ELEC 477

Kevin Yu: 20203451

Raatik Sharma: 20120770

Ainsley Taylor: 20210012

2024-03-17

Test

For this assignment, you should only check that the put messages are forwarded to the replica servers and the values are changed. You can do this with trace methods.

To test the replicas servers and ensure that the values are changed, I added trace methods (console logs) to the terminal to show what values were received and being stored in the kvPut() method for the replica servers. Afterwards, I would call kvGet() to illustrate that the value was stored in the GDBM file by the replica. In Figure 1, we can see that I have a replica server called “kvserver2” which is the replica server for “kvserver1”. Once the kvput method is called, the primary server forwards the kvput call to its replica server and we have trace methods to showcase what values are received and stored on the replica server as seen in Figure 2. Lastly, in the trace logs in Figure 3, we see the put request from the client to the primary server and then the kvPut method is forwarded to the replica server where the key and value pairs are stored and retrieved respectively.

main.cpp 2 ×

A2SolutionNetworkV2NewSemantics > main.cpp > main(int, char * [])

```
22  int main(int argc, char *argv[])
26      std::stringstream ss;
27
28      // start all of the servers first. This will let them get up
29      // and running before the client attempts to communicste
30      std::cout << "Main: ***********************************" << std::endl;
31      std::cout << "Main: starting server" << std::endl;
32
33      std::cout << "Main: starting directory server" << std::endl;
34      shared_ptr<SrvcDirServer> svcDirServer = make_shared<SrvcDirServer>("ServiceDirectory");
35      svcDirServer->setAddress("10.0.0.2");
36      svcDirServer->init();
37      svcDirServer->startServices();
38
39      ServerData::ServerInfo replica1 = {"kvserver2", 5000};
40      ServerData::ServerInfo replica2 = {"kvserver3", 5001};
41      vector<ServerData::ServerInfo> *replicas = new vector<ServerData::ServerInfo>;
42      ;
43      replicas->push_back(replica1);
44      replicas->push_back(replica2);
45
46      ServerData::ServerInfo *primaryServer = new ServerData::ServerInfo{"kvserver1", 5193};
47
48      // Primary Server
49      shared_ptr<KVServer> kvServer1 = make_shared<KVServer>("kvserver1", replicas, primaryServer, 1);
50
51      kvServer1->setAddress("10.0.0.3");
52      kvServer1->setDBMFileName("server1");
53      kvServer1->setSvcDirServer("ServiceDirectory");
54      kvServer1->setSvcName("kv1");
55      kvServer1->setPort(5193);
56      kvServer1->init();
57
58      // Replica Server 1
59      shared_ptr<KVServer> kvServer2 = make_shared<KVServer>("kvserver2", nullptr, primaryServer, 0);
60      kvServer2->setAddress("10.0.0.4");
61      kvServer2->setDBMFileName("server2");
62      kvServer2->setSvcDirServer("ServiceDirectory");
63      kvServer2->setSvcName("replica");
64      kvServer2->setPort(5000);
65      kvServer2->init();
66
```

