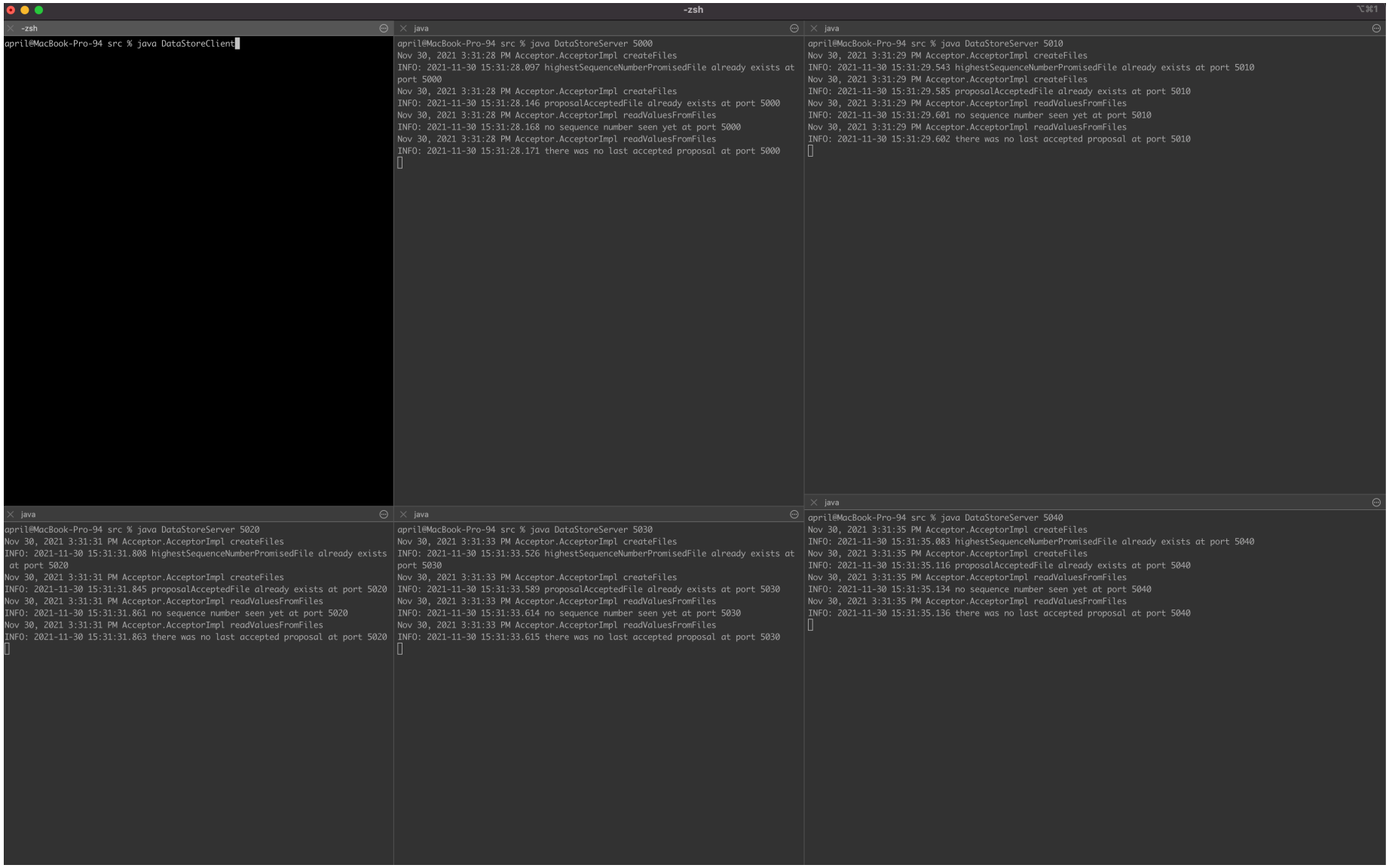
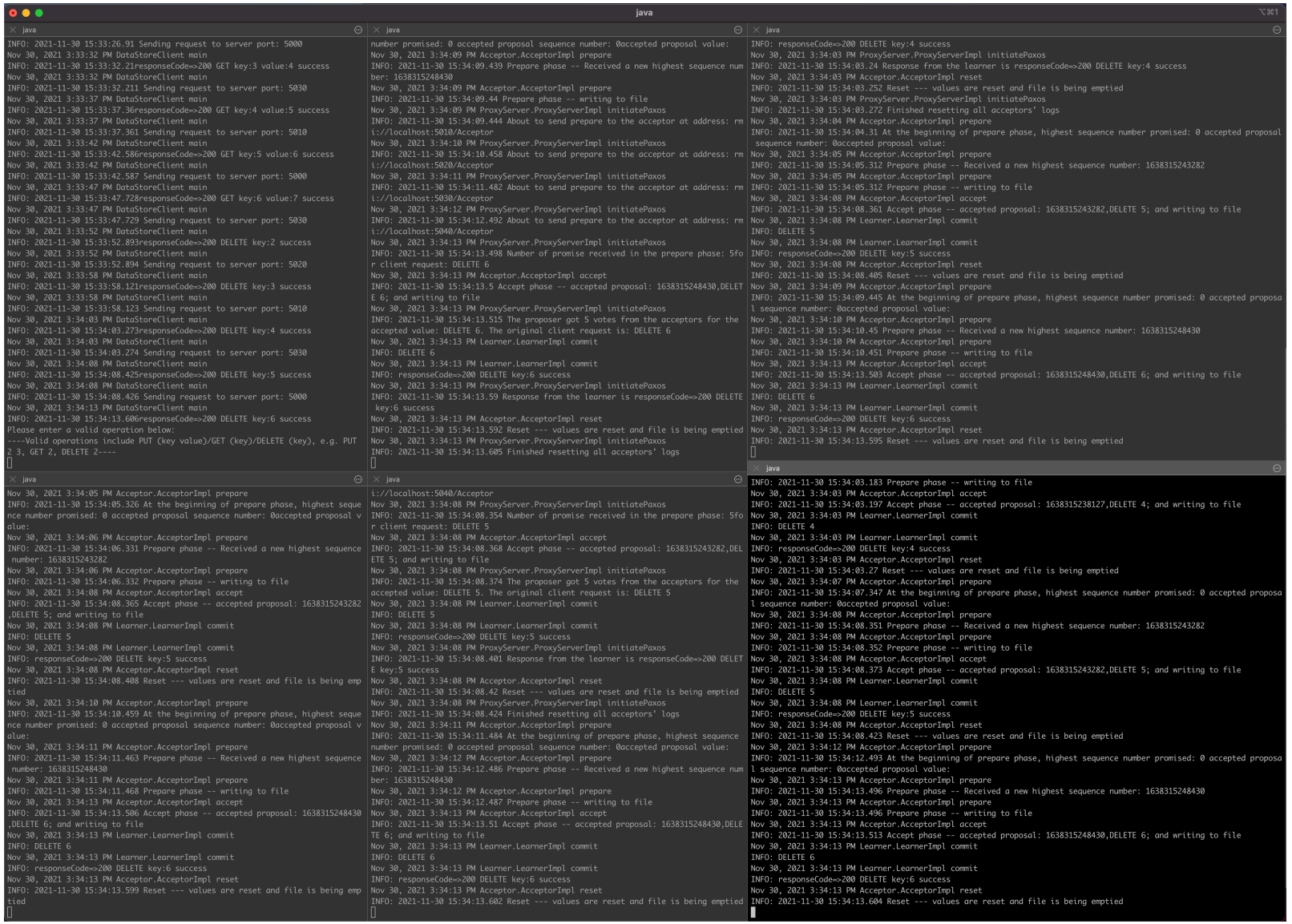
Code result walkthrough (please zoom in to show