

CSC 255

FALL 2016

VISHNU PRASANTH SEERANGARAJAN  
SIVABALAN BALASUBRAMANIAN

ASSignment #3 MULTISERVER CHAT- PART2

INSTRUCTOR: Dr. ISAAC GHANSAH

Table of Contents

Introduction2

Part 1: Assignment Objective2

Part 2: Design approach and Justification2

a)Frontend Server Code:3

b) Client Code:7

c) Backend Server 1 Code:8

d) Backend Server 2 Code:8

e) Backend Server 3 Code :9

f) Backend Server 4 Code:9

g) Execution Steps:10

h) Screenshots:10

Part 3: Document Ownership11

Part 4: Challenges11

Part 5: Lessons Learned12

**Introduction:**

The objective of this assignment is to develop a multi-server chat i.e. multiple co-operating servers with location transparency. The application consists of multiple clients who can join to a front-end server using its IP address. Front-end sever manages multiple backend servers for **reliability** and **Load Balancing**. There are two or more backend servers which handle client requests and respond to all active clients. We have also **encrypted** and **decrypted** the chat for security.

**Assignment Objective:**

For this assignment, we have implemented multi-server chat with **reliability** and **Load Balancing**. For reliability, if one of the backend servers is crashed or terminated, the other server handles the request from clients. For Load Balancing, the requests from clients are balanced to the backend server by the front end server based on a **number of clients** and **size of the message** as the critical factors. If the message length is greater than 7 then by default the messages are sent to Primary Server 1. If the primary server is not running, then the request is sent to Primary Server 2.

**Design Approach and Justification:**

We developed this application in python. We have three applications for

* Client
* Front-end server
* Back-end server (1-4)

**Client:**

Client will be prompted to enter the IP address of the front-end server to connect to the chat application. Once correct IP address is entered, front-end server adds the client address to a list and it creates a connection with the client. Multiple clients can send request to frontend server. Client application runs two threads- one for sending messages and other for receiving message from different clients.

**Front-end server:**

There is one front-end server running in our application. It receives the entire request from clients and forwards it to multiple backend servers. It keeps track of connected backend servers in a list and a flag to check the connection status of the backend server. It has three threads running in parallel. First thread is to monitor backend server. Second thread is to receive messages from clients. Third is to update the client list to active backend servers.

**Backend server:**

There are four replicated backend servers running in different machines to handle reliability and load balancing. These backend servers receives request from frontend server and sends only to connected clients.

**Config file:**

The config file has the list of primary and secondary server with its corresponding IP address. The data is given to frontend server.

**Frontend Server Code:**

from socket import \*

from socket import \*

import threading

from Crypto.Cipher import AES

Host = ''

Porta = 4000

Portb = 4005

Portc = 4006

Portd = 4007

BEPort1 = 4001

BEPort2 = 4002

BEPort3 = 4003

BEPort4 = 4004

Port3=4002

flagForConnectedPrimaryBEserver1 = 1

flagForSecondaryBEserver1 = 1

flagForConnectedPrimaryBEserver2 = 1

flagForSecondaryBEserver2 = 1

def do\_encrypt(message):

obj = AES.new('key1234567890123', AES.MODE\_CBC, '1234567890123456')

ciphertext = obj.encrypt(message)

return ciphertext

def do\_decrypt(ciphertext):

obj2 = AES.new('key1234567890123', AES.MODE\_CBC, '1234567890123456')

message = obj2.decrypt(ciphertext)

return message

connectedClients=[]

file1=open('configfile','r')

for line in file1:

linelist = line.split()

if linelist[0] == "primary1":

connectedPrimaryBEserver1 = linelist[1]

elif linelist[0] == "primary2":

connectedPrimaryBEserver2 = linelist[1]

elif linelist[0] == "secondary1":

connectedSecondaryBEserver1 = linelist[1]

elif linelist[0] == "secondary2":

connectedSecondaryBEserver2 = linelist[1]