Figure 1: Main.cpp file

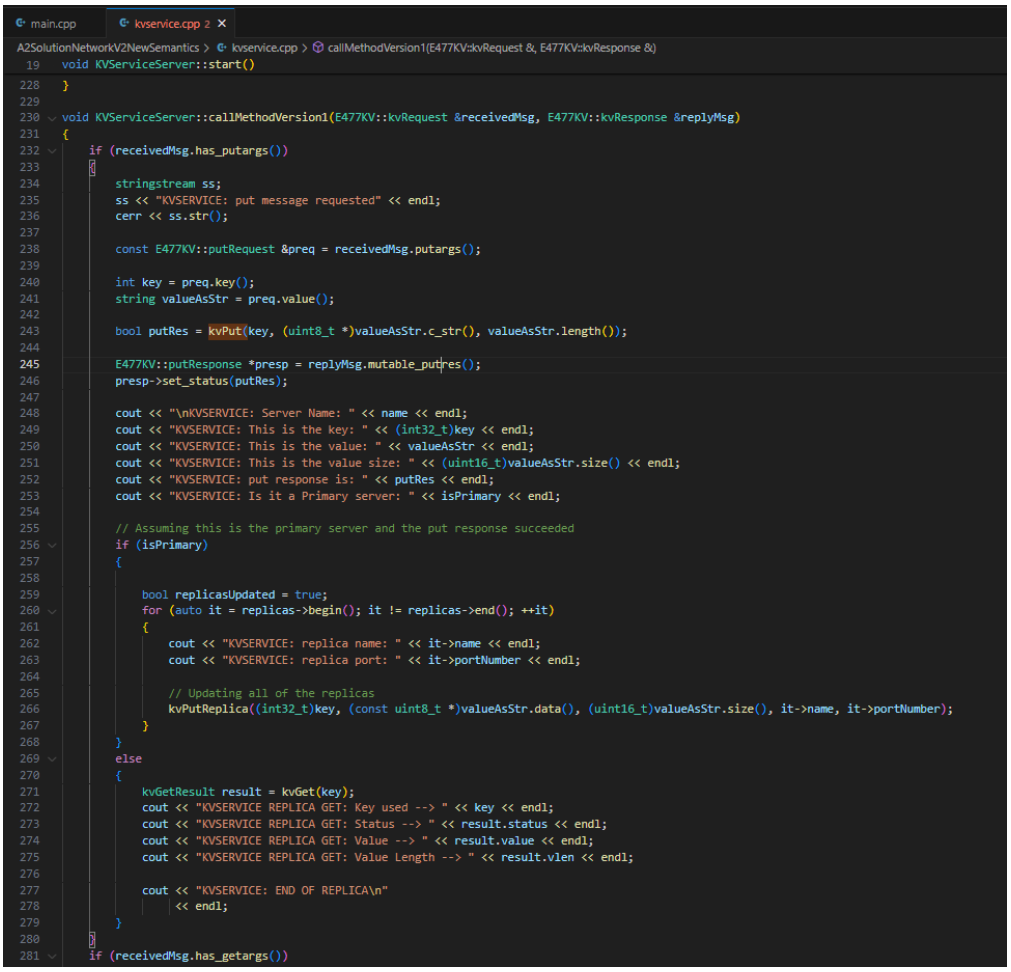

Figure 2: kvservice.cpp

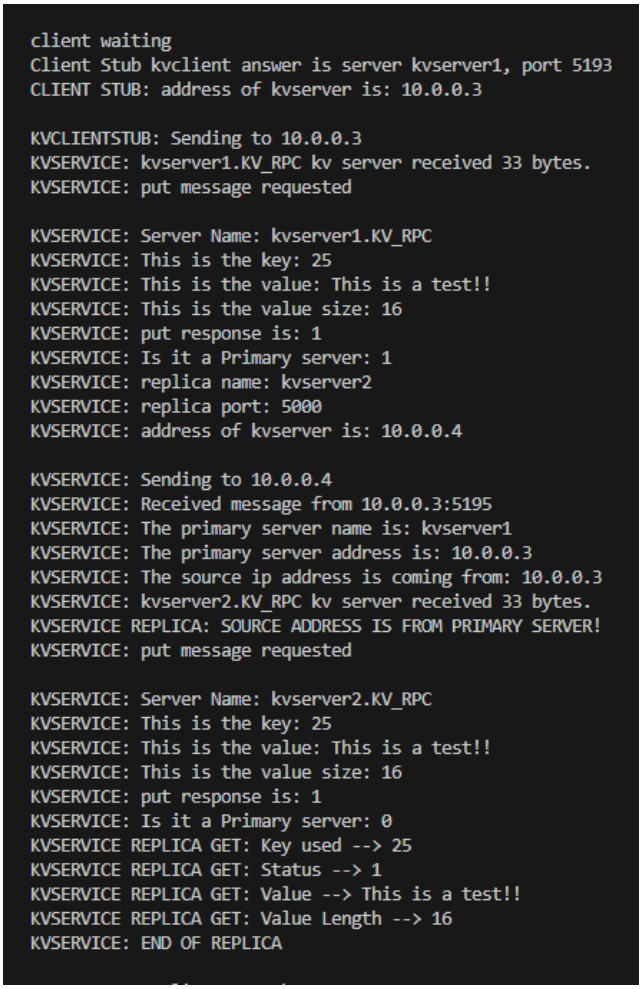

Figure 3: Trace Values