



MEENAKSHI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to Anna University, Chennai
ISO 9001:2015 Certified and Accredited by NAAC with 'A' Grade



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LABORATORY MANUAL

Course Code & Name	:	CS3591-Computer Networks
Programme	:	B. E-CSE
Regulation	:	2021
Academic year	:	2025-26

Course Outcomes

At the end of the course, students

1

Explain the basic layers and their functions in computer networks.

2

Understand the basics of how data flows from one node to another.

3

Analyze routing algorithms

4

Describe protocols for various functions in the network.

5

Analyze the working of various application-layer protocols.

Prepared by	Verified by	Submitted to
Rejitha D/AP-CSE	Dr. CH Sarada Devi Head of the Department CSE	Internal Quality Assurance Cell (IQAC)

Instruction for the Students: Laboratory Assessment

- (a) The maximum marks for Internal Assessment shall be 60 marks in case of practical courses. Every practical exercise/experiment shall be evaluated based on the conduct of the experiment / exercise and the records to be maintained. There shall be at least one test. The criteria for arriving at the Internal Assessment marks of 60 are as follows: 75 marks shall be awarded for successful completion of all the prescribed experiments done in the Laboratory, and 25 marks for the test. The total mark shall be converted into a maximum of 60 marks and rounded to the nearest integer.

Internal Assessment (100 Marks)*	
Evaluation of Laboratory Observation, Record	Test
75	25

* Internal assessment marks shall be converted into 60 marks

- (b) Each experiment must be evaluated for 100 marks with the following evaluation heads.

Pre-lab assessment	(A)	10
Pre-lab work	(B)	20
Conduct of Experiment	(C)	20
Data observation	(D)	20
Analysis and Interpretation	(E)	20
Post-lab assessment/Viva Voce	(F)	10
Total (=A+B+C+D+E+F)		100

- (c) The experiments prescribed by Anna University shall be divided equally into two cycles. Upon completion of each cycle, a model examination (for a total of 100 marks) will be conducted.
- (d) The average marks obtained in Model Examinations I and II (each out of 100 marks) shall be computed and converted to an equivalent score out of 25 marks, which will be considered as the internal test mark.
- (e) The evaluation of the Laboratory Observation and Record will be carried out for a total of 100 marks. The marks obtained will then be converted to an equivalent score out of 75 marks.
- (f) The internal mark (out of 100) will be calculated as the sum of the equivalent marks obtained in Step 4 (Test mark out of 25) and Step 5 (Observation and Record mark out of 75). This total will then be converted to an equivalent of 60 marks, which will be considered as the final internal mark for the laboratory course.

Note:

- (i) **Pre-Lab assessment (10 Marks):** For each laboratory experiment, a set of basic questions are prepared, and it may consist of either five two-mark descriptive questions with adequate space provided for students to write their answers, or ten objective-type questions, based on the faculty's discretion.

- (ii) **Viva (10 Marks):** A minimum of five viva questions are prepared and printed in the laboratory observation notebook, positioned before the 'Result/Conclusion' section of each experiment.

Instructions to Students: Laboratory Practice

1. General Conduct

- Students must enter the laboratory on time and maintain discipline throughout the session.
- Lab coat/apron, identity card, and shoes must be worn as per departmental regulations.
- Students should maintain silence and avoid unnecessary movement or conversations in the lab.

2. Preparation

- Pre-lab preparation is mandatory. Students must read the theory and procedure of the experiment in advance.
- Bring the laboratory manual, record notebook, and observation book to every session.
- Students should come with a clear idea of the experiment and discuss doubts with the faculty beforehand.

3. During the Experiment

- Students must work only at their assigned workstation or system.
- No student should operate any equipment without the presence or permission of the instructor/lab assistant.
- Handle all instruments and components with care. Any damage due to negligence will be charged.
- Avoid bringing bags, food, water bottles, or mobile phones near the experimental setup.
- Record readings neatly and accurately in the observation book during the experiment.

4. Safety Instructions

- Follow all safety protocols specific to the laboratory (electrical safety, chemical handling, machine safety, etc.).
- Report immediately to the faculty/laboratory assistant in case of equipment malfunction or accidents.
- Do not try to repair or modify any connection or device without guidance.

5. Submission & Assessment


- Observation notebooks must be signed by the faculty at the end of each laboratory session.
- Laboratory records should be completed and submitted within the stipulated time (usually by the next lab class).
- Internal assessments will include the evaluation of observation & record, and marks scored through the Model Examinations I and II.

6. Attendance & Conduct

- Minimum 75% attendance is mandatory for eligibility to attend the end semester laboratory examination.

- Misconduct or malpractice during laboratory sessions/end semester examinations will lead to disciplinary action as per the College/Anna University norms.

DO'S AND DON'TS IN THE LABORATORY

 DO's	✗ DON'Ts
Follow the Lab Rules	Do not Change Settings Without Permission
Log In Properly	Do not Install Unauthorized Software
Save Your Work Regularly	Do not eat or Drink Near Computers
Report Problems Immediately	Do not damage or Remove Hardware
Maintain Cleanliness	Do not Access Inappropriate Content
Practice Good Posture	Do not Use Flash Drives Without Scanning
Shut Down Properly	Do not Leave Without Logging Out



MEENAKSHI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to Anna University, Chennai
ISO 9001:2015 Certified and Accredited by NAAC with 'A' Grade



INSTITUTION

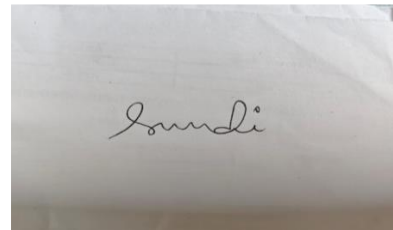
VISION

To make students envision and excel at the global market with moral ethics and good academic excellence.

MISSION

To Endeavour, nurture excellence in learning, fostering research and development towards global competence.

- To imbibe, install integrity and esteem to blossom with latent potential.
- To inculcate the habit of lifelong learning.





MEENAKSHI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to Anna University, Chennai
ISO 9001:2015 Certified and Accredited by NAAC with 'A' Grade



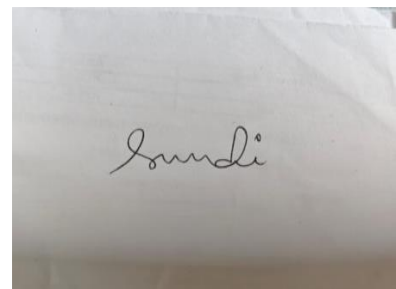
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION STATEMENT

To become a Centre of excellence in Computer Science with strong research and adaptable teaching environment.

MISSION STATEMENT

- Provide quality education and generate new knowledge by engaging in research.
- Promote teaching and learning processes resulting in the integration of research results and innovations by applying new technologies that lead to useful products.
- Provide ethical and value-based global education by promoting activities addressing societal needs.
- Provide placement opportunities and motivate higher studies and entrepreneurship.





MEENAKSHI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to Anna University, Chennai
ISO 9001:2015 Certified and Accredited by NAAC with 'A' Grade



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

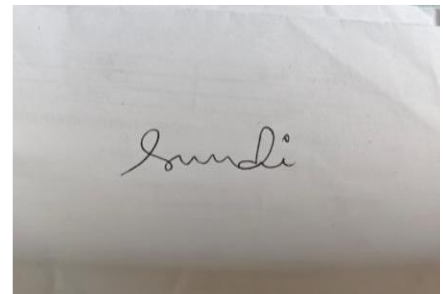
B.E. (CSE) graduates are able to

PEO-1: Exhibit students with a strong foundation in mathematics, engineering, and basic science fundamentals required to solve advanced engineering problems.

PEO-2: Develop an ability to understand the technical specifications, analyze the requirements, and design appropriate solutions, to adapt to the organizational needs by learning advanced technologies.

PEO-3: Promote the student graduates the ability to acquire multidisciplinary knowledge through projects and industrial training leading to gain domain knowledge.

PEO-4: Transform towards the issues of social relevance, group work, team management skills, and professional ethics and practice.





MEENAKSHI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to Anna University, Chennai
ISO 9001:2015 Certified and Accredited by NAAC with 'A' Grade

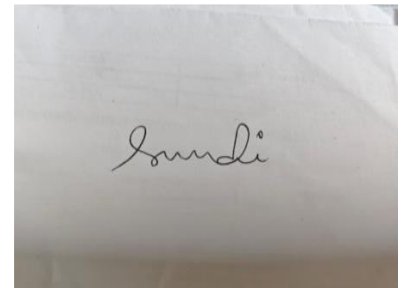


PROGRAMME SPECIFIC PROGRAM OUTCOMES (PSOs)

PSO1: Apply the acquired knowledge of basic skills, principles of computing, mathematical foundations, algorithmic principles, modeling, and design of computer-based systems in solving engineering Problems.

PSO2: Understand and analyze the interdisciplinary problems for developing innovative sustained solutions with environmental concerns.

PSO3: Update knowledge continuously in tools like Rational Rose, Android, NS2, Argo UML, and technologies like storage, Computing, communication to meet the industry requirements.





MEENAKSHI COLLEGE OF ENGINEERING

Approved by AICTE and Affiliated to Anna University, Chennai
ISO 9001:2015 Certified and Accredited by NAAC with 'A' Grade



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

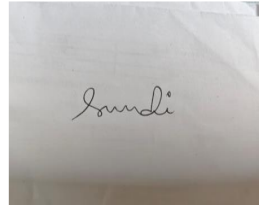
PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply this to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



COURSE OBJECTIVES:

- To understand the concept of layering in networks.
- To know the functions of the protocols of each layer of the TCP/IP protocol suite.
- To visualize the end-to-end flow of information.
- To learn the functions of the network layer and the various routing protocols
- To familiarize with the functions and protocols of the Transport layer

Week No.	Experiment to be covered
1	Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and trace route PDUs using a network protocol analyzer and examine.
2	Write an HTTP web client program to download a web page using TCP sockets.
3	Applications using TCP sockets like: a) Echo client and echo server b) Chat
4	Simulation of DNS using UDP sockets.
5	Use a tool like Wireshark to capture packets and examine the packets.
6	Write a code simulating ARP /RARP protocols.
7	Study of Network Simulator (NS) and Simulation of Congestion Control Algorithms using NS.
8	Study of TCP/UDP performance using a Simulation tool.
9	Simulation of Distance Vector/ Link State Routing algorithm.
10	Simulation of an error correction code (like CRC)

TEXT BOOKS

1. James F. Kurose, Keith W. Ross, Computer Networking, A Top-Down Approach Featuring the Internet, Eighth Edition, Pearson Education, 2021.
2. Behrouz A. Forouzan, Data Communications and Networking with TCP/IP Protocol Suite, Sixth Edition TMH, 2022

REFERENCES

1. Larry L. Peterson, Bruce S. Davie, Computer Networks: A Systems Approach, Fifth Edition, Morgan Kaufmann Publishers Inc., 2012.
2. William Stallings, Data and Computer Communications, Tenth Edition, Pearson Education, 2013.
3. Nader F. Mir, Computer and Communication Networks, Second Edition, Prentice Hall, 2014.
4. Ying-Dar Lin, Ren-Hung Hwang, Fred Baker, "Computer Networks: An Open Source Approach", McGraw Hill, 2012.

Course Outcomes:

CO1: Explain the basic layers and their functions in computer networks.

CO2: Understand the basics of how data flows from one node to another.

CO3: Analyze routing algorithms

CO4: Describe protocols for various functions in the network

CO5: Analyze the working of various application layer protocols

COs, POs AND PSOs MAPPING

[illegible]

LIST OF EXPERIMENTS

S. No.	Name of the Experiment	Marks secured (Out of 100)	Signature of the Faculty
I Cycle Experiments			
1	Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and trace route PDUs using a network protocol analyzer and examine.		
2	Write an HTTP web client program to download a web page using TCP sockets.		
3	Applications using TCP sockets, like: a) Echo client and echo server b) Chat		
4	Simulation of DNS using UDP sockets.		
5	Use a tool like Wireshark to capture packets and examine them.		
6	Write a code simulating ARP /RARP protocols.		
II Cycle Experiments			
7	Study of Network Simulator (NS) and Simulation of Congestion Control Algorithms using NS.		
8	Study of TCP/UDP performance using a Simulation tool.		
9	Simulation of Distance Vector/ Link State Routing algorithm.		
10	Simulation of an error correction code (like CRC)		
Average of Marks			
Content beyond the Syllabus*			
11	Implementing Simple File Transfer using FTP		
12	DNS Resolver using Sockets		

Exp No: 1	LEARN TO USE COMMANDS LIKE TCPDUMP, NETSTAT, IFCONFIG, NSLOOKUP AND TRACEROUTE. CAPTURE PING AND TRACEROUTE PDU's USING A NETWORK PROTOCOL ANALYZER AND EXAMINE
Date:	

PROBLEM STATEMENT:

To learn the basic UNIX commands, use a network protocol analyzer, and examine.

Commands:

ifconfig

ipconfig is a console application program of some computer operating systems that displays all current TCP/IP network configuration values and refreshes the Dynamic Host Configuration Protocol and Domain Name System settings.

Command name: ifconfig

tcpdump

It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.

Command name: tcpdump

netstat

netstat is a command-line network utility that displays network connections for Transmission Control Protocol, routing tables, and a number of network interface and network protocol statistics.

Command name: netstat -a | -e | -n | -p | -r | -s

nslookup

nslookup is a network administration command-line tool for querying the Domain Name System to obtain the mapping between domain name and IP address, or other DNS records.

Command name: nslookup google.com

Traceroute / tracert / tracepath

traceroute and tracert are computer network diagnostic commands for displaying possible routes and measuring transit delays of packets across an Internet Protocol network.

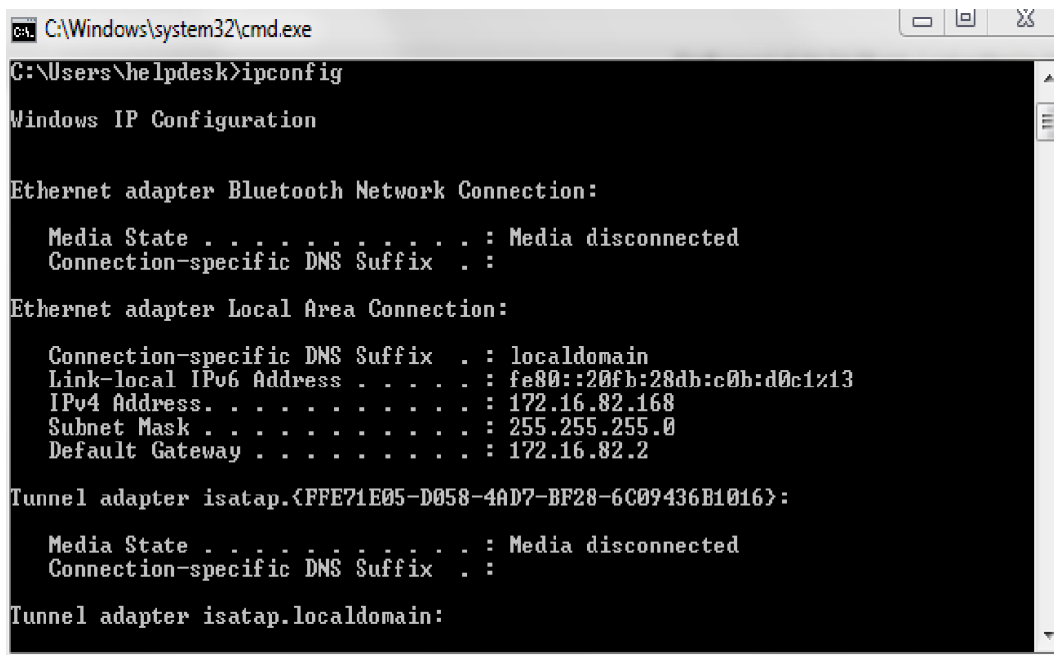
Command name: tracert 1.1.1.1

Ping

Ping is a computer network administration software ability used to test the reachability of a host on an internet protocol network.

