P2P File Sharing

Due: Tuesday, December 3rd at 11:59pm

Problem Setup

There are N peers. Each peer has a sub-folder, “Shared”, which to start, contains a single file. Each peer will start with a different file. I.e., P0 starts with file F0, P1 starts with file F1, etc.

Goal

Every peer should obtain a copy of every file that is available while the peer is alive.

Assumptions

1. Files are never deleted from the system.
2. Files are never modified.
3. Files to be shared are in the “Shared” subdirectory.
4. Each peer Px stays alive, and in communication with the tracker, until:
    a. no other peers are currently connected to Px and acquiring data, i.e., all other living peers have a copy of Px’s data
    b. Px has a copy of all available data
    c. at least S seconds has elapsed (passed as a parameter)
5. Peers do not reconnect.
6. There will be no more than 8 peers.

Implementation Requirements

You must implement a:

1. tracker
2. peer

Tracker

The tracker must keep track of each file currently available. For each file, the tracker should keep track of:

1. the file size
2. the number of chunks

For each file chunk, the tracker should keep track of:

1. the list of peers (IP and port) with that chunk.

The tracker must also assign each peer an identifying integer, which will be communicated back to that peer. It is sufficient to use an integer counter.

Initially, the tracker knows nothing. When a peer connects to the tracker, it should tell the tracker what data it has available to share. The tracker should then display:

[PEER N CONNECT:][1 SPACE][OFFERS][1 SPACE][NUMFILES][NEWLINE]
[N][4 SPACES][FILENAME1][1 SPACE][NUMCHUNKS][NEWLINE]
…

[N][4 SPACES][FILENAMEX][1 SPACE][NUMCHUNKS][NEWLINE]
where N is the peer's ID, assigned by the tracker, NUMFILES is the number of files offered by the new peer, FILENAME is the name of each file, and NUMCHUNKS is the number of file chunks.

The tracker should be updated as peers download data. When a peer completes the download of a file chunk, the tracker should display:
[PEER N ACQUIRED:][1 SPACE][CHUNK X][/][NUMCHUNKS][1 SPACE][FILENAME][NEWLINE]

When a peer Px has received all currently available data and all other peers have received Px's data, Px will say goodbye to the tracker. **The tracker will display**:
[PEER N DISCONNECT:][1 SPACE][RECEIVED][1 SPACE][NUMFILES][NEWLINE]
[N][4 SPACES][FILENAME1][NEWLINE]
…
[N][4 SPACES][FILENAMEX][NEWLINE]

Output to the screen should be done with a mutex to ensure an uninterrupted/corrupted message display.

When a peer connects, the tracker should update its internal information about file availability. When a peer disconnects, the tracker should again update its internal information about file availability.

The tracker has no parameters. The tracker must choose a port number and store it in a file called port.txt. To ensure this value is flushed immediately to disk for our scripts, after writing the port number to the file, close the file right away.

Create a shell script, **tracker.sh**, to run your tracker.
Note: the square bracket notation above [X] is used to delineate text. Do NOT display square brackets in your output.

Peer
The peer wants to share all of the files in its Shared folder with every other living peer. It also wants to acquire copies of files owned by other peers. When a peer starts, it connects to the tracker and receives an ID number from the tracker, which is used for logging. The peer should also inform the tracker about all file data it has.

Next, the peer should discover and retrieve all files that it does not have from other living peers. If the peer has completed this task, and, some minimum of S seconds have elapsed and NO NEW FILES HAVE ARRIVED at the tracker, then the peer may **disconnect from the tracker** and **shut down**.

On shut down, the peer should display:
[PEER N SHUTDOWN:][1 SPACE][HAS][NUMFILES][NEWLINE]
[N][4 SPACES][FILENAME1][NEWLINE]
…
[N][4 SPACES][FILENAMEX][NEWLINE]

The peer has three parameters:

1. the address of the tracker
2. the port number of the tracker
3. the minimum alive time in seconds

Create a shell script called **peer.sh** to run your peer.
Your peer should be run as follows:

```
sh peer.sh [IP] [PORT] [MIN ALIVE TIME]
./peer.sh [IP] [PORT] [MIN ALIVE TIME]
```

Implementation Details
You may implement the tracker and peer however you like so long as it adheres to the assumptions and output specified. Additionally, your implementation must:

1. use a file chunk size of 512 bytes
2. place all complete files in the Shared directory
3. the tracker may NOT be used to store and distribute files

Please submit a single ZIP file containing your submission.
You may use any language you like but may not use higher-level libraries which may trivialize this assignment (such as a P2P library). You can use TCP, UDP, or both.

As always, your programs MUST run on the school servers.

Directory Structure
Inside of your submission, use the following directory structure:

- peer.sh ←- the script to run your peer
- tracker.sh ←- the script to run your tracker
- Shared/ ←- leave this folder empty
- code/ ←- put all code in here
- Makefile ←- your makefile

Grading

Marks	Objective
10	peer and tracker compile and execute
20	two peers and a tracker
20	four peers and a tracker
20	eight peers and a tracker
30	following instructions/rules