print "Primary BE server is", connectedPrimaryBEserver1

print "Secondary BE server is", connectedSecondaryBEserver1

print "Primary BE server2 is", connectedPrimaryBEserver2

print "Secondary BE server2 is", connectedSecondaryBEserver2

print "flagForConnectedPrimaryBEserver1", flagForConnectedPrimaryBEserver1

print "flagForSecondaryBEserver1", flagForSecondaryBEserver1

print "flagForConnectedPrimaryBEserver2", flagForConnectedPrimaryBEserver2

print "flagForSecondaryBEserver2", flagForSecondaryBEserver2

socket1 = socket(AF\_INET, SOCK\_DGRAM)

socket2 = socket(AF\_INET, SOCK\_DGRAM)

socket3 = socket(AF\_INET, SOCK\_DGRAM)

socket4 = socket(AF\_INET, SOCK\_DGRAM)

socket5 = socket(AF\_INET, SOCK\_DGRAM)

socket1.bind((Host,Porta))

socket3.bind((Host,Portb))

socket4.bind((Host,Portc))

socket5.bind((Host,Portd))

socket2.bind((Host,Port3))

socket1.settimeout(5.0)

# socket2.settimeout(5.0)

socket3.settimeout(5.0)

socket4.settimeout(5.0)

socket5.settimeout(5.0)

def serverside1():

global flagForConnectedPrimaryBEserver1

while 1:

data = "SYN-SENT"

# Primary Back End server 1

socket1.sendto(data,(connectedPrimaryBEserver1, BEPort1))

# print "sent1"

try:

data2, address = socket1.recvfrom(2048)

# print "received 1"

if data2 == "SYN-RECEIVED":

flagForConnectedPrimaryBEserver1 = 1

else:

flagForConnectedPrimaryBEserver1 = 0

except Exception:

flagForConnectedPrimaryBEserver1 = 0

def serverside2():

global flagForSecondaryBEserver1

while 1:

data = "SYN-SENT"

# Primary Back End server 1

socket3.sendto(data,(connectedSecondaryBEserver1, BEPort2))

# print "sent2"

try:

data2, address = socket3.recvfrom(2048)

# print "received 2"

if data2 == "SYN-RECEIVED":

flagForSecondaryBEserver1 = 1

else:

flagForSecondaryBEserver1 = 0

except Exception:

flagForSecondaryBEserver1 = 0

def serverside3():

global flagForConnectedPrimaryBEserver2

while 1:

data = "SYN-SENT"

# Primary Back End server 1

socket4.sendto(data,(connectedPrimaryBEserver2, BEPort3))

# print "sent3"

try:

data2, address = socket4.recvfrom(2048)

# print "received 3"

if data2 == "SYN-RECEIVED":

flagForConnectedPrimaryBEserver2 = 1

else:

flagForConnectedPrimaryBEserver2 = 0

except Exception:

flagForConnectedPrimaryBEserver2 = 0

def serverside4():

global flagForSecondaryBEserver2

while 1:

data = "SYN-SENT"

socket5.sendto(data,(connectedSecondaryBEserver2, BEPort4))

# print "sent 2"

try:

data2, address = socket5.recvfrom(2048)

# print "received 4"

if data2 == "SYN-RECEIVED":

flagForSecondaryBEserver2 = 1

else:

flagForSecondaryBEserver2 = 0

except Exception:

flagForSecondaryBEserver2 = 0

def clientside():

global flagForConnectedPrimaryBEserver1, flagForSecondaryBEserver1

global flagForConnectedPrimaryBEserver2, flagForSecondaryBEserver2

global even, odd

# print "entered client recv thread outside"

while 1:

# print "entered client recv thread inside"

data, address = socket2.recvfrom(2048)

print "client joined from address", address

data2 = do\_decrypt(data)

data3 = data2.split('^')

data = data3[0]

load = len(data)

data = data.ljust(16, '^')

data = do\_encrypt(data)

if (address not in connectedClients):

print "new client is sent to BE", address

connectedClients.append(address)

update\_data="add!"+str(address)

socket1.sendto(update\_data,(connectedPrimaryBEserver1, BEPort1))

socket4.sendto(update\_data,(connectedPrimaryBEserver2, BEPort2))

socket3.sendto(update\_data,(connectedSecondaryBEserver1, BEPort3))

