CNT 5106 Computer Networks

Project 2 Team Project (Due Dec 4)
Implementation of P2P network

This project is a team project and each team consists of **one or two** students. The goal is to create a peer-to-peer network for file downloading. It resembles some features of Bit-torrent, but much simplified. There are two pieces of software – ***peer*** and ***file owner***.

The **file owner** has a file, and it breaks the file into chunks of 100KB, each stored as a separate file. The **minimum** number of chunks that the file can be split into is 5. The file owner listens on a TCP port. It should be designed as a server that can run multiple threads to serve multiple clients simultaneously.

Each **peer** should be able to connect to the file owner to download some chunks. It then should have two threads of control, one acting as a server that uploads the local chunks to another peer (referred to as *upload neighbor*), and the other acting as a client that downloads chunks from a third peer (referred to as *download neighbor*). So each peer has two neighbors, one of which will get chunks from this peer and the other will send chunks to this peer. You can arbitrarily decide on the neighboring relationship as long as the network is connected --- there is a direct path from any peer to any other peer. The neighboring relationship may be encoded through input parameters (see below).

1. Start the file owner process, giving a listening port.
2. Start five peer processes, one at a time, giving the file owner's listening port, the peer's listening port, and its download neighbor's (another peer) listening port.
3. Each peer connects to the server's listening port. The latter creates a new thread to upload one or several file chunks to the peer, while its main thread goes back to listening for new peers.
4. After receiving chunk(s) from the file owner, the peer stores them as separate file(s) and creates a summary file, listing the IDs of the chunks it has.
5. The peer then proceeds with two new threads, with one thread listening to its upload neighbor to which it will upload file chunks, and the other thread connecting to its download neighbor.
6. The peer requests for the chunk ID list from the download neighbor, compares with its own to find the missing ones, and RANDOMLY requests a missing chunk from the neighbor/file owner. In the meantime, it sends its own chunk ID list to its upload neighbor/file owner, and upon request uploads chunks to the neighbor.
7. After a peer has all file chunks, it combines them for a single file.
8. A peer MUST output its activity to its console **whenever** it receives a chunk, sends a chunk, receives a chunk ID list, sends out a chunk ID list, requests for chunks, or receives such a request.

Other Information

Programming Environment

Programming language: Java, C, C++, C#, Python
Operating System: Windows, Mac OS or Linux
Programming Tool: Eclipse, IntelliJ, Jcreator, Netbeans, … whatever you like.

To use Eclipse, please go through the following list:

1. Download JDK from: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
2. Download Eclipse from: http://www.eclipse.org/downloads/
3. Here is a link for eclipse tutorial: http://eclipsetutorial.sourceforge.net/totalbeginner.html
4. Here is a tutorial for socket programming in Java: http://java.sun.com/docs/books/tutorial/networking/sockets/

Demo Policy:

1. Before Demo

- Each group of student(s) must sign up a time.
- A sign-in sheet will be posted on Canvas before demo. You can pick an idle time slot and put down **ALL** team members' names and UFIDs.
- Please do not replace another group's information in a time slot and you must show up during the time slot that you sign up for.

2. During Demo

- 10 minutes presentation.
- Whole team must be present and demo your project during December 5, 6 at E309/E312.
- You can use a CISE computer or your laptop for the demo.
- We will use the source code you **uploaded** on Canvas and a large file (larger than 10MB) of certain formats (PDF, pptx, doc, mp3, mp4, jpg) to test your code.
- Run file owner and peers, and explain your code.
- **Minimum** Expectation from the program:
  1) All log information on the activities of the file owner and five peers;
  2) Each peer **concurrently** downloads file chunks from **both** the file owner and its download neighbor;
  3) The file is fully recovered in all peers.