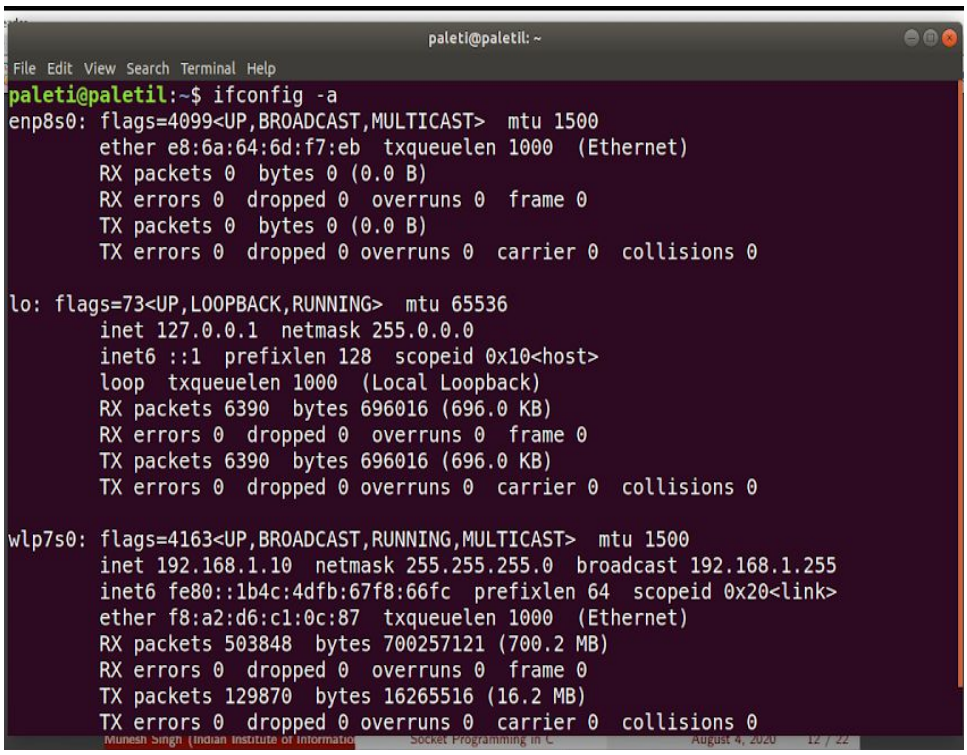
Paleti Krishnasai
CED18I039

COMPUTER NETWORKS PRACTICE
LAB 1

LINUX NETWORKING COMMANDS

1 .) **ifconfig -a** : checks ip address, MAC address , MTU (maximum transmission unit)



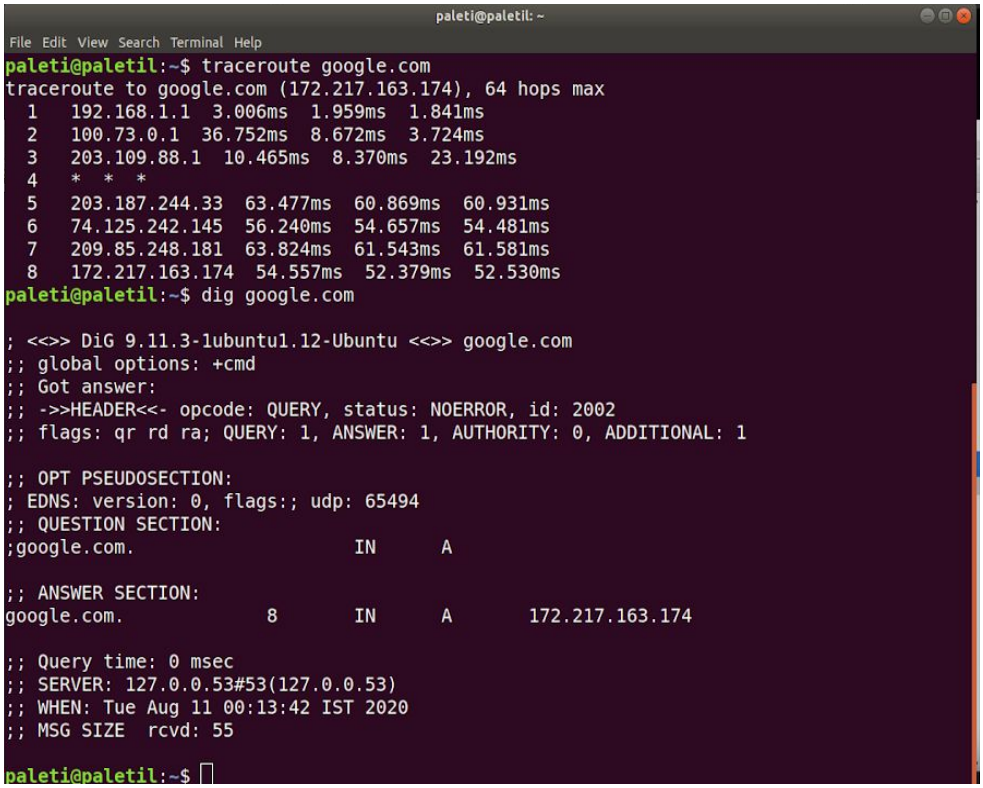
2.) **traceroute** : displays the number of hops or travelling path for the desired destination. It also shows the routes , IP addresses and hostnames of routers over a network.