socket5.sendto(update\_data,(connectedSecondaryBEserver2, BEPort4))

odd = 0

even = 0

for i in range(len(connectedClients)):

if address == connectedClients[i]:

j = i%2

if j == 0:

even = 1

break

else:

odd = 1

break

else:

even = 0

odd = 0

data="message!"+str(address)+"!"+data

if load >= 7:

odd = 0

even = 0

if flagForConnectedPrimaryBEserver1 == 1:

print "data sent to P1"

socket1.sendto(data,(connectedPrimaryBEserver1, BEPort1))

elif flagForSecondaryBEserver1 == 1:

print "data sent to S1"

socket3.sendto(data,(connectedSecondaryBEserver1, BEPort2))

elif flagForConnectedPrimaryBEserver2 == 1:

print "data sent to P2"

socket4.sendto(data,(connectedPrimaryBEserver2, BEPort3))

elif flagForSecondaryBEserver2 == 1:

print "data sent to s2"

socket5.sendto(data,(connectedSecondaryBEserver2, BEPort4))

print "data received is", data

print "even is", even

print "odd is", odd

print "flagForConnectedPrimaryBEserver1", flagForConnectedPrimaryBEserver1

print "flagForSecondaryBEserver1", flagForSecondaryBEserver1

print "flagForConnectedPrimaryBEserver2", flagForConnectedPrimaryBEserver2

print "flagForSecondaryBEserver2", flagForSecondaryBEserver2

if even == 1:

if flagForConnectedPrimaryBEserver1 == 1:

print "data sent to P1"

socket1.sendto(data,(connectedPrimaryBEserver1, BEPort1))

elif flagForSecondaryBEserver1 == 1:

print "data sent to S1"

socket3.sendto(data,(connectedSecondaryBEserver1, BEPort2))

elif flagForConnectedPrimaryBEserver2 == 1:

print "data sent to P2"

socket4.sendto(data,(connectedPrimaryBEserver2, BEPort3))

elif flagForSecondaryBEserver2 == 1:

print "data sent to S2"

socket5.sendto(data,(connectedSecondaryBEserver2, BEPort4))

if odd == 1:

if flagForConnectedPrimaryBEserver2 == 1:

print "data sent to P2"

socket4.sendto(data,(connectedPrimaryBEserver2, BEPort3))

elif flagForSecondaryBEserver2 == 1:

print "data sent to S2"

socket5.sendto(data,(connectedSecondaryBEserver2, BEPort4))

elif flagForConnectedPrimaryBEserver1 == 1:

print "data sent to P1"

socket1.sendto(data,(connectedPrimaryBEserver1, BEPort1))

elif flagForSecondaryBEserver1 == 1:

print "data sent to S1"

socket3.sendto(data,(connectedSecondaryBEserver1, BEPort2))

def updateClient():

# print "entered updating client recv thread outside"

global flagForConnectedPrimaryBEserver, flagForconnectedSecondaryBEserver

while 1:

# print "entered updating client recv thread inside"

for i in range(len(connectedClients)):

update\_data="add!"+str(connectedClients[i])

socket1.sendto(update\_data,(connectedPrimaryBEserver1, BEPort1))

socket3.sendto(update\_data,(connectedSecondaryBEserver1, BEPort2))

socket4.sendto(update\_data,(connectedPrimaryBEserver2, BEPort3))

socket5.sendto(update\_data,(connectedSecondaryBEserver2, BEPort4))

Timer(3,updateClient).start()

first1 = threading.Thread(target = serverside1)

first1.daemon = True

first2 = threading.Thread(target = serverside2)

first2.daemon = True

first3 = threading.Thread(target = serverside3)

first3.daemon = True

first4 = threading.Thread(target = serverside4)

first4.daemon = True

second = threading.Thread(target = clientside)

second.daemon = True

third = threading.Thread(target = updateClient)

third.daemon = True

# second.setDaemon(False)

# third.setDaemon(False)

first1.start()

first2.start()

first3.start()

first4.start()

second.start()

third.start()

first1.join()

first2.join()

first3.join()

first4.join()

second.join()

third.join()

sys.exit()

