

Project - P2P File Sharing System

Due: April 17, 2018 23:55

In this project you can work in a team of up to 2 students.

In this project you will develop a peer-to-peer (P2P) file sharing network application in Java with GUI¹. The application should allow users to share and download files from each other.

In general, the system should work as follows. In the system, one designated machine, let's call it **FileTracker (FT)**, should always be on and waiting for others' (peers) requests. FT should contain *information about the files which are shared by online peers* and *information about the peers*. If a peer *A* wants to download a file *X*, *A* asks FT to send information about the online peers that share *X*. If there exist peers that share *X*, FT sends information about these peers and files to *A* (Note that different files with the same name may exist (difference could be in size, type, etc.)). *A* connects to one of the peers that it has received from FT and *may* download from *X* that peer.

The rest explains how the system should work in detail. See Figure 1 for illustration.

- ~~• The IP address of FT and the port number on which the application is running should be public, i.e., users should know the IP address and port number of FT.~~
- ~~• When a peer *A* connects to FT, *A* should automatically send information about its shared files as a list of records to FT in the following format: <file name, file type (e.g., text, jpg, etc), file size, file last modified date (DD/MM/YY), IP address, port number>. *A* should send "HELLO" and receive "HI" before sending the information.~~
- ~~• If *A* does not send any file information to FT while joining the system, i.e., if *A* does not share any file, then FT should not accept *A*. FT should not respond to *A*.~~
- Only accepted peers should be able to use the services offered by this system.
- ~~• When *A* wants to download a file with the name "File Name", *A* requests the file from FT by sending "SEARCH: " + "File Name".~~
- ~~• When FT receives "SEARCH: " + "File Name", it tries to find the file in a hash table where 'key' is filename and 'value' is a list which contains records of this format: <file type, file size, file last modified date (DD/MM/YY), IP address, port number>.~~
 - ~~– If FT finds the file, it should send "FOUND: " + list of records.~~
 - ~~– If FT does not find the file, it should send "NOT FOUND".~~
- ~~• After receiving a list of records, *A* should choose one of the peers (records) from the list, say *B*, and connect to *B* (using IP and port number) to request and download the file. For that *A* should send "DOWNLOAD: " + "FileName" to *B*.~~
- ~~• When *B* receives a "DOWNLOAD" message, it should generate a random number $1 \leq p \leq 100$, and if $p < 50$, then *A* sends "FILE: " + file, otherwise sends "NO!". In other words, as mentioned above, *B may not* send the requested file to *A*. This requirement is just to test the scoring method of the system.~~
- The scoring method is used to evaluate peers trustworthiness. The method works as follows: FT keeps track of two variables - 'NumOfRequests' and 'NumOfUploads'. Assume peer *A* requests a file from peer *B*.
 - If *B* sends the requested file to *A*, *A* will send "SCORE: 1" to FT.
 - If *B* does not send the requested file to *A*, *A* will send "SCORE: 0" to FT.

¹ A sample Java program with GUI will be given on the course's moodle page.

- When FT receives “SCORE: n”, it increments ‘NumOfRequests’ by 1 and increments ‘NumOfUploads’ by n.
- When *A* wants to leave the system, *A* should notify FT about this so that FT can update the hash table accordingly. *A* should send “BYE” to FT to do so.
- FT should also store a list of all peers that have joined the system at least once. A peer in the list should include the following information: <IP address, port number, score>. FT should keep the information about a peer even the peer leaves the system.

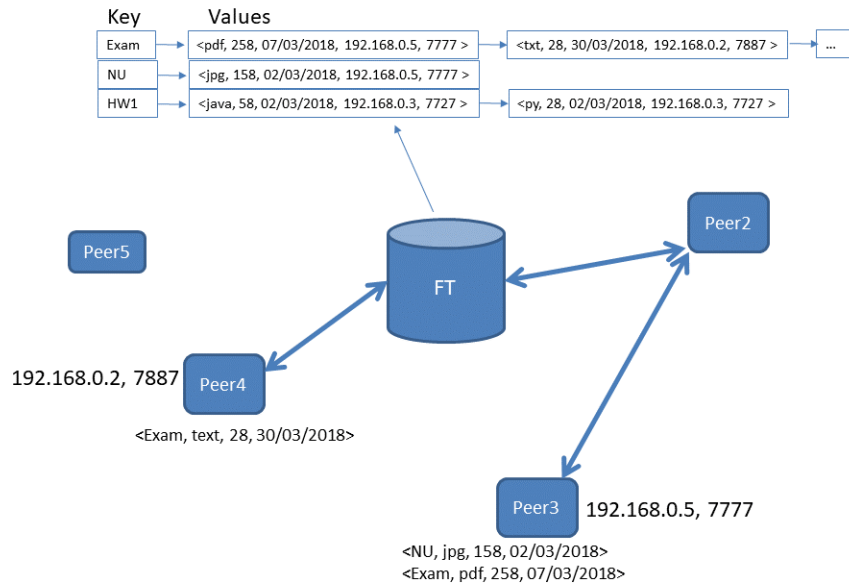


Figure 1: Illustration of P2P File Sharing System.