the details of the logs in the screenshots)

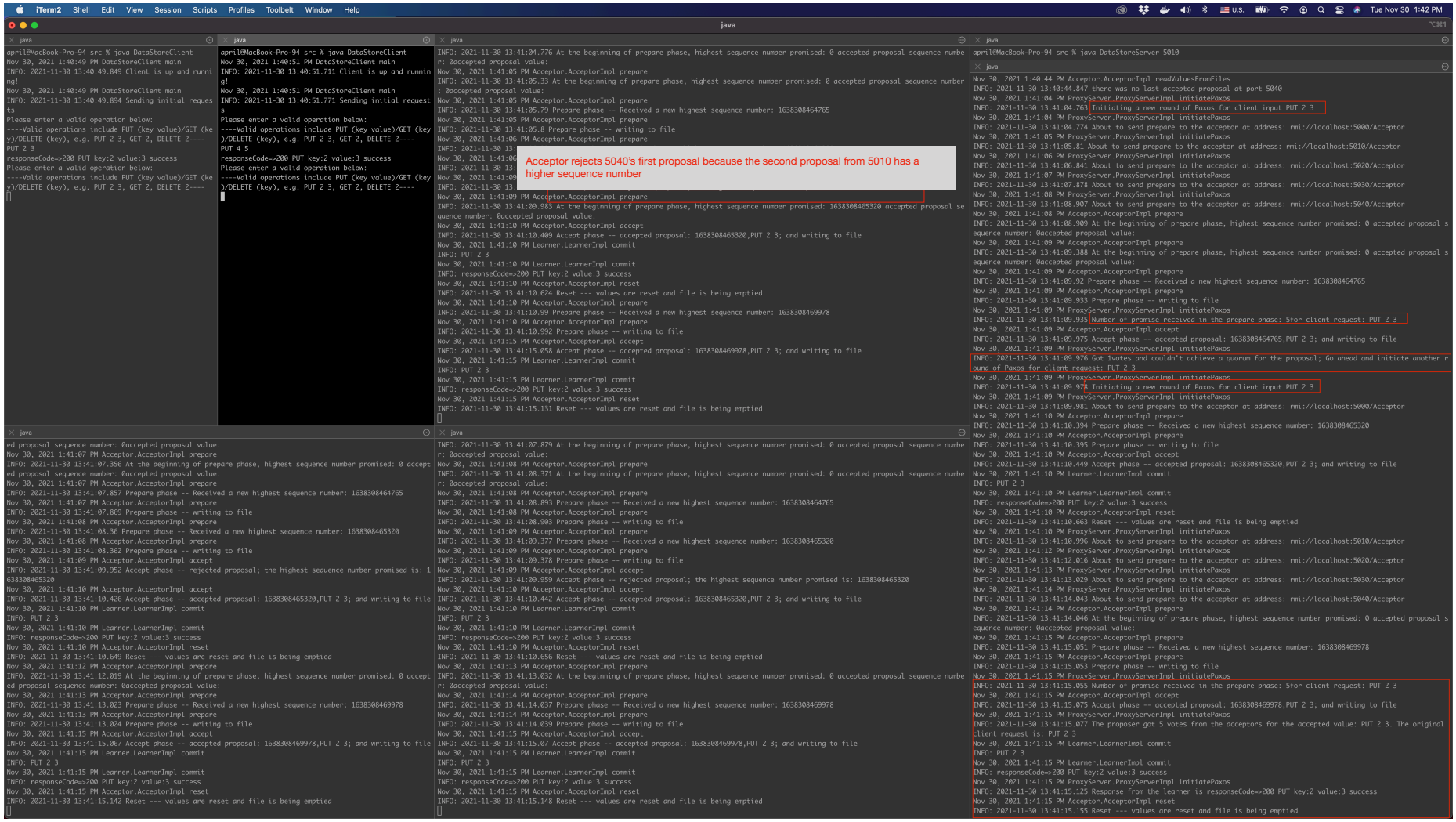
1. Below is when servers start up. They'll create the persistence logs (if they're non-existent) and read the previous accepted values and highest sequence number promised.



