Terna Engineering College
Computer Engineering Department
Program: Sem VIII

Course: Distributed Computing Lab (CSL802)

Faculty: Rohini Patil

Experiment No. 10

A.1 Aim: To Implement Name Resolution.

PART B
(PART B: TO BE COMPLETED BY STUDENTS)

Roll No. 50	**Name:** AMEY MAHENDRA THAKUR
Class: BE COMPS B 50	**Batch:** B3
Date of Experiment: 31-03-2022	**Date of Submission:** 31-03-2022
Grade:	

B.1 Software Code written by a student:

- **RpcServer.py**

```
import time, socket, sys
import os.path
import os

print("\nWelcome to Chat Room\n")
print("Initialising....\n")
time.sleep(1)

s = socket.socket()
host = socket.gethostname()
ip = socket.gethostbyname(host)
port = 1234
s.bind((host, port))
print(host, "(", ip, ")\n")
name = input(str("Enter your name: "))
```

```
s.listen(1)
print("\nWaiting for incoming connections...\n")
conn, addr = s.accept()
print("Received connection from ", addr[0], "(", addr[1], ")\n")

s_name = conn.recv(1024)
s_name = s_name.decode()
print(s_name, "has connected to the chat room\nEnter [e] to exit chat room\n")
conn.send(name.encode())

# commands=["request-file-status","request-file-dir","request-file-data"]

comands = '''COMMANDS:
          request-file-status:
          request-file-dir:
          request-file-data:
        '''
conn.send(comands.encode())
while True:
  message = conn.recv(1024)
  message = message.decode()
  if message == "[e]":
    message = "Left chat room!"
    conn.send(message.encode())
    print("\n")
    break
  # conn.send(message.encode())

  buffer = message.split(':')
  cmnd = buffer[0]
  filename = buffer[1]
  # print(filename)

  if cmnd == "request-file-status":
    print("Command accepted")
    path="data/"
    filepath=path+filename
    status = os.path.exists(filepath)
    print(status)
    if status:
      response = filename + "  exists"
    else:
      response = filename + "  does not exist"
    conn.send(response.encode())

elif cmnd == "request-file-dir":
    print("Command accepted")
    cwd = os.getcwd()
    path = "/data/"
    filepath=cwd+path+filename
    status = os.path.exists(filepath)
    print(status)
    if status:
        response = filepath
    else:
        response = filename + "  does not exist"

    conn.send(response.encode())

elif cmnd == "request-file-data":
    print("Command accepted")
    path = "data/"
    filepath=path+filename
    status = os.path.exists(filepath)
    print(status)
    if status:
        file = open(filepath, "r")
        file_data = file.read()
        response = "sending-file:"+filename
        conn.send(response.encode())
        message = conn.recv(1024)
        message = message.decode()
        if message == "ready":
            response = file_data
            conn.send(response.encode())
        else:
            pass
    else:
        response = filename + "  does not exist"
        conn.send(response.encode())

# print(s_name, ":", message)
```

- **RpcClient.py**

```
import time, socket, sys
import re

print("\nWelcome to Chat Room\n")
print("Initialising....\n")
time.sleep(1)

s = socket.socket()
shost = socket.gethostname()
ip = socket.gethostbyname(shost)
print(shost, "(", ip, ")\n")
host = input(str("Enter server address: "))
name = input(str("\nEnter your name: "))
port = 1234
print("\nTrying to connect to ", host, "(", port, ")\n")
time.sleep(1)
s.connect((host, port))
print("Connected...\n")

s.send(name.encode())
s_name = s.recv(1024)
s_name = s_name.decode()
print(s_name, "has joined the chat room\nEnter [e] to exit chat room\n")

while True:
   message = s.recv(1024)
   message = message.decode()
   print(s_name, ":", message)
   message = input(str("Me : "))
   if message == "[e]":
      message = "Left chat room!"
      s.send(message.encode())
      print("\n")
```

