



UNIVERSITY OF DHAKA

CSE:3111 COMPUTER NETWORKING LAB

Lab Report 2

Introduction To Socket Programming and Simple
Exercise on Client-Server Communication

Shariful Islam Rayhan

Roll: 41

Md Sakib Ur Rahman

Roll: 37

Submitted to:

Dr. Md. Abdur Razzaque

Dr. Muhammad Ibrahim

Dr. Md. Redwan Ahmed Rizvee

Dr. Md. Mamun Or Rashid

Submitted on: 2024-02-01

1 Introduction

Socket programming forms the backbone of network communication, facilitating the exchange of data between systems over a network. In this lab we delve into the essential concepts and practical implementation of server-client communication using sockets. By implementing basic server and client communication we were able to see how two host can be connected to send message by the client and accept response by the server.

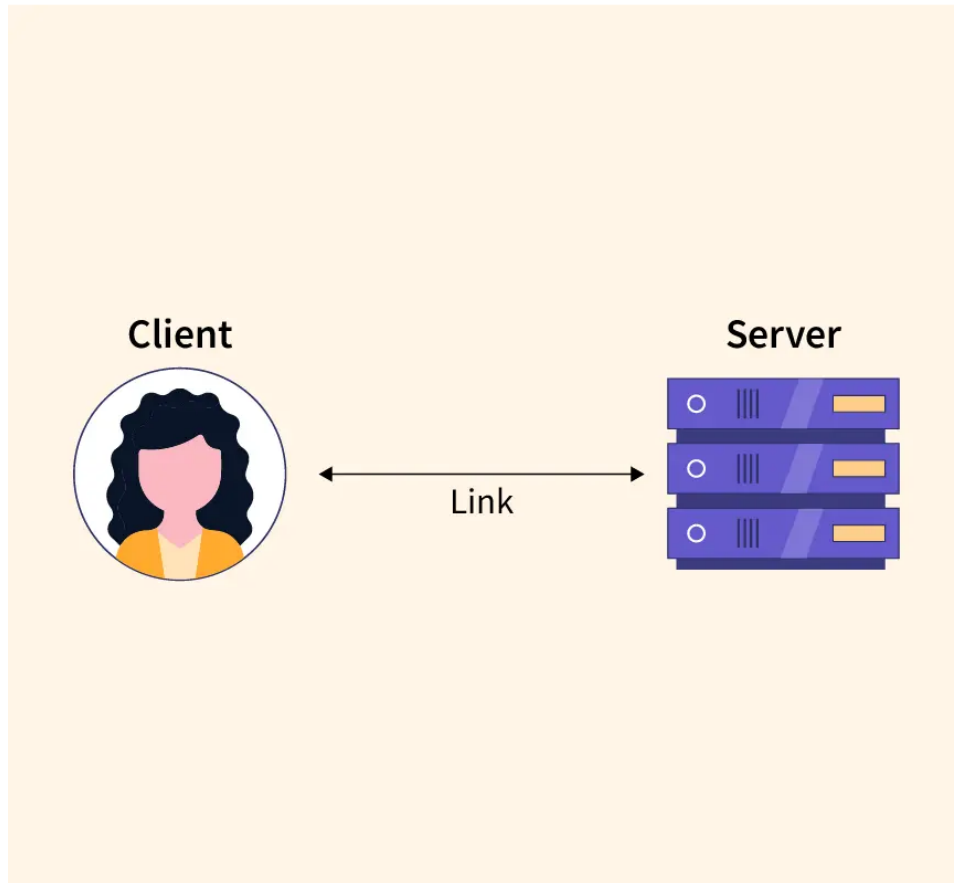


Figure 1: Server and Client is Communicating

1.1 Objectives

- To be familiar with socket programming by python
- To implement server and client by socket programming
- To know how to create connection and communicate between server and client
- Sending request to server and sending response from server

2 Theory

Sockets are used in socket programming to establish a connection between a client and a server program. When we talk about client and server, a server is software that waits for the client's requests and processes them when they arrive. The server can be on the same device or locally connected to some other computers, or could be remote. And the client is the requestor who requests the server for resources, and the server responds to those requests. A socket is a network endpoint for communication between servers and clients. The socket can communicate within the same process, across processes on the same machine, or between processes on other machines.