Command name: ping google.com

Sample Output:



```
C:\Windows\system32\cmd.exe
C:\Users\helpdesk>ipconfig

Windows IP Configuration

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : localdomain
    Link-local IPv6 Address . . . . . : fe80::20fb:28db:c0b:d0c1%13
    IPv4 Address. . . . . : 172.16.82.168
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.82.2

Tunnel adapter isatap.{FFE71E05-D058-4AD7-BF28-6C09436B1016}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Tunnel adapter isatap.localdomain:
```

C:\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.19045.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PS8 Network>netstat -a

Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:80	PS8-NETWORK:0	LISTENING
TCP	0.0.0.0:135	PS8-NETWORK:0	LISTENING
TCP	0.0.0.0:445	PS8-NETWORK:0	LISTENING
TCP	0.0.0.0:808	PS8-NETWORK:0	LISTENING
TCP	0.0.0.0:1042	PS8-NETWORK:0	LISTENING

C:\Windows\system32\cmd.exe

C:\Users\PS8 Network>netstat -e

Interface Statistics

	Received	Sent
Bytes	2923318896	2673126660
Unicast packets	16598610	9601140
Non-unicast packets	72642	28142
Discards	0	0
Errors	0	0
Unknown protocols	0	

C:\Windows\system32\cmd.exe

C:\Users\PS8 Network>netstat -n

Active Connections

Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:1042	127.0.0.1:49746	ESTABLISHED
TCP	127.0.0.1:1042	127.0.0.1:49751	ESTABLISHED
TCP	127.0.0.1:9012	127.0.0.1:49738	ESTABLISHED
TCP	127.0.0.1:17532	127.0.0.1:49741	ESTABLISHED
TCP	127.0.0.1:49738	127.0.0.1:9012	ESTABLISHED
TCP	127.0.0.1:49741	127.0.0.1:17532	ESTABLISHED
TCP	127.0.0.1:49746	127.0.0.1:1042	ESTABLISHED

C:\Windows\system32\cmd.exe

C:\Users\PS8 Network>netstat -s

IPv4 Statistics

Packets Received	= 10902082
Received Header Errors	= 0
Received Address Errors	= 3290
Datagrams Forwarded	= 1104
Unknown Protocols Received	= 0
Received Packets Discarded	= 19458
Received Packets Delivered	= 10971798
Output Requests	= 5261028
Routing Discards	= 0
Discarded Output Packets	= 3198
Output Packet No Route	= 216
Reassembly Required	= 2
Reassembly Successful	= 1
Reassembly Failures	= 0
Datagrams Successfully Fragmented	= 1
Datagrams Failing Fragmentation	= 0
Fragments Created	= 2

C:\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.19045.2364]

(c) Microsoft Corporation. All rights reserved.

C:\Users\PS8 Network>nslookup google.com

DNS request timed out.

timeout was 2 seconds.

Server: UnKnown

Address: fd01:db8:1111::2

Non-authoritative answer:

Name: google.com

Addresses: 2a00:1450:4001:803::200e
142.250.186.174

C:\Windows\system32\cmd.exe

```
C:\Users\PS8 Network>tracert 1.1.1.1
```

```
Tracing route to one.one.one.one [1.1.1.1]  
over a maximum of 30 hops:
```

```
  1    151 ms    149 ms    149 ms    one.one.one.one [1.1.1.1]
```

```
Trace complete.
```

C:\Windows\system32\cmd.exe

```
C:\Users\PS8 Network>ping google.com
```

```
Pinging google.com [2a00:1450:4001:803::200e] with 32 bytes of data:  
Request timed out.
```

```
Reply from 2a00:1450:4001:803::200e: time=150ms
```

```
Reply from 2a00:1450:4001:803::200e: time=151ms
```

```
Reply from 2a00:1450:4001:803::200e: time=150ms
```

```
Ping statistics for 2a00:1450:4001:803::200e:
```

```
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
    Minimum = 150ms, Maximum = 151ms, Average = 150ms
```

OUTPUT

PRE-LAB ASSESSMENT QUESTIONS WITH ANSWERS

1. Which command is used to display network interface configurations in Linux?
2. What does the netstat command display?
3. Which command is used to check the route packets take to a destination?
4. What is the default protocol used by the ping command?
5. Which tool captures live network traffic for analysis on the command line?
6. Which command helps you query DNS servers for domain name resolution?
7. In tcpdump, which option allows you to capture only 10 packets?
8. What is the purpose of the -i option in tcpdump?
.
9. What type of PDU is generated when you run the traceroute command?
10. Which network protocol analyzer can be used with a GUI for packet inspection?

PRE-LAB WORK:

To learn the basic WINDOW commands and use a network protocol analyzer, and examine

POST LAB ASSESSMENT QUESTIONS:

1. What is the function of the tcpdump command?
2. How does traceroute determine the path packets take to a destination?
3. What type of packets does the ping command send, and why?
4. How does nslookup help in network troubleshooting?
5. Why is a network protocol analyzer like Wireshark useful in this experiment?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, the above commands were learned and examined successfully.

Exp No:2	IMPLEMENTATION OF SIMPLE HTTP PROTOCOL
Date:	

PROBLEM STATEMENT:

To implement the HTTP protocol using sockets in Python.

ALGORITHM:

Server:

1. Start
2. Create a socket and accept the connection request from the client
3. Read the HTML file to be uploaded from the disk
4. Send the HTML file to the client
5. Terminate the connection
6. Stop

Client:

1. Start
2. Establish a connection with the server
3. Receive the HTML file from the server
4. Decode the HTML file and display it in the browser
5. Stop

PROGRAMS:

Index.html

```
<html>

    <body>

    </body>

</html>
```

Server.py

```
import socket

s = socket.socket()

print ("Socket successfully created")
```



```
port = 12345
s.bind(('', port))
print ("socket binded to %s" %(port))
s.listen(5)
print ("socket is listening") while
True:
    c, addr = s.accept()
    print ('Got connection from', addr )

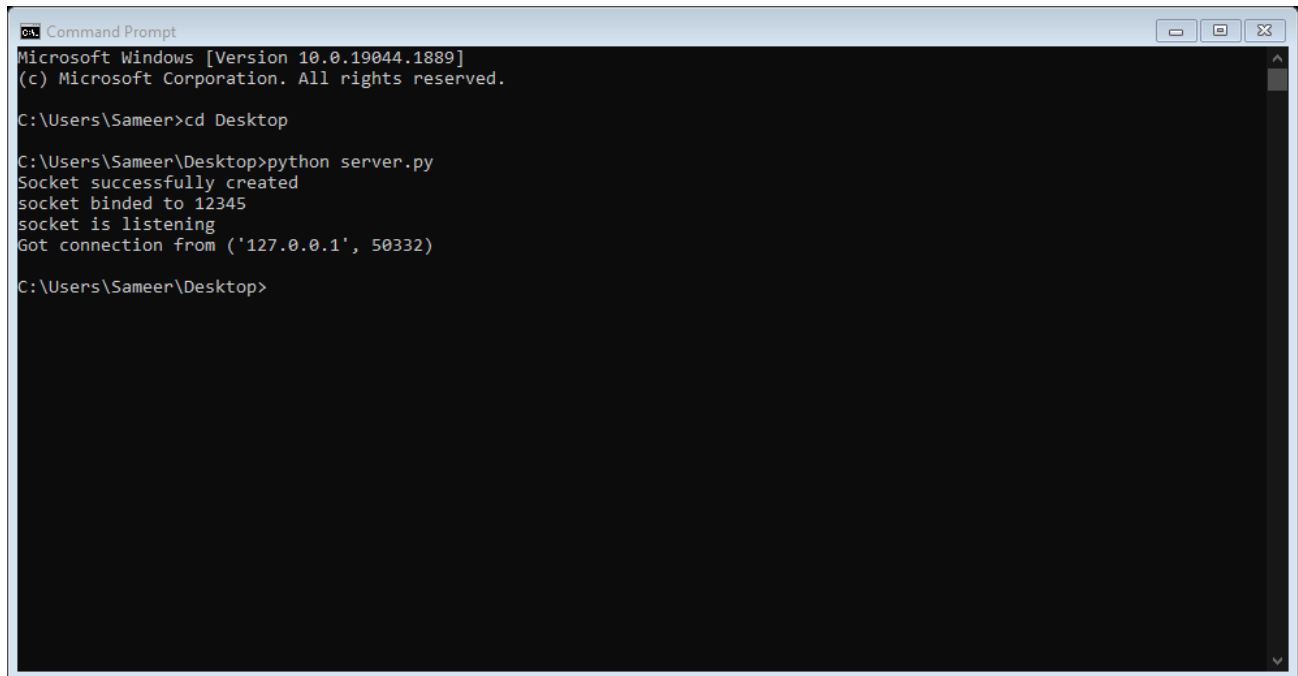
    c.send('Thank you for connecting!'.encode())
    c.send('index.html'.encode())
    c.close()
    break
```

Client.py

```
import socket
import webbrowser
s = socket.socket()
port = 12345
s.connect(('localhost', port))
print(s.recv(1024).decode())
filename = s.recv(1024).decode()
webbrowser.open_new_tab(filename)
s.close()
```

SAMPLE OUTPUT:

Server.py



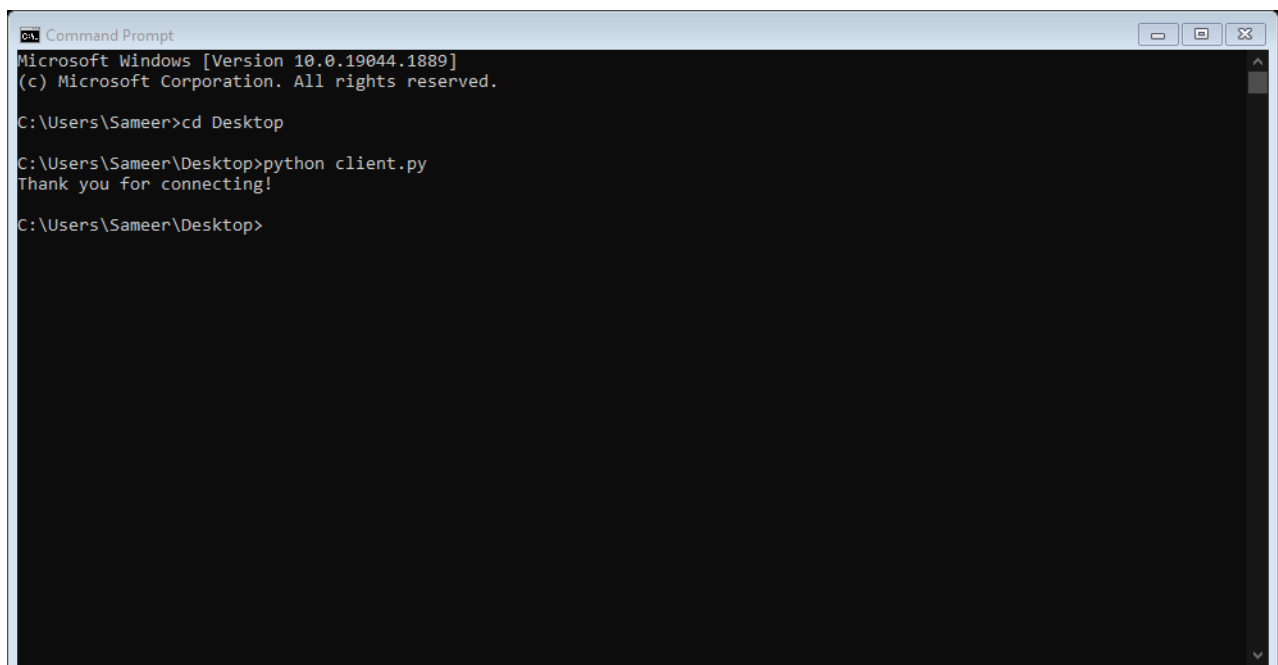
```
CA Command Prompt
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sameer>cd Desktop

C:\Users\Sameer\Desktop>python server.py
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 50332)

C:\Users\Sameer\Desktop>
```

Client.py



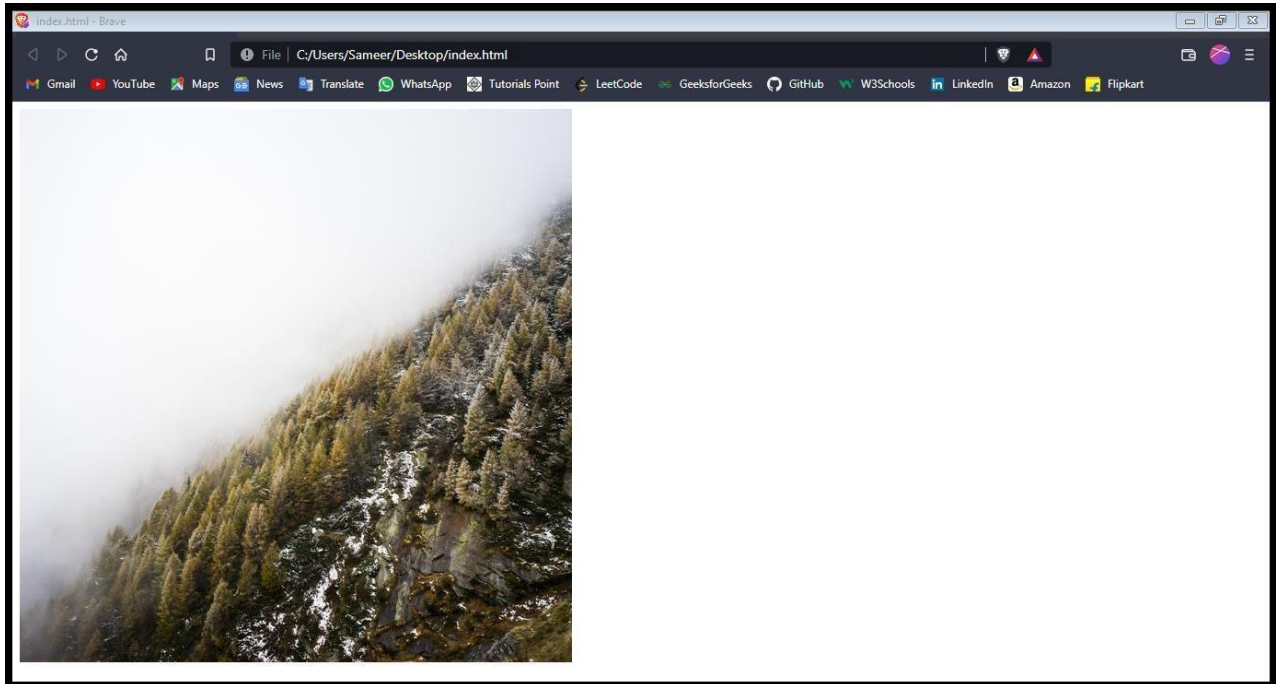
```
CA Command Prompt
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sameer>cd Desktop

C:\Users\Sameer\Desktop>python client.py
Thank you for connecting!

C:\Users\Sameer\Desktop>
```

Browser



OUTPUT:

PRE-ASSESSMENT QUESTIONS WITH ANSWERS:

1. Which protocol does HTTP run on by default?
2. What is the default port number for HTTP?
3. In Python, which module can be used for low-level socket programming?
4. What does the connect() method in a TCP client socket do?
5. What is the purpose of the HTTP GET request method?
6. Which header in HTTP specifies the domain name of the server?
7. In HTTP/1.0, what happens to the TCP connection after the response is sent?
8. Which HTTP version introduced persistent connections by default?
9. What command in Python's socket library is used to send data to the server?
10. What function is used to receive data from a TCP socket in Python?