2. Below is a snapshot of the client start up and sending initial requests sequentially



3. Here is a test on having two different concurrent client requests (*"PUT 2 3" and "PUT 4 5"*) sending to two different servers (proposers). In this example, there's no acceptor failure yet. An example with acceptor failures will be illustrated in the following sections. Also note that I added a "Thread.sleep(1000)" in the server side to slow down the process so that the different proposals will overlap and we can see the effect of the Paxos algorithm more clearly. As highlighted below. The proposer at port 5040 received



client request “PUT 2 3” and initiated a round of Paxos. It received 5 promised (from all acceptors) in the prepare stage (because it’s sent slightly earlier than “PUT 4 5”. However, it only received 1 vote in the accept stage (because by this time, the proposer for the request “PUT 4 5” has sent out another proposal with a higher sequence number). Only 1 acceptor accepted the proposal for “PUT 2 3”. Therefore, this proposer will initiate another round of Paxos with a higher sequence number.

As indicated by the highlighted log, we can see that at this point, the other proposer at port 5010 (the proposer for the request “PUT 4 5”) has got a quorum in the prepare phase. However, 1 acceptor already accepted the proposal for “PUT 2 3” earlier, the proposer at port 5010 then went ahead and proposed “PUT 2 3” instead of its original client request “PUT 4 5” and the value got committed. A consensus is achieved. This round of Paxos is complete. All the persisted values are cleared.

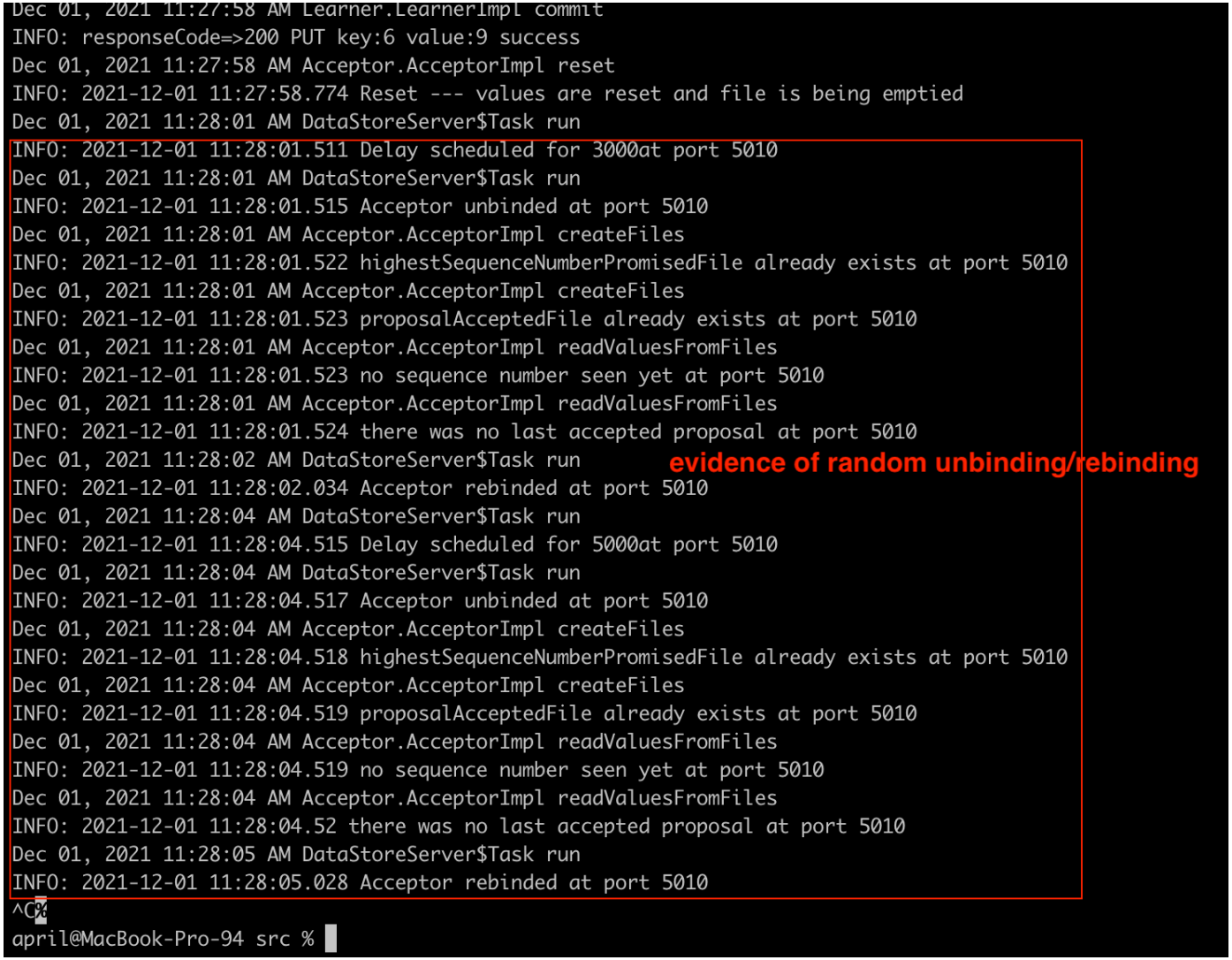
