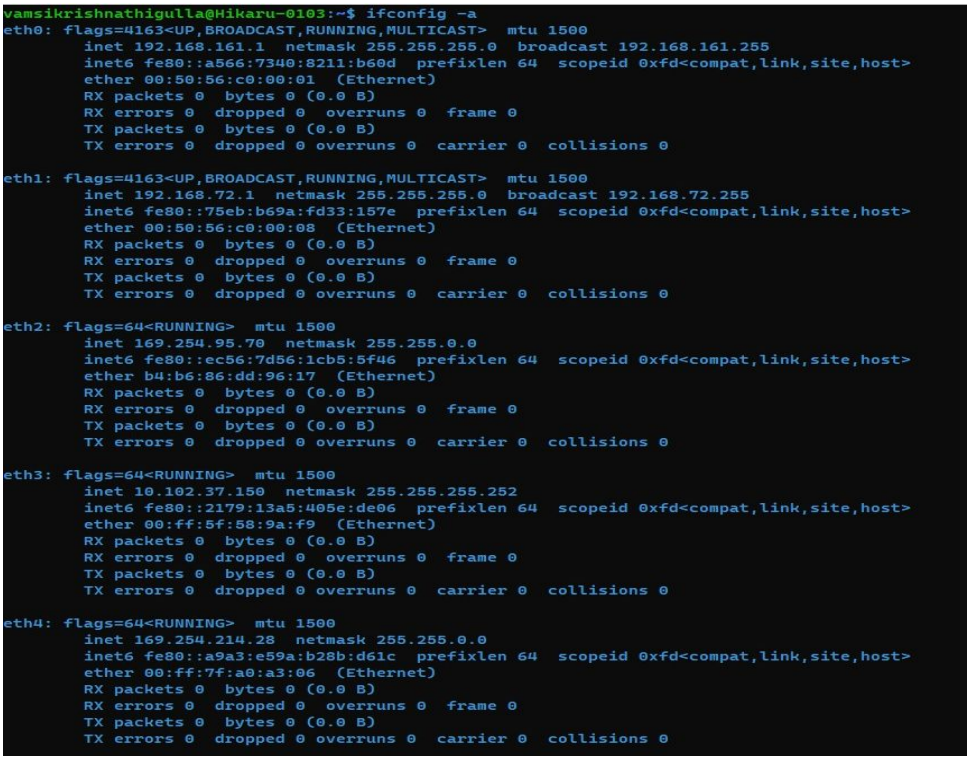
Computer Networking Practice
COM302P
Assignment 1

Thigulla Vamsi Krishna
COE18B056

Commands:

- **ifconfig -a:**
  - This command gives us the configuration of the different ports of the network interface in the device.
  - 
```
vamsikrishnathigulla@Hikaru-0103:~$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.161.1  netmask 255.255.255.0  broadcast 192.168.161.255
        inet6 fe80::a566:7340:8211:b60d  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether 00:50:56:c0:00:01  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.72.1  netmask 255.255.255.0  broadcast 192.168.72.255
        inet6 fe80::75eb:b69a:fd33:157e  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether 00:50:56:c0:00:08  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth2: flags=64<RUNNING>  mtu 1500
        inet 169.254.95.70  netmask 255.255.0.0
        inet6 fe80::ec56:7d56:1cb5:5f46  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether b4:b6:86:dd:96:17  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth3: flags=64<RUNNING>  mtu 1500
        inet 10.102.37.150  netmask 255.255.255.252
        inet6 fe80::2179:13a5:405e:de06  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether 00:ff:5f:58:9a:f9  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth4: flags=64<RUNNING>  mtu 1500
        inet 169.254.214.28  netmask 255.255.0.0
        inet6 fe80::a9a3:e59a:b28b:d61c  prefixlen 64  scopeid 0xfd<compat,link,site,host>
        ether 00:ff:7f:a0:a3:06  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- **Traceroute:**
  - This command measures the speed and the route that data takes to a destination server.
  - It works by sending packets of data to the specified destination address, and records every single intermediate router in the path between the source and the desired destination.

