Delhi Technological University

Department of Computer Engineering

Subject: **Computer Networks**

Synopsis of the project: **Implementation of a chat-based file transfer system using socket programming**

B. TECH (SEMESTER VI)

SUBMITTED BY:

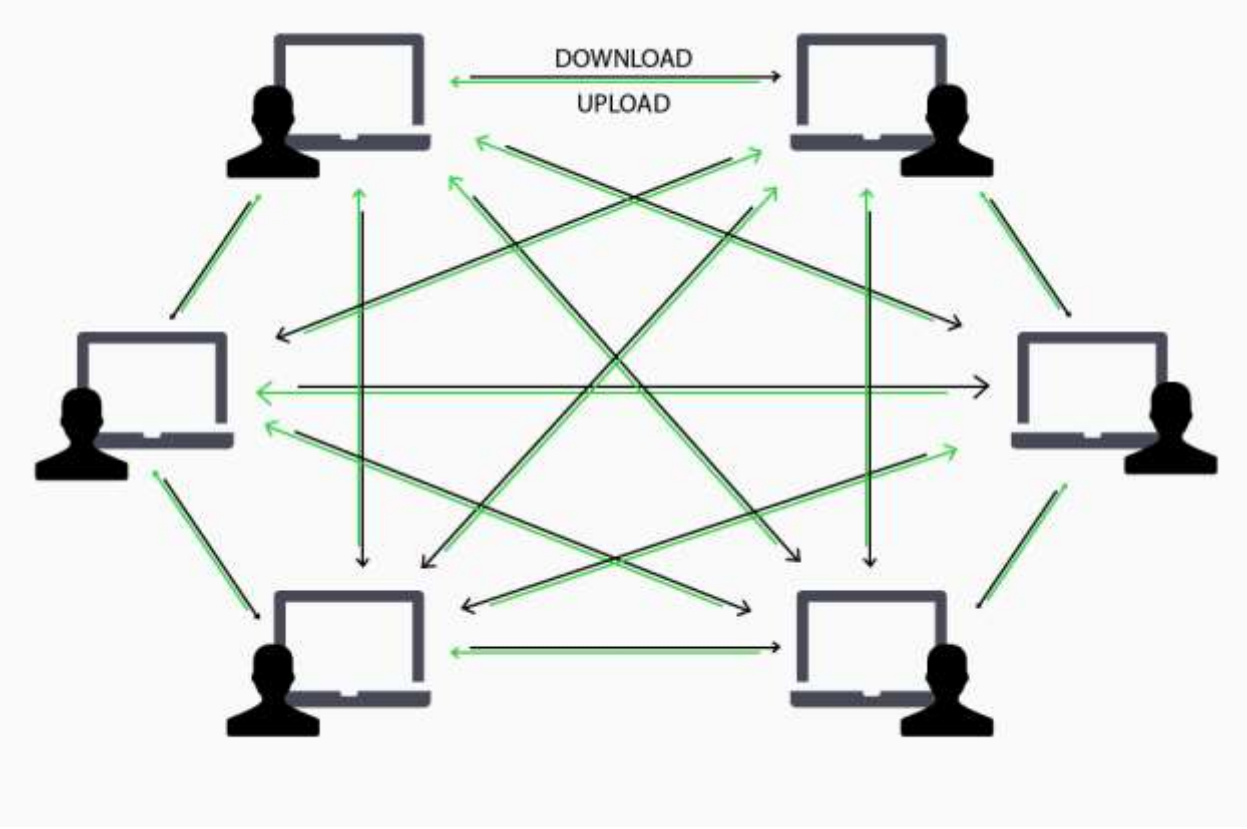
Prithish Kumar Rath (2K18/CO/262)

Nilesh Nishant (2K18/CO/235)

Introduction

We often come across chatrooms where peers come together in a virtual chat environment and exchange messages with each other. Chatrooms provide an easy-to-use interface for peers to communicate with each other. However, the functionality of a chatroom would be much more if the chat can be used by peers to send and receive files from other peers. Thus, a chatroom-based communication system that provides file transfer functionality is the ideal utilization of the chatroom environment.

We use several servers and the peer connects to the closest server which allows it to communicate to the peers connected to the same server or to other servers (by hopping). Also, all peers use a chat room identification code while connecting which is visible only to other peers in the same chat room. User Datagram Protocol (UDP), a transport layer protocol, is used to transfer files so that unnecessary latency during hopping and traffic is avoided. Also, data link layer protocols such as Selective Repeat and Go Back N are used to ensure reliability.



General Representation of a Peer-to-Peer System

Objectives

The primary objective of the project is to create a chatroom-based file transfer system, where peers can come together and pass messages as well as transfer files to each other. A client would connect to a server and then decide whether to enter an existing chatroom or a new one. The server would provide information to the client, such as the peers already in the room, clients who join or leave the chatroom, and the messages being sent in the chatroom. To leverage the file transfer functionality, a client would ask his/ her peers about who has a particular file, and the clients with the requested file can respond so that the requester can select a peer to obtain the file from. File transfer is done by UDP that does not involve the server, and Go Back N is used at the same time for reliability. Thus, the objectives include-

- Simulating a chatroom with multiple servers and multiple clients.
- Informing every peer whenever another peer enters or leaves the chatroom.
- Allow peers to take requests to transfer files to another peer
- Allow peers to refuse connections when it is already uploading a file at its maximum capacity.
- Allow peers to choose the next peer when a client refuses to entertain a request.

Theory of Socket Programming

To access the network services, python gives two degrees of approach. At a seemingly low level, one may approach the basic socket support in the fundamental operating system, which authorizes one to put both the client and servers for both connection-oriented and connectionless protocols. The termination of a bidirectional communications channel are sockets and they may transmit with a process, between the same machine or different continents.
The *socket* library provides unique classes for grasping common transports and a generic interface for grasping the due.

A course of action of interjoining two nodes on a network to convey messages with each other is called socket programming. One node understands a particular port at an IP and co-operates to merge with the other to form a connection. This is how web browsing works.

A socket is created by importing the socket library and running the command below:

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Here, AF_INET points to the address family IPv4. The SOCK_STREAM means the connection-oriented TCP protocol.

There are many common socket methods:

- s. recv()- it receives the TCP message
- s.send()- it transmits the TCP message
- s.recvfrom()- it receives the UDP message
- s.sendto()- it transmits the UDP message
- s.close()- the socket is closed
- s.gethostname()- it returns the host name
- s.connect()- it starts the TCP server connection (for client only)
- s.bind()- it binds address (hostname, port number etc.) to socket (for server only)
- s.listen()- it sets up and initiates TCP listener (for server only)
- s.accept()-it accepts TCP client connection, waits for the connection to arrives (for server only)