At this point, the proposer at port 5040 is initiating on its new round of Paxos algorithm. Since at this time, all acceptors are starting at a clean slate (because the last round of Paxos has ended). Therefore, the proposer is able to get its way this time and achieved quorum in both prepare and accept phases. “PUT 2 3”, which is its original client request, is committed.

Note that “PUT 4 5” was unable to be executed because of the algorithm, but this is fine as long as only the value that all acceptors agreed on was executed across all learners.

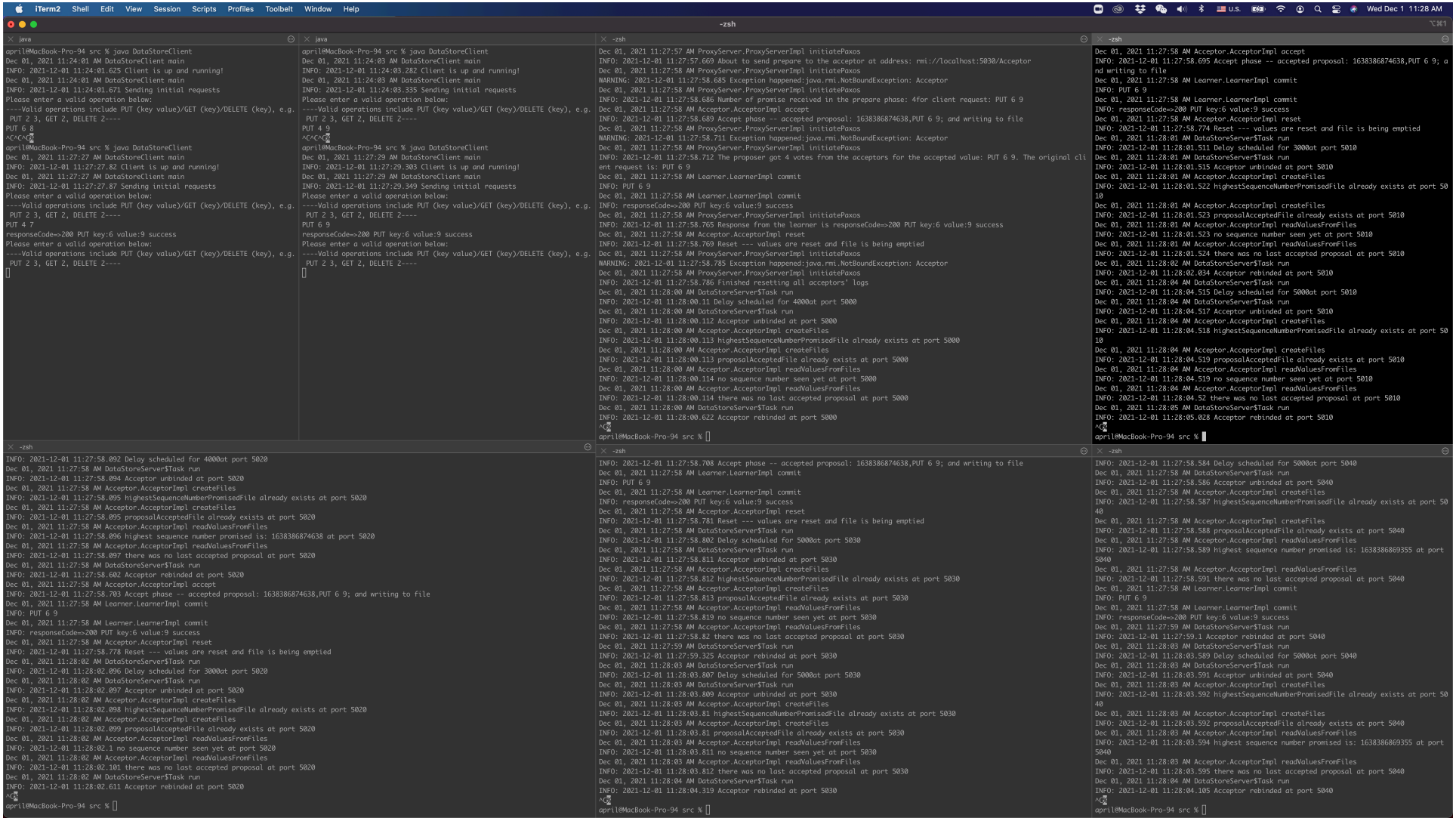
A further demonstration of how the acceptor’s failure can be overcame through the persisted logs is in the next page.

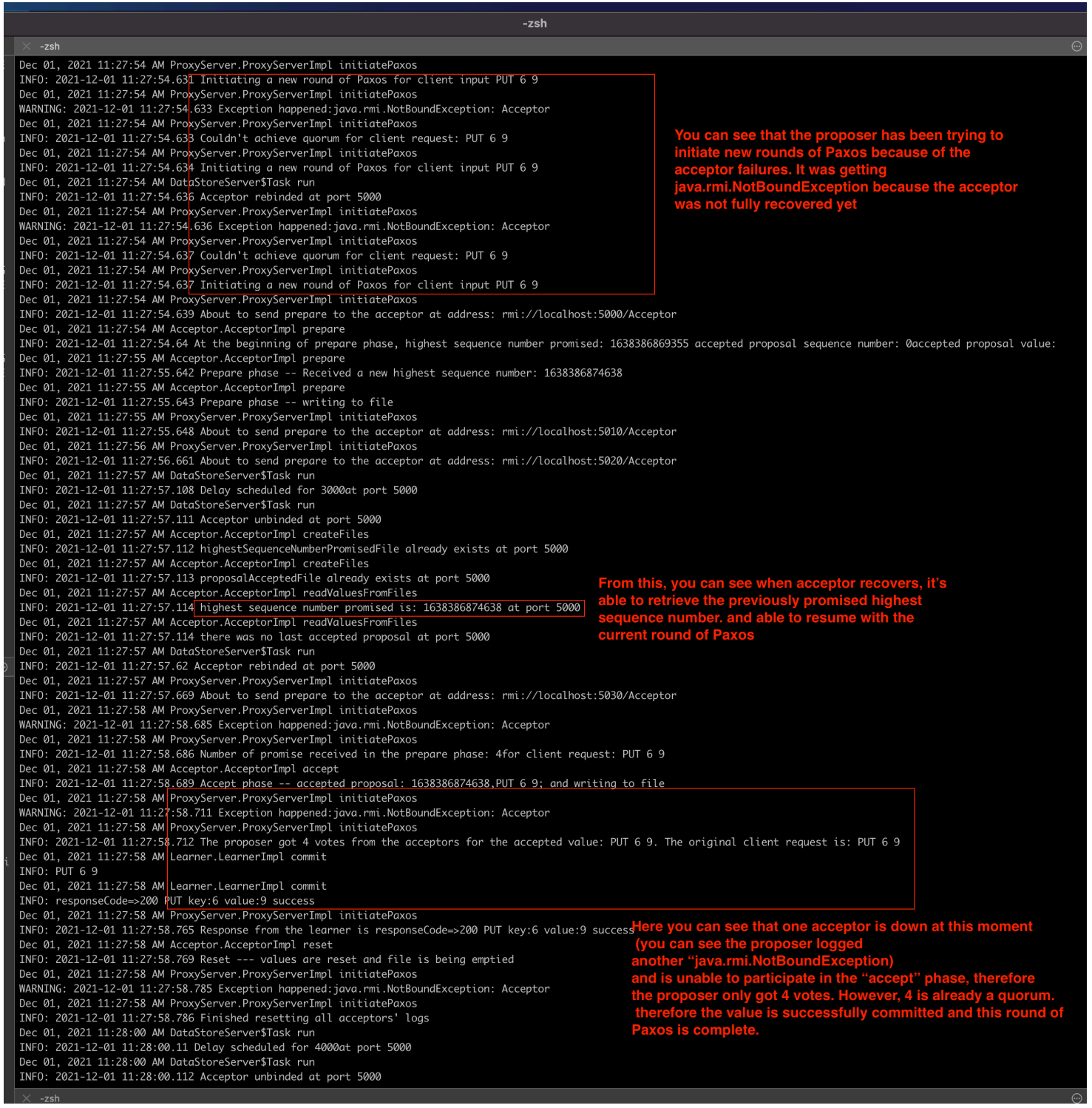
4. Demonstration of Acceptor failure and restarting

The way I choose to kill and restart the acceptor is by unbind/rebind the acceptors, as shown below. Unbind/rebind is an expensive operation, therefore a further way to improve this project would be to utilize thread kill/restart. I set each acceptor to repeatedly fail (unbind)/restart(rebind) at random time intervals. For now, Every time an acceptor rebinds, it will create the log files and read from them to retrieve the previous state. See below screenshot for the evidence of random acceptor failure/restart at random intervals.



The following two screenshots show the details of an example test run of the Paxos algorithm. In this example, two clients send concurrent requests (“PUT 4 7” and “PUT 6 9”) to two random servers and despite the randomly failing acceptors, a consensus was still reached in the end. The highlighted logs and explanation in the screenshots show that, when acceptors restart, through reading from the persisted logs, they can retrieve previous promised highest sequence number and accepted proposals and rejoin the paxos process. Also, if a quorum can still be achieved despite small number of acceptor failure, the algorithm can still move forward and reach a consensus.




