CNT5106c (Computer Networks)

Project

This project creates 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 size of the file is 5 chunks. 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 listening port.
3. Each peer connects to the server's listening port. The latter creates a new thread to download 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, compare with its own to find the missing ones, and download those from the neighbor. At the mean time, it sends its own chunk ID list to its upload neighbor, 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 should 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.

Edge Students:

Here is a guideline for project submission. Since remote students cannot come to demonstrate the project in person, our TAs will do the demonstration for you. They will use the CISE computers (http://www.cise.ufl.edu/help/access/index.shtml). You are also able to access these machines using Telnet or SSH. We do not limit the programming environment that you use. But to eliminate the possibility that you lose some points because the TAs cannot successfully run your projects, you are required to run and test your projects on these computers before submitting. Your submission must include executable/compiled files, Makefile, README and source files. Note that it is your responsibility to make it sure that your executable/compiled files are run-able or compilable on CISE computers. If you have any more questions regarding the submission, please send an email to the TA Mr. Zhen Mo (zmo@cise.ufl.edu).