Northeastern University – Silicon Valley

CS 6650 Scalable Distributed Systems

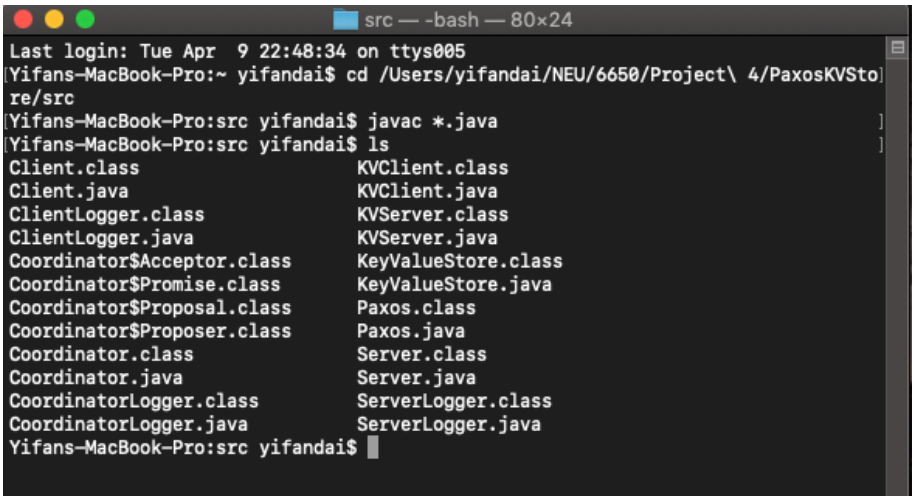
Project #4 By Yifan Dai

Paxos-Based, Multi-threaded, Replicated Key-Value Store using RPC

README

Build Code

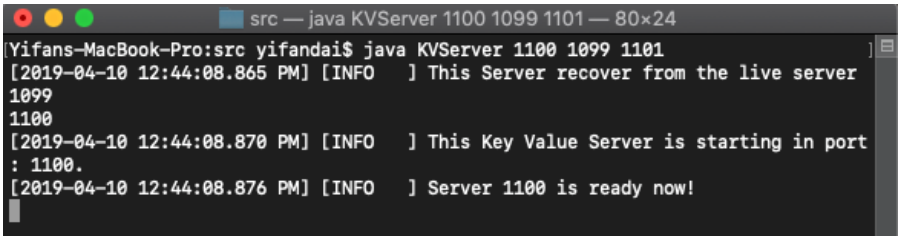
Compile: ***javac *.java***



Run Server

1) Run a server at the first time, you can simply run it by: ***java KVServer <portNumber>***

2) If you want to run a server and recover it from the live server, you can run:
java KVServer <portNumber> <livePort1> <livePort2> …

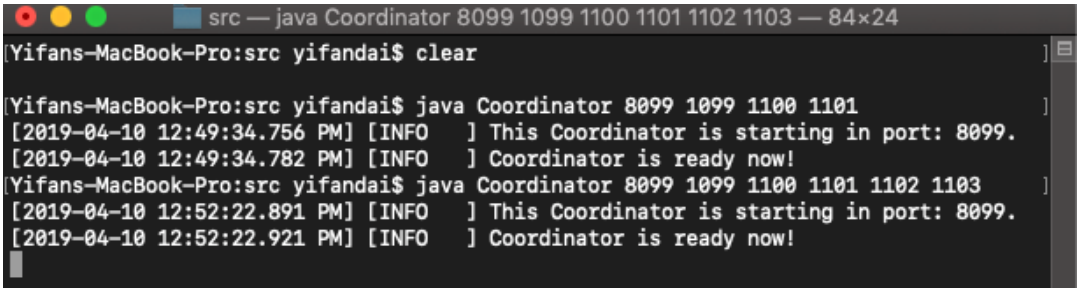


The ***portNumber*** is the port number of current server, the ***livePort1, livePort2*** and etc is the port number of the live ports which are currently running.

The server will recover the data from ***<livePort1>, <livePort2>*** and etc. The recover logic is: **the server will first try to connect to the live server 1 and recover. If it recover successfully then it will complete the recovery, otherwise, it will try the live server 2 and etc until it is success.**

Run Coordinator

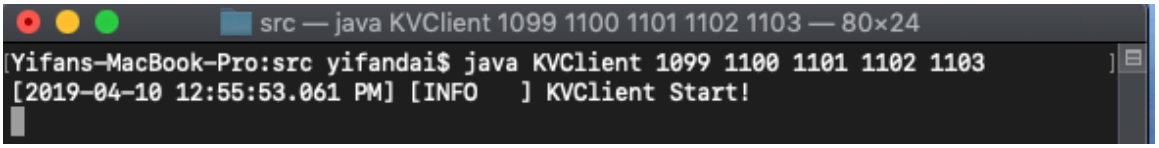
Run Coordinator by: ***java Coordinator <portNumber> <serverPort1> … <serverPort5>***



The ***serverPort1*** to ***serverPort5*** are the port numbers of the live servers.

