
















I N D E X

NAME: A. Kavin STD: 11th year SEC: CSE-B ROLL NO: 220704122

S.No.	Date	Title	Page No.	Teacher's Sign/Remarks
1.	16/7/24	Study of various network commands used in linux & windows		
2.	23/7/24	Study of network cables		
3.	30/7/24	Experiments of CISCO PACKET TRACER (simulation tools)		
4.	6/8/24	Setup and configure a LAN using a switch and Ethernet cable		
5.	9/8/24	Experiments on packet capturing tool; Wireshark		
6.	16/8/24	Error correction at data link layer (Hamming code)		
7.	23/8/24	Flow control at data link layer (Sliding window protocol)		
8.	10/9/24	Stimulate virtual LAN		
		CISCO Packet Tracer		
9.	30/9/24	Implementation of subnetting in CISCO Packet tracer		
10.	4/10/24	Internetworking using router DHCP server and internet cloud		
11.	8/10/24	Stimulate static routing Protocol		
		Configuration using CISCO Packet & RIP		
12.	15/10/24	echo client TCP/UDP sockets chat client server TCP/UDP		
13.	22/10/24	write own Ping Problem		
14.	25/10/24	Raw sockets to implement		
		Packet Sniffing		
15.	29/10/24	weblizer tool		

Completed

Exp. no : 12

Practical - 12

Date : 15/10/24

Aim

a) Implement echo client server using TCP/UDP sockets

client

```
import socket
```

```
import time
```

```
def ping_server(host='127.0.0.1', port=12345)
```

```
    with socket.socket(socket.AF_INET)
```

```
        socket.SOCK_DGRAM) as s:
```

```
    try
```

```
        s.sendto(b"Hello", (host, port))
```

```
    except
```

```
        print("Request timed out")
```

```
if __name__ == "__main__":
```

```
    ping_server()
```

Server:

```
import socket
```

```
def start_server(host='127.0.0.1', port=12345)
```

```
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
```

```
        s.bind((host, port))
```

```
        print(f"UDP server running on {host}")
```

```
    while True:
```

```
        data, addr = s.recvfrom(1024)
```

```
        print(f"Received message from {addr}: {data.decode()}")
```

```
if __name__ == "__main__":
```

```
    start_server()
```

Output

Python server.py

UDP server running on 127.0.0.1, 12345

Received message from (127.0.0.1, 59290) Hello

python client.py

Received reply from server: Hello, client

b) Implement chat client server using TCP/UDP sockets.

chat-server.py

```
import socket
```

```
def server():
```

```
    port = 12345
```

```
    host = '127.0.0.1', 59290) Hello
```

python client.py

Received reply from server: Hello, client

with socket. socket(socket.AF_INET, socket.

SOCK_DGRAM) as s

s.bind((host, port))

while(True)

d: add = s.recvfrom(1024)

print("client", {d.decode()})

a = input("Enter reply")

s.sendto(a.encode(), add)

if (a == "end")

break

exit

server()

recvr2.py

```
import socket
```

```
import time
```

```
def recvr2(a)
```

```
    host = '127.0.0.1'
```

```
    port = 12345
```

```
    with socket.socket(socket.AF_INET,  
        socket.SOCK_DGRAM)
```

```
    s: sendto(a.encode(), (host, port))
```

```
    d: add = s.recvfrom(1024)
```

```
    print({d.decode()})
```


while (True):

a = input("Enter message")

if (a == "end"):

recv2(a)

break

else

recv2(a)

Output

Python chat-srv.py

Client {hi}

Enter reply hello

Client {"How are you?"}

Enter Reply I'm fine

Python .1recv2.py

Enter message hi

{'hello'}

Enter messages How are you

{'I'm fine'}

Enter message

[Signature]

Output

The program is successfully ~~is~~ executed
and output is verified.