2.1 Server

To implement a server we followed some steps. Those are mentioned below:

- Import the socket module
- We get the hostname by the `socket.gethostname()` function.(optional)
- Fixed the available port number for the server
- We used `socket.socket()` function to create a socket object.
- We then bind the IP/host and the port number using `bind()` function
- Call `listen()` to specify how many clients the server can listen simultaneously
- Accept the connection from client by `accept()` method
- We used `send()` to send and `recv()` to receive message from the client

2.2 Client

To implement client we followed some steps. Those are mentioned below:

- Import the socket module
- Fixed the available port number for the server
- We used `socket.socket()` function to create a socket object.
- We then bind the IP/host and the port number using `bind()` function
- We then connect to the server by `connect()`

3 Methodology

3.1 Problem A

By following the steps needed to implement Server and Client for Problem A we create the both files. The code snippets are given below

3.1.1 Server:

```
1 import socket
2
3
4 def primechek(num):
5
6     for i in range (2,int(num/2)):
7         if num%i==0:
8             return False
9     return True
10
11
12 def palchek(pal):
13
14     l=len(pal)
15
16     for i in range (0,int (l/2)):
17         if pal[i]!=pal[-(i+1)]:
18             return False
19
20     return True
21
22
23 def connection():
24     hostn=socket.gethostname()
25     host=socket.gethostbyname(hostn)
26     port=4553
27     print(host)
28
29     serv_sock=socket.socket()
30     serv_sock.bind(('',port))
31     serv_sock.listen(2)
32
33     con,addr=serv_sock.accept()
34
35     print("Connected to : "+str(addr))
36
37     return con
38
39
```

```

40 def serv_driver(c):
41     while True:
42
43
44         text=c.recv(1024).decode()
45         print("Client wants to convert uppercase : "+str(text))
46         c.send(text.upper().encode())
47
48
49         #recieving option
50         opt=int(c.recv(1024).decode())
51
52         print(opt)
53         if opt==1:
54
55             numb=int(c.recv(1024).decode())
56             print(f"Checking wheter {numb} is prime")
57             if primechek(numb):
58                 c.send(bytes("Your number is a prime number",'
utf-8'))
59             else:
60                 c.send(bytes("Your number is not prime",'utf-8'
))
61
62         elif opt==2:
63             pal=c.recv(1024).decode()
64             print(f"Checking wheter {pal} is palindrome ")
65             if palchek(pal):
66                 c.send(bytes("Your number is a palindrome
number",'utf-8'))
67
68             else:
69                 c.send(bytes("Your number is not a palindrome
number",'utf-8'))
70         else:
71             c.close()
72             break
73
74
75
76
77
78 c=connection()
79
80 serv_driver(c)

```

3.1.2 Client

```
1 import socket
2 def connection():
3     host=' '
4     port=4553
5
6     client_sock=socket.socket()
7
8     client_sock.connect((host,port))
9
10    return client_sock
11
12
13 def client_driver(c):
14     while True:
15         #sending text
16         text=input("Enter a line of text : ")
17         c.send(text.encode())
18
19         #recieving uppercase
20         msg = c.recv(1024).decode()
21         print(msg)
22
23         #recieving option printing things
24         print("Choice Your Option ")
25         print("1. Prime Check")
26         print("2. Palindrome Check")
27         print("3. No more ")
28
29         opt=input("Enter : ")
30         c.send(opt.encode())
31
32         num=input("Enter your number: ")
33         c.send(num.encode())
34
35         msg=c.recv(1024).decode()
36         print(msg)
37
38         if opt=="3":
39             c.close()
40             break
41
42
43 c=connection()
44
45 client_driver(c)
```