Run Client

Run Client by: ***java Client <serverPort1> … <serverPort5>***



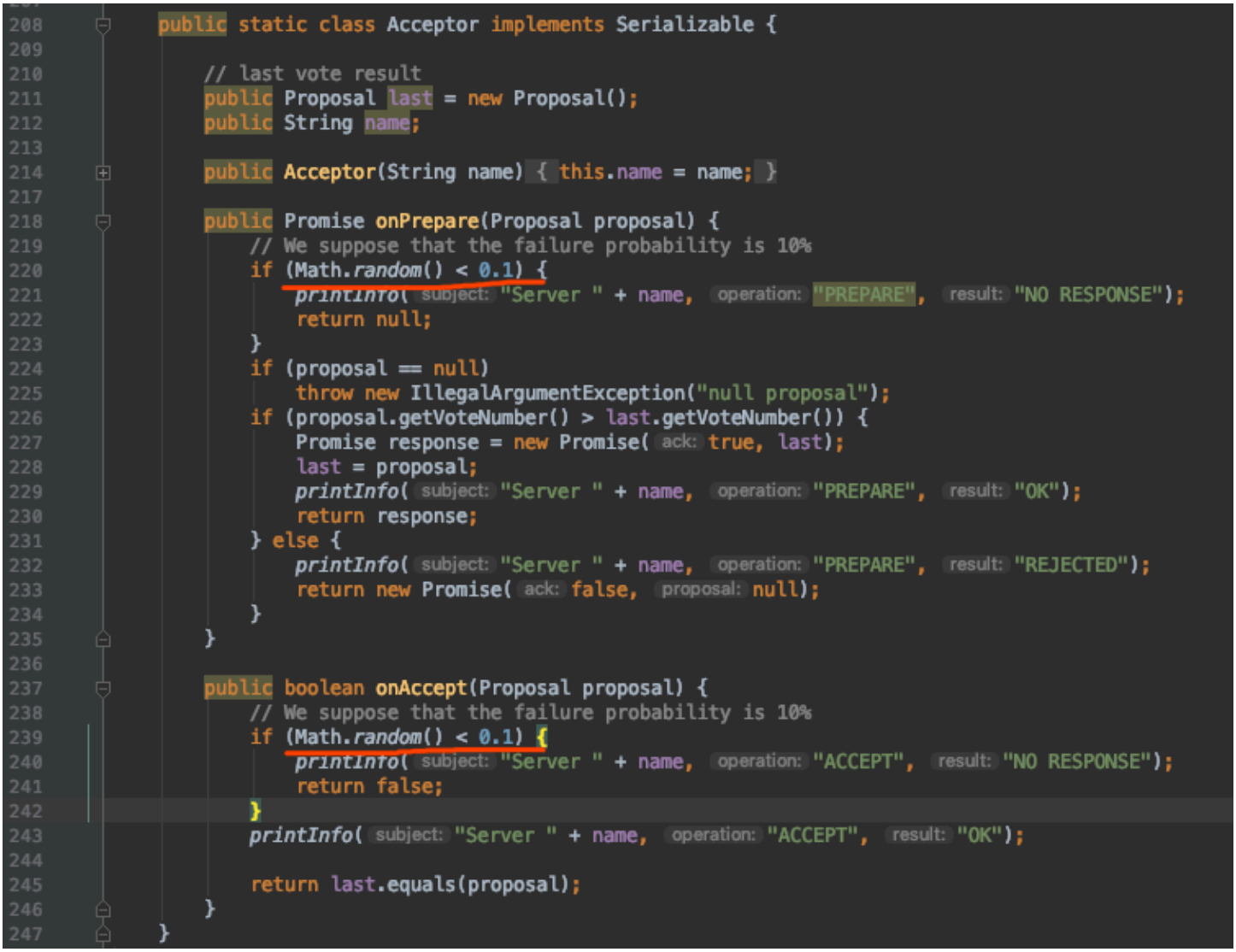
The ***serverPort1*** to ***serverPort5*** are the port numbers of the live servers.

Notice

1. If some servers are down, then the quorum is based on the number of live servers, not including the servers which are already down.
2. When starting a new server by running the ***java KVServer <portNumber> <livePort1> <livePort2> …*** this current server will try to recover data from the live servers. If all the servers provided here are all down, then the current server will start with an empty KV.
3. The live servers are always consistent (the data in every live servers are always the same). The client could send request and receive correct response if there is at least one live server. If there is no live server currently, then the client will retry until a live server is available and send request to the live server.

How to make the servers randomly fail

There is a ***Coordinator.java*** file in the source code directory.
As the screenshot showed below, we can change the float which is underlined by the red line. This float represent the failure probability (the probability to reject the update) when clients send request to the servers.
Please feel free to change the float value when testing.



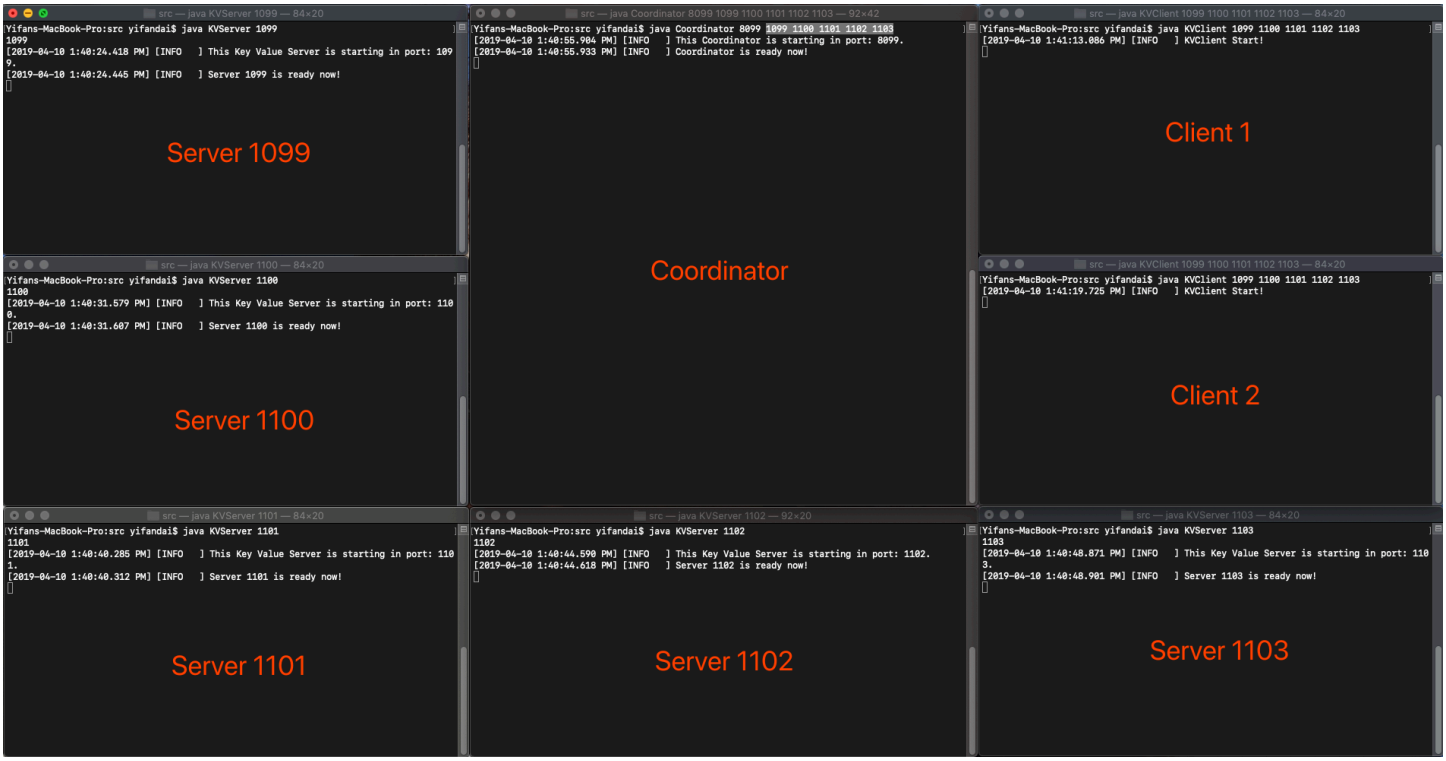
```
208     public static class Acceptor implements Serializable {
209
210         // last vote result
211         public Proposal last = new Proposal();
212         public String name;
213
214         public Acceptor(String name) { this.name = name; }
217
218         public Promise onPrepare(Proposal proposal) {
219             // We suppose that the failure probability is 10%
220             if (Math.random() < 0.1) {
221                 printInfo( subject: "Server " + name,  operation: "PREPARE",  result: "NO RESPONSE");
222                 return null;
223             }
224             if (proposal == null)
225                 throw new IllegalArgumentException("null proposal");
226             if (proposal.getVoteNumber() > last.getVoteNumber()) {
227                 Promise response = new Promise( ack: true, last);
228                 last = proposal;
229                 printInfo( subject: "Server " + name,  operation: "PREPARE",  result: "OK");
230                 return response;
231             } else {
232                 printInfo( subject: "Server " + name,  operation: "PREPARE",  result: "REJECTED");
233                 return new Promise( ack: false,  proposal: null);
234             }
235         }
236
237         public boolean onAccept(Proposal proposal) {
238             // We suppose that the failure probability is 10%
239             if (Math.random() < 0.1) {
240                 printInfo( subject: "Server " + name,  operation: "ACCEPT",  result: "NO RESPONSE");
241                 return false;
242             }
243             printInfo( subject: "Server " + name,  operation: "ACCEPT",  result: "OK");
244
245             return last.equals(proposal);
246         }
247     }
```