PRE-LAB WORK:

Implementation of a Simple HTTP Protocol using a Python Program.

POST-LAB QUESTIONS:

1. Which transport protocol is used by HTTP and why?
2. What is the default port for HTTP and HTTPS?
3. In a TCP socket program, what does the `recv()` function do?
4. Why must an HTTP request include a Host header in HTTP/1.1?
5. What is the difference between HTTP GET and POST requests?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, the HTTP web client program to download a web page using TCP sockets was executed successfully.

Exp No:3a	APPLICATIONS USING TCP SOCKETS (ECHO CLIENT & SERVER)
Date:	

PROBLEM STATEMENT:

To create an echo client & server application using TCP sockets in Python.

ALGORITHM:**SERVER:**

1. Start
2. Create a TCP socket and listen for connection requests
3. Accept the connection from the client
4. Receive the message from the client
5. Echo it back to the client
6. Repeat step 4 and 5 until the quit message is received.
7. Stop

CLIENT:

1. Start
2. Create a TCP and socket and establish a connection with server
3. Get the input message from the user and send it to the server
4. Receive the echo reply from the server
5. Decode and print it
6. Send the message quit to terminate the connection
7. Stop

PROGRAM:**Server.py**

```
import socket
```

```
server = socket.socket()
```

```
server.bind(('',12345))
```

```
server.listen()
```

```
print("Server is listening...")
```

```
c,addr = server.accept()
```

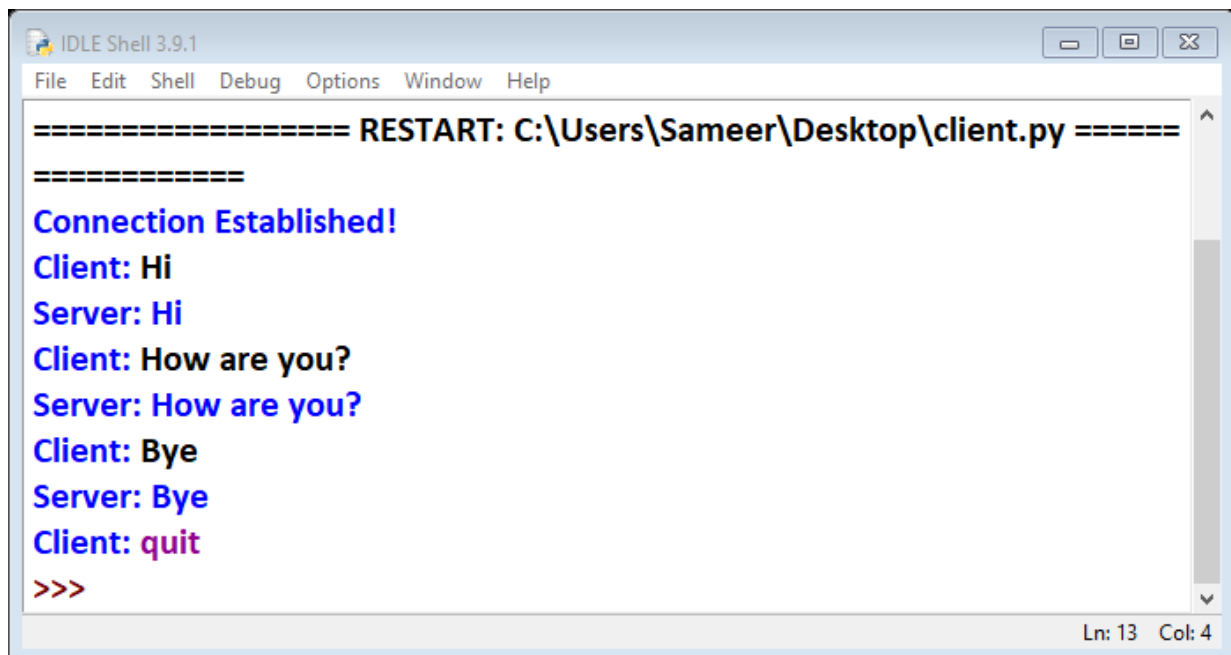


```
    print("Got Connection from ",addr)
while(True):
    data = c.recv(1024)
    msg = data.decode()
    if(msg!="quit"):
        print("Recieved:",msg)
        c.sendall(data)
    else:
        c.close()
        server.close()
        break
```

Client.py

```
import socket
client = socket.socket()
client.connect(("localhost",12345))
print("Connection Established! ")
while(True):
    msg = input("Client: ")
    client.sendall(msg.encode())
    if(msg!='quit'):
        data = client.recv(1024).decode()
        print("Server:",data)
    else:
        client.send('quit'.encode())
        client.close()
        break
```

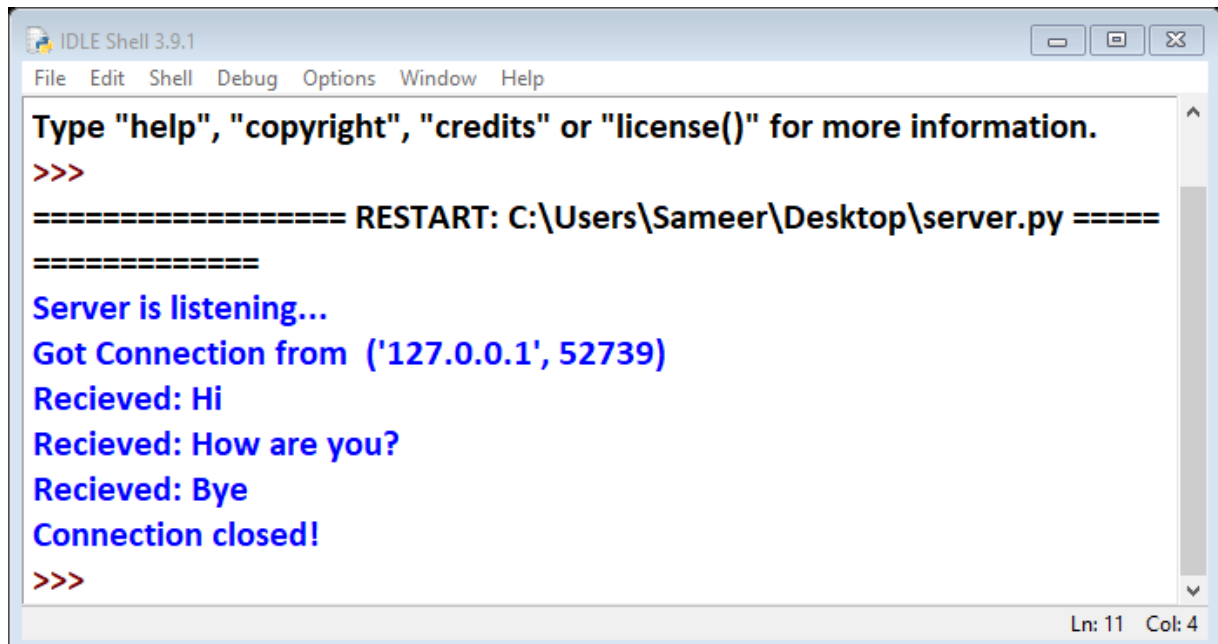
SAMPLE OUTPUT



A screenshot of the IDLE Shell 3.9.1 window. The title bar reads "IDLE Shell 3.9.1". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following output:

```
===== RESTART: C:\Users\Sameer\Desktop\client.py =====  
=====  
Connection Established!  
Client: Hi  
Server: Hi  
Client: How are you?  
Server: How are you?  
Client: Bye  
Server: Bye  
Client: quit  
>>>
```

The status bar at the bottom right shows "Ln: 13 Col: 4".



A screenshot of the IDLE Shell 3.9.1 window. The title bar reads "IDLE Shell 3.9.1". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following output:

```
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\Sameer\Desktop\server.py =====  
=====  
Server is listening...  
Got Connection from ('127.0.0.1', 52739)  
Recieved: Hi  
Recieved: How are you?  
Recieved: Bye  
Connection closed!  
>>>
```

The status bar at the bottom right shows "Ln: 11 Col: 4".

OUTPUT:

PRE-LAB ASSESSMENT QUESTIONS WITH ANSWERS:

1. Which function in Python is used to create a socket?
2. What does the bind() function do in a TCP server?
3. What is the purpose of the listen() method in a TCP server?
4. Which method is used by a TCP server to accept a new incoming connection?
5. What is the default transport protocol used by TCP sockets?
6. Which function is used by the client to connect to a TCP server?
7. What is the main difference between an echo server and a chat application?
8. Which socket method is used to send data to the other end of the connection?
9. Which method is used to close a socket connection in Python?
10. In a chat application using TCP sockets, what feature allows simultaneous sending and receiving of messages?

PRE-LAB WORK:

To develop an echo client program using TCP sockets

POST-LAB ASSESSMENT QUESTIONS:

1. What is the main purpose of an echo server?

2. Which TCP socket method is used by a server to wait for incoming connections?

3. In a chat application, how can you allow sending and receiving messages simultaneously?

4. What is the difference between TCP and UDP in socket communication?

5. Why is `sendall()` preferred over `send()` in some TCP applications?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT

Thus, the echo client and server application using TCP sockets in Python was executed successfully.

Exp No:3b	APPLICATIONS USING TCP SOCKETS (CHAT APPLICATION)
Date:	

PROBLEM STATEMENT:

To create a chat application using TCP sockets in Python.

ALGORITHM:

Server:

1. Start
2. Create a TCP socket and listen for connection requests
3. Accept the connection from the client
4. Receive the message from the client, decode and display it.
5. Send a reply message to the client
6. Repeat step 4 and 5 until the quit message is received.
7. Stop

Client:

1. Start
2. Create a TCP and socket and establish a connection with server
3. Get the input message from the user and send it to the server
4. Receive the reply from the server
5. Decode and display it
6. Send the message quit to terminate the connection
7. Stop

PROGRAM:

Server.py

```
import socket

server = socket.socket()
server.bind(('',12345))
print("Socket is created successfully")
server.listen()
print("Waiting for Connection...")
```

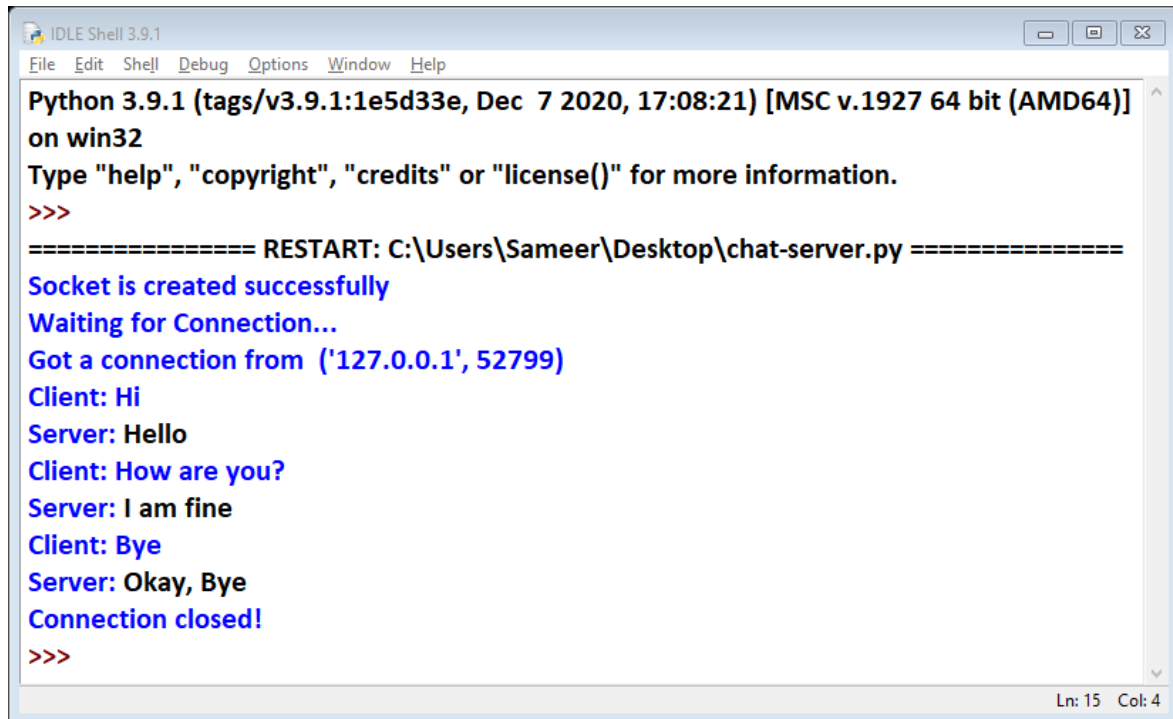


```
c,addr = server.accept()
print("Got a connection from ",addr)
while(True):
    clientmsg = c.recv(1024).decode()
    if(clientmsg!="quit"):
        print("Client:",clientmsg)
        msg = input("Server: ")
        c.send(msg.encode())
    else:
        c.close()
        server.close()
        break
```

