Project Logistics

Due Dates

Midpoint check: Nov. 2, 11:59 pm
Final Project: Dec. 9, 11:59 pm
Late Policy: No late project will be accepted.

The project is fairly large (2000 lines of code or more). Please start early.

There will be a midpoint check. You are supposed to implement about half of the lines for your project (800 lines or more), which can pass compilation (see suggestions later). This is worth 5% of your course grade.

Group Size

- Three students per group.
- If you need project partners, the TA is helping. You should have received announcements on this process. If you haven't participated, please do so ASAP.

Project Submission

- Submit on Canvas for both the midpoint check and the final project.
- Combine all your files into one archive and/or compressed file with winzip on Windows PC or with the tar command on UNIX/Linux machines. An example:

```
tar cvf proj1.tar foo.java bar.java
```

Here, `proj1.tar` is the archive file, and `foo.java` and `bar.java` are your source files.

- The zip or tar file should contain all source files, all files needed to compile your program, and all files needed to run your program. Your zip file should also contain files for starting your remote processes, even if you use the files from the course web site without changing anything. Your zip file should not contain executable files or object files which can be generated by compiling your program. It should not contain Common.cfg, PeerInfo.cfg, and any other sample files for testing.

Demo

- After the submission, each group will present a demo of their project. This may take the form of 20-minute live demo via zoom, or a much shorter pre-recorded demo if your project works well. More details will come later.

Questions

- If you have difficulty understanding the protocol description, you may search the BitTorrent protocol description on the Internet and study it. There are many sources of information that explain the protocol. Then, you can come back and read the project description again.

Grading

There will be a common part of the grade for each project group. On top of that, each group member will be given an individual grade for the amount of effort spent.

Sample Files

On Canvas, under the direction '`Files/Project`', you can find three sample files related to the project.

`Sample Client.java`, `Sample Server.java`: These are self-explanatory.

`StartRemotePeers.zip`: Code for starting multiple peers automatically. It is not essential. The code used to work in the CISE environment. But, it is now known to have problems due to SSH. If you can get it to work, it can be helpful (see also the next file). Otherwise, you can always start peers manually. See the project description for explanation.

`startpeers-from-students.zip`: Code and instruction for getting around the SSH problem. This was discovered by some student group.

Academic honesty

You should not have someone else to write your code or copy the code from someone else. But, you are allowed to use the Internet resources and learn how to do network programming in general. Anyone who violates the above rule will get penalized. Please note that there are tools for us to check the similarity between two codes.

FAQ

Q1. Can I implement the project on Windows environment?

You may. But, in the end, you must be able to demonstrate your project on 5 different machines at the same time.

Q2. Can I assume that the subdirectories for peers have already been set up before I run the peer processes?

Yes. You may create subdirectories with proper names before you run your peer processes. That saves you the burden for managing subdirectories in your program. See the relevant discussion in the project description document.

Q3. How many TCP connections should be used between two peers?

The design may vary. But, consider using one TCP connection for each pair of peers. Consequently, a peer maintains only one socket for each neighboring peer. However, you may have two threads associated to the socket, one for getting messages from the neighboring peer and the other for sending messages to the peer. The socket will be shared by those two threads.

Q5. How is the ‘have’ message different from the ‘bitfield’ message?

The ‘bitfield’ message is useful to inform a neighboring peer the availability of many file chunks. You can use it at the start of a process. The ‘have’ message is used to inform a neighbor the availability of a single chunk. It is useful for incremental update between the neighboring peers. For instance, when a peer receives a new chunk, it can use the ‘have’ message to inform other neighbors.

Q6. How can I submit the final project?

Please read the section "Project Submission" carefully.