- ```
  Tracing route to google.com [216.58.197.46]
  over a maximum of 30 hops:

    1     5 ms     2 ms     5 ms  192.168.0.1
    2     3 ms     5 ms     8 ms  10.130.128.1
    3    20 ms    19 ms    16 ms  broadband.actcorp.in [183.83.250.1]
    4    36 ms    28 ms    33 ms  72.14.194.18
    5    19 ms    18 ms    18 ms  108.170.253.97
    6    19 ms    19 ms    16 ms  108.170.234.109
    7    77 ms    95 ms    84 ms  maa03s20-in-f46.1e100.net [216.58.197.46]

  Trace complete.
  ```
  traceroute google.com

- **telnet:**
  - A facility for bidirectional interactive text-oriented communication using a virtual terminal connection is provided by telnet.
  - In terms of security, as telnet does not encrypt data by default, SSH is preferred over telnet for security.
  - ```
    vamsikrishnathigulla@Hikaru-0103:~$ telnet google.com 443
    Trying 216.58.197.46...
    Connected to google.com.
    Escape character is '^]'.
    Connection closed by foreign host.
    ```

- **dig:**
  - dig command is used to query for information in the domain that is provided.
  - It is a DNS lookup of the domain that is provided.
  - vamsikrishnathigulla@Hikaru-0103: ~
    ```
    vamsikrishnathigulla@Hikaru-0103:~$ dig google.com

    ; <<>> DiG 9.16.1-Ubuntu <<>> google.com
    ;; global options: +cmd
    ;; Got answer:
    ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2058
    ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

    ;; OPT PSEUDOSECTION:
    ; EDNS: version: 0, flags:; udp: 4096
    ;; QUESTION SECTION:
    ;google.com.                    IN      A

    ;; ANSWER SECTION:
    google.com.             54      IN      A       216.58.197.46

    ;; Query time: 6 msec
    ;; SERVER: 192.168.0.1#53(192.168.0.1)
    ;; WHEN: Sun Aug 09 13:14:53 IST 2020
    ;; MSG SIZE  rcvd: 55
    ```
  - Adding the trace command will allow the dig command to make multiple DNS queries, starting from the root of the domain.

- ```
  vamsikrishnathigulla@Hikaru-0103:~$ dig google.com trace

  ; <<>> DiG 9.16.1-Ubuntu <<>> google.com trace
  ;; global options: +cmd
  ;; Got answer:
  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14857
  ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

  ;; OPT PSEUDOSECTION:
  ; EDNS: version: 0, flags:; udp: 4096
  ;; QUESTION SECTION:
  ;google.com.                    IN      A

  ;; ANSWER SECTION:
  google.com.             76      IN      A       216.58.197.46

  ;; Query time: 92 msec
  ;; SERVER: 192.168.0.1#53(192.168.0.1)
  ;; WHEN: Sun Aug 09 13:19:31 IST 2020
  ;; MSG SIZE  rcvd: 55

  ;; Got answer:
  ;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 51511
  ;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

  ;; OPT PSEUDOSECTION:
  ; EDNS: version: 0, flags:; udp: 4096
  ;; QUESTION SECTION:
  ;trace.                         IN      A

  ;; AUTHORITY SECTION:
  .                       3600    IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2020080900 1800 900 604800 86400

  ;; Query time: 57 msec
  ;; SERVER: 192.168.0.1#53(192.168.0.1)
  ;; WHEN: Sun Aug 09 13:19:31 IST 2020
  ;; MSG SIZE  rcvd: 109
  ```