-zsh
-zsh
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:54.631 Initiating a new round of Paxos for client input PUT 6 9
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
WARNING: 2021-12-01 11:27:54.633 Exception happened:java.rmi.NotBoundException: Acceptor
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:54.633 Couldn't achieve quorum for client request: PUT 6 9
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:54.634 Initiating a new round of Paxos for client input PUT 6 9
Dec 01, 2021 11:27:54 AM DataStoreServer$Task run
INFO: 2021-12-01 11:27:54.636 Acceptor rebinded at port 5000
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
WARNING: 2021-12-01 11:27:54.636 Exception happened:java.rmi.NotBoundException: Acceptor
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:54.637 Couldn't achieve quorum for client request: PUT 6 9
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:54.637 Initiating a new round of Paxos for client input PUT 6 9
Dec 01, 2021 11:27:54 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:54.639 About to send prepare to the acceptor at address: rmi://localhost:5000/Acceptor
Dec 01, 2021 11:27:54 AM Acceptor.AcceptorImpl prepare
INFO: 2021-12-01 11:27:54.64 At the beginning of prepare phase, highest sequence number promised: 1638386869355 accepted proposal sequence number: 0accepted proposal value:
Dec 01, 2021 11:27:55 AM Acceptor.AcceptorImpl prepare
INFO: 2021-12-01 11:27:55.642 Prepare phase -- Received a new highest sequence number: 1638386874638
Dec 01, 2021 11:27:55 AM Acceptor.AcceptorImpl prepare
INFO: 2021-12-01 11:27:55.643 Prepare phase -- writing to file
Dec 01, 2021 11:27:55 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:55.648 About to send prepare to the acceptor at address: rmi://localhost:5010/Acceptor