How to make the servers down

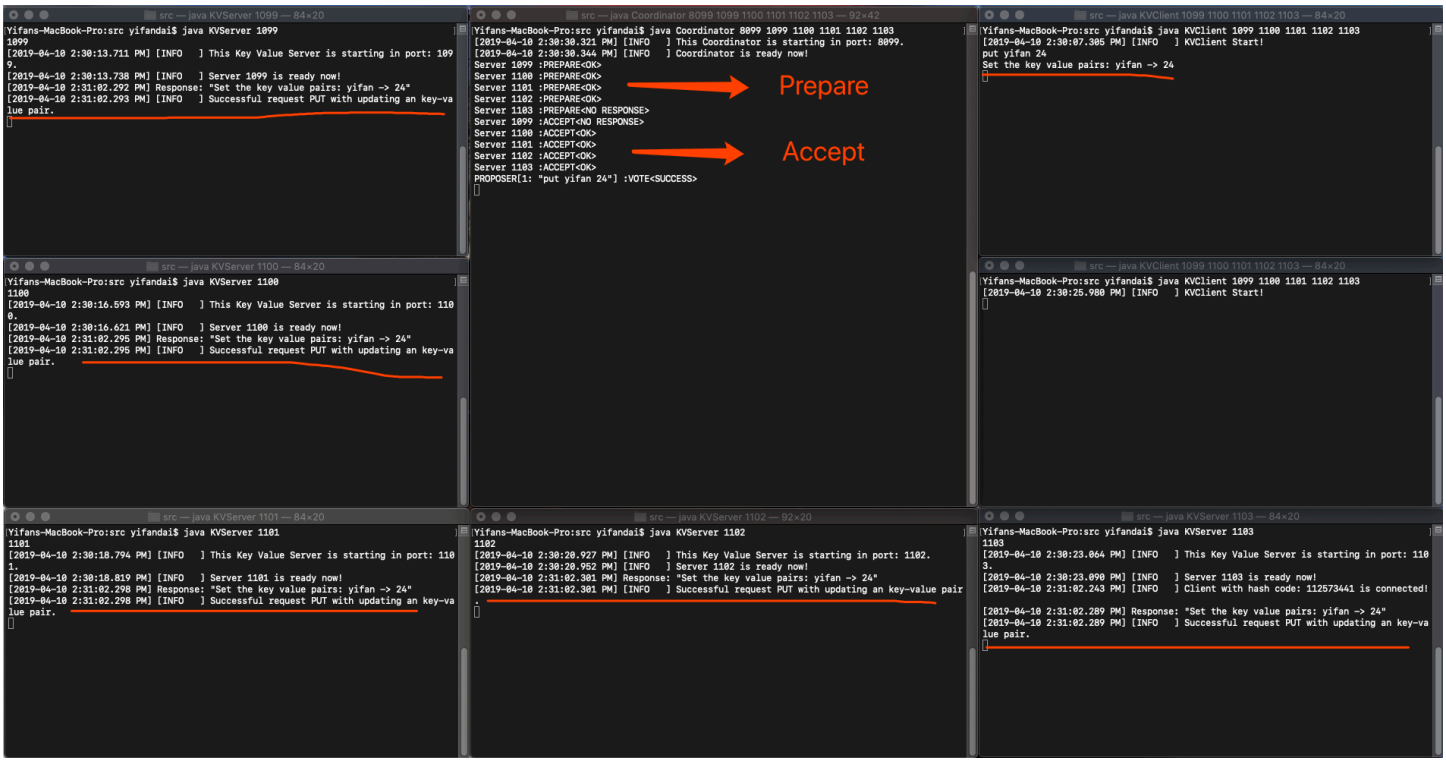
We could simulate that by simply shut down the server (ctrl + c or close the terminal).