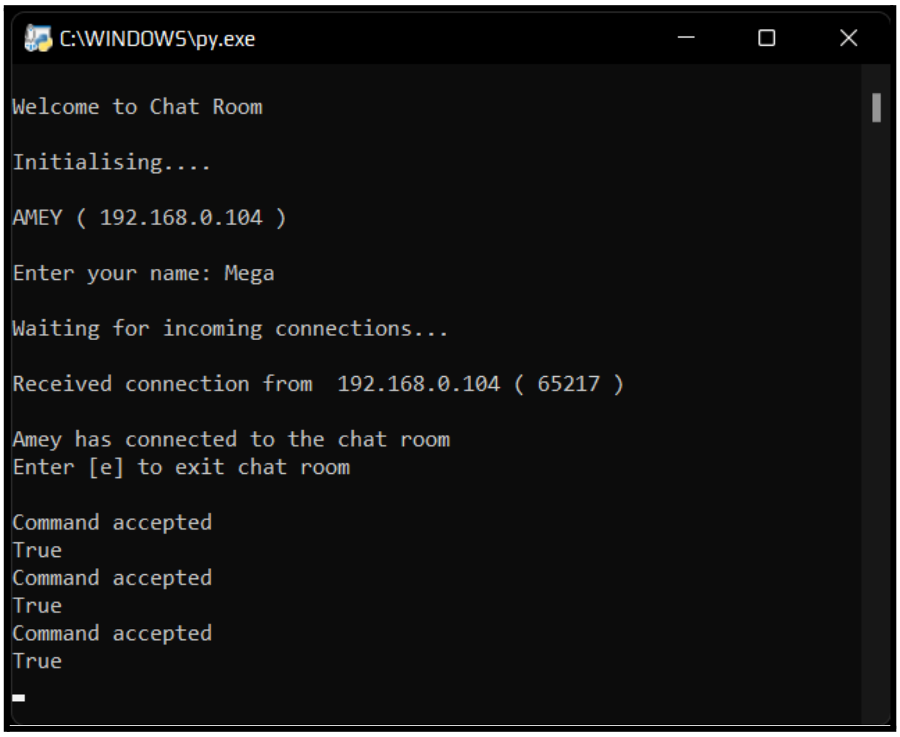
```
        break
    elif re.search("^sending-file",message):
        buffer = message.split(':')
        filename = buffer[1]
        print(filename)
        rqst_data = "ready"
        s.send(rqst_data.encode())
        file_data = s.recv(1024)
        file_data = file_data.decode()
        file = open(filename, "w")
        file.write(file_data)

    s.send(message.encode())
```

- **Test.txt**

This is test file.

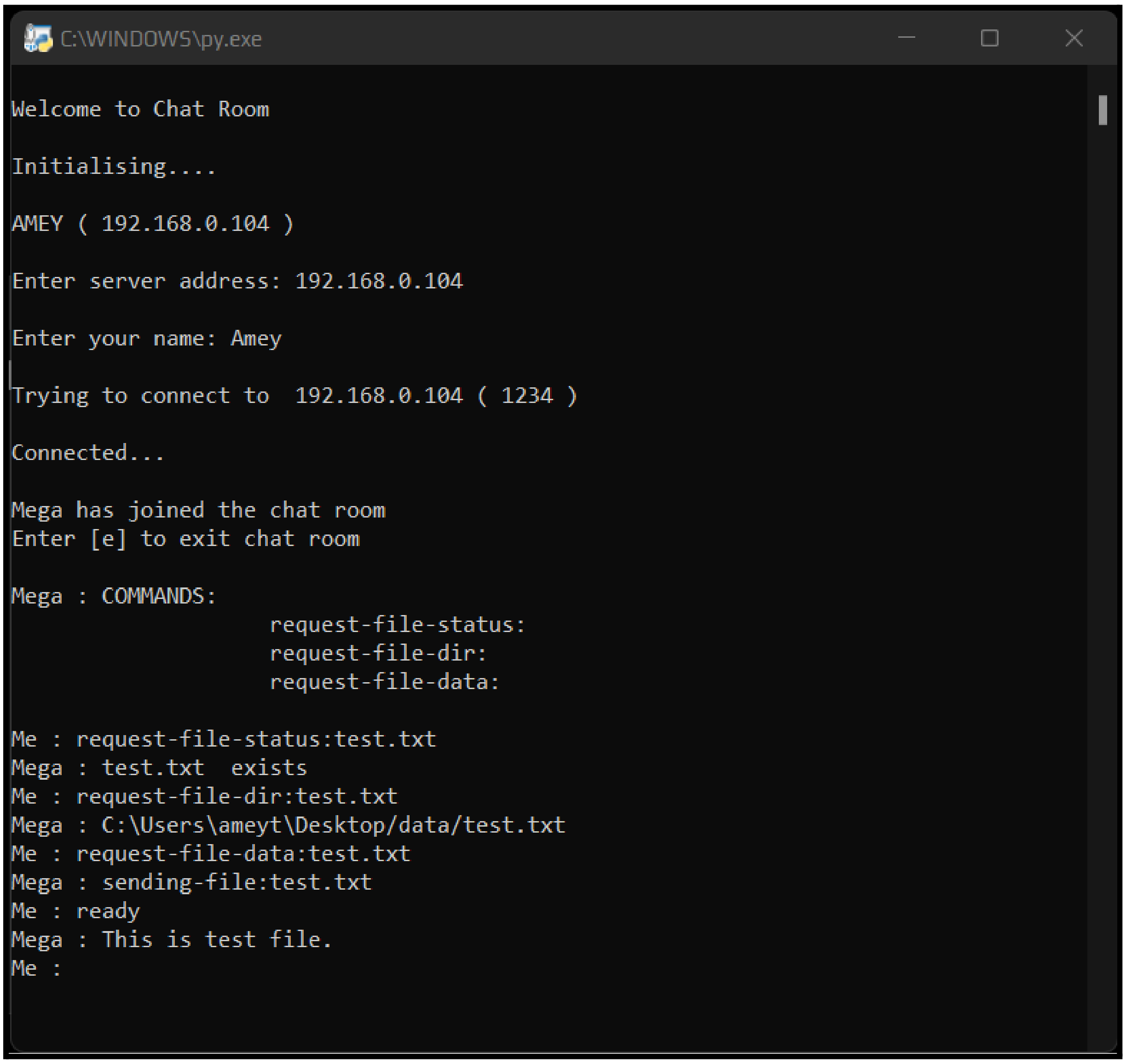
B.2 Input and Output:

