CS 3251: Computer Networking I (Section A and B) **Spring 2021**

Programming Assignment 2

Handed Out: March 14^{th}*,* 2021 *Due: 1:59pm, April* 21^{st}*,* 2021

1 Early Release

This is the early release version of the assignment to get you started with the idea and the required baseline code. We would like you to write your code in Python, even though we thought earlier that we would give you the freedom to choose your own language. If the Python requirement gives you serious heartburn, let your faculty instructor know.

2 Introduction

BitTorrent is a peer-to-peer (P2P) file sharing service that transfers files between hosts without the involvement of a central server, providing scalability benefits compared to client-server architectures, where the server can be a performance bottleneck. Peer-to-peer systems are also popular for an unethical reason, namely that they may help evade law enforcement when distributing copyrighted material. You can read an overview of P2P systems in general and BitTorrent in particular in the Kurose and Ross textbook, 8th Edition, Section 2.5.

In this assignment you will be creating a mini-version of a file sharing service that has some characteristics in common with BitTorrent. There are two primary components that you will code: (1) a **P2PBootstrapper** responsible for keeping track of *registered* P2PClients, and (2) a **P2PClient** that stores files, answers requests about which files it has, and (in a full system) delivers files to peers (other P2PClients) upon request. You will not have to deliver the actual files. The P2PClient can also make requests about which files a peer has, and register/deregister with the P2PBootstrapper. As you can see from this description, the P2PClient acts as both a client (requesting file information) and a server (delivering file information). You may notice that the P2PBootstrapper is a server, thus the system is not purely peer-to-peer. The file transfers, however, only happen between peers. More detail on the functionality of the P2PBootstrapper (also called the Bootstrapper) and the P2PClient (also called the Client or peer) is below.

We will grade the assignment using an autograder. It is to your advantage to read the assignment in its entirety, before commencing work on it. Though not intentional, there will undoubtedly be some details that are not clear from the English language description. This is part of why finite state machines are useful to specify protocols! You should expect some clarifications to emerge as students start to work on the assignment.

3 Overview of the System

Each file to be shared is identified by a 32 hex-character key that is the MD5 hash of the file content. We will give you the keys; you will not need to compute them. For our purposes, we will assume that these hash value keys are unique. You will also not need to keep actual files, since the point of this assignment is to model the tracking and finding of content.

Each Client keeps two data structures, a *Content List (CL)*, which is simply a list of the keys for the content the client currently holds, and a *Content Originator List (COL)* which is a list of the keys for content the Client has received from other Clients. For each key in the COL, the IP address and port number of the sending client is kept, as illustrated in the table.

Content List	Content Originator List		
	Content	IP	Port
28a92bc8ef18ac92b1...	28a92bc8ef18ac92b1...	127.0.0.1	21312
36df53be1ef189c67f1...	abababababccc112332...	127.0.0.1	12315
718bdeca812a52834...	8881818188818818...	127.0.0.1	21531
720361bac36edcab1...	720361bac36edcab1...	127.0.0.1	32144
...	...	127.0.0.1	12314
237ba23dc189a392...	237ba23dc189a392...	127.0.0.1	42141

Clients may delete files from their content list at any time. When they do so, the corresponding key is immediately removed from the CL, but news of the deletion is not propagated any further. As a result, the COL list at other peers may be out of date, i.e., a list may indicate that a particular peer has content when in fact that peer has deleted the content. This leads to a searching and routing problem, checking to see if a peer has content, and if not, discovering other peers to ask.

The P2PBootstrapper is the mechanism that allows a P2PClient to find out about currently registered peers in the system. The P2PBootstrapper keeps a list of all currently registered peers (via a Client ID and the IP address and port number for the client) and provides that list upon request. The P2PBootstrapper also sends a start command to all peers it knows (registered and deregistered) to trigger the start of peer action and to enforce some command ordering across peers, making it easier to test.

4 P2PClient Behavior

A P2PClient starts by reading its command line parameters, which specify its globally unique ClientID, the P2PBootstrapper IP address and port number, the filename for the local file containing the Content List, and filename for the local input-script file containing commands for the client to execute.

After reading the command line parameters, the client registers with the P2PBootstrapper using the IP address of the host machine it is running on and an OS-provided port number. It then waits for the start message from the bootstrapper. After receiving the start message, the client executes the commands in the input-script in order pausing as required by the script to respect the command timing. The possible commands are contained in the table below.

