# File Transfer using Socket Programming

# Server Side

- The server accepts connections from the clients to establish a network interface.
- We need to ensure that clients/users are unique.
- Hence we assign a unique IP address to each client.
- However many users prefer to be identified by a username. Hence we will assign usernames as well.
- The role of the server is to collect any incoming messages and deliver them to the other client/clients.
- Let's begin coding the server-side.
- Firstly, create a file named filetcpserver.py(for example).

# Server Socket Creation

- import socket # Import socket module
- port = 60000 # Reserve a port for your service.
- s = socket.socket() # Create a socket object
- host = socket.gethostname() # Get local machine name

The **socket()** function is a constructor of the socket library to create a socket. Once the socket is created, we retrieve the hostname/local device name using the **gethostname()**, which is again a function of the socket library.

# Binding to the port

- s.bind((host, port))

Now we will **bind** the **port** and **host** together using the bind function

# Listen to Client

- s.listen(1)  # Now wait for client connection.
- print ('Server listening....')

Here, we use the **listen()** function which takes in one argument namely number_of_connections.
This parameter can be any whole number such as 1,2,3,…

# Accepting the Connection and Reading the File

- while True:
- conn, addr = s.accept()  # Establish connection with client.
- print ('Got connection from', addr)
- data = conn.recv(1024)
- print('Server received', repr(data))
- filename='Server-Code.txt'
  f = open(filename,'rb')
  l = f.read(1024)

The first variable which is '**conn**' is connected to the socket and the variable '**addr**' is assigned to the IP address of the client.

•The **repr()** function **returns the string representation of the value passed to eval function by default**.

•To open a file in binary format, add 'b' to the mode parameter. Hence **the "rb" mode opens the file in binary format for reading**

# Sending the Acknowledgement and Close the Connection

- while (l):
    conn.send(l)
    print('Sent ',repr(l))
    l = f.read(1024)
  f.close()
  print('Done sending')
  conn.send('Thank you for connecting')
  conn.close()

# Client Side

- Create a file name as filetcpclient.py(for example)

# Importing Libraries

- • import socket
- • import sys
- • import time

# Client Socket Creation

- s = socket.socket()  # Create a socket object
- host = socket.gethostname()

# Get local machine name

- port = 60000  # Reserve a port for your service.
- s.connect((host, port))
- s.send("Hello server!")

# File Writing

- with open('received_file', 'wb') as f:
- print ('file opened')

To open a file in binary format, add 'b' to the mode parameter. Hence **the “wb" mode opens the file in binary format for writing**

# Writing to the File

```
while True:
        print('receiving data...')
        data = s.recv(1024)
        print('data=%s', (data))
        if not data:
            break
        f.write(data) # write data to a file
```

# Close the File and Connection

- f.close()
- print('Successfully get the file')
- s.close()
- print('connection closed')