

The serverController.py was intended to be the driver code so you can pick between eventual, sequential, etc. But that was not finished.

Client.py is almost the same as the first KV store:

```
client.py > ...
1  from fileinput import filename
2  from socket import *
3  from curses import raw
4  import sys
5
6  # Port and server info
7  serverName = 'localhost'
8  serverPort = int(sys.argv[1])
9
10 #Connecting
11 clientSocket = socket(AF_INET, SOCK_STREAM)
12 clientSocket.connect((serverName, serverPort))
13
14 # User input
15 sentence = input(str(r'Input set/get space key space value:')).encode("utf-8")
16 # set 8 ploy
17
18 #sending it to sever
19 clientSocket.send(sentence)
20
21 #This is where you receive info from server after connecting and sending to server
22 modifiedSentence = clientSocket.recv(1024)
23 removeString = str(modifiedSentence, 'utf-8')
24 print('From Server: ', removeString)
25 clientSocket.close()
```

Eventual.py is the eventual consistency one. It starts with the socket connections and initializes a dictionary and makes a new file called ploytest.txt to test it.

```
eventual.py × test.py sequential.py serverController.py
eventual.py > getSet
1  from socket import *
2  import threading
3  import time
4  import socketserver
5  from _thread import *
6  import os.path
7  import sys
8  import socket
9  from threading import Thread
10
11  from eventual import getSet
12
13  # Socket connections
14  serverPort = int(sys.argv[1])
15  serverName = 'localhost'
16
17  # need diff serverPorts for each replicas for each new consistency
18
19  serverSocket = socket(AF_INET,SOCK_STREAM)
20  serverSocket.bind(("127.0.0.1", serverPort))
21  print("Ready to receive \n")
22  serverSocket.listen()
23  # locks = threading.Lock()
24
25  # Initialize dictionary and make new file
26  kvstore = {}
27  file = open('poytest.txt', 'a+')
28
29  # dont need a lock for eventual consistency
30
31  def getSet():
32      #Client input received here
33      sentence = connectionSocket.recv(1024).decode()
34
35      # Checks input request type if's get or set by using split sentence
```

In the getSet() method, It splits up the request type and writes it into the kvstore and encodes it. To connect it to different servers, there is a for loop that reads each line in serverPorts.txt and sends that key value to the different ports.

```

def getSet():
    #Client input received here
    sentence = connectionSocket.recv(1024).decode()

    # Checks input request type if's get or set by using split sentence
    request_type = sentence.split()[0]
    print(str(request_type))
    if str(request_type) == "set":

        keyword = sentence.split()[1]
        value = sentence.split()[2]

        # print(keyword, value)
        ## set ops
        #print(str(request_type))
        kvstore[keyword] = value

        # locks.acquire()

        file.write('%s:%s\n' % (keyword, value))

        print(kvstore[keyword])
        connectionSocket.send('successfully saved!'.encode())

    # read server port numbers and send to key value to that port
    f = open('serverPorts.txt', "r")
    for x in f:
        clientSocket = socket(AF_INET, SOCK_STREAM)
        clientSocket.connect((serverName, x))
        print(x)
    f.close

    # locks.release()

```

≡ serverPorts.txt

1	1207
2	1206
3	1205

Finally, you create a new thread at the very end:

```
# You get connected here
while 1:
    connectionSocket, addr = serverSocket.accept()
    new_thread = Thread(getSet, connectionSocket)
    new_thread.start()
    new_thread.join()
```

Here is one test I did:

```
test.py > ...
1  import socket
2  from threading import Thread
3
4  from eventual import getSet
5
6  HOST = "127.0.0.1" # Standard loopback interface address (localhost)
7  PORT = 65432 # Port to listen on (non-privileged ports are > 1023)
8
9  with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
10     s.bind((HOST, PORT))
11     s.listen()
12     while True:
13         connectionSocket, addr = s.accept()
14         new_thread = Thread(getSet, connectionSocket)
15         new_thread.start()
16         new_thread.join()
17
```