**Client Code:**

from socket import \*

import threading

import sys

from Crypto.Cipher import AES

server=raw\_input('Please enter IP address to connect ')

# send an empty message only during the beginning of connection

def do\_encrypt(message):

obj = AES.new('key1234567890123', AES.MODE\_CBC, '1234567890123456')

ciphertext = obj.encrypt(message)

return ciphertext

def do\_decrypt(ciphertext):

obj2 = AES.new('key1234567890123', AES.MODE\_CBC, '1234567890123456')

message = obj2.decrypt(ciphertext)

return message

data = ' '

data = data.ljust(16,'^')

Port = 4002

socket1 = socket(AF\_INET, SOCK\_DGRAM)

socket1.sendto(data, (server, Port))

# socket1.timeout(5)

print "----------------Chat Begins---------------------"

def transmit():

while 1:

data = raw\_input("You:")

data2 = data.upper()

data3 = data2.ljust(16, '^')

data4 = do\_encrypt(data3)

socket1.sendto(data4, (server, Port))

if data == "EXIT" or data == "exit":

print "\n Exiting chat"

print "----------------------------------------------"

second.exit()

socket1.close()

break

def get():

while 1:

new\_data, address = socket1.recvfrom(2048)

new\_data = new\_data.split(':')

# print "length of new\_data", len(new\_data)

if len(new\_data[1]) == 16:

new\_data2 = do\_decrypt(new\_data[1])

new\_data3 = new\_data2.split('^')

print "\n", new\_data3[0]

if new\_data3[0] == "EXIT" or new\_data3[0] == "exit":

print "\n Exiting chat"

print "----------------------------------------------"

first.exit()

socket1.close()

break

else:

continue

first = threading.Thread(target=transmit)

second = threading.Thread(target=get)

first.start()

second.start()

first.join()

second.join()

sys.exit()

**Backend Server Code:**

from socket import \*

Host = ''

Port = 4001

socket1 = socket(AF\_INET, SOCK\_DGRAM)

socket1.bind((Host,Port))

connectedClients=[]

while 1:

# print connectedClients

data, address = socket1.recvfrom(2048)

# print data, address

request = data.split("!")

if request[0] == "add":

new\_client = eval(request[1])

if new\_client not in connectedClients:

connectedClients.append(new\_client)

elif request[0] == "message":

print data

for i in range(len(connectedClients)):

# print "connectedClients", connectedClients

if (connectedClients[i] != eval(request[1])):

socket1.sendto(request[1]+":"+request[2],connectedClients[i])

elif request[0] == "SYN-SENT":

socket1.sendto("SYN-RECEIVED", address)

else:

print "no case matched"

**Backend Server Code 2:**

from socket import \*

Host = ''

Port = 4002

socket1 = socket(AF\_INET, SOCK\_DGRAM)

socket1.bind((Host,Port))

connectedClients=[]

while 1:

# print connectedClients

data, address = socket1.recvfrom(2048)

# print data, address

request = data.split("!")

if request[0] == "add":

new\_client = eval(request[1])

if new\_client not in connectedClients:

connectedClients.append(new\_client)

elif request[0] == "message":

print data

for i in range(len(connectedClients)):

# print "connectedClients", connectedClients

if (connectedClients[i] != eval(request[1])):

socket1.sendto(request[1]+":"+request[2],connectedClients[i])

elif request[0] == "SYN-SENT":

socket1.sendto("SYN-RECEIVED", address)

else:

print "no case matched"

**Backend Server Code 3:**

from socket import \*

Host = ''

Port = 4003

socket1 = socket(AF\_INET, SOCK\_DGRAM)

socket1.bind((Host,Port))

connectedClients=[]

while 1:

# print connectedClients

data, address = socket1.recvfrom(2048)

# print data, address

request = data.split("!")

if request[0] == "add":

new\_client = eval(request[1])

if new\_client not in connectedClients:

connectedClients.append(new\_client)

elif request[0] == "message":

print data

for i in range(len(connectedClients)):