Code Test Example

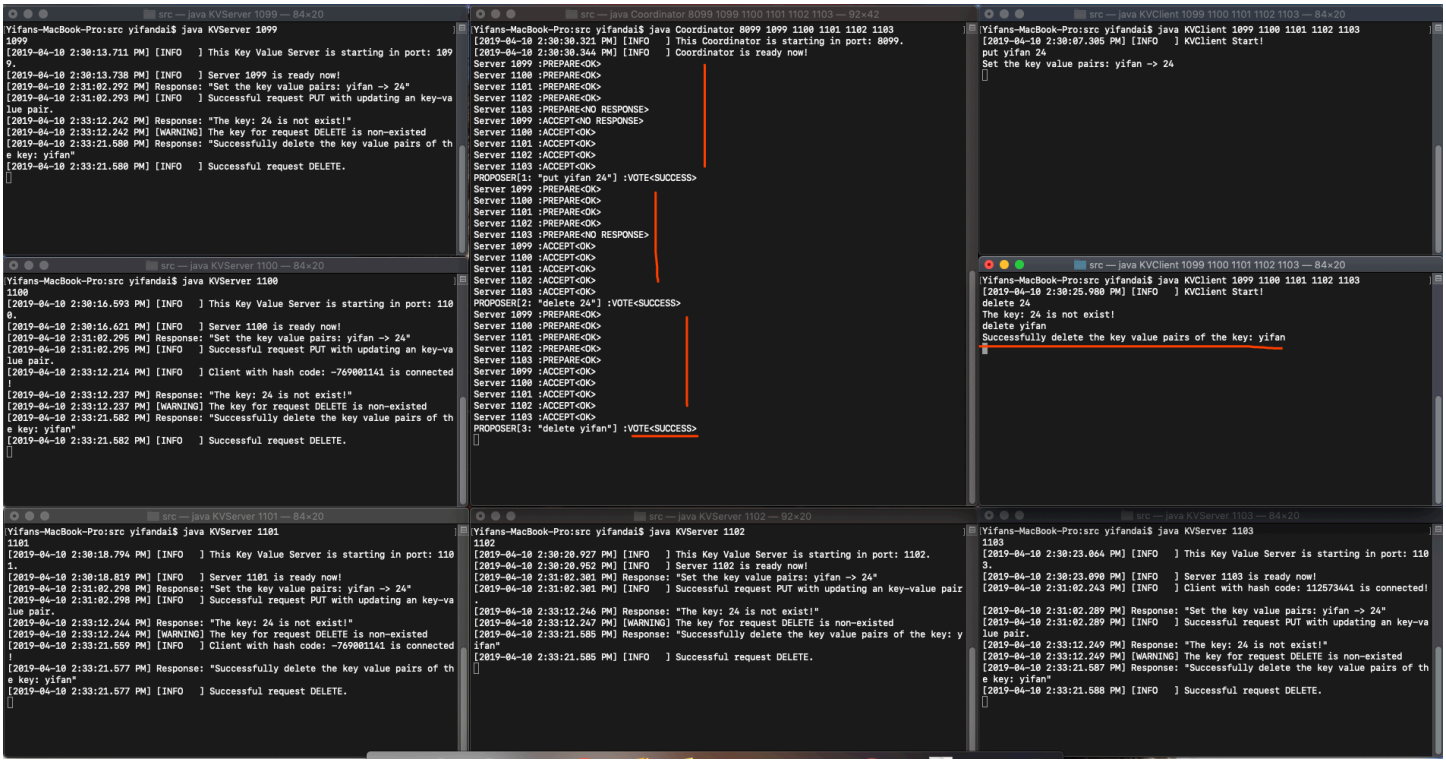
We could start 5 servers, 1 coordinator and 2 clients for testing first (I use the screenshot on my Mac as the example):



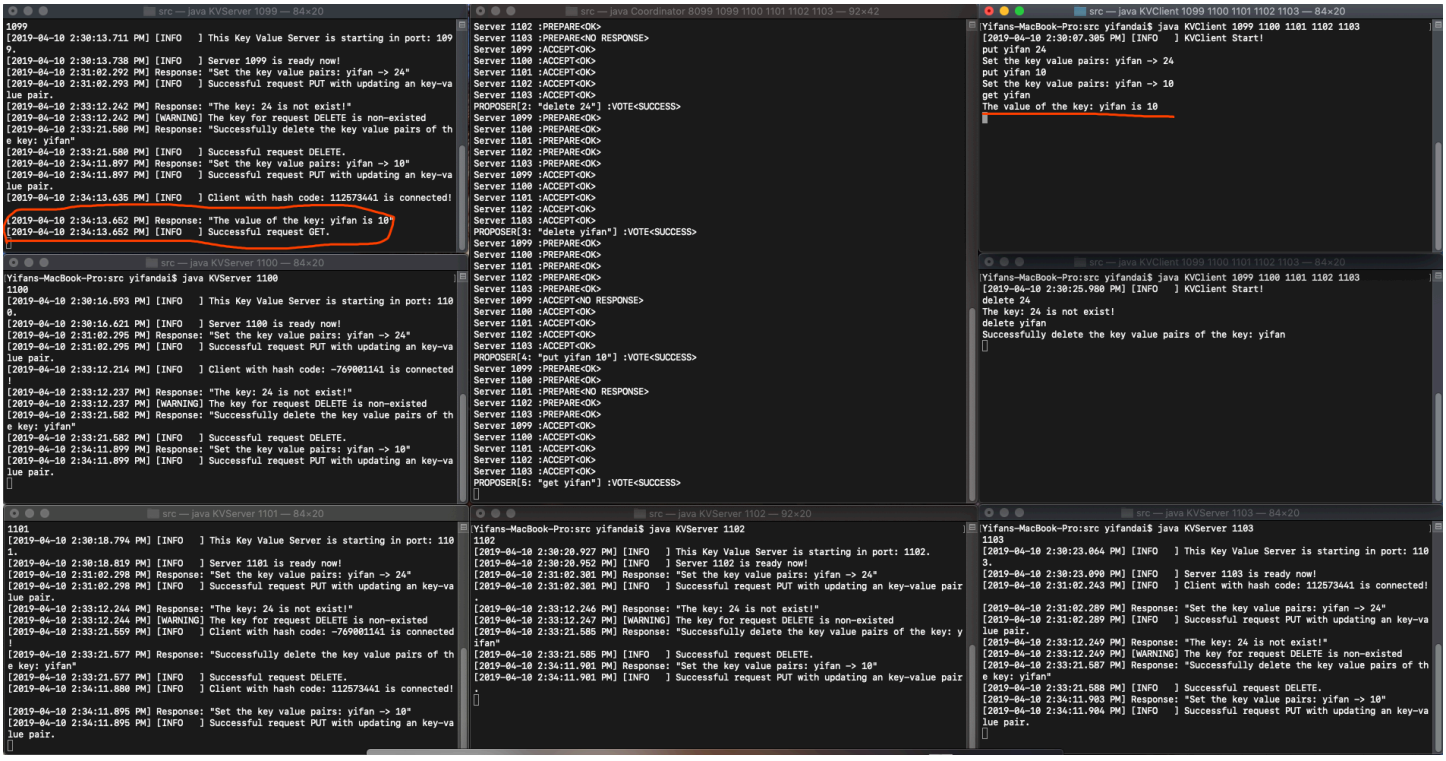
The client 1 send requests PUT:



We can see that for the requests PUT, all the servers update at the same time. It is also the same for the delete:

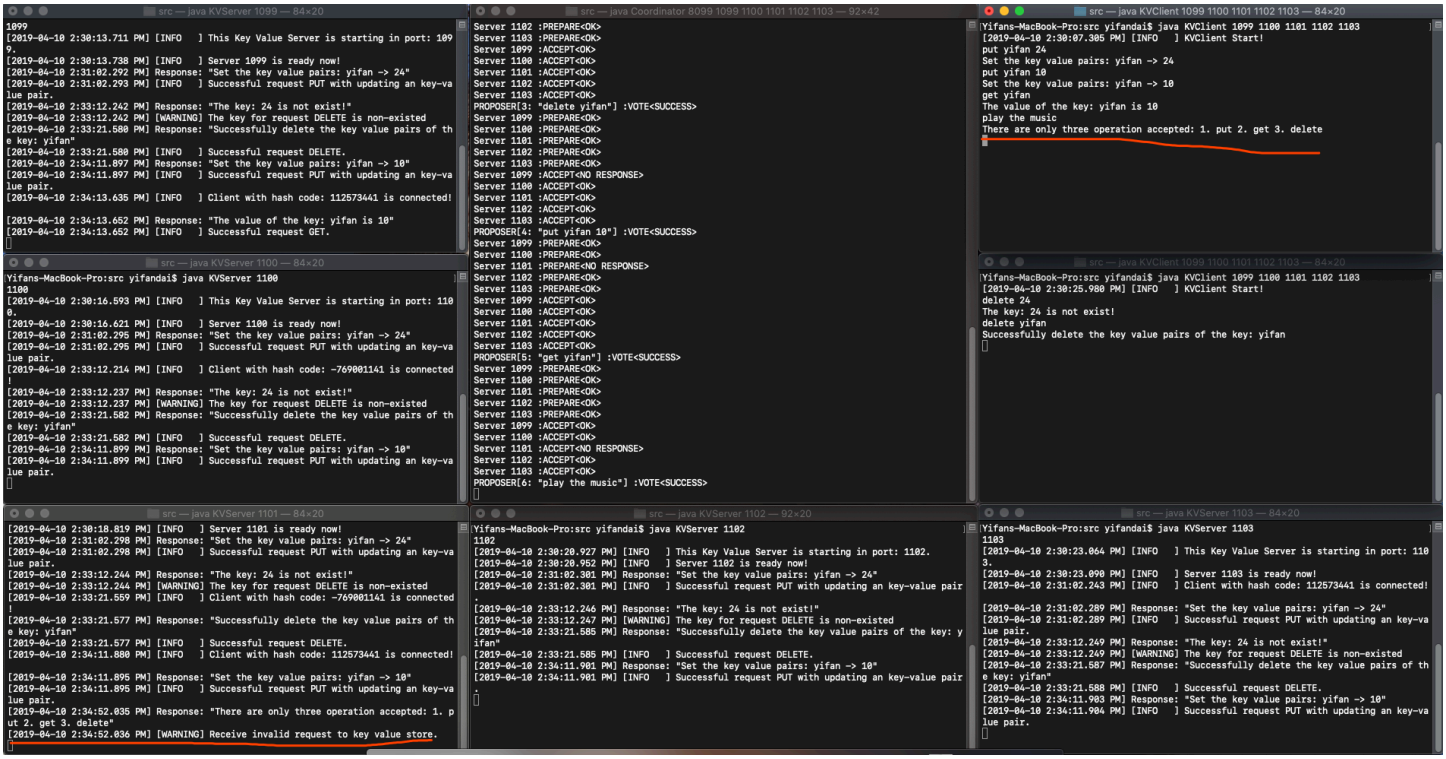


You can see that the coordinator has some log for the result of the **Prepare** and **Accept** phase. It will show whether a server accept, reject or has no response to the proposal. Every proposal will have a proposal number and a string which is same as the request the client sent.

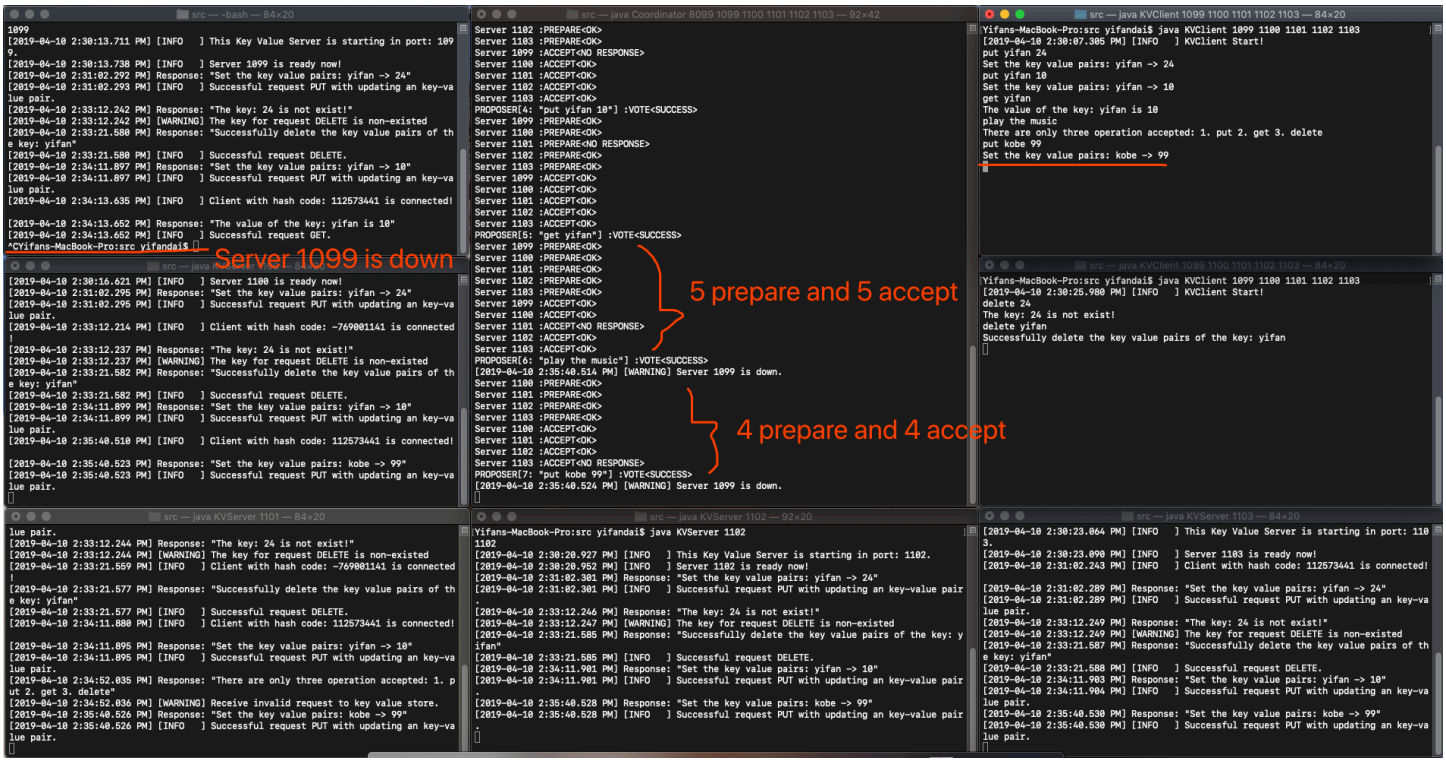


Above is an example of the **GET**, the **GET** operation will only be executed by the server which the client connect to but no need to synchronize with other servers (The server which process the request is underlined by the red line).

