20CYS402 – Distributed Systems and Cloud Computing

Lab Exercise – 1

Name: R Subramanian

Roll Number: CH.EN.U4CYS22043

Date: 18/06/2025

🔷 Program 1: Client–Server Communication using Different IPs

1. Objective

To implement a basic client–server communication model that allows message exchange using different IP addresses via socket programming.

2. Program Code

Server Code (Python):

```
import socket

server_socket = socket.socket()
server_socket.bind(('localhost', 12345))
server_socket.listen(1)
print("Server is listening...")

conn, addr = server_socket.accept()
print(f"Connected to {addr}")

while True:
    data = conn.recv(1024).decode()
    if not data or data.lower() == 'exit':
        break
    print("Client:", data)
    reply = input("Server: ")
    conn.send(reply.encode())

conn.close()
```

Client Code (Python):

```
import socket

client_socket = socket.socket()
client_socket.connect(('localhost', 12345))
```

```
while True:
    message = input("Client: ")
    client_socket.send(message.encode())
    if message.lower() == 'exit':
        break
    reply = client_socket.recv(1024).decode()
    print("Server:", reply)

client_socket.close()
```

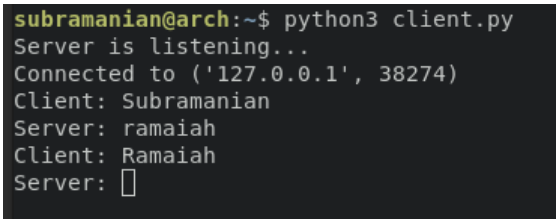
3. Explanation

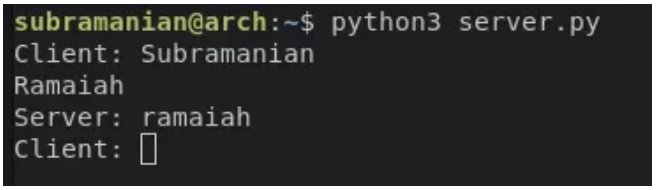
- The server listens on a specified IP and port for incoming connections.
- The client connects to the server and exchanges messages in a loop.
- The communication terminates when either side types **“exit”**.

4. Sample Input/Output

```
Client: Subramanian
Server: Ramaiah
```

5. Output Screenshots





6. Conclusion

This program demonstrated two-way message transmission using socket programming across different IPs, establishing a foundational understanding of client–server communication.

🔷 Program 2: Peer-to-Peer File Listing from Local Folder

1. Objective

To establish a peer-to-peer communication system where one peer can list files from another peer’s local directory.

2. Program Code

Server Peer Code (Python):

```
import socket
import os

server_socket = socket.socket()
server_socket.bind(('localhost', 5555))
server_socket.listen(1)
print("Peer listening for requests...")

conn, addr = server_socket.accept()
print(f"Connected to {addr}")
files = os.listdir('.')
file_list = "\n".join(files)
conn.send(file_list.encode())
conn.close()
```

Client Peer Code (Python):

```
import socket

client_socket = socket.socket()
client_socket.connect(('localhost', 5555))

data = client_socket.recv(4096).decode()
print("Files available on server peer:\n", data)

client_socket.close()
```

---

3. Explanation

- The server peer uses `os.listdir()` to retrieve files in its directory.
- The client peer connects to the server and receives the file list as plain text.

---

4. Sample Input/Output

```
Files available on server peer:
main.py
data.csv
report.pdf
```

---

5. Output Screenshots

6. Conclusion

This program demonstrated peer-to-peer communication for file sharing — a decentralized approach useful for distributed and collaborative systems.

🔷 Program 3: RPC-Based Age Calculator

1. Objective

To implement a distributed system using Remote Procedure Call (RPC), where the client sends a date of birth and the server returns the calculated age.

2. Program Code

Server Code (Python):

```
from xmlrpc.server import SimpleXMLRPCServer
from datetime import datetime

def calculate_age(dob):
    birth_date = datetime.strptime(dob, "%Y-%m-%d")
    today = datetime.today()
    return today.year - birth_date.year - ((today.month, today.day) < (birth_date.month, birth_date.day))

server = SimpleXMLRPCServer(("localhost", 8000))
server.register_function(calculate_age, "calculate_age")
print("RPC Server is running...")
server.serve_forever()
```

Client Code (Python):

```
import xmlrpc.client

proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")
dob = input("Enter your date of birth (YYYY-MM-DD): ")
age = proxy.calculate_age(dob)
print("Your age is:", age)
```

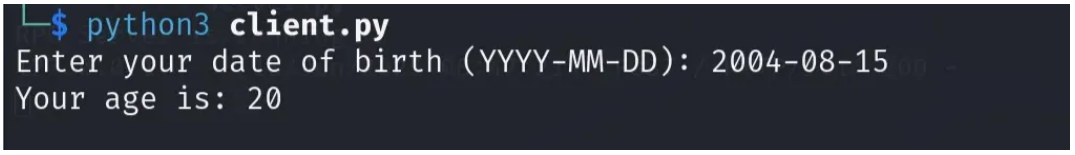
3. Explanation

- The **RPC server** hosts a remote function to calculate age from a date of birth.
- The **client** connects to the server and invokes this function remotely.
- The server computes the result and sends it back to the client.

4. Sample Input/Output

```
Enter your date of birth (YYYY-MM-DD): 2004-08-15
Your age is: 20
```

5. Output Screenshots



6. Conclusion

This exercise successfully simulated a distributed application using RPC, reinforcing the concept of **service-based architecture** and remote function invocation in distributed systems.