# print "connectedClients", connectedClients

if (connectedClients[i] != eval(request[1])):

socket1.sendto(request[1]+":"+request[2],connectedClients[i])

elif request[0] == "SYN-SENT":

socket1.sendto("SYN-RECEIVED", address)

else:

print "no case matched"

**Backend Server Code 4:**

from socket import \*

Host = ''

Port = 4004

socket1 = socket(AF\_INET, SOCK\_DGRAM)

socket1.bind((Host,Port))

connectedClients=[]

while 1:

# print connectedClients

data, address = socket1.recvfrom(2048)

# print data, address

request = data.split("!")

if request[0] == "add":

new\_client = eval(request[1])

if new\_client not in connectedClients:

connectedClients.append(new\_client)

elif request[0] == "message":

print data

for i in range(len(connectedClients)):

# print "connectedClients", connectedClients

if (connectedClients[i] != eval(request[1])):

socket1.sendto(request[1]+":"+request[2],connectedClients[i])

elif request[0] == "SYN-SENT":

socket1.sendto("SYN-RECEIVED", address)

else:

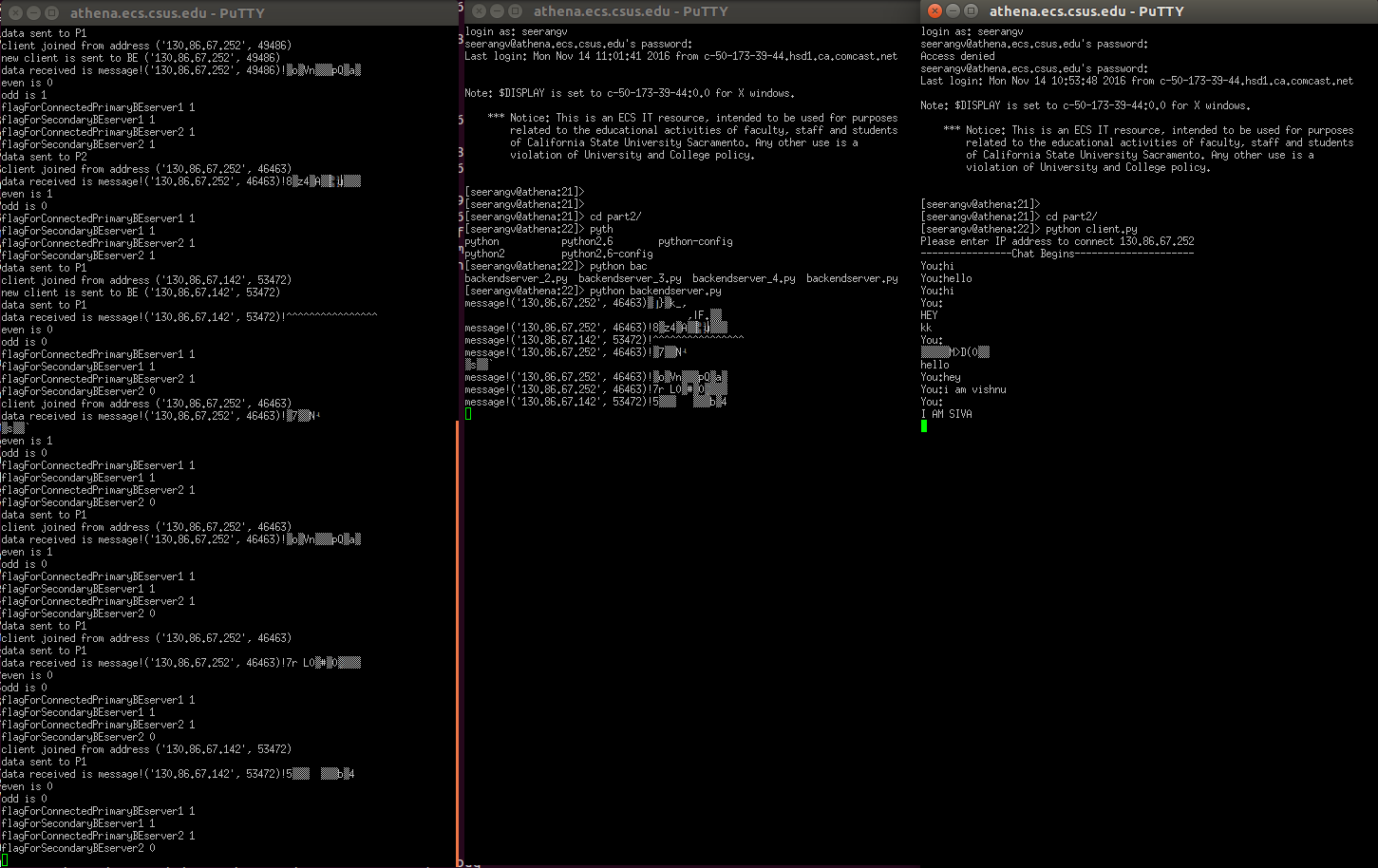
print "no case matched"