- **nslookup:**
  - nslookup means ‘name server lookup’.
  - This command is a command-line tool for network administration.
  - It is used to query the domain name for information like the domain name, IP address and is used for address mapping.
  - If a server is not specified, it uses the default server of the device.
  - ```
    vamsikrishnathigulla@Hikaru-0103:~$ nslookup google.com
    Server:         192.168.0.1
    Address:        192.168.0.1#53

    Non-authoritative answer:
    Name:   google.com
    Address: 216.58.197.46
    Name:   google.com
    Address: 2404:6800:4007:807::200e
    ```

- **netstat:**
  - This command shows the current status of all network connections of the device.
  - The fields shown for the network status are the protocol, local address, foreign address and the current state of the network.
  - ```
    vamsikrishnathigulla@Hikaru-0103:~$ netstat
    Active Internet connections (w/o servers)
    Proto Recv-Q Send-Q Local Address           Foreign Address         State
    Active UNIX domain sockets (w/o servers)
    Proto RefCnt Flags       Type       State         I-Node   Path
    ```
    ```

```
C:\Users\VAMSIKRISHNATHIGULLA>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:6000         Hikaru-0103:62202      ESTABLISHED
  TCP    127.0.0.1:6000         Hikaru-0103:62203      ESTABLISHED
  TCP    127.0.0.1:6000         Hikaru-0103:62204      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:49999      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:50118      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58178      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58179      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58182      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58183      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58184      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58189      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58229      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58230      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58244      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58276      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58277      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:58280      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:60551      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:60552      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:60708      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:60724      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:60725      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:60754      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:61544      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:62290      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:64402      ESTABLISHED
  TCP    127.0.0.1:49688        Hikaru-0103:64863      ESTABLISHED
  TCP    127.0.0.1:49712        Hikaru-0103:49713      ESTABLISHED
  TCP    127.0.0.1:49713        Hikaru-0103:49712      ESTABLISHED
  TCP    127.0.0.1:49999        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:50118        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58143        Hikaru-0103:58144      ESTABLISHED
  TCP    127.0.0.1:58144        Hikaru-0103:58143      ESTABLISHED
  TCP    127.0.0.1:58151        Hikaru-0103:58152      ESTABLISHED
  TCP    127.0.0.1:58152        Hikaru-0103:58151      ESTABLISHED
