

Computer Networking Mini-Project

Develop a TCP-socket based concurrent client-server system where the server can communicate with 7 clients simultaneously. The services provided by the server are file sharing, and distributed mathematical calculations

Dept : Information Technology

Year : 3rd | Sem : 6th

Kaustabh Ganguly	: Roll 55
Sayani Sarkar	: Roll 35
Debolina Saha	: Roll 61
Tanisha Paul	: Roll 23

ACKNOWLEDGEMENT

We want to thank PSB sir for guiding and helping us through every steps in this project.

Without his clear and vivid theory classes it wouldn't be possible.

Also we would like to address Stack Overflow forum for being a big helping hand in coding.

We used Python , Socket library and UNIX system.

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

They are the real backbones behind web browsing. In simpler terms there is a server and a client.

Client.py

```
import socket

s = socket.socket()
port = 12348
s.connect(('localhost', port))

def math():
    s.send(b'1')
    data = input("Enter mathematical operation ( +, - , / , * , ^ ) :")
    s.send(str(data).encode())
    r = s.recv(1024).decode()
    print("Result is :", r)
    s.close()

def ftp():
    s.send(b'2')

    fname = input("Enter Filename :")
    f = open (fname, "rb")
    l = f.read(1024)
    while (l):
        print("sending....")
        s.send(l)
        l = f.read(1024)
    f.close()
    s.close()

def ftp2():
    s.send(b'3')

    f =
open("server2client.txt", 'wb')
    l = s.recv(1024)
    while (l):
        f.write(l)
```

```
l = s.recv(1024)
f.close()
print("Success")
s.close()

while True :

    opt = int(input("1. Mathematical Operation -
press 1\n2. Send File -
press 2\n3. Recieve file
- press 3\n4. Quit -
press 4\n---> "))

    if opt == 1 :
        math()
        break
    elif opt == 2:
        ftp()
        break
    elif opt == 3:
        ftp2()
        break
    else :
        break
    s.close()
```

We used socket library in python version 3.x to write client.py.
We used our own machine for the server-client setup - so we used localhost and a random port number (here 12348) for our system.

We created the **distributed math function** , which first sends the server the input user entered . Then the user enters the mathematical function in the form of a string and then the math function encodes it to byte and sends to server for processing . Server processes the result and the client receives the result and displays back to the user.

We created the **ftp** function to send a text file (or any file , but we worked here with txt to make things simpler).First it sends the option the user selected for the server to execute the respective function . It prompts the user to input the filename . Then it opens the file and read it in bytes . It then sends the server 1024 bytes as recv/send function is only a **TCP** wrapper and there is no python level direct implementation of send till it's done without loop .

We created the **ftp2** function to do the exact opposite and receive the file from the server . It first sends the server the option user selected . Then receives the file server sends 1024 bytes at a time.

Server.py

```
import socket
from _thread import *
import threading

def calc(s):
    if '+' in s :
        a,b =
s.split('+')
        return
float(a)+float(b)
    elif '-' in s:
        a,b =
s.split('-')
        return
float(a)-float(b)
    elif '*' in s:
        a,b =
s.split('*')
        return
float(a)*float(b)
    elif '/' in s:
        a,b =
s.split('/')
        return
float(a)/float(b)
    elif '^' in s:
        a,b =
s.split('^')
        return
float(a)**float(b)
```

```
else :
    return 0
s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(('localhost', 12348))
s.listen(7)
print("Socket is
listening...")
def thread_func(c):
    print('Got connection
from', addr)
    opt =
int(str(c.recv(1024).decode()
))
    print(opt)
    while True :
        if opt == 1:
            msg =
str(c.recv(1024).decode())
            k =
calc(msg)
            c.send(str(k).encode())
        if opt == 2:
            f =
open("client2server.txt", 'wb'
)
            l =
c.recv(1024)
```

```
c.recv(1024)
while (l):
    f.write(l)
    l =
c.recv(1024)
    f.close()
print("Success")
    break
    if opt == 3:
        fname =
input("Enter Filename :")
        f = open (fname,
"rb")
        l = f.read(1024)
        while (l):
            c.send(l)
            l =
f.read(1024)
            f.close()
print("Success....\nListening
Again....")
        break
    else :
        break
    c.close()
while True:
    c, addr = s.accept()
    start_new_thread(thread_func,
(c,))
    s.close()
```