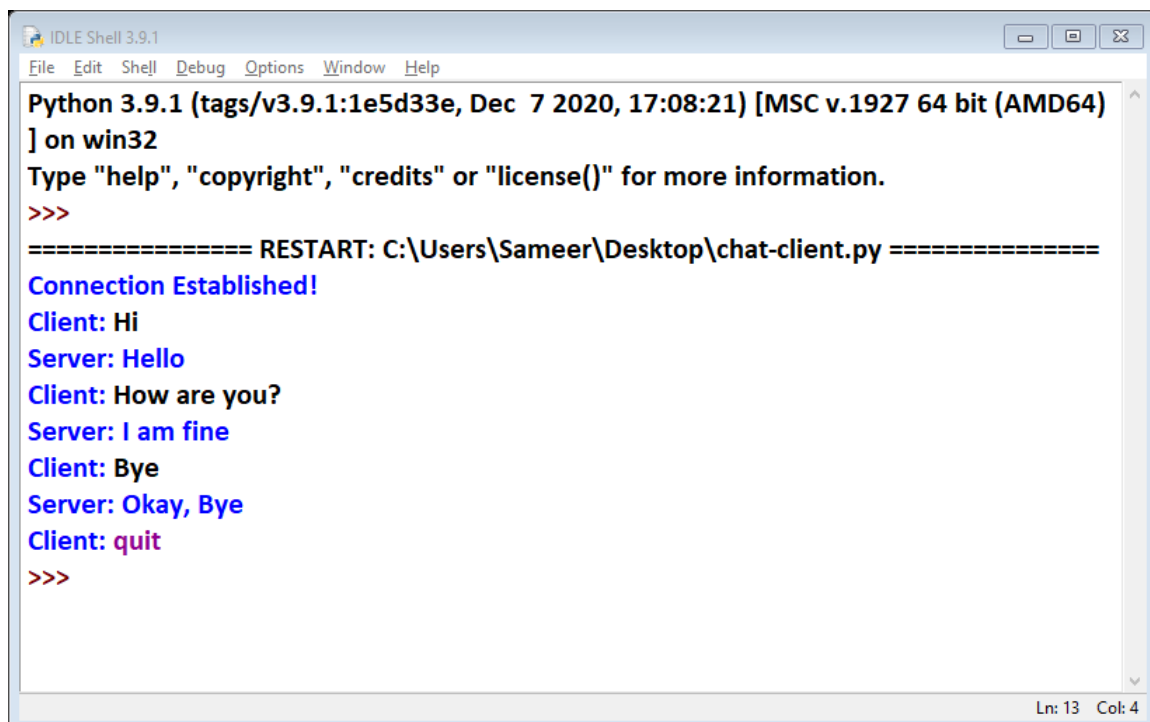
Client.py

```
import socket
client = socket.socket()
client.connect(('localhost',12345))
print("Connection Established!")
while True:
    msg = input("Client: ")
    if(msg!="quit"):
        client.send(msg.encode())
        print("Server:",client.recv(1024).decode())
    else:
        client.send("quit".encode())
        client.close()
        break
```

SAMPLE OUTPUT:



```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\chat-server.py =====
Socket is created successfully
Waiting for Connection...
Got a connection from ('127.0.0.1', 52799)
Client: Hi
Server: Hello
Client: How are you?
Server: I am fine
Client: Bye
Server: Okay, Bye
Connection closed!
>>>
```



```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]
] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\chat-client.py =====
Connection Established!
Client: Hi
Server: Hello
Client: How are you?
Server: I am fine
Client: Bye
Server: Okay, Bye
Client: quit
>>>
```

OUTPUT

PRE-LAB ASSESSMENT QUESTIONS WITH ANSWERS:

1. Which protocol will be used for this chat application?
2. Which Python module is used for creating sockets?
3. What is the default port range for TCP connections?
4. Which function is used to bind a socket to an address?
5. In a TCP server program, which function is used to accept incoming connections?
6. Which socket method is used to send data to the connected socket?
7. What is the purpose of the listen() function in TCP socket programming?
8. Which method is used to close a socket in Python?
9. What type of communication does TCP provide?
10. Which method is used by the client to connect to a server in TCP?

PRE-LAB WORK:

Develop a chat program using TCP sockets in Python.

POST-LAB ASSESSMENT QUESTIONS:

1. What is the main difference between TCP and UDP?
2. Why do we use sockets in network programming?
3. What is the role of `bind ()` in a TCP server program?
4. In a chat application, why is TCP preferred over UDP?
5. How does the `accept ()` function work in a TCP server?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT

Thus, a chat application using TCP sockets in Python was executed successfully.

Exp No:4	SIMULATION OF DNS USING UDP SOCKETS.
Date:	

PROBLEM STATEMENT:

To simulate the working of DNS server using python.

ALGORITHM:**Server:**

1. Start
2. Create a DNS table with domains and their respective addresses.
3. Create a UDP Socket and listen for requests
4. Accept the client's request and decode the message
5. Send the IP address back to the client
6. Stop

Client:

1. Start
2. Create a UDP socket with the server's IP and port number.
3. Get the website name from the user and send it to the server
4. Receive the reply from the server, decode and display it.
5. Stop.

Program:**Server.py**

```
import socket
dns_table = {

    "www.google.com":"110.10.1.1",
    "www.youtube.com":"11.11.11.11",
    "www.instagram.com":"12.12.12.12",
    "www.netflix.com":"110.10.1.100",
    "www.amazon.com":"110.19.15.12"}
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
print("Starting Server...")
s.bind(("127.0.0.1",1234))
while True:
    data,addr = s.recvfrom(1024)
    print(f'{addr} wants to fetch the data!')
```



```
data = data.decode()
ip = dns_table.get(data,"Not Found").encode()
send = s.sendto(ip,addr)
reply,addr = s.recvfrom(1024)
if(reply.decode()=="n"):
    break
```

Client.py

```
import socket

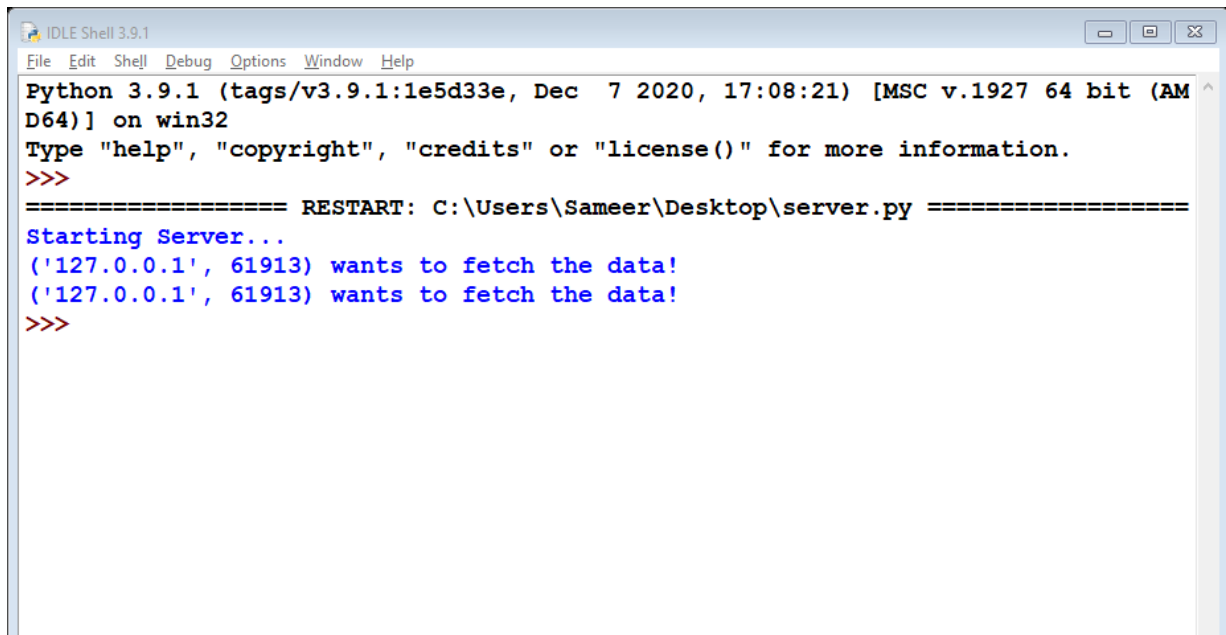
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
address = ("127.0.0.1",1234)
c = "Y"
while(c.upper() == "Y"):
    req_domain = input("Enter the domain name: ")
    s.sendto(req_domain.encode(),address)

    data,ipaddr = s.recvfrom(1024)
    reply_ip = data.decode().strip()
    print(f"The IP Address of {req_domain} is: {reply_ip}")

    c = input("Do you want to continue?(y/n): ")
    s.sendto(c.encode(),address)
s.close()
```

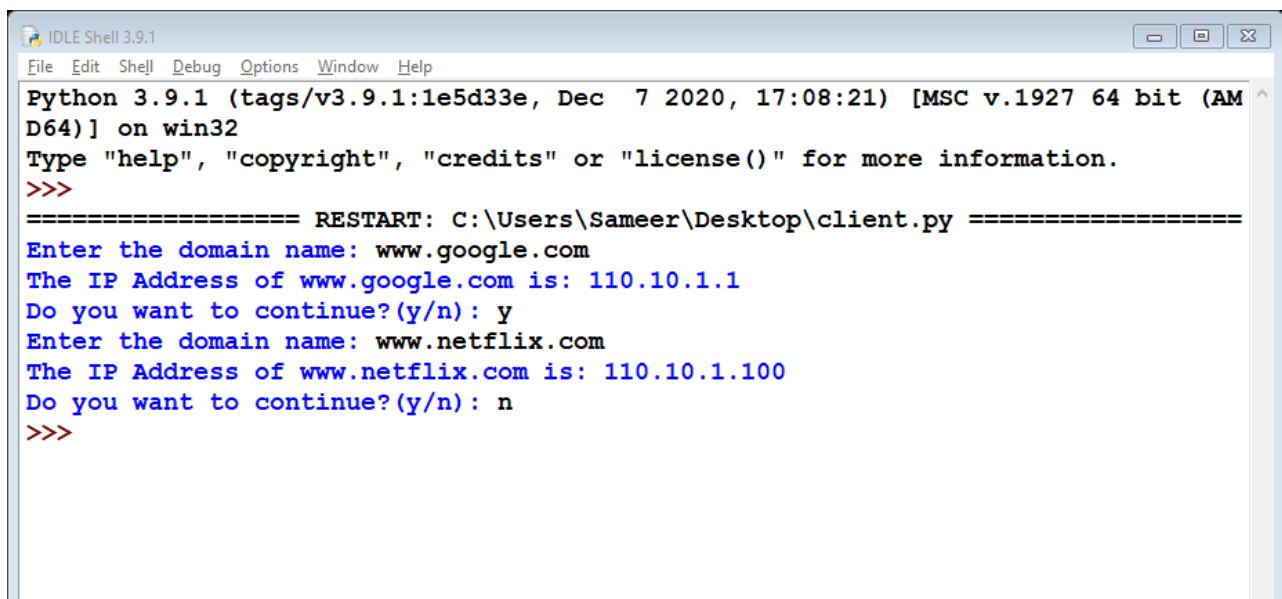
SAMPLE OUTPUT:

Server.py



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\server.py =====
Starting Server...
('127.0.0.1', 61913) wants to fetch the data!
('127.0.0.1', 61913) wants to fetch the data!
>>>
```

Client.py



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\client.py =====
Enter the domain name: www.google.com
The IP Address of www.google.com is: 110.10.1.1
Do you want to continue?(y/n): y
Enter the domain name: www.netflix.com
The IP Address of www.netflix.com is: 110.10.1.100
Do you want to continue?(y/n): n
>>>
```

OUTPUT

PRE-LAB ASSESSMENT QUESTIONS & ANSWERS

1. Which transport protocol does DNS primarily use for queries and responses?
2. What is the default port number for DNS?
3. Why does DNS typically use UDP instead of TCP?
4. In which cases does DNS use TCP instead of UDP?
5. What is the main purpose of DNS?
6. Which Python module is commonly used for low-level UDP socket programming?
7. Q: What is the term for a DNS server that resolves queries recursively on behalf of a client?
8. What type of DNS record maps a domain name to an IPv4 address?
9. What type of DNS record maps a domain name to an IPv6 address?
10. What type of DNS record maps a domain to another domain name?

PRE-LAB WORK:

Design the structure of DNS Query and Response Messages.

POST-LAB ASSESSMENT QUESTIONS:

1. Why does DNS usually use UDP instead of TCP?
2. On which port does DNS operate by default?
3. When does DNS use TCP instead of UDP?
4. What is the role of a recursive DNS server?
5. What is the difference between an A record and a CNAME record in DNS?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, simulating the working of a DNS server using Python was executed successfully.

Exp No:5	USE A TOOL LIKE WIRESHARK TO CAPTURE PACKETS AND EXAMINE THE PACKETS
Date:	

PROBLEM STATEMENT:

To capture and analyze network packets using a tool like Wireshark to understand protocol behavior and detect issues.

ALGORITHM:

- a. Start
- b. Launch the Wireshark application on the system.
- c. Select the desired network interface (e.g., Ethernet, Wi-Fi) to monitor.
- d. Click Start Capture to begin recording live network traffic.
- e. Generate network activity (e.g., open websites, send pings) to produce packets.
- f. Stop the capture after collecting sufficient packet data.
- g. Use filters in Wireshark to view specific protocol traffic (e.g., http, tcp, udp, icmp).
- h. Select an individual packet to inspect its details (headers, source/destination addresses, payload).
- i. Analyze the captured packets to identify protocol behavior and detect any unusual activity.
- j. Save the capture file for documentation or future analysis.
- k. End

PROGRAM:

```
# Packet Capture and Analysis using Scapy

from scapy.all import sniff

# Function to process each captured packet
def analyze_packet(packet):
    print("\n--- Packet Captured ---")
    print(packet.summary())      # Brief summary of the packet
    packet.show()                # Detailed packet information

# Main program
```



```
if __name__ == "__main__":  
    print("Starting packet capture... Press Ctrl+C to stop.")  
  
    # Capture 10 packets (you can change count or remove it for continuous capture)  
    packets = sniff(count=10, prn=analyze_packet)  
  
    print("\nPacket capture complete. Saving to file...")  
    packets.summary()  
  
    # Save packets to a file (Wireshark-compatible format)  
    packets.wrpcap("captured_packets.pcap", packets)  
    print("Packets saved to captured_packets.pcap")
```

SAMPLE OUTPUT:

```
--- Packet Captured ---  
Ether / IP / TCP 192.168.1.5:53012 > 142.250.182.142:http S  
####[ Ethernet ]####  
dst      = 84:3a:4b:5c:6d:7e  
src      = 12:34:56:78:9a:bc  
type     = IPv4  
####[ IP ]####  
version  = 4  
ihl      = 5  
tos      = 0x0  
len      = 60  
id       = 12345  
flags    = DF  
ttl      = 64  
proto    = tcp  
src      = 192.168.1.5  
dst      = 142.250.182.142  
####[ TCP ]####
```

sport = 53012
dport = http
seq = 123456789
ack = 0
flags = S
window = 64240

--- Packet Captured ---

Ether / IP / UDP 192.168.1.5:5678 > 192.168.1.1:domain / DNS Qry "www.example.com."