3.2 Problem B

By following the steps needed to implement Server and Client for Problem B we create the both files. The code snippets are given below

```
1 import socket
2 import time
3 import random
4
5 usa='rayhan'
6 pasa='1234'
7 bala=120000
8
9 usb='sakib'
10 pasb='123'
11 balb=50000
12
13 d={}
14 serv_sock=socket.socket()
15 def connection():
16
17     hostn = socket.gethostname()
18     host = socket.gethostbyname(hostn)
19     port = 6990
20
21     print(host)
22
23     serv_sock.bind(('', port))
24     serv_sock.listen(5)
25     c, addr = serv_sock.accept()
26
27     print("Connected to: " + str(addr))
28     return c
29
30
31 def servertransaction(bala):
32     while True:
33         opt = c.recv(1024).decode()
34         print('choice ', opt)
35
36         if int(opt) == 1:
37             print("Holder wants to know balance")
38             c.send(("Your current balance is : " + str(bala)).
39 encode())
40
41         elif int(opt) == 3:
42             am = int(c.recv(1024).decode())
43             bala = bala + am
```



```

43         c.send(("After depositing current balance is: " + str
(bala)).encode())
44
45     elif int(opt)==2:
46         am = int(c.recv(1024).decode())
47         print(am)
48
49         if am <= 0:
50             continue
51
52         id = c.recv(1024).decode()
53
54         if d.get(id) is not None:
55             print('Error')
56             c.send(('555')).encode())
57
58     else:
59         print("Withdrawn amnt: ", am)
60
61         if bala < am:
62             c.send(('401')).encode())
63             continue
64         else:
65             d[id] = {usa, am}
66
67             ra = random.randint(0, 50)
68             print(ra)
69
70             if ra > 50:
71                 c.send(('401')).encode())
72                 continue
73             bala = bala - am
74             c.send(("Withdraw successful!After withdraw
your balance is: " + str(bala)).encode())
75             print(d)
76             any=c.recv(1024).decode()
77
78             if any=='1':
79                 continue
80             else:
81                 break
82
83
84
85
86
87 def serverdriver(c,bala):
88     while True:
89         user = c.recv(1024).decode()

```

```

90         if not user:
91             break
92
93         if user == usa:
94             pas = c.recv(1024).decode()
95             if pas == pasa:
96                 c.send('40'.encode())
97
98                 servertransiction(bala)
99
100            else:
101                print("Invalid Password")
102                c.send(('404').encode())
103                c.close()
104        else:
105            print('Invalid User')
106            c.send(('404').encode())
107            c.close()
108
109
110
111 c=connection()
112
113 serverdriver(c,bala)

```

3.2.1 Client

```
1 import socket
2 import time
3
4 cl_sock = socket.socket()
5 id = 0
6
7 def connection():
8     host = ''
9     port = 6990
10    cl_sock.connect((host, port))
11    user = input("Enter Your Username: ")
12    cl_sock.send(user.encode())
13
14    pasw = input("Enter Your Password: ")
15    cl_sock.send(pasw.encode())
16
17    cc = cl_sock.recv(1024).decode()
18
19    if cc == '404':
20        print("Server Error: ")
21        cl_sock.close()
22
23
24 def clientdriver(id):
25     while True:
26         print('Please Select')
27         print('1. Check Balance')
28         print('2. Cash Withdraw')
29         print('3. Cash Deposit')
30
31         opt = input("Enter : ")
32         cl_sock.send(opt.encode())
33
34         if int(opt) == 1:
35             msg = cl_sock.recv(1024).decode()
36             print(msg)
37
38         elif int(opt) == 3:
39             am = input("Enter the amount to deposit: ")
40             cl_sock.send(am.encode())
41             msg = cl_sock.recv(1024).decode()
42             print(msg)
43
44         elif int(opt) == 2:
45             withdrawhelper(id)
46
47
```

```

48     print('Anything else') # after executing one request
49     print('1. YES')
50     print('2. NO')
51     a = input('ENTER : ') # enter the desired option
52     cl_sock.send(a.encode())
53     if a == '2':
54         break
55
56 def widra_req(id, wit, client_socket):
57     client_socket.send(wit.encode())
58     time.sleep(1)
59     id=str(id)
60     client_socket.send(id.encode())
61
62
63
64 def withdrawhelper(id):
65     id = id + 1
66     wam = input("Enter the amount of withdrawal: ")
67
68     if int(wam) <= 0:
69         print("Enter Correct Amount to withdraw")
70         msg = cl_sock.recv(1024).decode()
71         print(msg)
72
73     else:
74         while True:
75             widra_req(id, wam, cl_sock)
76
77             r = cl_sock.recv(1024).decode()
78             print(r)
79
80             if r == '401':
81                 print("Insufficient Balance")
82             elif r == '555':
83                 print("Failed Attempt")
84                 print("Try Again!")
85                 print('1. Yes')
86                 print('2. No')
87                 trya = input('Enter : ')
88
89                 if trya == '1':
90                     cl_sock.send(('2').encode())
91                     time.sleep(0.5)
92                     continue
93                 else:
94                     break
95             else:
96                 print(r)

