CSC/ECE 573 – Internet Protocols
Course Project Proposal
Fall 2016

Chat Room Based File Transfer

Submitted on 14 October 2016,
by

amaleth • Aparna Maleth • 200153053
aaugust4 • Arjun Augustine • 200112749
dfermi • Danis Fermi • 200151858
ekokkil • Eswara Rao Kokkiligadda • 200109998

Introduction

There are implementations of chat rooms where peers who know each other come together in a single virtual chat room and exchange messages. The chat rooms offer a simple interface to log on to the chat room and communicate with one another. But it is much more beneficial if the peers can leverage the same chat interface to request for and receive files from other peers.

Problem Statement

We propose a chat room based communication and file transfer system. There are several servers, that gives the peer the capability to connect to the closest server, and communicate with peer connected to the same server or to other servers using multi-hopping. Additionally, peers mention a chat room ID while connecting, making it visible only to others peers in the same chat room. We propose a small set of commands which the peers can use to mutually communicate and request for files. We plan to use UDP to transfer files, so that the latency for multi hop communication and unnecessary handshake traffic is avoided. The application is responsible for handling reliability and performance by implementing the Selective Repeat or the Go-Back-N algorithms.

Project Objectives

1. Server informs every other peer, of any peer that joins/leaves a chat room.
2. To successfully simulate a chat room with multiple servers and many clients.
3. To implement chat based file transfer where
    a. A client, C sends a message targeted ‘@all’ peers in the chat room or a message targeted ‘@<hostname>’ requesting for a particular file.
    b. Server, S either broadcasts the ‘@all’ message or unicasts the ‘hostname’ asking for the file and routes replies back to the requester.
    c. The client C will select a peer amongst the list of replies and requests the file from that peer.
4. A peer can take incoming requests while receiving a file from another peer
5. A peer can refuse a connection when it reaches its max upload connections.
6. A peer chooses a next suitable peer and requests for a file when a client refuses a connection.

How do you plan to carry out the project

We plan to use the existing socket programming libraries in Python to implement the project. We have separate programs for the client and the server. We will be executing various scenarios from the client end. Some error generation mechanisms are required so that we can test the reliability of the file transfer and chat system.

How will you evaluate your project

The project will be evaluated based on the implementation of the following features:-

1. Test the functionality of chat system
   a. ping a server and get a response
   b. send requests to a peer identifying it with an IP address and get a response
   c. broadcast a request to all peers in the network using @all tag
2. Download a file from the network.
   a. Flood the network with a request for a file
   b. Connect with a peer who gives a positive response and measure throughput of file download.
3. Stress the network with a lot of traffic and see if it breaks (test robustness)
   a. Generate unique file transfer requests such that each peer’s upload link is exhausted. Measure the time taken and hence the throughput
4. Vary the traffic between the two extremes in cases 2 and 3 and plot throughput.
5. Test the setup from different internal networks and confirm they work.

What exactly will be shown during the demo

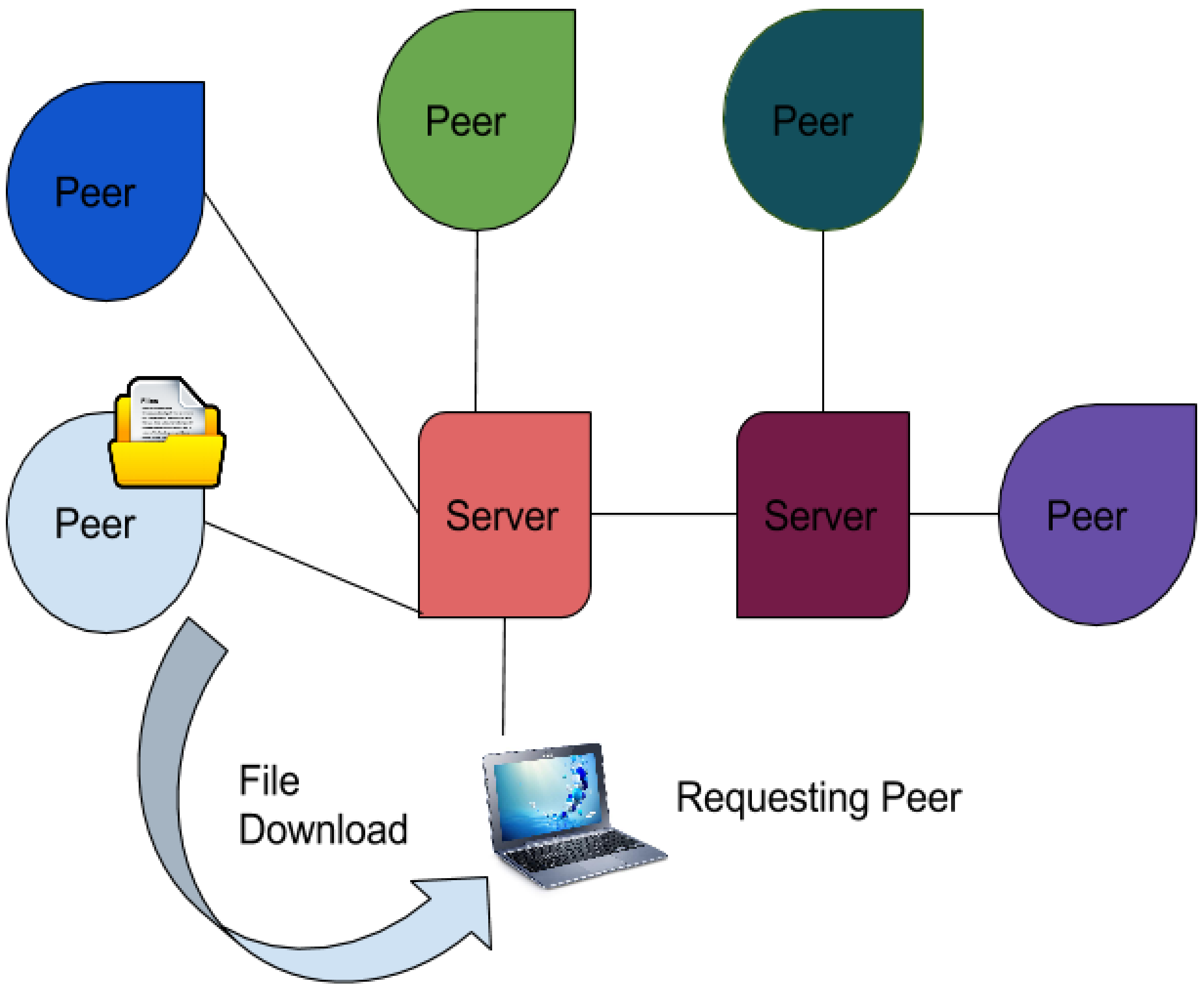
Demo setup will be as follows:-

a) The Server: The server will be a terminal running a python program (server.py) running on a laptop, labelled as the server. Our demo will start by running this code by calling it from the command line. There can be multiple instances of server running on different machines.

b) Now, we simulate some peers who connect to the server. For this, we open multiple command line instances from multiple machines, each running an instance of the python program named client.py.

c) Once a client is connected to the server, it is joining a chat room and will see a list of all peers connected to the servers. Then it sends out a chat broadcast message requesting a particular file, which is received by the individual peers.

The peers who have the file will respond to the requestor. The requester then sends a direct one-to-one request to the peer who has the file, establishes a direct connection and downloads the file. Alternatively, the requester also has the option to chat with a specific peer or peers using the *@ipaddress* tag name.


Peer
Peer
Peer
Peer
Server
Server
Peer
File Download
Requesting Peer