####[Ethernet]####

dst = 00:1a:2b:3c:4d:5e
src = 12:34:56:78:9a:bc
type = IPv4

####[IP]####

version = 4
ihl = 5
tos = 0x0
len = 78
id = 67890
flags = DF
ttl = 64
proto = udp
src = 192.168.1.5
dst = 192.168.1.1

####[UDP]####

sport = 5678
dport = domain

####[DNS]####

id = 1234
qr = 0
qdcount = 1
qd = 'www.example.com.'

Packet capture complete. Saving to file...

Packets saved to captured_packets.pcap

OUTPUT:

PRE-ASSESSMENT QUESTIONS WITH ANSWERS:

1. What is Wireshark primarily used for?
.
2. What type of tool is Wireshark classified as?
3. Which file extension is commonly used to save Wireshark capture files?
4. What is the default capture mode used by Wireshark?
5. Which filter type in Wireshark is used while capturing packets?
.
6. Which filter type in Wireshark is applied after packets have been captured?
7. What is the purpose of the `tcp.port == 80` display filter in Wireshark?
8. What layer of the OSI model does Wireshark allow you to examine?
9. Which command-line tool is similar to Wireshark for packet capturing?
10. Can Wireshark be used to capture encrypted HTTPS traffic?
.

PRE-LAB WORK:

Analysis of Network Traffic through Packet Sniffing Techniques

POST-ASSESSMENT QUESTIONS:

1. What is Wireshark used for?
2. What is the difference between a capture filter and a display filter in Wireshark?
3. What does the term "promiscuous mode" mean in packet capturing?
4. How can you filter packets for only HTTP traffic in Wireshark?
5. Can Wireshark analyze both TCP and UDP traffic?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, to capture and analyze network packets using a tool like Wireshark was executed successfully.

Exp No:6	SIMULATION OF ARP/RARP PROTOCOLS
Date:	

PROBLEM STATEMENT:

To write a program to simulate the ARP/RARP protocols using python.

ALGORITHM:

Server:

1. Start
2. Create a table with IP addresses and their respective MAC addresses
3. Create a TCP Socket and listen for connection requests
4. Accept the request and establish a connection
5. Receive the address from the client and decode it
6. Send the respective IP/MAC address back to the client
7. Stop

Client:

1. Start
2. Create a TCP socket and establish a connection with the server
3. Get the address from the user and Send it to the server
4. Receive the reply from the server, decode and display the address.
5. Stop.

PROGRAM:

Server.py

```
import socket
```

```
table= {
```

```
    '192.168.1.1':'1E.4A.4A.11',
```

```
    '192.168.2.1':'5E.51.4B.01',
```

```
    '192.168.1.3':'4B.35.CD.32',
```

```
    '4B.35.CD.32' : '192.168.1.3',
```

```
    '5E.51.4B.01': '192.168.2.1',
```

```
    '1E.4A.4A.11':'192.168.1.1'}
```

```
s = socket.socket()
```

```
ip = socket.gethostname()
```

```
s.bind((ip,1234))
s.listen()
clientsocket, address = s.accept()

ip = clientsocket.recv(1024)
ip = ip.decode('utf-8')
mac = table.get(ip,"no entry for given address")
clientsocket.send(mac.encode())
```

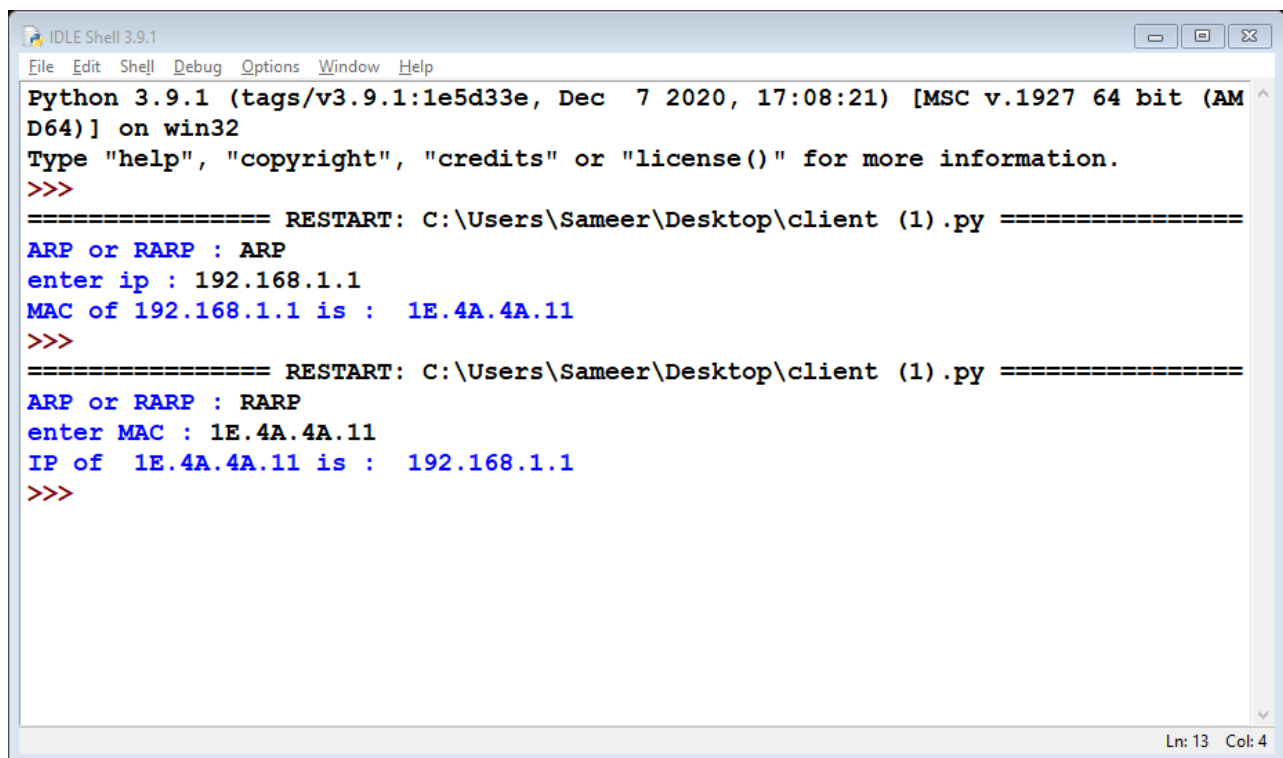
Client.py

```
import socket

s = socket.socket()

ip = socket.gethostname()
s.connect((ip,1234))
a = input('ARP or RARP : ')
if(a=="ARP"):
    add = input("enter ip : ")
elif(a=="RARP"):
    add = input("enter MAC : ")
s.send(add.encode())
mac = s.recv(1024)
mac = mac.decode("utf-8")
if(a=="ARP"):
    print("MAC of",add,'is : ',mac)
else:
    print('IP of ',add,'is : ',mac)
```

SAMPLE OUTPUT:



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\client (1).py =====
ARP or RARP : ARP
enter ip : 192.168.1.1
MAC of 192.168.1.1 is : 1E.4A.4A.11
>>>
===== RESTART: C:\Users\Sameer\Desktop\client (1).py =====
ARP or RARP : RARP
enter MAC : 1E.4A.4A.11
IP of 1E.4A.4A.11 is : 192.168.1.1
>>>
```

Ln: 13 Col: 4

OUTPUT

PRE-ASSESSMENT QUESTIONS & ANSWERS:

1. What does ARP stand for?
2. What does RARP stand for?
3. Which layer of the OSI model does ARP operate in?
4. What is the main function of ARP?
5. What is the main function of RARP?
6. Which protocol replaced RARP in modern networks?
7. What type of message is sent when a device requests MAC information using ARP?
8. What type of message is sent in response to an ARP Request?
9. What is the default ARP cache timeout in many systems (approximate value)?
.
10. Which protocol is commonly used alongside ARP in IPv6 networks?

PRE-LAB WORK:

Design Address Resolution Mechanisms in Computer Networks

POST-LAB ASSESSMENT QUESTIONS:

1. What is the purpose of ARP in a network?
2. How does RARP differ from ARP?
3. What type of ARP message is sent to all devices on a LAN?
4. Why is RARP rarely used today?
5. In which OSI layer do ARP and RARP operate?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, the program to simulate the ARP/RARP protocols using Python was executed successfully.

Exp No:7	STUDY OF NETWORK SIMULATOR (NS) AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS
Date:	

PROBLEM STATEMENT:

To study Network Simulator (Ns) And Simulation of congestion control algorithm using NS2 Simulator.

ALGORITHM:

1. Create a simulator object
2. Set routing as dynamic
3. Open the trace and nam trace files
4. Create nodes and links between them
5. Create agents and attach them to the nodes
6. Create the application and attach them to the TCP agent
7. Create a variable to store the RTT of the acknowledgement
8. Define a procedure to plot the congestion graph
9. Define the finish procedure
10. Connect TCP and TCP sink
11. Run the simulation

PROGRAM:

Congestion.tcl

```

set ns [new Simulator]
set f [ open congestion.tr w ]
$ns trace-all $f
set nf [ open congestion.nam w ]
$ns namtrace-all $nf
$ns color 1 Red
$ns color 2 Blue
$ns color 3 White
$ns color 4 Green
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail

```

```
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp2
set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $n2 $tcp3
set tcp4 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp4
$tcp1 set fid_ 1
$tcp2 set fid_ 2
$tcp3 set fid_ 3
$tcp4 set fid_ 4
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1 set
sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2 set
sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3 set
sink4 [new Agent/TCPSink]
$ns attach-agent $n4 $sink4
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3
$ns connect $tcp4 $sink4
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set type_ FTP
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ftp3 set type_ FTP
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ftp4 set type_ FTP
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
set p1 [new Agent/Ping]
$ns attach-agent $n4 $p1
$ns connect $p0 $p1
Agent/Ping instproc recv {from rtt} {
```

```

$self instvar node_
puts "node [$node_ id] received ping answer from \
$from with round-trip-time $rtt ms."
}

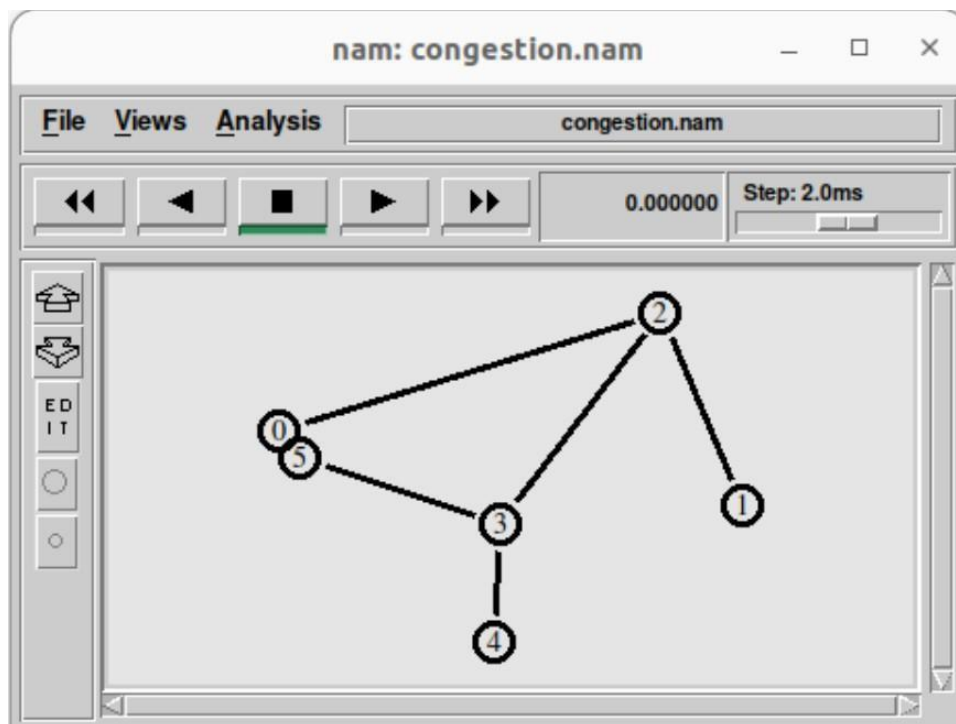
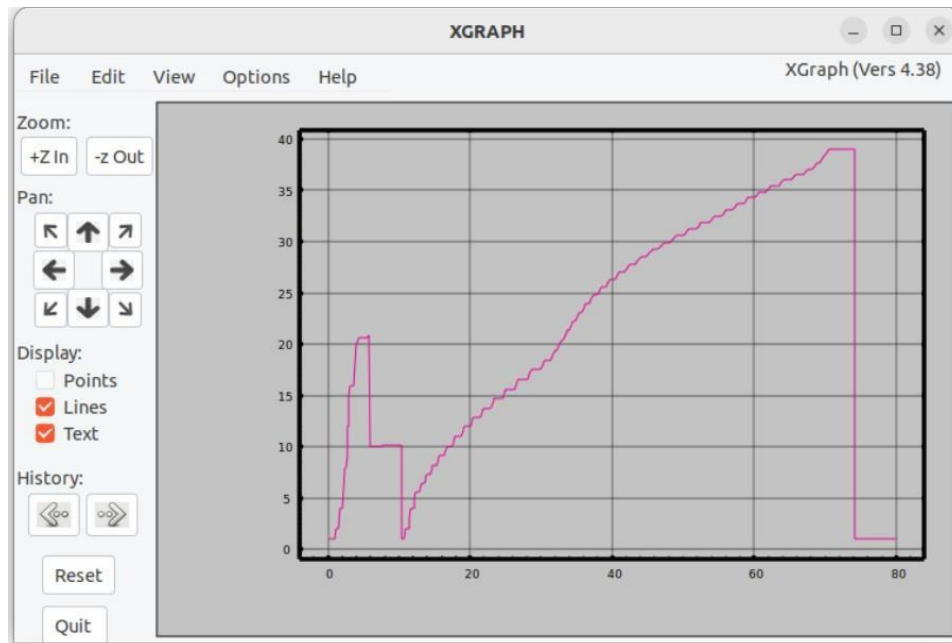
$ns at 0.2 "$p0 send"
$ns at 0.3 "$p1 send"
$ns at 0.5 "$ftp1 start"
$ns at 0.6 "$ftp2 start"
$ns at 0.7 "$ftp3 start"
$ns at 0.8 "$ftp4 start"
$ns at 66.0 "$ftp4 stop"
$ns at 67.0 "$ftp3 stop"
$ns at 68.0 "$ftp2 stop"
$ns at 70.0 "$ftp1 stop"
$ns at 70.1 "$p0 send"
$ns at 70.2 "$p1 send"
$ns at 80.0 "finish"
proc plotWindow {tcpSource outfile} {
global ns
set now [$ns now]
set cans_ [$coerce set cans_]
puts $out file "$now $cwnd_"
$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"
}

set outfile [open "congestion.xg" w]
$ns at 0.0 "plotWindow $tcp1 $outfile"
proc finish {} {
exec nam congestion.nam &
exec xgraph congestion.xg -geometry 300x300 &
exit 0
}