3.) **dig :** Domain Information Groper. It is a utility for examining Domain Name Servers (DNS).

4.) **route** : shows IP routing table

5.) **nslookup :** DNS related queries

6.) **telnet :** Telnet is a network protocol used to virtually access a computer and to provide a two-way, collaborative and text-based communication channel between two machines. It follows a user command Transmission Control Protocol/Internet Protocol (TCP/IP) networking protocol for creating remote sessions.



```
paleti@paletil: ~
File  Edit  View  Search  Terminal  Help
paleti@paletil:~$ traceroute google.com
traceroute to google.com (172.217.163.174), 64 hops max
  1   192.168.1.1  3.006ms  1.959ms  1.841ms
  2   100.73.0.1  36.752ms  8.672ms  3.724ms
  3   203.109.88.1  10.465ms  8.370ms  23.192ms
  4   *  *  *
  5   203.187.244.33  63.477ms  60.869ms  60.931ms
  6   74.125.242.145  56.240ms  54.657ms  54.481ms
  7   209.85.248.181  63.824ms  61.543ms  61.581ms
  8   172.217.163.174  54.557ms  52.379ms  52.530ms
paleti@paletil:~$ dig google.com

; <<>> DiG 9.11.3-1ubuntu1.12-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2002
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             8       IN      A       172.217.163.174

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue Aug 11 00:13:42 IST 2020
;; MSG SIZE  rcvd: 55

paleti@paletil:~$
```

```
Activities  Terminal                    Tue 00:17
                                   paleti@paletil: ~
File  Edit  View  Search  Terminal  Help
paleti@paletil:~$ telnet google.com 443
Trying 172.217.163.174...
Connected to google.com.
Escape character is '^]'.
Connection closed by foreign host.
paleti@paletil:~$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.163.174
Name:   google.com
Address: 2404:6800:4007:80f::200e

paleti@paletil:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 paletil:52854           103.103.197.68:https    ESTABLISHED
tcp        0      0 paletil:39344           maa05s02-in-f14.1:https ESTABLISHED
tcp        0      0 paletil:59606           maa05s13-in-f14.1:https ESTABLISHED
tcp        0      0 paletil:33568           221.42.105.34.bc.:https ESTABLISHED
tcp   3602124      0 paletil:50836           74.125.171.73:https     ESTABLISHED
tcp        0      0 paletil:60124           74.125.24.189:https     ESTABLISHED
tcp        0      0 paletil:58818           maa05s13-in-f14.1:https ESTABLISHED
tcp        0      0 paletil:52668           maa03s31-in-f14.1:https ESTABLISHED
tcp        0      0 paletil:55960           ec2-34-216-82-69.:https ESTABLISHED
tcp        0      0 paletil:44540           45.55.41.223:http       CLOSE_WAIT
tcp        0      0 paletil:58816           maa05s13-in-f14.1:https ESTABLISHED
tcp        0      0 paletil:39846           103.103.196.88:https    ESTABLISHED
tcp        0      0 paletil:47508           maa05s13-in-f14.1e:http ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    39031    /run/user/1000/systemd/notify
unix  2      [ ]         DGRAM                    32903    /run/user/121/systemd/notify
unix  3      [ ]         DGRAM                    2672     /run/systemd/notify
unix  2      [ ]         DGRAM                    2685     /run/systemd/journal/syslog
unix  9      [ ]         DGRAM                    2694     /run/systemd/journal/socket
unix  22     [ ]         DGRAM                    2752     /run/systemd/journal/dev-log
unix  2      [ ]         DGRAM                    28669    /run/wpa_supplicant/wlp7s0
unix  2      [ ]         DGRAM                    34831    /run/wpa_supplicant/p2p-dev-wlp7s0
```

7.) scp allows you to secure copy files to and from another host in the network.

8.) w

00:19:14 up 2:03, 1 user, load average: 1.69, 1.44, 1.37

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

paleti :0 :0 22:17 ?xdm? 2:48 0.01s /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu gnome-session -

```
Activities  Terminal                    Tue 00:24
                                  paleti@paletil: ~
File  Edit  View  Search  Terminal  Help
paleti@paletil:~$ nmap
Nmap 7.60 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
```

9 .) **nmap :** checks the opened ports on the server.

SOCKET API

A socket is an abstraction through which an application may send and receive data.

A socket API is a collection of socket calls that enable us to set up connections to users over the network .