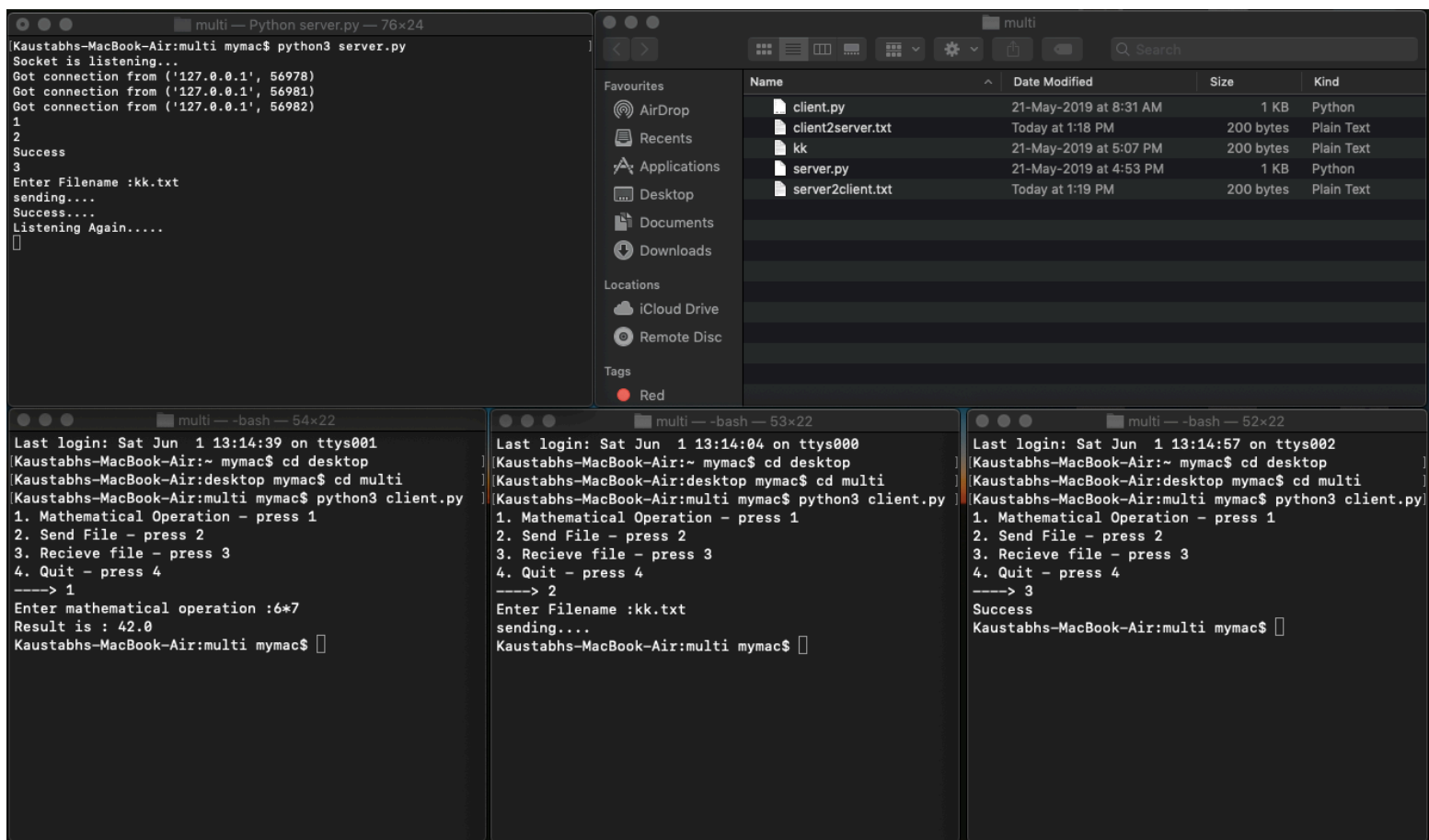
We used socket library in python version 3.x to write client.py.
We used our own machine for the server-client setup - so we used localhost and a random port number (here 12348) for our system.

We used **calc()** function to calculate the mathematical string that is coming from the client.py module and send the result back to the client.

For **ftp** we opened a file named client2server.txt and write it 1024 bytes at a time in a while loop till the content keeps coming .

Now the **s.listen(7)** is not responsible for the concurrency that our problem description demanded . It only works as the maximum number of connections allowed . So for **concurrency** we used Multithreading and for that we imported the **_threads** library . For every connection from a client we are opening a new thread in server machine . That way it can compute tasks in parallel .

Output



```
[Kastabhs-MacBook-Air:multi mymac$ python3 server.py
Socket is listening...
Got connection from ('127.0.0.1', 56978)
Got connection from ('127.0.0.1', 56981)
Got connection from ('127.0.0.1', 56982)
1
2
Success
3
Enter Filename :kk.txt
sending....
Success....
Listening Again....
[]

Last login: Sat Jun 1 13:14:39 on ttys001
[Kastabhs-MacBook-Air:~ mymac$ cd desktop
[Kastabhs-MacBook-Air:desktop mymac$ cd multi
[Kastabhs-MacBook-Air:multi mymac$ python3 client.py
1. Mathematical Operation - press 1
2. Send File - press 2
3. Recieve file - press 3
4. Quit - press 4
----> 1
Enter mathematical operation :6*7
Result is : 42.0
Kastabhs-MacBook-Air:multi mymac$

Last login: Sat Jun 1 13:14:04 on ttys000
[Kastabhs-MacBook-Air:~ mymac$ cd desktop
[Kastabhs-MacBook-Air:desktop mymac$ cd multi
[Kastabhs-MacBook-Air:multi mymac$ python3 client.py
1. Mathematical Operation - press 1
2. Send File - press 2
3. Recieve file - press 3
4. Quit - press 4
----> 2
Enter Filename :kk.txt
sending....
Kastabhs-MacBook-Air:multi mymac$

Last login: Sat Jun 1 13:14:57 on ttys002
[Kastabhs-MacBook-Air:~ mymac$ cd desktop
[Kastabhs-MacBook-Air:desktop mymac$ cd multi
[Kastabhs-MacBook-Air:multi mymac$ python3 client.py
1. Mathematical Operation - press 1
2. Send File - press 2
3. Recieve file - press 3
4. Quit - press 4
----> 3
Success
Kastabhs-MacBook-Air:multi mymac$
```

CONCLUSION

We have shown simultaneously running 3 (out of 7) terminal windows representing 3 different machines , as running this in different machines is difficult .

We have learned a lot about socket programming , client-server system , and doing this project also polished our python skills.