- RPCServer.py:



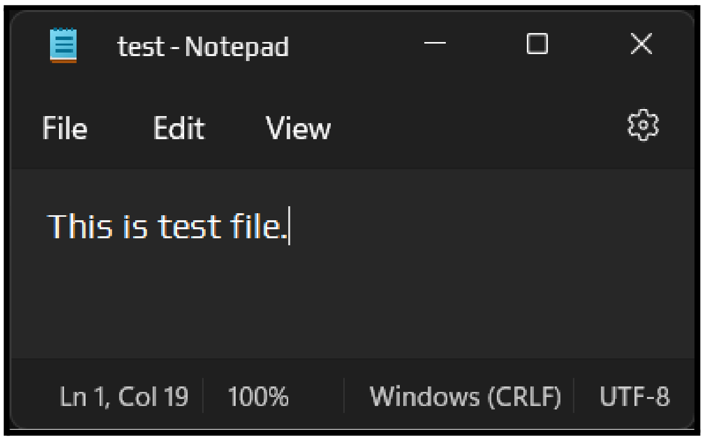
```
C:\WINDOWS\py.exe

Welcome to Chat Room

Initialising....

AMEY ( 192.168.0.104 )

Enter your name: Mega

Waiting for incoming connections...

Received connection from  192.168.0.104 ( 65217 )

Amey has connected to the chat room
Enter [e] to exit chat room

Command accepted
True
Command accepted
True
Command accepted
True
```

- RPCClient.py:



```
Welcome to Chat Room

Initialising....

AMEY ( 192.168.0.104 )

Enter server address: 192.168.0.104

Enter your name: Amey

Trying to connect to  192.168.0.104 ( 1234 )

Connected...

Mega has joined the chat room
Enter [e] to exit chat room

Mega : COMMANDS:
                    request-file-status:
                    request-file-dir:
                    request-file-data:

Me : request-file-status:test.txt
Mega : test.txt  exists
Me : request-file-dir:test.txt
Mega : C:\Users\ameyt\Desktop/data/test.txt
Me : request-file-data:test.txt
Mega : sending-file:test.txt
Me : ready
Mega : This is test file.
Me :
```

- Test.txt



```
This is test file.
```

B.3 Observations and learning:

- Name Resolution is the process of looking up a name. It uses a Closure mechanism: knowing where and how to start a name.
- Name Resolution can be Iterative vs recursive.
- Recursive name resolution puts a higher performance demand on each name server.
  - Too high for global layer name servers.
  - Advantages of recursive name resolution.
  - Caching is more effective.
  - Communication costs may be reduced.

B.4 Conclusion:

We implemented a name resolution protocol using Python and discussed mechanisms used for naming and locating objects in distributed systems.

