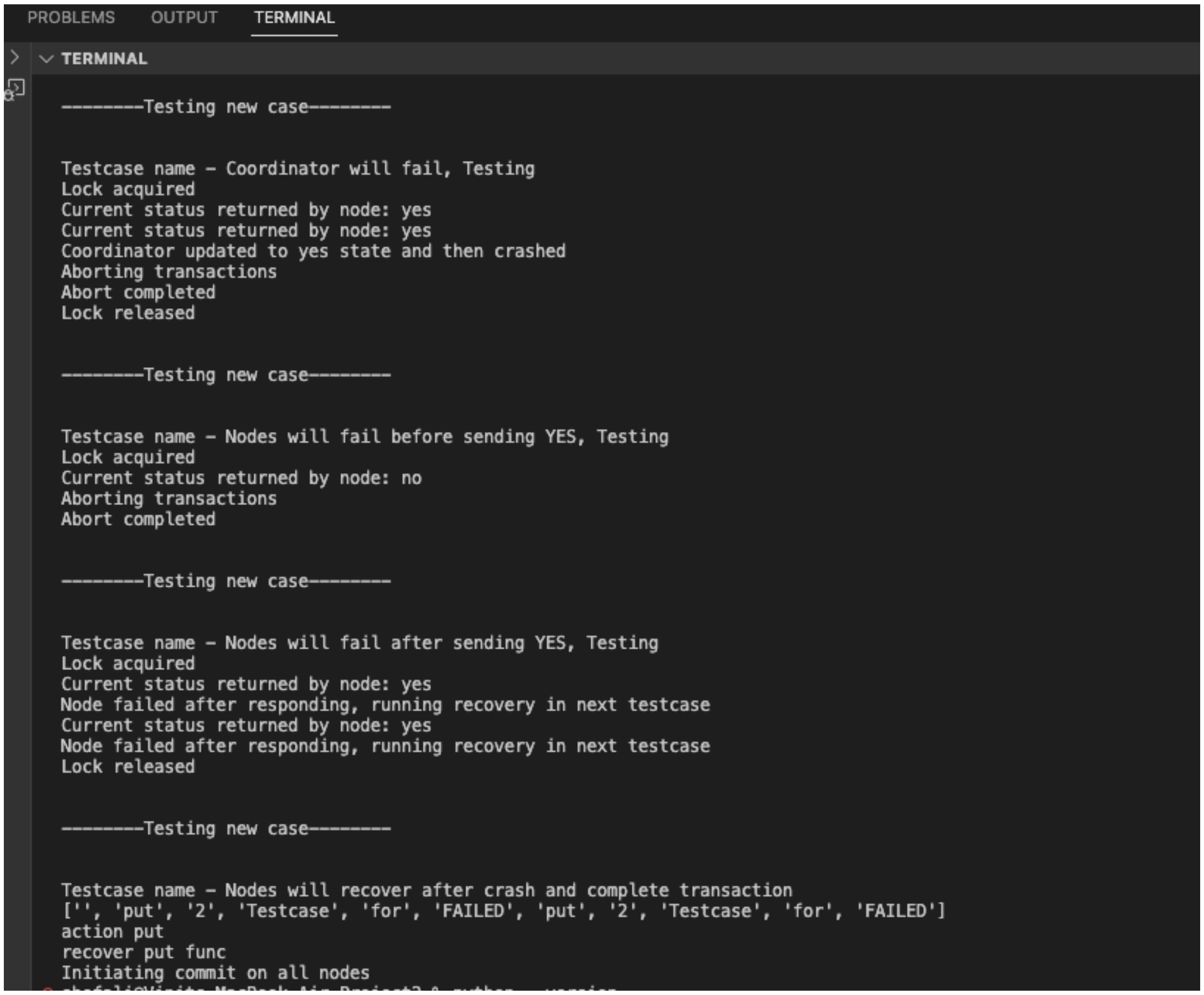


```
○ shefali@Vinits-MacBook-Air Project2 % python3 main.py
  Starting a node on port: 5001
  Starting a node on port: 5002
  Node on port 5001 is listening...
  Node on port 5002 is listening...
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  127.0.0.1 - - [06/Aug/2023 19:41:22] "POST /RPC2 HTTP/1.1" 200 -
  ▯
```

Then run coordinator.py:



```
shefali@Vinits-MacBook-Air Project2 % python3 coordinator.py


New connection for node -  http://localhost:5001
New connection for node -  http://localhost:5002


--------Testing new case--------


Testcase name - Coordinator fail before sending prepare message
Lock acquired
Current status returned by node: no
Aborting transactions
Abort completed
Lock released


--------Testing new case--------


Testcase name - All cases are true, Testing
Lock acquired
Current status returned by node: yes
Current status returned by node: yes
Initiating commit on all nodes
Lock released
```



```
--------Testing new case--------


Testcase name - Coordinator will fail, Testing
Lock acquired
Current status returned by node: yes
Current status returned by node: yes
Coordinator updated to yes state and then crashed
Aborting transactions
Abort completed
Lock released


--------Testing new case--------


Testcase name - Nodes will fail before sending YES, Testing
Lock acquired
Current status returned by node: no
Aborting transactions
Abort completed


--------Testing new case--------


Testcase name - Nodes will fail after sending YES, Testing
Lock acquired
Current status returned by node: yes
Node failed after responding, running recovery in next testcase
Current status returned by node: yes
Node failed after responding, running recovery in next testcase
Lock released


--------Testing new case--------


Testcase name - Nodes will recover after crash and complete transaction
['', 'put', '2', 'Testcase', 'for', 'FAILED', 'put', '2', 'Testcase', 'for', 'FAILED']
action put
recover put func
Initiating commit on all nodes
```