- **int sockid = socket (family , type, protocol) ;**
  This creates a socket/ interface for data transfer.

  sockid: socket descriptor (integer , similar to a file handle)
  family : AF_INET for ipv4
  Type : SOCK_STREAM for TCP/IP , SOCK_DGRAM for UDP.
  Protocol : usually set to 0 , returns -1 in case of failure.

- **Int status = bind(sockid , &addrport , size);**
  Associates and reserves a port for use by the socket(like assigning a telephone number).

  sockid : file descriptor of socket.
  addrport : IP address and port of the machine, usually set to INADDR_ANY as we can take any incoming interface.
  Size : size in bytes of addrport (sizeof(server))

- **Int status = connect (sockid , &foreignAddr, addrlen);**
  The client establishes a connection with the server by calling connect(). (done in TCP/IP)

  sockid : file descriptor of socket.
  foreignAddr : address of passive participant.
  addrlen : sizeof(sockaddr)

- **Int status = accept (sockid, &clientAddr , &addrlen);**
  The server gets a socket for an incoming client connection by calling accept()

  STREAM SOCKETS
  - **int count = send(sockid, msg, msgLen, flags);**
    msg: const void[], message to be transmitted
    msgLen: integer, length of message (in bytes) to transmit
    flags: integer, special options, usually just 0
    count: # bytes transmitted (-1 if error)

  - **int count = recv(sockid, recvBuf, bufLen, flags);**
    recvBuf: void[], stores received bytes
    bufLen: # bytes received
    flags: integer, special options, usually just 0
    count: # bytes received (-1 if error)

  DATAGRAM SOCKETS
  - **int count = sendto(sockid, msg, msgLen, flags,&foreignAddr, addrlen);**
    msg, msgLen, flags, count : same with send()
    foreignAddr: struct sockaddr, address of the destination
    addrLen: sizeof(foreignAddr)

  - **int count = recvfrom(sockid, recvBuf, bufLen,flags, &clientAddr, addrlen);**
    recvBuf, bufLen, flags, count: same with recv()
    clientAddr: struct sockaddr, address of the client
    addrLen: sizeof(clientAddr)

- **status = close(sockid);**
  The socket should be closed after usage.
  closes a connection (for stream socket)
  frees up the port used by the socket

  sockid: the file descriptor (socket being closed)
  status: 0 if successful, -1 if error

UDP _CLIENT_SERVER_CHAT

UDP_CLIENT.C

```
// udp_client.c
// author : paleti krishnasai CED18I039

#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>

int main()
{
    int c_socket;
    char buf[20]="hello server";
    c_socket=socket(AF_INET,SOCK_DGRAM,0);
    struct sockaddr_in client;
    socklen_t *client_len;
    client.sin_family=AF_INET;
    client.sin_port = htons(9009);
    client.sin_addr.s_addr =INADDR_ANY;
    while(1)
      {
       fflush(stdin);
       fgets(buf,20,stdin);

             sendto(c_socket,buf,sizeof(buf),0,(struct
sockaddr*)&client,sizeof(client));
       recvfrom(c_socket,buf,sizeof(buf),0,(struct
sockaddr*)&client,client_len);
       printf("mes from server : %s",buf );
    }
      close(c_socket);
      return 0;
}
```

UDP_SERVER.C

```c
// udp_server.c
// author : Paleti Krishnasai CED18I039

#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>

int main()
{
    int s_socket;
    char buf[20]="hello client";
    s_socket = socket(AF_INET, SOCK_DGRAM, 0);
    struct sockaddr_in server ,client;
    server.sin_family = AF_INET;
    server.sin_port = htons(9009);
    server.sin_addr.s_addr=INADDR_ANY;
    bind(s_socket,(struct sockaddr*)&server,sizeof(server));
    socklen_t client_len;
    client_len=sizeof(client);
    while(1)
    {
       recvfrom(s_socket,buf,sizeof(buf),0,(struct
sockaddr*)&client,&client_len);
       printf("msg from client : %s",buf);
       fflush(stdin);
       fgets(buf,20,stdin);
       sendto(s_socket,buf,sizeof(buf),0,(struct
sockaddr*)&client,sizeof(client));
    }
    close(s_socket);
    return 0;
}
```

OUTPUT :

```
paleti@paletil:~/CN_LAB$ ./udp_client
hi!
mes from server : hello
how are you?
mes from server : im fine , what abou
mes from server : t you?
im good
mes from server : how is the sem goin
mes from server : g
^C
paleti@paletil:~/CN_LAB$ ./udp_client
hello server
mes from server : hi client!
wassup?
mes from server : CN lab work!
wow fun!
mes from server : YES!
```

```
paleti@paletil:~/CN_LAB$ ./udp_server
msg from client : hi!
hello
msg from client : how are you?
im fine , what about you?
msg from client :
msg from client : im good
how is the sem going
msg from client :
^C
paleti@paletil:~/CN_LAB$ ./udp_server
msg from client : hello server
hi client!
msg from client : wassup?
CN lab work!
msg from client : wow fun!
YES!
```