$ns run

```

SAMPLE OUTPUT:



OUTPUT:

PRE-LAB ASSESSMENT QUESTIONS & ANSWERS:

1. What does NS in networking stand for?
2. Which version of NS introduced support for both wired and wireless simulation?
3. Which language is used to write simulation scripts in NS-2?
4. Name one congestion control algorithm implemented in TCP.
5. In TCP congestion control, what does AIMD stand for?
6. Which TCP phase is used to find the network capacity gradually?
7. In NS-2, which file format is typically used to store simulation trace results?
8. What is the main goal of congestion control in a network?
9. Which TCP variant uses fast retransmit and fast recovery mechanisms?
10. In NS simulations, which command is used to start the event scheduler?

PRE-LAB WORK:

Develop a simple code using a congestion control algorithm in a network

POST-LAB ASSESSMENT QUESTIONS:

- a. What is Network Simulator (NS) and why is it used?

- b. Which programming languages are primarily used in NS-2?

- c. Name two TCP congestion control algorithms available in NS.

- d. What is the role of the trace file in NS simulation?

- e. What is the difference between Slow Start and Congestion Avoidance in TCP?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, the congestion control algorithm has been simulated, and the output has been obtained successfully.

Exp No:8	STUDY OF TCP/UDP PERFORMANCE USING SIMULATION TOOL
Date:	

PROBLEM STATEMENT:

To simulate the performance of TCP/UDP using NS2.

ALGORITHM:

TCP:

1. Create a simulator object
2. Set routing as dynamic
3. Open the trace and nam trace files
4. Create nodes and links between them
5. Create agents and attach them to the nodes
6. Create the application and attach them to the TCP agent
7. Define the finish procedure
8. Connect TCP and TCP sink
9. Run the simulation

UDP:

1. Create a simulator object
2. Set routing as dynamic
3. Open the trace and nam-trace files
4. Define the finish procedure
5. Create nodes and links between them
6. Create the agents and attach them with the nodes
7. Create the applications and attach them to the UDP agent
8. Connect the UDP and Null Agents
9. Run the simulation

PROGRAM:

TCP.tcl

```
set ns [new Simulator]
$ns color 0 Blue
$ns color 1 Red
$ns color 2 Yellow
set n0 [$ns node]
```

```

set n1 [$ns
node] set n2
[$ns node] set
n3 [$ns node]
set f [open tcpout.tr w]
$ns trace-all $f
set nf [open tcpout.nam w]
$ns namtrace-all $nf
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5 set
tcp [new Agent/TCP]
$tcp set class_ 1
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
proc plotWindow {tcpSource outfile} {
global ns
set now [$ns now]
set cwnd_ [$tcpSource set cwnd_] puts
$outfile "$now $cwnd_"
$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"
}

set outfile [open "tcpout.xg" w]
$ns at 0.0 "plotWindow $tcp $outfile"
$ns at 1.2 "$ftp start"
$ns at 1.35 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n3 $sink"
$ns at 3.0 "finish"
proc finish {} {
global ns f nf
$ns flush-
trace close $f
close $nf
puts "Running nam.."
exec xgraph tcpout.xg -geometry 600x800 & exec
nam tcpout.nam &
exit 0
}

$ns run

```

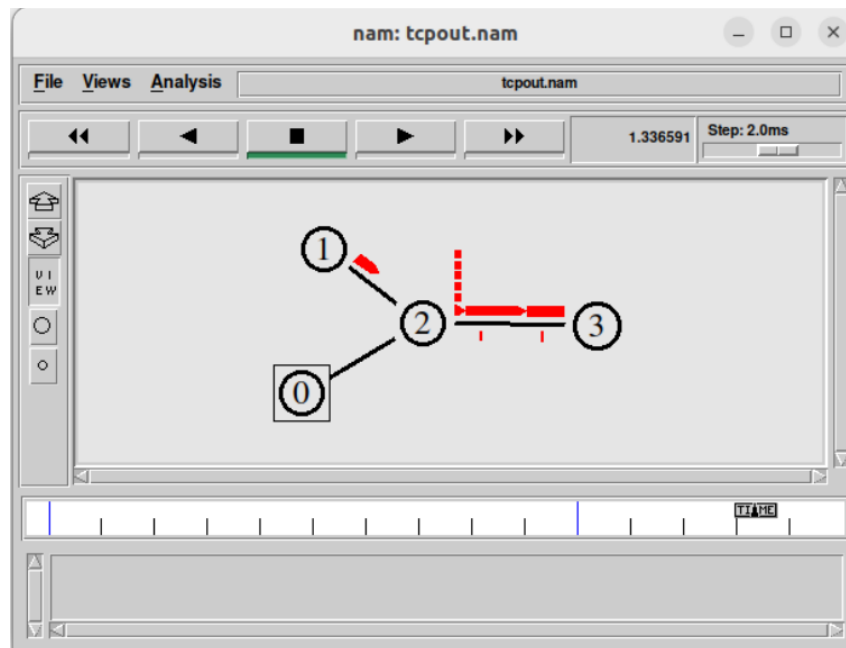
UDP.tcl

```
set ns [new Simulator]
$ns color 0 Blue
$ns color 1 Red
$ns color 2
Yellow set n0
[$ns node] set n1
[$ns node] set n2
[$ns node] set n3
[$ns node]
set f [open udpout.tr w]
$ns trace-all $f
set nf [open udpout.nam w]
$ns namtrace-all $nf
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n3 queuePos 0.5 set
udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$scr0 attach-agent $udp0
set udp1 [new Agent/UDP]
$ns attach-agent $n3 $udp1
$udp1 set class_ 0
set cbr1 [new Application/Traffic/CBR]
$scr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0 set
null1 [new Agent/Null]
$ns attach-agent $n1 $null1
$ns connect $udp0 $null0
$ns connect $udp1 $null1
$ns at 1.0 "$scr0 start"
$ns at 1.1 "$scr1 start" puts
[$scr0 set packetSize_] puts
[$scr0 set interval_]
$ns at 3.0 "finish"
proc finish {} {
global ns f nf
$ns flush-
trace close $f
close $nf
puts "Running nam.."
exec nam udpout.nam &
```

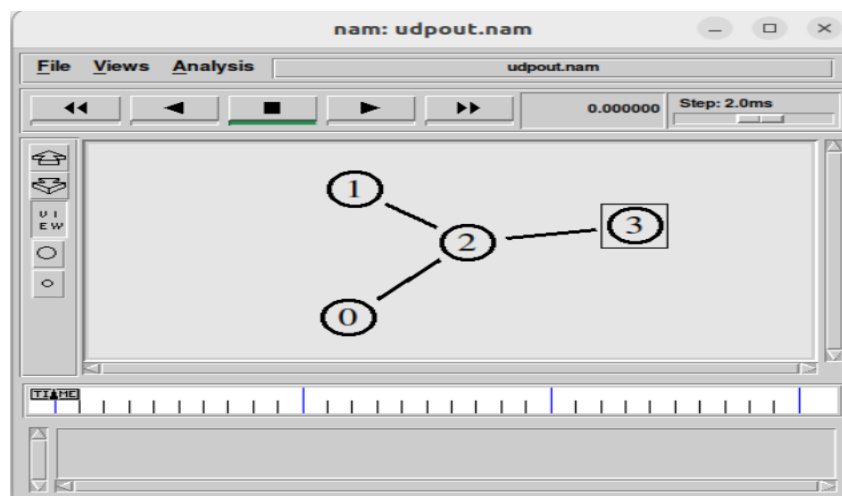
```
exit 0
}
$ns run
```

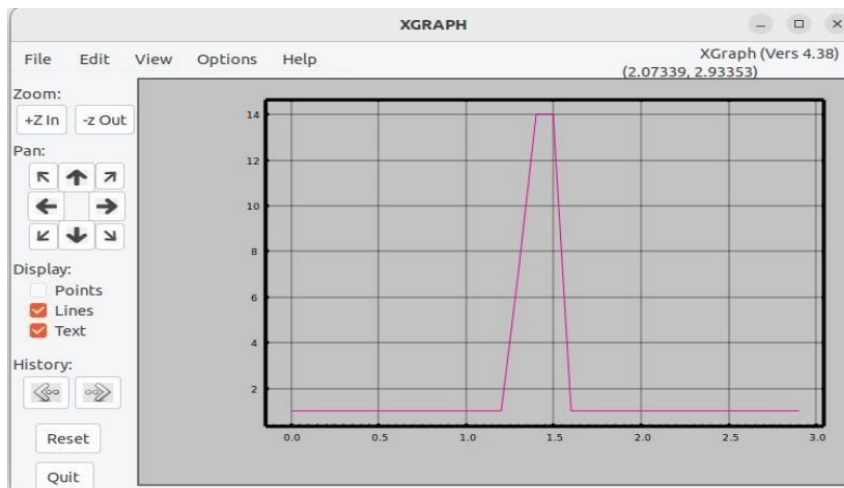
SAMPLE OUTPUT:

TCP.tcl



UDP.tcl





OUTPUT

PRE-ASSESSMENT QUESTIONS & ANSWERS

1. Which of the following is a connection-oriented protocol?
2. Which protocol provides faster data transfer but without guaranteed delivery?
3. TCP ensures reliability by using:
4. UDP is best suited for:
5. In TCP, the congestion control mechanism is primarily handled by:
6. Which of the following fields is not present in the TCP header?
7. UDP header size is:
8. TCP is preferred over UDP for:
9. Which simulation tool is commonly used to compare TCP and UDP performance?
10. In a simulation comparing TCP and UDP, a high packet loss rate will affect:

PRE-LAB WORK:

Design a simple simulation tool to be used (e.g., running simple NS2 scripts)

POST-ASSESSMENT QUESTIONS

1. What is the main difference between TCP and UDP?
2. Why is UDP preferred in real-time applications?
3. Name a common simulation tool used to study TCP/UDP performance.
4. What parameters are typically compared when studying TCP vs UDP performance?
5. How does TCP handle congestion?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, simulating the performance of TCP/UDP using NS2 was executed successfully.

Exp No:9	SIMULATION OF DISTANCE VECTOR/ LINK STATE ROUTING ALGORITHM.
Date:	

PROBLEM STATEMENT:

To simulate the Distance vector / Link State Routing Algorithm using NS2.

ALGORITHM:

LINK STATE ROUTING:

- i. Create a simulator object
- ii. Set routing as dynamic
- iii. Open the trace and nam-trace files
- iv. Define the finish procedure
- v. Create nodes and links between them
- vi. Create the agents and attach them with the nodes
- vii. Create the applications and attach them to the UDP agent
- viii. Connect the UDP and Null Agents
- ix. At 1 sec the link between node 1 and 2 is broken
- x. At 2 sec the link is up again
- xi. Run the simulation

DISTANCE VECTOR ROUTING:

- i. Create a simulator object
- ii. Set routing protocol to Distance Vector routing
- iii. Trace packets on all links onto NAM trace and text trace file
- iv. Define finish procedure to close files, flush tracing and run NAM
- v. Create eight nodes
- vi. Specify the link characteristics between nodes
- vii. Describe their layout topology as a octagon
- viii. Add UDP agent for node n1
- ix. Create CBR traffic on top of UDP and set traffic parameters.
- x. Add a sink agent to node n4
- xi. Connect source and the sink
- xii. 12.Schedule events as follows:
 - a. Start traffic flow at 0.5
 - b. Down the link n3-n4 at 1.0
 - c. Up the link n3-n4 at 2.0
 - d. Stop traffic at 3.0

- e. Call finish procedure at 5.0
- xiii. Start the scheduler
- xiv. Observe the traffic route when link is up and down
- xv. View the simulated events and trace file analyze it
- xvi. Stop

PROGRAM:

LINK STATE ROUTING PROTOCOL:

```

set ns [new Simulator]
$ns rtproto LS
set nf [open linkstate.nam w]
$ns namtrace-all $nf
set f0 [open linkstate.tr w]
$ns trace-all
$f0 proc
finish {} {
global ns f0
nf
$ns flush-
trace close
$f0
close $nf
exec nam linkstate.nam &
exit 0
}