  TCP    127.0.0.1:58178        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58179        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58182        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58183        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58184        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58189        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58229        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58230        Hikaru-0103:49688      ESTABLISHED
  TCP    127.0.0.1:58244        Hikaru-0103:49688      ESTABLISHED
```

```
  TCP    192.168.0.102:49787    maa03s28-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:49807    172.217.194.189:https  ESTABLISHED
  TCP    192.168.0.102:49824    maa05s10-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:49853    maa03s28-in-f3:https   ESTABLISHED
  TCP    192.168.0.102:49866    maa05s05-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:49898    whatsapp-cdn-shv-02-maa2:https  ESTABLISHED
  TCP    192.168.0.102:49957    162.125.19.131:https   ESTABLISHED
  TCP    192.168.0.102:49987    maa03s28-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:50075    maa05s06-in-f10:https  TIME_WAIT
  TCP    192.168.0.102:50081    maa03s28-in-f3:https   TIME_WAIT
  TCP    192.168.0.102:50086    maa05s10-in-f4:https   ESTABLISHED
  TCP    192.168.0.102:50088    maa03s28-in-f10:https  ESTABLISHED
  TCP    192.168.0.102:50089    maa05s05-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:50090    maa05s10-in-f4:https   ESTABLISHED
  TCP    192.168.0.102:50092    maa05s02-in-f1:https   ESTABLISHED
  TCP    192.168.0.102:50100    maa05s06-in-f3:https   TIME_WAIT
  TCP    192.168.0.102:50105    maa03s26-in-f2:https   ESTABLISHED
  TCP    192.168.0.102:50106    maa03s26-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:50107    maa05s01-in-f3:https   TIME_WAIT
  TCP    192.168.0.102:50108    maa03s29-in-f14:https  TIME_WAIT
  TCP    192.168.0.102:50110    maa03s26-in-f22:https  ESTABLISHED
  TCP    192.168.0.102:50111    broadband:https        ESTABLISHED
  TCP    192.168.0.102:50116    180.87.4.163:https     TIME_WAIT
  TCP    192.168.0.102:50117    180.87.4.163:https     TIME_WAIT
  TCP    192.168.0.102:50119    maa03s26-in-f2:https   ESTABLISHED
  TCP    192.168.0.102:50120    maa05s03-in-f3:https   ESTABLISHED
  TCP    192.168.0.102:50121    maa03s31-in-f1:https   ESTABLISHED
  TCP    192.168.0.102:50122    maa05s06-in-f6:https   ESTABLISHED
  TCP    192.168.0.102:50123    maa03s20-in-f46:https  ESTABLISHED
  TCP    192.168.0.102:50124    e2a:https              ESTABLISHED
  TCP    192.168.0.102:50125    180.87.4.163:https     TIME_WAIT
  TCP    192.168.0.102:50126    maa05s05-in-f14:https  ESTABLISHED
  TCP    192.168.0.102:50127    162.125.19.130:https   ESTABLISHED
  TCP    192.168.0.102:50128    52.114.76.35:https     TIME_WAIT
  TCP    192.168.0.102:50129    162.125.81.17:https    ESTABLISHED
  TCP    192.168.0.102:50130    180.87.4.163:https     ESTABLISHED
  TCP    192.168.0.102:50131    180.87.4.163:https     TIME_WAIT
  TCP    192.168.0.102:50132    52.114.76.35:https     TIME_WAIT
  TCP    192.168.0.102:50133    20.189.73.166:https    TIME_WAIT
  TCP    192.168.0.102:50134    whatsapp-cdn-shv-02-maa2:https  ESTABLISHED
  TCP    192.168.0.102:50135    maa03s20-in-f46:https  ESTABLISHED
  TCP    192.168.0.102:50136    66.110.49.68:https     ESTABLISHED
  TCP    192.168.0.102:58040    40.90.189.152:https    ESTABLISHED
  TCP    192.168.0.102:58703    sa-in-f188:5228        ESTABLISHED
  TCP    192.168.0.102:58935    82.202.184.214:https   ESTABLISHED
  TCP    192.168.0.102:58941    82.202.184.214:https   ESTABLISHED
  TCP    192.168.0.102:58969    82.202.184.202:https   ESTABLISHED
  TCP    192.168.0.102:61410    117.18.237.29:http     CLOSE_WAIT
  TCP    192.168.0.102:61715    a104-120-173-121:https  CLOSE_WAIT
  TCP    192.168.0.102:61814    192.168.0.101:8009     ESTABLISHED
```