The server will not process all other invalid request (red line):



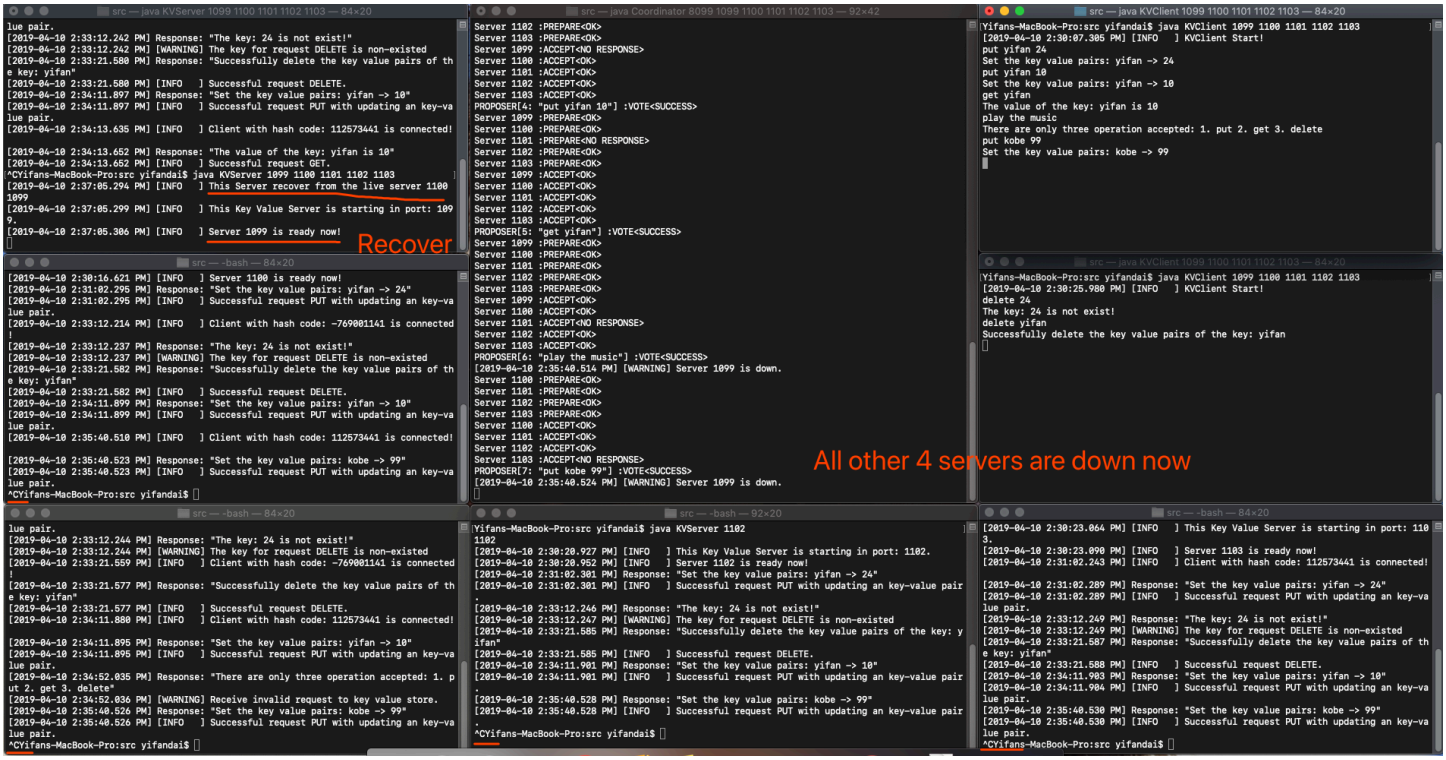
Then I will show how the servers work when a server is down.



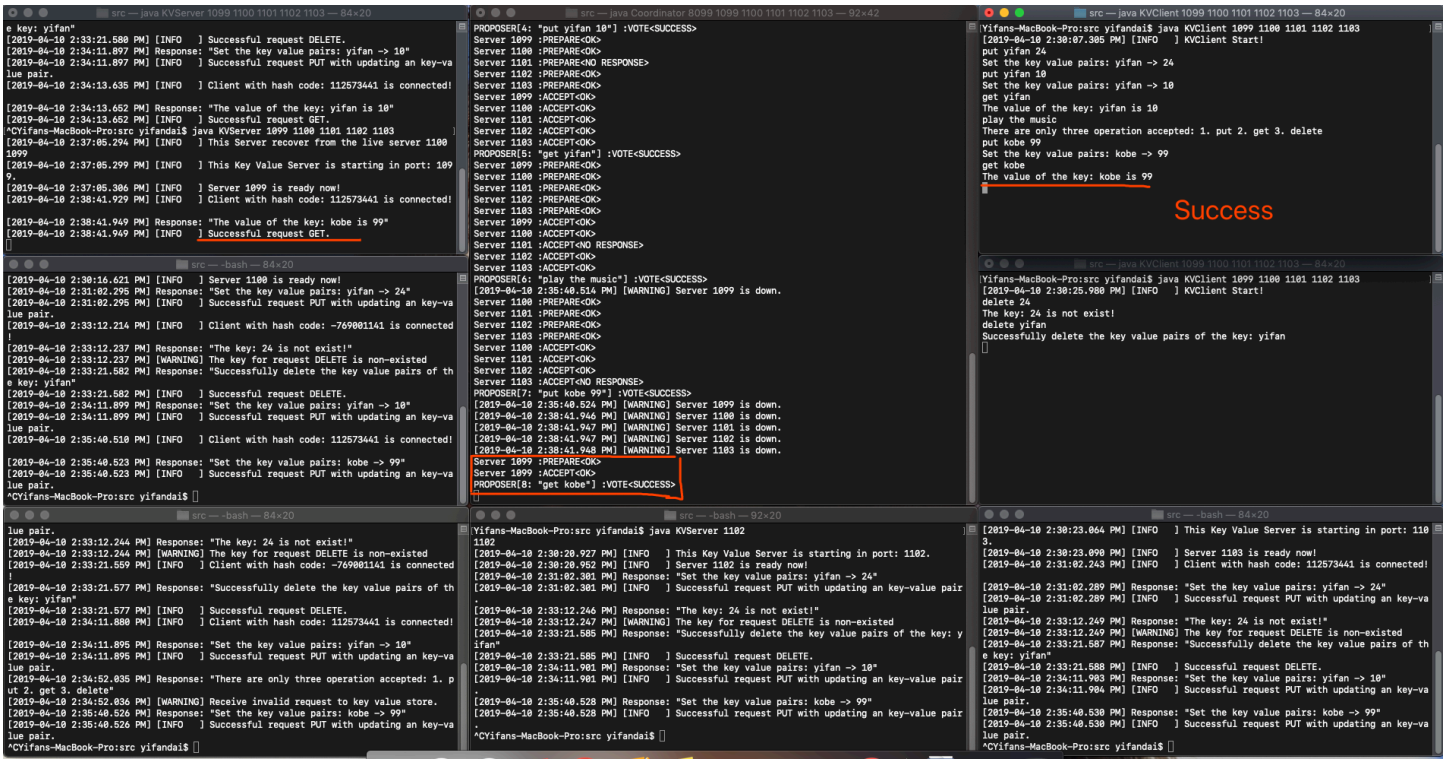
I shut down the **server 1099** and there are only 4 running servers. And the Coordinator only collect 4 responses but not 5 responses. If some servers are down, then the quorum is based on the number of live servers, not including the servers which are already down.

Now, we can test whether the recovery works well by this:
We first recover the **server 1099** and then close all other 4 servers and check whether the ***get kobe*** request works (If it works, then the recover is success, because all other 4 servers are down and the **server 1099** is the only available one).

First start **server 1099** and then shut down all other 4 servers:



Then we test whether the ***get kobe*** request works:

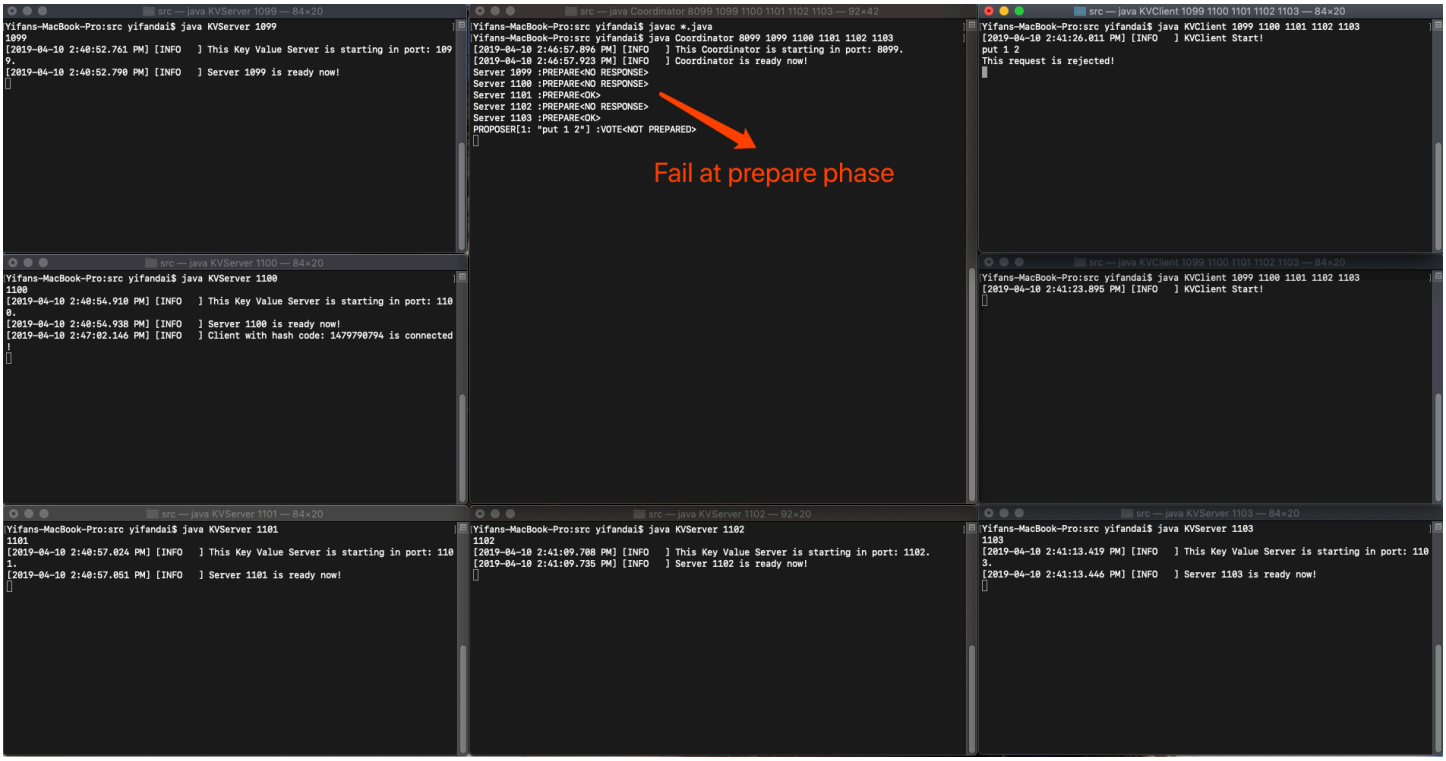


The ***get kobe*** can get the correct value. It means that the recovery works well!