for {set i 0} {$i < 7} {incr
i} { set n($i) [$ns node]
}

for {set i 0} {$i < 7} {incr i} {
$ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
$ns connect $udp0 $null0
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"

```

\$ns at 5.0 "finish"

\$ns run

DISTANCE VECTOR ROUTING ALGORITHM:

#Distance vector routing protocol – distvect.tcl

#Create a simulator object

set ns [new Simulator]

#Use distance vector routing

\$ns rtproto DV

#Open the nam trace file

set nf [open out.nam w]

\$ns namtrace-all \$nf

Open tracefile

set nt [open trace.tr w]

\$ns trace-all \$nt

#Define 'finish' procedure

proc finish { } {

global ns nf

\$ns flush-trace

#Close the trace

file close \$nf

#Execute nam on the trace file

exec nam -a out.nam &

exit 0

}

Create 8 nodes

set n1 [\$ns node]

set n2 [\$ns node]

set n3 [\$ns node]

set n4 [\$ns node]

set n5 [\$ns node]

set n6 [\$ns node]

set n7 [\$ns node]

set n8 [\$ns node]

Specify link characteristics

\$ns duplex-link \$n1 \$n2 1Mb 10ms DropTail

\$ns duplex-link \$n2 \$n3 1Mb 10ms DropTail

\$ns duplex-link \$n3 \$n4 1Mb 10ms DropTail

\$ns duplex-link \$n4 \$n5 1Mb 10ms DropTail

\$ns duplex-link \$n5 \$n6 1Mb 10ms DropTail

\$ns duplex-link \$n6 \$n7 1Mb 10ms DropTail

\$ns duplex-link \$n7 \$n8 1Mb 10ms DropTail

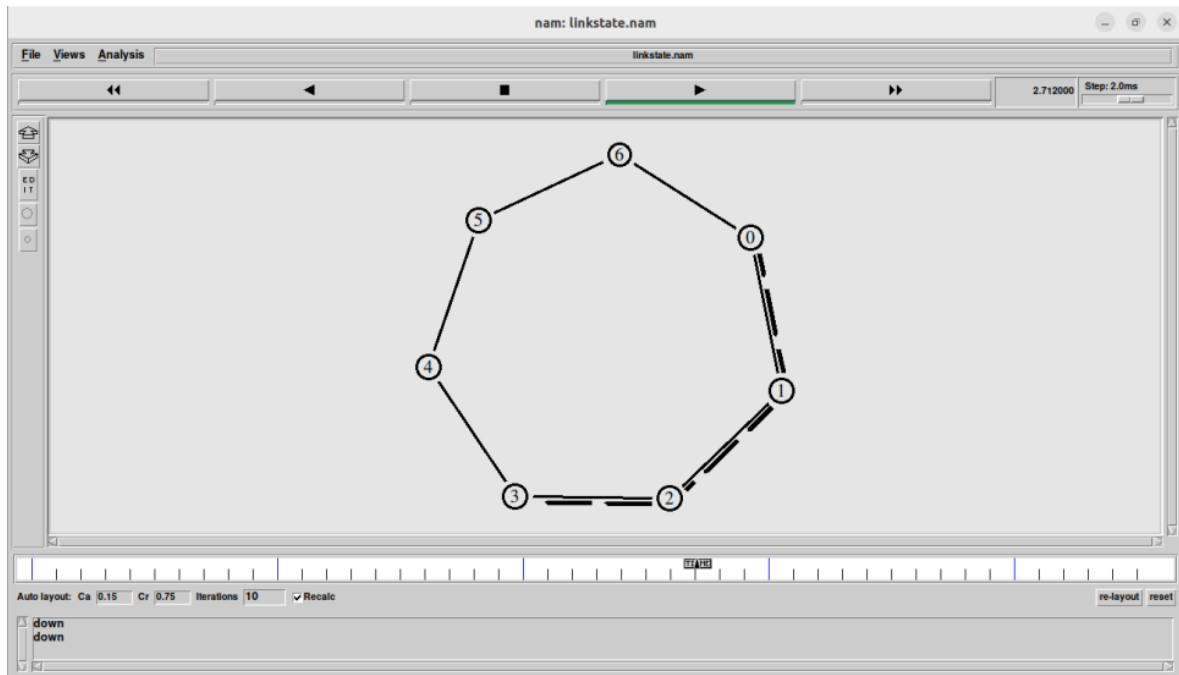
\$ns duplex-link \$n8 \$n1 1Mb 10ms DropTail #

specify layout as a octagon

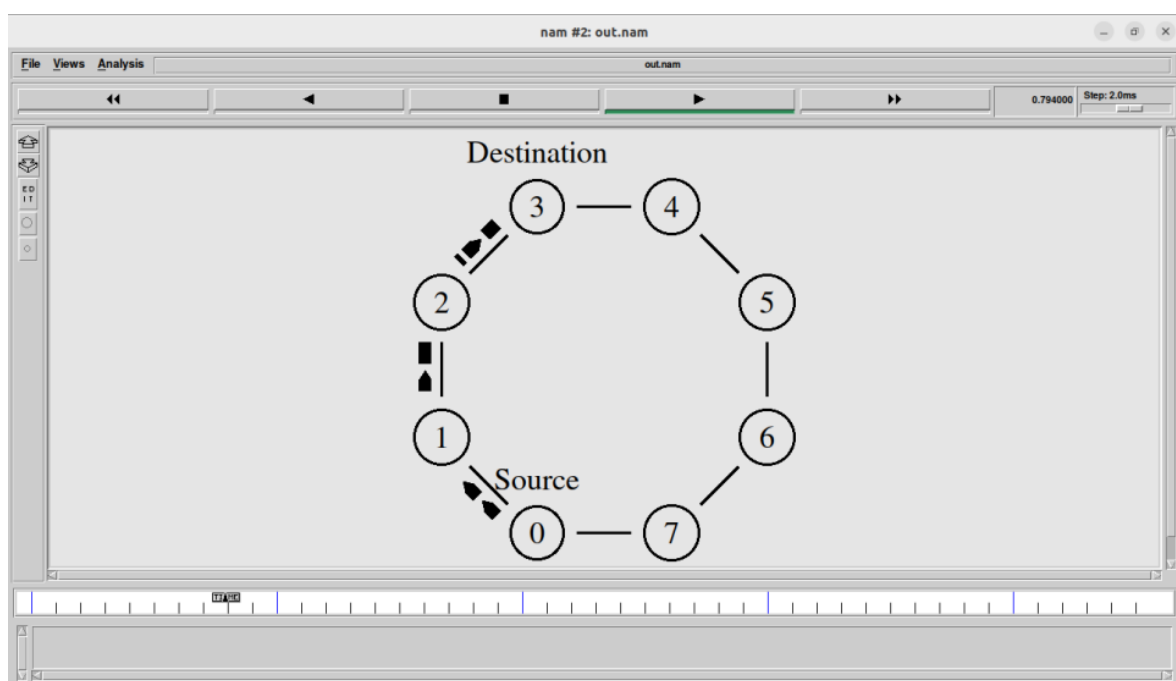
```
$ns duplex-link-op $n1 $n2 orient left-up
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right-down
$ns duplex-link-op $n6 $n7 orient down
$ns duplex-link-op $n7 $n8 orient left-down
$ns duplex-link-op $n8 $n1 orient left #Create a
UDP agent and attach it to node n1 set udp0
[new Agent/UDP]
$ns attach-agent $n1 $udp0
#Create a CBR traffic source and attach it to udp0 set
cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n4 set
null0 [new Agent/Null]
$ns attach-agent $n4 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent and the network dynamics
$ns at 0.0 "$n1 label Source"
$ns at 0.0 "$n4 label Destination"
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n3 $n4
$ns rtmodel-at 2.0 up $n3 $n4
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

SAMPLE OUTPUT:

LINK STATE ROUTING PROTOCOL



DISTANCE VECTOR ROUTING ALGORITHM



OUTPUT

PRE-LAB ASSESSMENT QUESTIONS & ANSWERS

- 1) Which algorithm is commonly used for Distance Vector Routing?
.
- 2) Which algorithm is commonly used for Link State Routing?
- 3) In Distance Vector Routing, what information is shared between routers?
- 4) In Link State Routing, what information is shared between routers?
- 5) What is the main drawback of Distance Vector Routing?
- 6) Which routing method generally converges faster?
- 7) In Link State Routing, which protocol is an example?
- 8) In Distance Vector Routing, which protocol is an example?
- 9) Which type of routing requires more memory and CPU resources?
.
- 10) What is the metric typically used in Distance Vector Routing?

PRE-LAB WORK:

Develop a simple program using Distance Vector and Link State routing algorithms.

POST-LAB ASSESSMENT QUESTIONS:

1. What is the main difference between Distance Vector and Link State routing?
2. Which algorithm does Link State routing use to calculate the shortest path?
3. What problem is associated with Distance Vector routing, and how is it mitigated?
4. Name one real-world protocol that uses Distance Vector routing.
5. Why does Link State routing converge faster than Distance Vector routing?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, the simulation of the Distance Vector / Link State Routing Algorithm using NS2 was executed successfully.

Exp No:10	SIMULATION OF ERROR CORRECTION CODE (LIKE
Date:	CRC)

PROBLEM STATEMENT:

To create a program to perform the simulation of an error correction code (like CRC).

ALGORITHM:

Server.py

- a. Start
- b. Create a socket and start listening for connections
- c. Accept the request and establish a connection with the client
- d. Receive the data from the client
- e. Decode the data with the key and check the remainder
- f. If $\text{rem} == 0 * (\text{len}(\text{key}) - 1)$ then send the acknowledgement to the client that the message is received without errors
- g. Else, send the client that there is some error in the received message.
- h. Stop

Client.py

- a. Start
- b. Create a socket and establish a connection with the server
- c. Get the message from the user
- d. Encode the message with the key and send it to the server
- e. Receive and display the acknowledgement from the server
- f. Stop

PROGRAM: Server.py

```
import socket
def xor(a, b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append("0")
        else:
            result.append("1")
    return "".join(result)

def mod2div(dividend, divisor):
    pick = len(divisor)
    tmp = dividend[0:pick]
```

```

while pick <
len(dividend):
    if tmp[0] == "1":
        tmp = xor(divisor, tmp) + dividend[pick]
    else:
        tmp = xor("0" * pick, tmp) + dividend[pick]
    pick += 1
if tmp[0] == "1":
    tmp = xor(divisor, tmp)
else:
    tmp = xor("0" * pick, tmp)
checkword = tmp
return checkword

def decodeData(data, key):
    l_key = len(key)
    appended_data = data + "0" * (l_key - 1)
    remainder = mod2div(appended_data, key)
    return remainder

s = socket.socket() port =
12345
s.bind(("127.0.0.1", port))
s.listen()
print("Server is listening...")
while True:
    c, addr = s.accept()
    print("Got connection from ", addr)
    data = c.recv(1024).decode()
    print("Received encoded data in Binary format:", data)
    if not data:
        break
    key = "1001"
    ans = decodeData(data, key)
    print("Data after decoding is: ", ans)
    temp = "0" * (len(key) - 1)
    if ans == temp:
        c.sendto(
            ("Thank you, Data " + data + " has been received and NO ERROR
FOUND").encode(),
            ("127.0.0.1", 12345),
        )
    else
        c.sendto(("Error in Data!").encode(), ("127.0.0.1", 12345))
    c.close()

```

```
s.close()
```

```
Client.py import
```

```
socket def xor(a,
```

```
b):
```

```
    result = []
```

```
    for i in range(1, len(b)):
```

```
        if a[i] == b[i]:
```

```
            result.append("0
```

```
        ") else:
```

```
            result.append("1")
```

```
    return "".join(result)
```

```
def mod2div(dividend, divisor):
```

```
    pick = len(divisor)
```

```
    tmp = dividend[0:pick]
```

```
    while pick <
```

```
len(dividend):
```

```
    if tmp[0] == "1":
```

```
        tmp = xor(divisor, tmp) + dividend[pick]
```

```
    else:
```

```
        tmp = xor("0" * pick, tmp) + dividend[pick]
```

```
    pick += 1
```

```
if tmp[0] == "1":
```

```
    tmp = xor(divisor, tmp)
```

```
else:
```

```
    tmp = xor("0" * pick, tmp)
```

```
checksum = tmp
```

```
return checksum
```

```
def encodeData(data, key):
```

```
    l_key = len(key)
```

```
    appended_data = data + "0" * (l_key - 1)
```

```
    remainder = mod2div(appended_data, key)
```

```
codeword = data + remainder
```

```
return codeword
```

```
s = socket.socket()
```

```
port = 12345
```

```
s.connect(("127.0.0.1", port))
```

```
input_string = input("Enter data you want to send->")
```

```
data = "".join((format(ord(x), "b") for x in input_string)) print("Entered
```

```
data in binary format :", data)
```

```
key = "1001"
```

```
ans = encodeData(data, key)
```

```
print("Encoded data to be sent to server in binary format :", ans) s.sendto(ans.encode(),
```

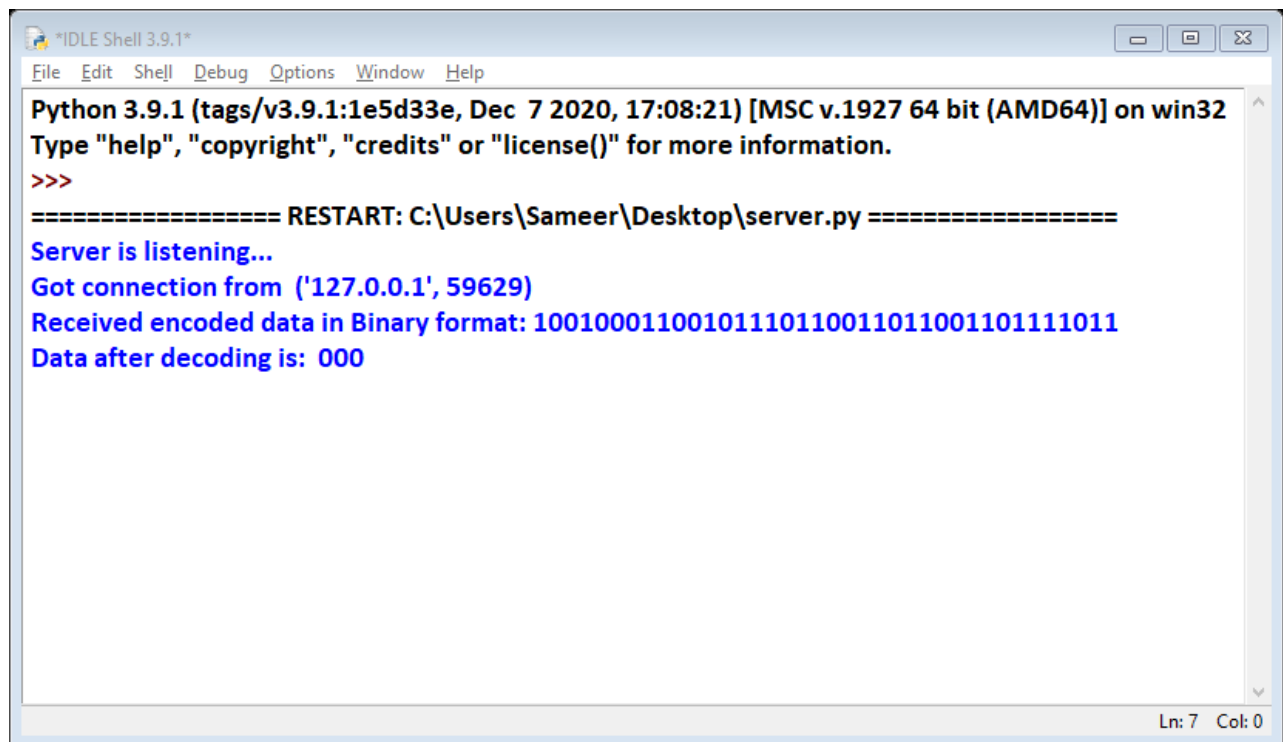
```
("127.0.0.1", 12345))
```

```
print("Received feedback from server :", s.recv(1024).decode())
```

```
s.close()
```