B.5 Question of Curiosity:

Q1: Mapping of file is managed by

A. file metadata
B. page table
C. virtual memory
D. file system

ANS: A. file metadata

Q2: Explain why DNS is not very suitable for providing the naming of mobile entities.
ANS:

- DNS is a distributed client/server networked database used by TCP/IP applications to map between hostnames and IP addresses (and vice versa), to provide capabilities like electronic mail routing information and service naming. The DNS provides the protocol that allows clients and servers to communicate with each other and also a protocol for allowing servers to exchange information.
- The DNS namespace is hierarchically organized as a rooted tree. A label is a case-insensitive string made up of alphanumeric characters. A label has a maximum length of 63 characters; the length of a complete pathname is restricted to 255 characters. The string representation of a pathname consists of listing its labels, starting with the rightmost one, and separating the labels by a dot ("."). The root is represented by a dot. So, for example, the pathname root:<nl, vu, cs, flits>, is represented by the string flits.cs.vu.nl., which includes the rightmost dot to indicate the root node. We generally omit this dot for readability.
- Because each node in the DNS namespace has exactly one incoming edge (with the exception of the root node, which has no incoming edges), the

labels attached to a node's incoming edge is also used as the name for that node. A subtree is called a domain; a pathname to its root node is called a domain name. Note that, just like a pathname, a domain name can be either absolute or relative.

- The contents of a node are formed by a collection of resource records. There are different types of resource records. The most important types of resource records form the contents of nodes in the DNS namespace.

Type of record	**Associated entity**	**Description**
SOA	Zone	Holds information on the represented zone.
A	Host	Contains an IP address of the host this node represents.
MX	Domain	Refers to a mail server to handles mail addressed to this node.
SRV	Domain	Refers to a server handling a specific service.
NS	Zone	Refers to a name server that implements the represented zone.
CNAME	Node	Symbolic link with the primary name of the represented node.
PTR	Host	Contains the canonical name of a host.
HINFO	Host	Holds information on the host this node represents.
TXT	Any kind	Contains any entity-specific information considered useful.

- A node in the DNS namespace often will represent several entities at the same time. For example, a domain name such as vu.nl is used to represent a domain and a zone. In this case, the domain is implemented by means of several (non-overlapping) zones.
- An SOA (start of authority) resource record contains information such as the e-mail address of the system administrator responsible for the represented zone, the name of the host where data on the zone can be fetched, and so on.
- An A (address) record represents a particular host on the Internet. The A record contains an IP address for that host to allow communication. If a host has several IP addresses, as is the case with multi-homed machines, the node will contain an A record for each address.
- Another type of record is the MX (mail exchange) record, which is like a symbolic link to a node representing a mail server. For example, the node representing the domain cs.vu.nl has an MX record containing the name

zephyr.cs.vu.nl, which refers to a mail server. That server will handle all incoming mail addressed to users in the cs.vu.nl domain. There may be several MX records stored in a node.

- Related to MX records are SRV records, which contain the name of a server for a specific service. SRV records are defined in Gulbrandsen (2000). The service itself is identified by means of a name along with the name of a protocol. For example, the Web server in the cs.vu.nl domain could be named by means of an SRV record such as _http._tcp.cs.vu.nl. This record would then refer to the actual name of the server (which is soling.cs.vu.nl). An important advantage of SRV records is that clients no longer need to know the DNS name of the host providing a specific service. Instead, only service names need to be standardized, after which the providing host can be looked up.
- Nodes that represent a zone, contain one or more NS (name server) records. Like MX records, an NS record contains the name of a name server that implements the zone represented by the node. In principle, each node in the namespace can store an NS record referring to the name server that implements it. However, as we discuss below, the implementation of the DNS namespace is such that only nodes representing zones need to store NS records.
- DNS distinguishes aliases from what are called canonical names. Each host is assumed to have a canonical, or primary name. An alias is implemented by means of a node storing a CNAME record containing the canonical name of a host. The name of the node storing such a record is thus the same as a symbolic link.
- DNS maintains an inverse mapping of IP addresses to hostnames by means of PTR (pointer) records. To accommodate the lookups of hostnames when given only an IP address, DNS maintains a domain named in-addr.arpa, which contains nodes that represent Internet hosts and which are named by the IP address of the represented host. For example, the host www.cs.vu.nl has an IP address 130.37.20.20. DNS creates a node named 20.20.37.130.in-addr.arpa, which is used to store the canonical name of that host (which happens to be soling.cs.vu.nl) in a PTR record.
- The last two record types are HINFO records and TXT records. A HINFO (host info) record is used to store additional information on a host such as its machine type and operating system. In a similar fashion, TXT records are used for any other kind of data that a user finds useful to store about the entity represented by the node.