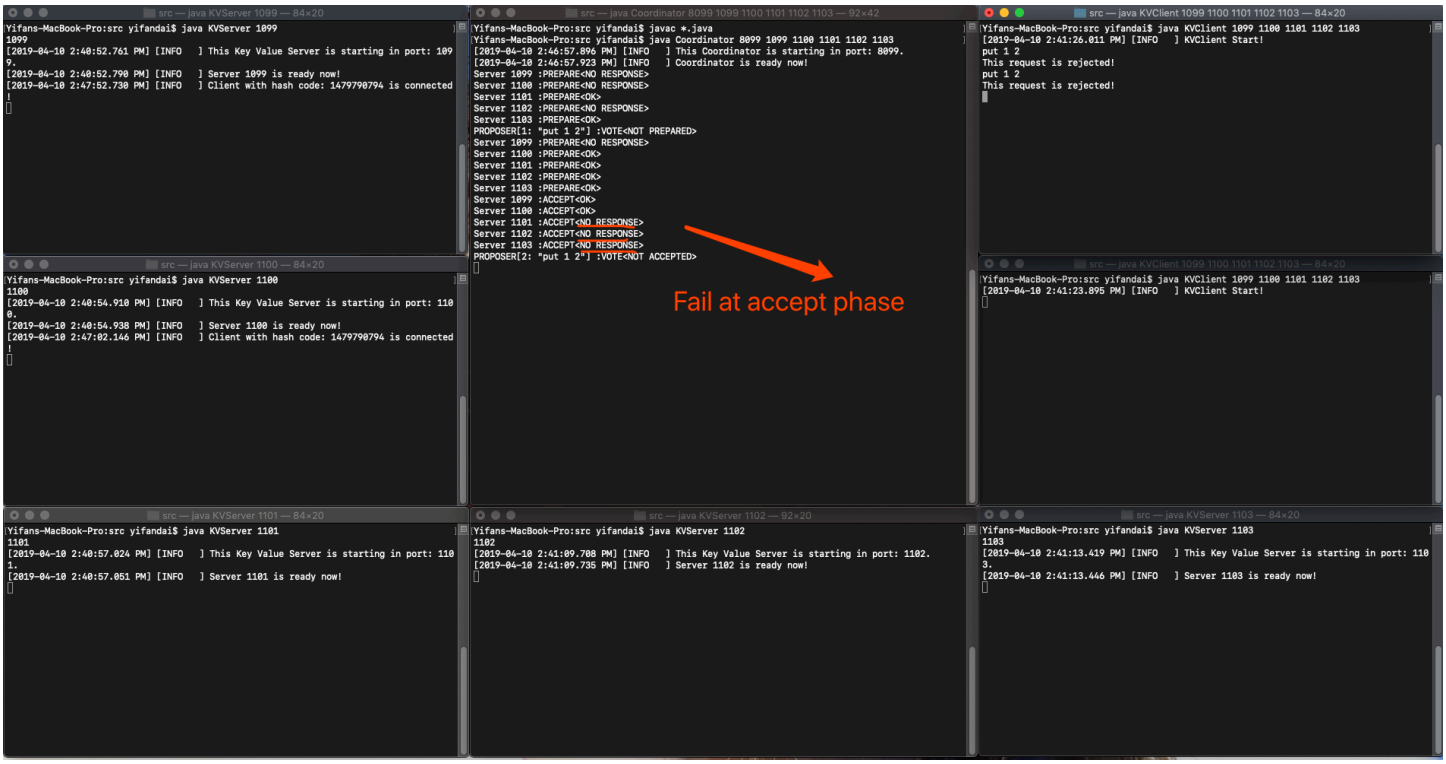
At last, I will show how the requests are rejected. We can see the ***Coordinator.java*** file. Increase the failure probability to 50% (To increase the probability to fail).

```java
    public static class Acceptor implements Serializable {

        // last vote result
        public Proposal last = new Proposal();
        public String name;

        public Acceptor(String name) { this.name = name; }

        public Promise onPrepare(Proposal proposal) {
            // We suppose that the failure probability is 10%
            if (Math.random() < 0.5) {
                printInfo( subject: "Server " + name,  operation: "PREPARE",  result: "NO RESPONSE");
                return null;
            }
            if (proposal == null)
                throw new IllegalArgumentException("null proposal");
            if (proposal.getVoteNumber() > last.getVoteNumber()) {
                Promise response = new Promise( ack: true, last);
                last = proposal;
                printInfo( subject: "Server " + name,  operation: "PREPARE",  result: "OK");
                return response;
            } else {
                printInfo( subject: "Server " + name,  operation: "PREPARE",  result: "REJECTED");
                return new Promise( ack: false,  proposal: null);
            }
        }

        public boolean onAccept(Proposal proposal) {
            // We suppose that the failure probability is 10%
            if (Math.random() < 0.5) {
                printInfo( subject: "Server " + name,  operation: "ACCEPT",  result: "NO RESPONSE");
                return false;
            }
            printInfo( subject: "Server " + name,  operation: "ACCEPT",  result: "OK");

            return last.equals(proposal);
        }
    }
```

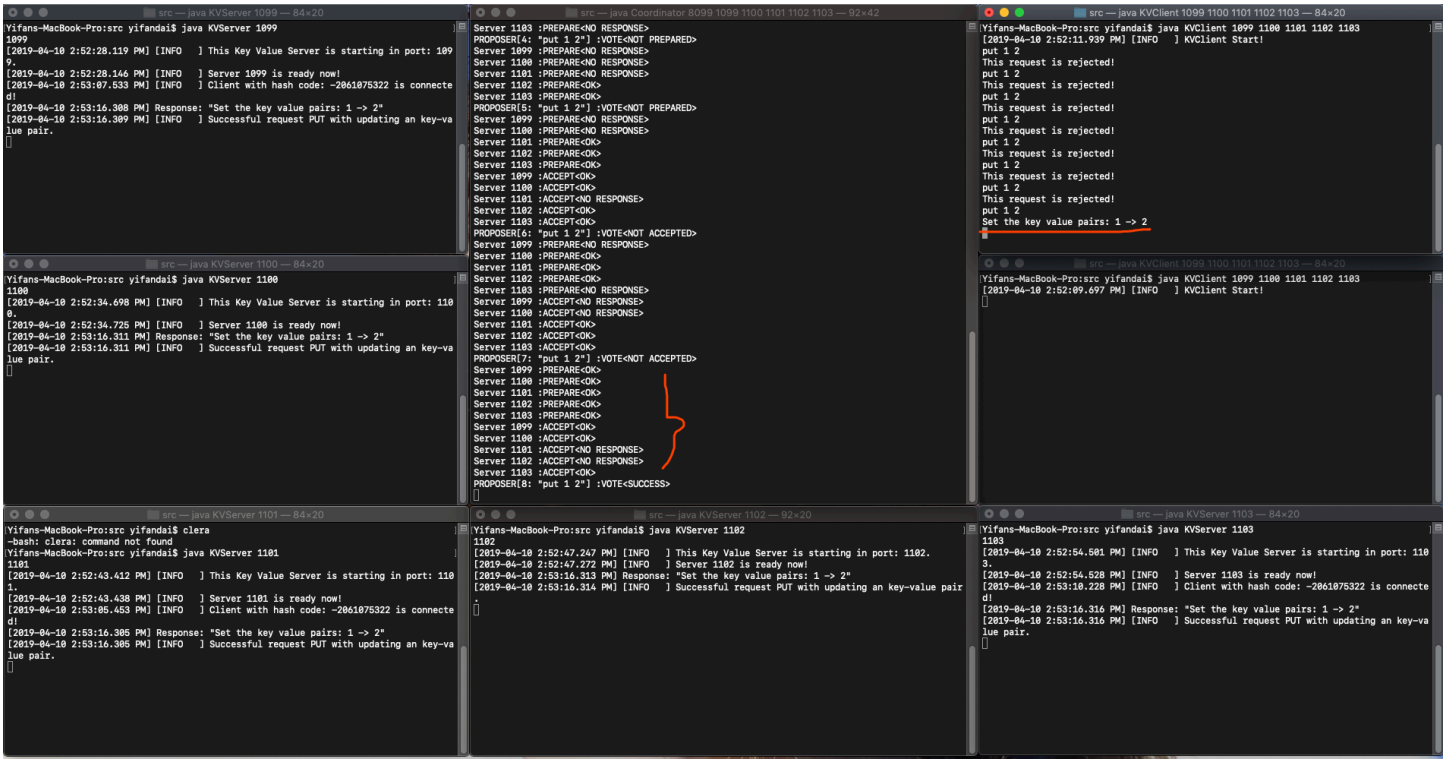
Then we could run the codes again to see what will happen when fails.
Below is an example of failing at the prepare phase (3 no response when prepare).



Below is an example of failing at the accept phase (3 no response when accept).



Last, we can see a successful request:



Only when both the prepare phase and the accept phase reach the quorum, the proposal will be accepted and the requests will be executed by the server.

This is all the code instruction of the project 4.
Thank you!

Yifan Dai
On 04/10/2019