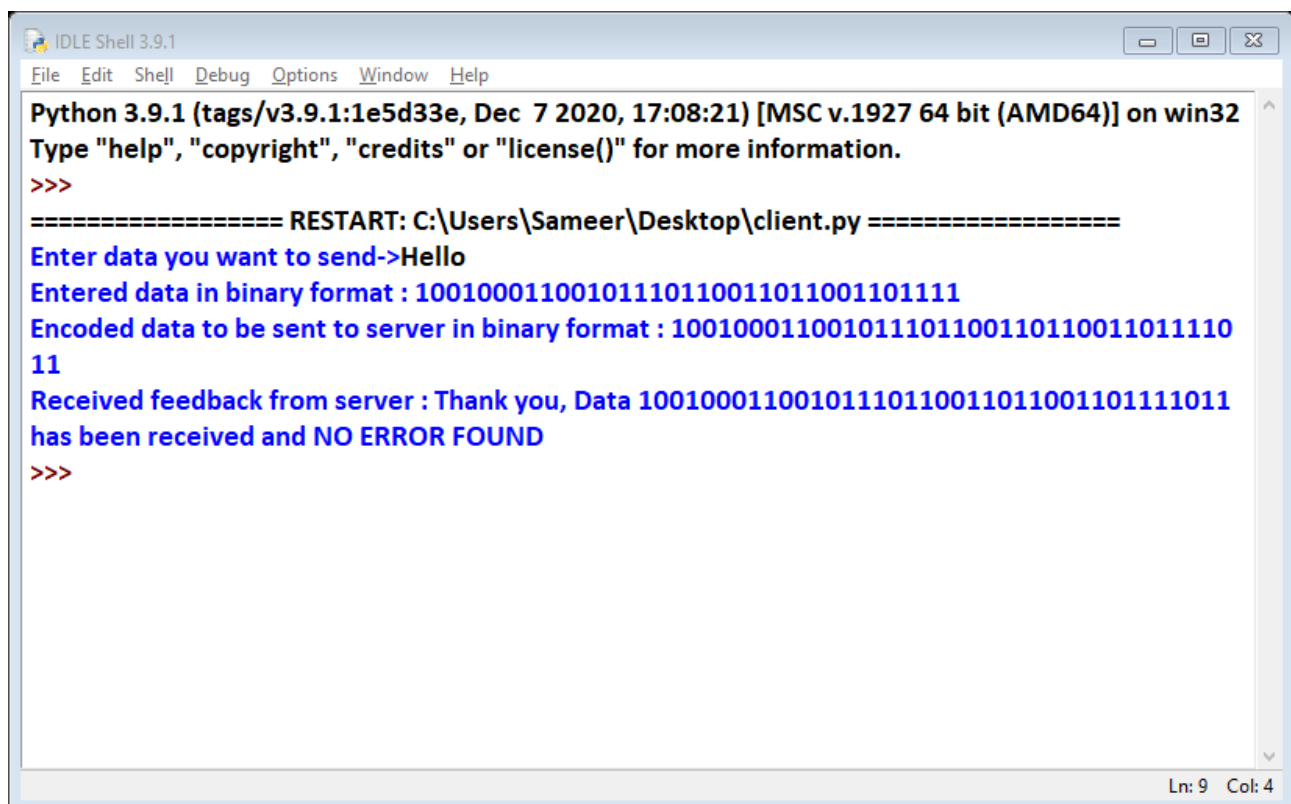

SAMPLE OUTPUT:

Server.py



```
*IDLE Shell 3.9.1*
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\server.py =====
Server is listening...
Got connection from ('127.0.0.1', 59629)
Received encoded data in Binary format: 1001000110010111011001101100110111011
Data after decoding is: 000
Ln: 7 Col: 0
```

Client.py



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Sameer\Desktop\client.py =====
Enter data you want to send->Hello
Entered data in binary format : 1001000110010111011001101100110111
Encoded data to be sent to server in binary format : 100100011001011101100110110011011110
11
Received feedback from server : Thank you, Data 1001000110010111011001101100110111011
has been received and NO ERROR FOUND
>>>
Ln: 9 Col: 4
```

OUTPUT

PRE-ASSESSMENT QUESTIONS AND ANSWERS:

1. What does CRC stand for?
2. CRC is primarily used for:
3. CRC uses which mathematical operation?
4. In CRC, the divisor is also called:
5. What is appended to the data before performing CRC computation?
6. The output of CRC calculation is called:
7. If the CRC remainder at the receiver is all zeros, it means:
8. Which layer of the OSI model typically uses CRC for error detection?
9. CRC is better than simple parity because it can:
10. Which of the following is NOT a common CRC polynomial?

PRE-LAB WORK:

Develop a program to perform error detection in a network

POST-ASSESSMENT QUESTIONS:

1. What is the primary purpose of CRC in data communication?
2. Which mathematical method is used to compute CRC?
3. What is the generator polynomial in CRC?
4. Can CRC correct errors or only detect them?
5. Why is CRC preferred over simple parity checking?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, the simulation of an error correction code (like CRC) was executed successfully.

Exp No:11	IMPLEMENTING SIMPLE FILE TRANSFER USING
Date:	FTP (BEYOND SYLLABUS)

PROBLEM STATEMENT:

To develop a simple, user-friendly file transfer system using FTP.

ALGORITHM:

Client.py:

- a. Start the Client Program.
- b. Connect to the FTP Server using IP and port.
- c. Authenticate the user (username/password).
- d. Display Menu with options:
 - List files.
 - Upload a file.
 - Download a file.
 - Delete a file.
 - Exit
- e. Process User Choice:
 - List: Request and display files from the server.
 - Upload: Send the selected file to the server.
 - Download: Request and save a file locally.
 - Delete: Remove a file on the server.
- f. Repeat or Exit.
- g. End Program.

Server.py:

- a. Start the Server Program.
- b. Wait for Client Connections.
- c. Authenticate the Client with username and password.
- d. Process Client Requests:
 - i. List: Send the list of files.

- ii. Upload: Receive and save the file.
- iii. Download: Send the requested file.
- iv. Delete: Remove the specified file.
- e. Handle Multiple Clients or exit.
- f. End Program.

PROGRAM:

Client.py:

```
from ftplib import FTP

# Function to connect to the server

def connect_to_server(server_address, username, password): ftp =
    FTP(server_address)
    ftp.login(user=username, passwd=password)
    print("Connected to the server.")
    return ftp

# Function to list files on the server

def list_files(ftp):
    print("Files on server:")
    ftp.retrlines('LIST')

# Function to upload a file to the server

def upload_file(ftp, file_path):
    with open(file_path, 'rb') as file:
        ftp.storbinary(f"STOR {file_path}", file)
    print(f"Uploaded: {file_path}")

# Function to download a file from the server def
download_file(ftp, file_name, save_as):
    with open(save_as, 'wb') as file:
        ftp.retrbinary(f"RETR {file_name}", file.write)
    print(f"Downloaded: {file_name} as {save_as}")
```



```
# Function to delete a file on the server

def delete_file(ftp, file_name):

    ftp.delete(file_name)

    print(f"Deleted: {file_name}")


# Main function for the client program

def client_program():

    server_address = input("Enter server address: ")

    username = input("Enter username: ") password
    = input("Enter password: ")


# Connect to the server

ftp = connect_to_server(server_address, username, password)
while True:

    # Display the menu options

    print("\nMenu:")

    print("1. List files")

    print("2. Upload file")

    print("3. Download
file") print("4. Delete
file") print("5. Exit")

    choice = input("Enter your choice: ")


# Process the user's

choice if choice == '1':

    list_files(ftp)

elif choice == '2':

    file_path = input("Enter the path of the file to upload: ") upload_file(ftp,
file_path)

elif choice == '3':

    file_name = input("Enter the name of the file to download: ")

    save_as = input("Enter the name to save the file as: ")
```

```

        download_file(ftp, file_name, save_as)
    elif choice == '4':
        file_name = input("Enter the name of the file to delete: ")
        delete_file(ftp, file_name)
    elif choice == '5':
        print("Goodbye!")
        ftp.quit()
        break
    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    client_program()

```

Server.py:

```

pip install pyftplib

from pyftplib.authorizers import DummyAuthorizer
from pyftplib.handlers import FTPHandler
from pyftplib.servers import FTPServer

# Function to start the FTP server
def start_ftp_server():
    authorizer = DummyAuthorizer()

    # Define a user with full permissions
    authorizer.add_user("user", "12345", ".", perm="elradfmw") # e = List files, f
    # Retrieve files, m = Create directories, w = Write

    # Define the anonymous user (optional)
    authorizer.add_anonymous(".")

```

```
handler = FTPHandler
```

```
handler.authorizer = authorizer
```

```
# Create and start the FTP server
```

```
server = FTPServer(("127.0.0.1", 21), handler)
```

```
print("FTP Server started at 127.0.0.1:21")
```

```
server.serve_forever()
```

```
if __name__ == "
```

```
__main__":
```

```
start_ftp_server()
```

SAMPLE OUTPUT:

Client.py:

```
Enter server address: 127.0.0.1
Enter username: user
Enter password: 12345
Connected to the server.
```

Server.py:

```
FTP Server started at 127.0.0.1:21

127.0.0.1: Connected.
127.0.0.1: Login successful as user.
127.0.0.1: Command 'LIST' received.
127.0.0.1: Command 'STOR' received for file 'localfile.txt'.
127.0.0.1: File 'localfile.txt' uploaded.
127.0.0.1: Command 'RETR' received for file 'testfile.txt'.
127.0.0.1: File 'testfile.txt' sent to client.
127.0.0.1: Command 'DELE' received for file 'example.pdf'.
127.0.0.1: File 'example.pdf' deleted.
127.0.0.1: Client disconnected.
```

OUTPUT

PRE-LAB ASSESSMENT QUESTIONS WITH ANSWERS:

1. Which protocol is primarily used for file transfer over a network in FTP?
2. What does FTP use the default control port number?
3. In FTP, which port is typically used for data transfer in active mode?
4. Which of the following is a secure alternative to FTP?
5. In FTP, which mode requires the client to open a data port and wait for the server to connect?
6. Which FTP command is used to upload a file from client to server?
7. Which FTP command is used to download a file from server to client?
8. Which of the following is *not* a valid FTP transfer mode?
9. In FTP, the command USER is used for:
10. Which Python library is commonly used for implementing an FTP client?

PRE-LAB WORK

Developing a Basic and User-Friendly FTP Client for File Transfer

POST-LAB ASSESSMENT QUESTIONS:

1. What is the primary function of a DNS resolver?
2. Which transport protocol is used by DNS by default, and why?
3. What is the purpose of the TC (Truncated) flag in a DNS response?
4. How is a domain name encoded in a DNS query packet?
5. Which Python socket method is used to send a UDP DNS query to a server?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, a user-friendly file transfer system using FTP was executed successfully.

Exp No:12	IMPLEMENTING DNS RESOLVER USING SOCKETS.
Date:	(BEYOND SYLLABUS)

PROBLEM STATEMENT:

To develop a DNS resolver using sockets that translates human-readable domain names into IP addresses by implementing the DNS protocol.

ALGORITHM:

Client.py:

1. Initialize the Socket for UDP.
2. Get Domain Name from the user.
3. Construct DNS Query packet.
4. Send a Query to a DNS server (e.g., 8.8.8.8).
5. Receive Response from the DNS server.
6. Parse the Response to extract the IP address.
7. Display the IP Address or an error message.
8. End Program.

Server.py

1. Initialize the Socket for UDP on port 53.
 2. Listen for Queries from clients.
 3. Receive DNS Query packet.
 4. Check Domain Name in the database:
 - If found, prepare a response with the IP address.
 - If not found, prepare an error response.
 5. Send the Response back to the client.
- Repeat for new queries.

PROGRAM:

Client.py:

```
import socket

import struct

def build_dns_query(domain):
    # Create a DNS query packet
    transaction_id = 1 # Arbitrary
    ID flags = 0x0100 # Standard
    query questions = 1
    query_type = 1 # A record
    query_class = 1 # IN (Internet)

    # Encode domain name
    domain_parts= domain.split('.')

    query_name = b''.join(struct.pack('B', len(part)) + part.encode() for part in
domain_parts) + b'\x00'

    # Build the DNS query packet
    packet = struct.pack('>HHHHHH', transaction_id, flags, questions,
    query_name, query_type, query_class)
    return packet

def dns_resolver(domain):
# Define the DNS server to query dns_server = '8.8.8.8'
    port = 53

    # Create a UDP socket
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as sock:

        query = build_dns_query(domain)
        sock.sendto(query, (dns_server, port))
```

```

# Receive the response

response, _ = sock.recvfrom(512) # Buffer size 512 bytes

# Parse the response

transaction_id, flags, questions, answer_rrs, authority_rrs, additional_rrs
= struct.unpack('>HHHHHH', response[:12])

# Extract the answer section

answer_offset = 12 + len(domain) + 5 # 12 bytes header + domain length
+ query type and class (2 bytes each)

ip_address = response[answer_offset:answer_offset + 4]

# Convert IP address to human-readable format

return socket.inet_ntoa(ip_address)

if __name__ == '__main__':

    domain = input("Enter a domain name (e.g., www.example.com): ")

    try:

        ip_address = dns_resolver(domain)
        print(f"The IP address for {domain} is: {ip_address}")

    except Exception as e:

        print(f"Error: {e}")

```

Server.py

```

import socket

import struct

def start_dns_server():

    # Create a UDP socket

    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:

        server_socket.bind(('0.0.0.0', 53)) # Bind to all interfaces on port 53

        print("DNS Server is running...")

```

```

while True:

    # Receive a DNS query

    query, addr = server_socket.recvfrom(512)

    print("Received query from:", addr)


    # Parse the DNS query

    transaction_id, flags, questions, answer_rrs, authority_rrs, additional_rrs
= struct.unpack('>HHHHHH', query[:12])

    domain = query[12:-4] # Domain name


    # Prepare a mock response (hardcoded for example purposes)

    ip_address = b'\x7f\x00\x00\x01' # 127.0.0.1 (localhost)

    response = struct.pack('>HHHHHH', transaction_id, 0x8000, 1, 1, 0, 0) +
domain + b'\x00' + struct.pack('>HH', 1, 1) + ip_address


    # Send the response back to the client

    server_socket.sendto(response, addr)


if __name__ == '
__main__':
    start_dns_server()

```

SAMPLE OUTPUT:

Server Console:

Arduino

```
DNS Server is running...  
Received query from: ('127.0.0.1', 54321)
```

Client Console:

```
Enter a domain name (e.g., www.example.com): www.example.com  
The IP address for www.example.com is: 127.0.0.1
```

OUTPUT:

PRE-LAB ASSESSMENT QUESTIONS WITH ANSWERS:

1. Which transport protocol is most commonly used for DNS queries?
2. On which port number do standard DNS servers listen?
3. In DNS, what does the 'A' record represent?
4. Which flag in the DNS header indicates that the response was truncated?
5. What is the purpose of using TCP instead of UDP in DNS queries?
6. Which function in Python's socket library is used to send data to a specific address?
7. In DNS message format, how is a domain name represented?
8. What does the Recursion Desired (RD) flag in a DNS query indicate?
9. Which DNS record type is used to store mail server information?
10. In a socket-based DNS resolver, which method is used to convert a binary IPv4 address into a human-readable string?

PRE-LAB WORK

Design and Development of a Socket-Based DNS Query System

POST-LAB ASSESSMENT QUESTIONS

1. What is the primary function of a DNS resolver?
2. Which transport protocol is used by DNS by default, and why?
3. What is the purpose of the TC (Truncated) flag in a DNS response?
4. How is a domain name encoded in a DNS query packet?
5. Which Python socket method is used to send a UDP DNS query to a server?

Pre-lab assessment	(A)	10	
Pre-lab work	(B)	20	
Conduct of Experiment	(C)	20	
Data observation	(D)	20	
Analysis and Interpretation	(E)	20	
Post-lab assessment/Viva Voce	(F)	10	
Total (=A+B+C+D+E+F)		100	

RESULT:

Thus, a DNS resolver using sockets that translates human-readable domain names into IP addresses was executed successfully.