Dec 01, 2021 11:27:56 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:56.661 About to send prepare to the acceptor at address: rmi://localhost:5020/Acceptor
Dec 01, 2021 11:27:57 AM DataStoreServer$Task run
INFO: 2021-12-01 11:27:57.108 Delay scheduled for 3000at port 5000
Dec 01, 2021 11:27:57 AM DataStoreServer$Task run
INFO: 2021-12-01 11:27:57.111 Acceptor unbinded at port 5000
Dec 01, 2021 11:27:57 AM Acceptor.AcceptorImpl createFiles
INFO: 2021-12-01 11:27:57.112 highestSequenceNumberPromisedFile already exists at port 5000
Dec 01, 2021 11:27:57 AM Acceptor.AcceptorImpl createFiles
INFO: 2021-12-01 11:27:57.113 proposalAcceptedFile already exists at port 5000
Dec 01, 2021 11:27:57 AM Acceptor.AcceptorImpl readValuesFromFiles
INFO: 2021-12-01 11:27:57.114 highest sequence number promised is: 1638386874638 at port 5000
Dec 01, 2021 11:27:57 AM Acceptor.AcceptorImpl readValuesFromFiles
INFO: 2021-12-01 11:27:57.114 there was no last accepted proposal at port 5000
Dec 01, 2021 11:27:57 AM DataStoreServer$Task run
INFO: 2021-12-01 11:27:57.62 Acceptor rebinded at port 5000
Dec 01, 2021 11:27:57 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:57.669 About to send prepare to the acceptor at address: rmi://localhost:5030/Acceptor
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
WARNING: 2021-12-01 11:27:58.685 Exception happened:java.rmi.NotBoundException: Acceptor
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:58.686 Number of promise received in the prepare phase: 4for client request: PUT 6 9
Dec 01, 2021 11:27:58 AM Acceptor.AcceptorImpl accept
INFO: 2021-12-01 11:27:58.689 Accept phase -- accepted proposal: 1638386874638,PUT 6 9; and writing to file
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
WARNING: 2021-12-01 11:27:58.711 Exception happened:java.rmi.NotBoundException: Acceptor