Command Label	Command Meaning
R,<bootstrapper IP>,<port>	Register itself with the Bootstrapper. Send ClientID
U,<bootstrapper IP>,<port>	Unregister itself from the Bootstrapper
L,<bootstrapper IP>,<port>	Obtain a list of all clients from bootstrapper. Get ClientID, ClientIP, ClientPort
Q,<content hashcode>	reQuest a content
P,<content hashcode>	Purge (remove) content from own list (but retain it in the content originator list)
O,<clientID>	Obtain a list of all other clients known to a particular client (clientID).
M,<clientID>	Obtain a list of all content hashes from a particular client (clientID).

Suppose a particular client has a ClientID 1. Here is an example input-script:

```
2,Q,898119abdc27abd898119abdc27abd13
3,Q,aaaaaaaaaaaaaaaaaaaaaaabbbbbbbbb
3,Q,77777777777777777777778888888888
3,Q,cccccbbbbccccccccccccbbbbbbbcbcb
4,P,aaaaaaaaaaaaaaaaaaaaaabbbbbbbbb
5,Q,ddddddddddddddddddddddddddddddddd
6,U,127.0.0.1,5001
7,L,127.0.0.1,5001
7,O,4
8,M,7
```

This script first requests fetching the content for hashcode `898119abdc27abd898119abdc27abd13`. The internal processing on this command can be quite complex. The program will first query the Bootstrapper for a set of all clients. It will then start sending requests for the content to each of these clients in a serial fashion. From any P2PClient, it might either receive confirmation that the peer has the content, or may receive a hint about where the content might be located. On receiving an hint, the client attempts to contact that P2Pclient before progressing ahead with its current list. If content is not found, then it proceeds to request every P2PClient for a list of all other clients it knows and creates a longer list for trying various P2PClients.

On time 3 seconds, the content for hashcode `aaaaaaaaaaaaaaaaaaaaaabbbbbbbbb` is requested. After that is complete, the next line is executed (same time 3), and so on. Then at time 4 seconds, the client is purging its own (or newly obtained) content. It just removes that entry from the "Content List" datastructure. At time 6, it unregisters from the bootstrapper.

At time 7 seconds, it asks for a list of all clients currently connected to the Bootstrapper (this operation is allowed even if this client is not currently registered).

The output of your program will follow a specific format. Example output for the above client is as follows:

```
0,Client ID 1 registered
2,Obtained 898119abdc27abd898119abdc27abd13 from 127.0.0.1#12348
3,Obtained aaaaaaaaaaaaaaaaaaaaaaabbbbbbbbb from 127.0.0.1#21441
3,Content 77777777777777777777778888888888 is unavailable
3,Obtained cccccbbbbccccccccccccbbbbbbbcbcb from 127.0.0.1#21441
4,Removed aaaaaaaaaaaaaaaaaaaaaaabbbbbbbbb
5,Obtained ddddddddddddddddddddddddddddddddd from 127.0.0.1#12379
6,Unregistered
7,Bootstrapper:<4,127.0.0.1,12348>,<5,127.0.0.1,21441>,<3,127.0.0.1,12379>
7,Client 4:<3,127.0.0.1,12379>,<7,127.0.0.1,43182>
8,Client 7:aaaaaaaaaaaaaaaaaaaaaaabbbbbbbbb,123456789012345678901abcdefabcdef
```

5 P2PBootstrapper Behavior

As described earlier, the P2PBootstrapper is a simple program that keeps a single list of all P2PClients it knows about. The list contains the ClientID, IP address, and port number, along with a status, either registered or deregistered. A P2PClient registers itself upon startup. It can then issue a deregistration or a registration in the future. (To be completely robust, the Bootstrapper could only allow registration for currently deregistered Clients, and vice versa, but you do not need to check for this.) A P2PClient may continue to exist and serve requests even after it deregisters itself from the Bootstrapper. The Bootstapper must send a start command to all clients to signal the beginning of time (t=0). All P2PClients will start executing their own input-scripts from this time.

6 Submission Guidelines and Grading

You will submit the P2PClient and P2PBootstrapper programs and a makefile that can compile these programs on Gradescope.

Your code will be checked for the following:

- Bootstrapper maintains a list and supports register and deregister commands (10 points)
- P2Pclient can obtain a list of clients from the Bootstrapper (5 points)
- P2PClient can check for content at the Clients listed by the Bootstrapper (10 points)
- P2PClient can provide hints about content it knows about (10 points)

- P2PClient can use hints and does so before iterating the list provided by the Bootstrapper. (10 points)
- P2PClient purges content from its own list (5 points).
- P2PClient still provides hints about purged content. (5 points)
- P2PClient can fetch data from a client unreachable from the Bootstrapper for which no hint was obtained (by obtaining a list of all clients known to all known clients). (10 points)
- P2PClient can obtain all content hashes from a particular peer (10 points).
- P2PClient can re-register with the Bootstrapper (5 points).
- More than 20 clients are supported (10 points).
- Some other criteria worth 10 points. (10 points).