**Execution steps:**

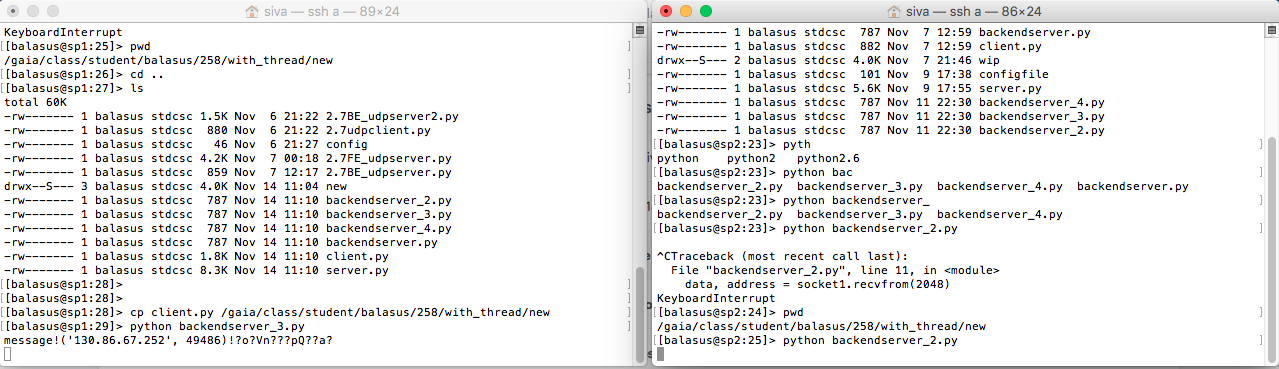
1. Run backendserver.py in IP address mentioned in configfile.
2. Run backendserver in (IP address: 130.86.67. 252) for primary server1 and mention the IP address in configfile.
3. Run one more backendserver 3 in sp1(IP address: 130.86.67.173) for secondary server 2
4. Run backendserver 2 in Sp2 (130.86.67.179) for Primary Server 2
5. Run backendserver 1 in Sp3 (130.86.67.142) for Secondary Server 1
6. Run server.py in any IP address(if it is Athena, it is 130.86.67.252).
7. Now, run client.py from any IP. After executing client.py, mention the IP address of server.py(if it is Athena, it is 130.86.67.252).
8. You can run any instance of client anywhere.
9. Now, you will be able to chat with other clients.

**Screenshots:**

Front end server displaying active backend servers, backend server running on Athena, one client.



Backend servers joined from Athena, Sp1, Sp2



Other client running on Athena.



**Document Ownership:**

* Front end server, encryption, decryption done by Vishnu Prasanth
* Back end server and Client done by Sivabalan
* Report done by both

**Challenges:**

There are lots of challenges which we faced during the implementation of this chat application. I am listing a few of them below:

* We faced problems in synchronization of the active backend servers.
* Managing clients and sending only to other clients.
* Managing threads in clients and front end server.
* Figuring out the factors on which message had to be load balanced.

**Lessons Learnt:**

* We learnt working as a team.
* We learnt implementing thread in python
* Using inbuilt functions to perform some actions in python.
* Implementing TCP using UDP with handshake.
* **Encryption** and **Decryption** of messages using **AES**.