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:58.712 The proposer got 4 votes from the acceptors for the accepted value: PUT 6 9. The original client request is: PUT 6 9
Dec 01, 2021 11:27:58 AM Learner.LearnerImpl commit
INFO: PUT 6 9
Dec 01, 2021 11:27:58 AM Learner.LearnerImpl commit
INFO: responseCode=>200 PUT key:6 value:9 success
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:58.765 Response from the learner is responseCode=>200 PUT key:6 value:9 success
Dec 01, 2021 11:27:58 AM Acceptor.AcceptorImpl reset
INFO: 2021-12-01 11:27:58.769 Reset --- values are reset and file is being emptied
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
WARNING: 2021-12-01 11:27:58.785 Exception happened:java.rmi.NotBoundException: Acceptor
Dec 01, 2021 11:27:58 AM ProxyServer.ProxyServerImpl initiatePaxos
INFO: 2021-12-01 11:27:58.786 Finished resetting all acceptors' logs
Dec 01, 2021 11:28:00 AM DataStoreServer$Task run
INFO: 2021-12-01 11:28:00.11 Delay scheduled for 4000at port 5000
Dec 01, 2021 11:28:00 AM DataStoreServer$Task run
INFO: 2021-12-01 11:28:00.112 Acceptor unbinded at port 5000
-zsh
You can see that the proposer has been trying to initiate new rounds of Paxos because of the acceptor failures. It was getting java.rmi.NotBoundException because the acceptor was not fully recovered yet
From this, you can see when acceptor recovers, it's able to retrieve the previously promised highest sequence number. and able to resume with the current round of Paxos
Here you can see that one acceptor is down at this moment (you can see the proposer logged another "java.rmi.NotBoundException) and is unable to participate in the "accept" phase, therefore the proposer only got 4 votes. However, 4 is already a quorum. therefore the value is successfully committed and this round of Paxos is complete.