```

```
97         break
98
99     connection()
100
101     clientdriver(id)
102
103     cl_sock.close()
```

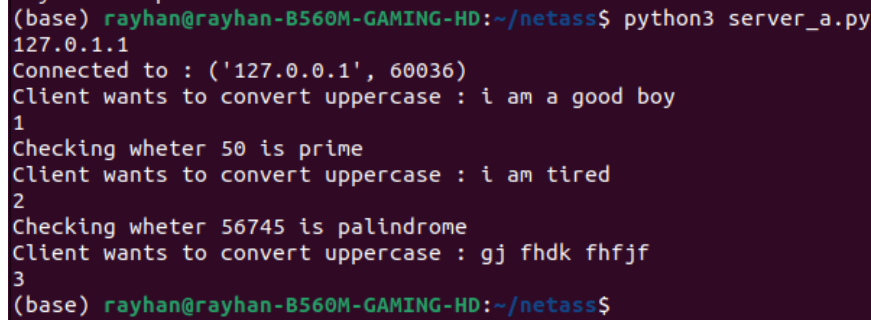
4 Experimental result

Some Snapshots of the Client and Server Side queries can be seen in the following figures for both Problem A and B:

4.1 Problem A

4.1.1 Server

Server takes a string and an integer from the client. It converts the string to an upper case string and send it back to the client. On the other hand, it check the integer if its a prime or not and whether it is palindrome or not and send the result to the client.

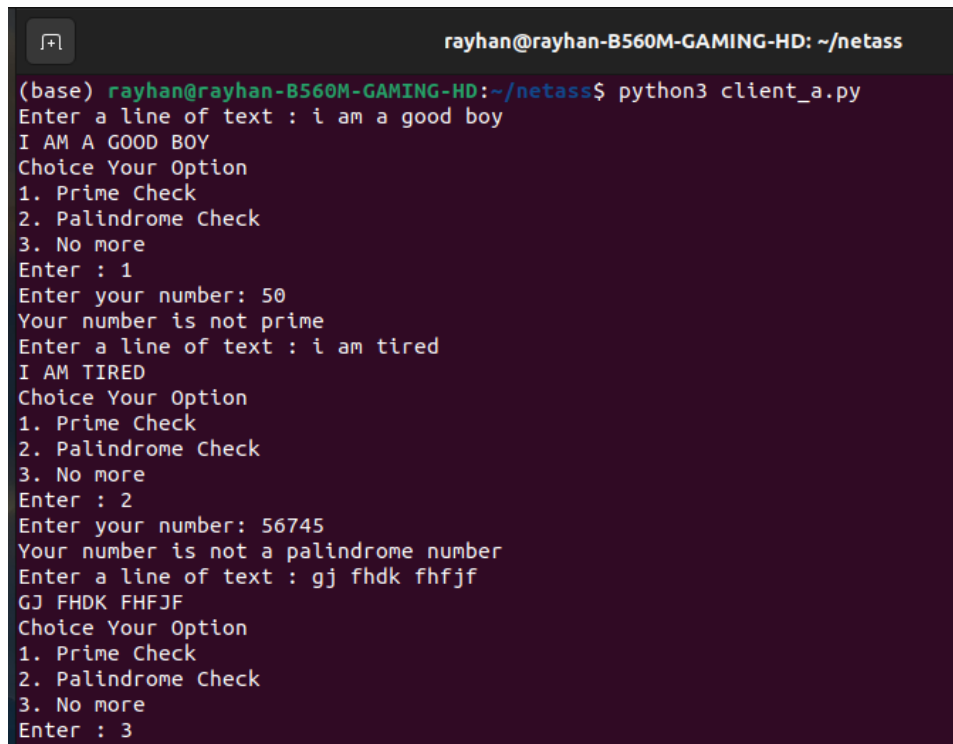
A terminal window with a dark purple background. The text shows the execution of a Python script named 'server_a.py'. It displays three client connections from 127.0.0.1. Each connection includes a string to be converted to uppercase and an integer to be checked for primality or palindromicity. The server's responses are shown for each case.

```
(base) rayhan@rayhan-B560M-GAMING-HD:~/netass$ python3 server_a.py
127.0.0.1
Connected to : ('127.0.0.1', 60036)
Client wants to convert uppercase : i am a good boy
1
Checking wheter 50 is prime
Client wants to convert uppercase : i am tired
2
Checking wheter 56745 is palindrome
Client wants to convert uppercase : gj fhdk fhfjf
3
(base) rayhan@rayhan-B560M-GAMING-HD:~/netass$
```

Figure 2: Content of Server for Problem A

4.1.2 Client

The client takes the input from the user and send it to the server. Then it takes the message form the server and show it in the console. Then client choice an option whether he wants to check prime or palindrome a given number from the user side. Then takes the result from the server. He choice 3 to stop. Some Snapshots of the Client Side queries can be seen in the following figures:



```
rayhan@rayhan-B560M-GAMING-HD: ~/netass
(base) rayhan@rayhan-B560M-GAMING-HD:~/netass$ python3 client_a.py
Enter a line of text : i am a good boy
I AM A GOOD BOY
Choice Your Option
1. Prime Check
2. Palindrome Check
3. No more
Enter : 1
Enter your number: 50
Your number is not prime
Enter a line of text : i am tired
I AM TIRED
Choice Your Option
1. Prime Check
2. Palindrome Check
3. No more
Enter : 2
Enter your number: 56745
Your number is not a palindrome number
Enter a line of text : gj fhdk fhjff
GJ FHDK FHFJF
Choice Your Option
1. Prime Check
2. Palindrome Check
3. No more
Enter : 3
```

Figure 3: Client Side View after successful request

4.2 Problem B

4.2.1 Balance Check

```
csedu@csedu-H97M-D3H:~/Downloads$ python3 client_b.py
Enter Your Username: rayhan
Enter Your Password: 1234
Please Select
1. Check Balance
2. Cash Withdraw
3. Cash Deposit
Enter : 1
Your current balance is : 120000
Anything else
1. YES
2. NO
ENTER : 2
```

Figure 4: Client Side View after Balance request

```
csedu@csedu-H97M-D3H:~/Downloads$ python3 server_b.py
127.0.1.1
Connected to: ('127.0.0.1', 35436)
choice 1
Holder wants to know balance
choice close
Traceback (most recent call last):
```

Figure 5: Server Side View after Balance Check request

4.2.2 Deposit

```
csedu@csedu-H97M-D3H:~/Downloads$ python3 server_b.py
127.0.1.1
Connected to: ('127.0.0.1', 60380)
choice 3
choice close
Traceback (most recent call last):
```

Figure 6: Client Side View after deposit request


```

csedu@csedu-H97M-D3H:~/Downloads$ python3 client_b.py
Enter Your Username: rayhan
Enter Your Password: 1234
Please Select
1. Check Balance
2. Cash Withdraw
3. Cash Deposit
Enter : 3
Enter the amount to deposit: 500
After depositing current balance is: 120500
Anything else
1. YES
2. NO
ENTER : 2

```

Figure 7: Server Side View after Deposit request

4.2.3 Withdraw

```

csedu@csedu-H97M-D3H:~/Downloads$ python3 client_b.py
Enter Your Username: rayhan
Enter Your Password: 1234
Please Select
1. Check Balance
2. Cash Withdraw
3. Cash Deposit
Enter : 2
Enter the amount of withdrawal: 400
Withdraw successful!After withdraw your balance is: 119600
Withdraw successful!After withdraw your balance is: 119600
Anything else
1. YES
2. NO
ENTER : 2

```

Figure 8: Client Side View after Withdraw request

```

csedu@csedu-H97M-D3H:~/Downloads$ python3 server_b.py
127.0.1.1
Connected to: ('127.0.0.1', 56882)
choice 2
400
Withdrawn amnt: 400
9
{'1': {400, 'rayhan'}}
choice close
Traceback (most recent call last):

```

Figure 9: Server Side View after Withdraw request

4.3 Grphical Analysis of Error

Time(ms) vs. Error(%)

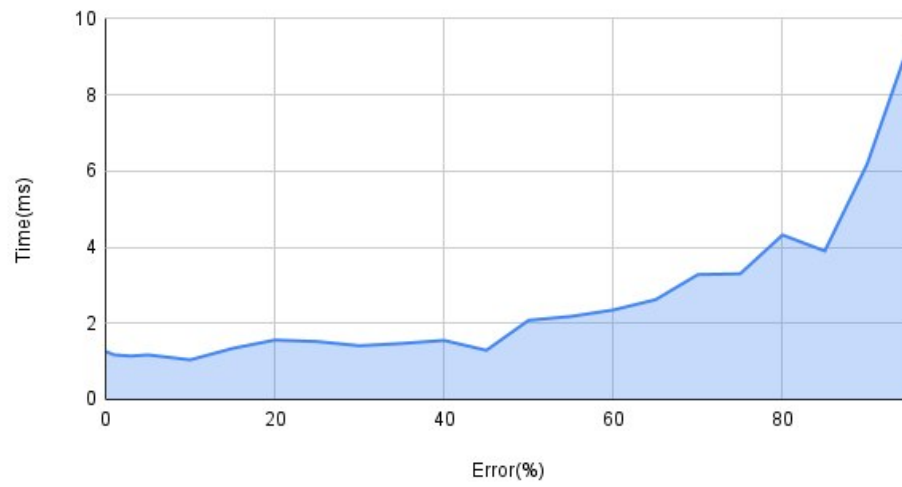


Figure 10: Server Error for Time vs Error

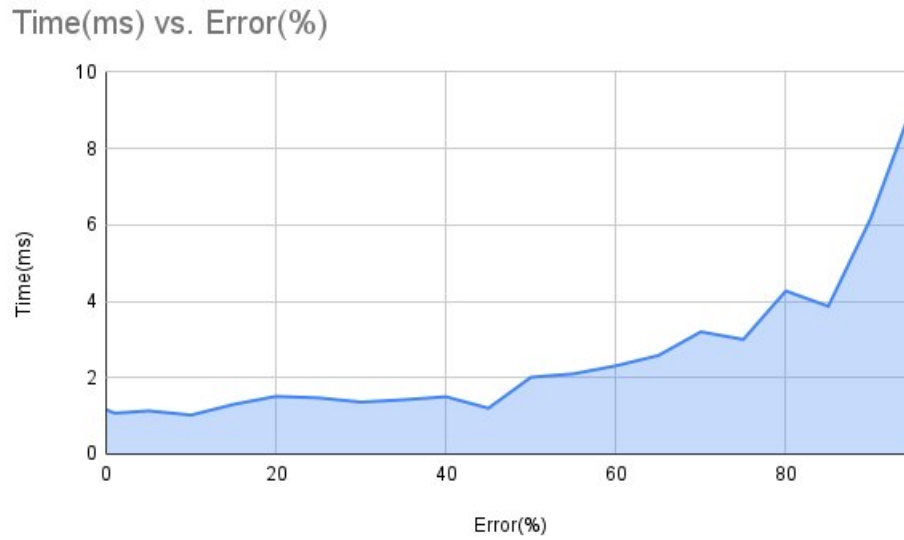


Figure 11: Client Error for Time vs Error

5 Experience

1. We have seen how server and client communicate with each other
2. Experienced real time communication
3. It was nice to implement server and client and make sense

References

- [1] Socket Programming : <https://www.scaler.com/topics/socket-programming-in-python/>
- [2] Tutorial : <https://www.youtube.com/watch?v=3QiPPX-KeSc>