Additions

To create the primary backup system, the Kvserver and respective Kvservice (that runs on the KVserver) was updated.

The KVservice saw the addition of:

* A variable to track whether or not a server was the primary (I assumed the primary server would be the holder of the primary value of every data item (key value))
  + As well as a function to set this variable: setPrimary()
* A vector containing pairs consisting of a name and a port, called ‘replicas’. This is for storing the name and port of every replica server as the assignment states
  + As well as a function to add replicas to this vector: addReplica()
* A propagatemessage() function. This function is used to send a received request to another server.
  + This function waits for a response to the sent request before ending. In our case, this is the confirmation that the replica executed the propagatedwrite command.
* Within kvput() a check was added after the storing of the key value to check whether or not a server was a primary server. If it is a primary server, kvput() will utilize propagatemessage() to propagate the kvput() request to all replicas of the server—found by iterating through ‘replicas’.

setPrimary and addReplica() were made public so that they could be implemented in KVserver. That way any instance of a KVserver created in main.cpp, could have it set to be a primary server and could have replicas added to it.

Finally, within main.cpp, in addition to the starting of kvserver1, I added the creation of kvserver2—a replica of kvserver1. Kvserver2 is not registered with the service directory server as instructed. After, kvserver2’s creation, kvserver2’s port was added to kvserver1’s ‘replias’ vector and kvserver1 was asset as a primary server.

Testing

By placing a kvput() request within the kvclient’s start(), then checking a replica’s server gdbm file for the requested key value you could determine whether or not the replication system created works.