-
-
- This command shows all the existing connections, including the various web and desktop applications present in the device (Ex: Whatsapp Web), and also shows all the various broadband connections available to the device.

- **ipconfig -all:**
  - Similar to ifconfig for Linux, this command for Windows displays the IP configuration for all the various ports for the device.
  - These ports include various types like:
    i. Host Configuration
    ii. Ethernet Adapters
    iii. Wireless LAN Adapters
    iv. Ethernet ports from Virtual Machines
    v. Ports for anti-virus softwares
    vi. Wireless LAN Wi-Fi port

- ```
  C:\Users\VAMSIKRISHNATHIGULLA>ipconfig -all

  Windows IP Configuration

     Host Name . . . . . . . . . . . . : Hikaru-0103
     Primary Dns Suffix  . . . . . . . :
     Node Type . . . . . . . . . . . . : Mixed
     IP Routing Enabled. . . . . . . . : No
     WINS Proxy Enabled. . . . . . . . : No

  Ethernet adapter Ethernet 2:

     Media State . . . . . . . . . . . : Media disconnected
     Connection-specific DNS Suffix  . :
     Description . . . . . . . . . . . : Realtek PCIe GBE Family Controller
     Physical Address. . . . . . . . . : B4-B6-86-DD-96-17
     DHCP Enabled. . . . . . . . . . . : Yes
     Autoconfiguration Enabled . . . . : Yes

  Wireless LAN adapter Local Area Connection* 9:

     Media State . . . . . . . . . . . : Media disconnected
     Connection-specific DNS Suffix  . :
     Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
     Physical Address. . . . . . . . . : C0-B6-F9-91-AA-CE
     DHCP Enabled. . . . . . . . . . . : Yes
     Autoconfiguration Enabled . . . . : Yes

  Wireless LAN adapter Local Area Connection* 10:

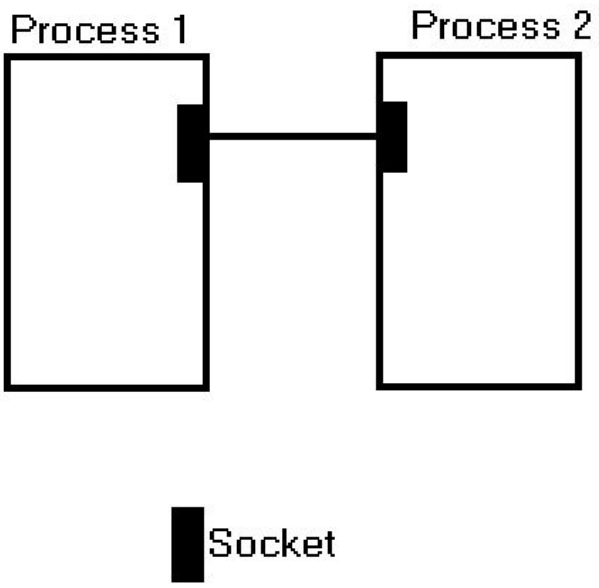
     Media State . . . . . . . . . . . : Media disconnected
     Connection-specific DNS Suffix  . :
     Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
     Physical Address. . . . . . . . . : C2-B6-F9-91-AA-CD
     DHCP Enabled. . . . . . . . . . . : Yes
     Autoconfiguration Enabled . . . . : Yes

  Ethernet adapter VMware Network Adapter VMnet1:

     Connection-specific DNS Suffix  . :
     Description . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet1
     Physical Address. . . . . . . . . : 00-50-56-C0-00-01
     DHCP Enabled. . . . . . . . . . . : No
     Autoconfiguration Enabled . . . . : Yes
     Link-local IPv6 Address . . . . . : fe80::a566:7340:8211:b60d%21(Preferred)
     IPv4 Address. . . . . . . . . . . : 192.168.161.1(Preferred)
     Subnet Mask . . . . . . . . . . . : 255.255.255.0
     Default Gateway . . . . . . . . . :
     DHCPv6 IAID . . . . . . . . . . . : 83906646
     DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-22-D9-65-AB-B4-B6-86-DD-96-17
  ```

Socket API:

- Socket API is an application programming interface for IP networking, and relates to the Transport Layer of OSI.
- This API provides a programming construct called a socket.
  - **Socket:-** the software abstraction used to represent the terminals of a connection between two machines.
- Any process, in order to communicate with another process, creates an instance of a socket.
- Two sockets that communicate must necessarily be of the same type (TCP or UDP).
  ```



-
- Two processes in which a socket is created then perform operations provided by the API to send or receive data.
- Socket types:
    - Stream sockets
    - Datagram sockets
    - Raw sockets
- Socket protocols which we use can be of two types:
    - TCP:
        - Sends data in forms of packets
        - Reliable form of data transmission
        - Data is received in same order it was sent
        - Connection-oriented
        - Bidirectional
        - User has to terminate the connection
        - Can transmit larger amounts of data compared to UDP
        - Notified if data is not delivered
        - sock_stream
    - UDP:
        - Data sent in form of datagrams
        - Data is not received in same order as it was sent
        - After sending data and receiving a reply, the connection is automatically terminated
        - Connectionless
        - No acknowledgement of data received and no notification if data is not received
        - No retransmission possible
        - Limited in size of data transmitted (minimum of 512 bytes)

        - sock_dgram
- Socket protocol types:
    - SOCK_DGRAM:- datagram socket
    - SOCK_STREAM:- stream socket
    - SOCK_RAW:- raw socket
    - SOCK_SEQPACKET:- sequenced packet socket
    - SOCK_RDM:- reliably delivered message socket
- **Types of socket API:**
    - Java datagram socket API
    - Stream mode socket API
- **Operations provided in the socket API:**
    - Create a socket
    - Assign a unique value to the socket (bind a socket to an address)
    - Wait for a connection
    - Initiate a connection
    - Accept an incoming connection
    - Send data
    - Receive data
    - Close the socket
- **Socket creation:**
    - *int socket(int domain, int type, int protocol)*
        - Domain is the IPV4 address
        - Type:
            - sock_stream for TCP sockets
            - sock_dgram for UDP sockets
        - Protocol is by default 0
        - This function will return 1 on successful creation of socket and -1 on encountering an error
- **Socket Binding:**
    - *int Bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen)*
        - sockfd is the description of the socket that is created with the socket() call
        - my_addr is a pointer to a valid sock address that is cast as a struct
        - addrlen is the length of the address in the struct
        - This function is called by the server
        - This function binds a socket to a specific port and IP address
- **Making a connection:**
    - *int Listen(int sock, int backlog)*
        - sock is the socket that is returned by socket()
        - backlog is the maximum length of the queue of pending connections for the socket
        - listen(sock, 5) will allow a maximum of 5 connections pending.
        - The server process is the one that calls this function to tell the kernel to initialize a wait queue of connections

    - *int Accept(int socket, (struct sockaddr *)&client, socklen_t *client_len)*
        - socket is returned by socket()
        - client will hold the new client's information after accept returns
        - Client_len points to size of the client structure
        - Accept is called by the server process to accept new connections from clients trying to connect
- **Data transmission:**
    - send(), recv(), write() are functions called to perform operations for sending and receiving data.
    - *int send(int sock, void *mesg, size_t len, int flags)*
    - *int recv(int sock, void *mesg, size_t len, int flags)*
    - sock is a socket that is connected
    - mesg is the pointer to a buffer through which we send/receive data
    - len represents the length/size of the data to be sent
- **Initialize a connection:**
    - The client calls connect() to make a connection with a server port.
    - *int connect(int sock, (struct sockaddr *)&server_addr, socklen_t len)*
        - sock is the socket that is returned by socket()
        - server_addr is the address of the server port to which the connection is to be made
            - It has all the remote server details and is cast as a sockaddr struct pointer
        - len denotes the size of the server addr struct
- **Close the connection:**
    - *int close(int sock)*
        - sock is the socket that needs to be closed.
    - After the work needed is done, the socket needs to be closed, or else this may create a security problem.

UDP Chat Program:

Server program:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
#include <string.h>

int main()
{
   int socket_chat;
   char buffer[1024] = "";
        socket_chat = socket(AF_INET , SOCK_DGRAM , 0);
        struct sockaddr_in servaddr,cliaddr;

        if (socket_chat == -1)
        {
                printf("Could not create socket");
        }

        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(8080);
        servaddr.sin_addr.s_addr = inet_addr("0");

        bind(socket_chat, (const struct sockaddr *)&servaddr,sizeof(servaddr));

        int len = sizeof(cliaddr);

        while(1)
        {
                printf("Input Ctrl+C to end chat\n");
                int n=recvfrom(socket_chat, (char *)buffer, 1024, MSG_WAITALL, ( struct
sockaddr *) &cliaddr, &len);
                buffer[n]='\0';

                printf("Client : %s", buffer);
                printf("Server : ");

                fflush(stdin);
                fgets(buffer,1024,stdin);

                sendto(socket_chat, (const char *)buffer, strlen(buffer), 0, (const struct sockaddr
*) &cliaddr,sizeof(cliaddr));
        }
        close(socket_chat);
        return 0;
}
```

Client Program:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>

int main()
{
   int socket_chat,len;
   char buffer[1024]="hello";
        socket_chat = socket(AF_INET , SOCK_DGRAM , 0);
        struct sockaddr_in servaddr;

        if (socket_chat == -1)
        {
                printf("Could not create socket");
        }

        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(8080);
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.9");

        while(1)
        {
                printf("Input Ctrl+C to end chat\n");
                printf("Client : ");
                fflush(stdin);
                fgets(buffer,1024,stdin);
                sendto(socket_chat, (const char *)buffer, strlen(buffer), 0, (const struct sockaddr
*) &servaddr,sizeof(servaddr));

                int n=recvfrom(socket_chat, (char *)buffer, 1024, MSG_WAITALL, ( struct
sockaddr *) &servaddr, &len);
                buffer[n]='\0';

                printf("Server : %s", buffer);

        }
```

```
        close(socket_chat);
        return 0;
}
```

Output: