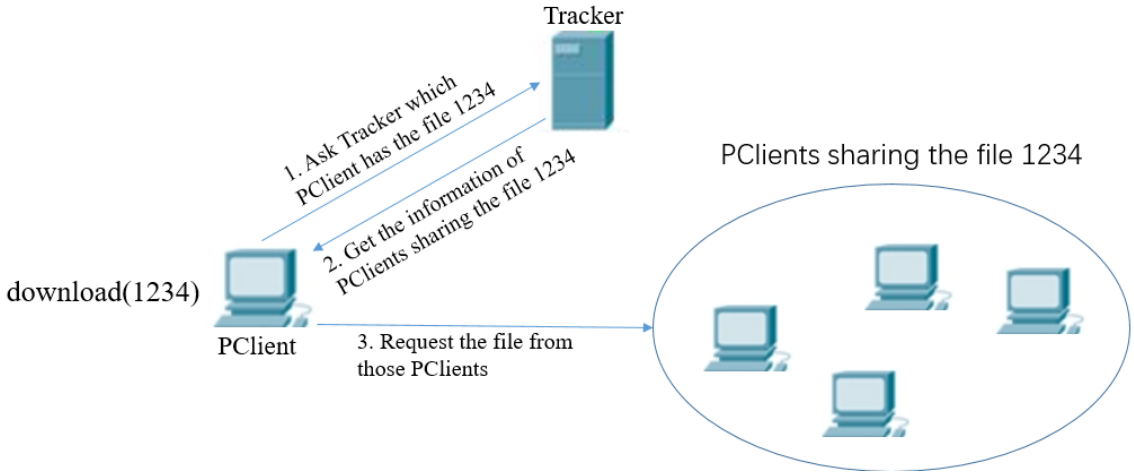
Project 2: P2P File Sharing System

People in charge: Jiaxi Zhang, Muzhen Yang

Introduction

Compared with the traditional Server-Client model, P2P network can make full use of the upload and download bandwidth of different nodes, and achieve that the larger the network, the faster the file sharing speed. **In this project, you are going to build a P2P file transfer system, design and implement the functions of Tracker and PClient as required.** In this system, PClient can use the information recorded on Tracker to achieve the purpose of file sharing; when a PClient wants to download a file from the P2P network, he needs to request the tracker for the information of the node that owns the target file, and then request files from these nodes. The following figure shows the process of download() function.



Except the restrictions and requirements in this document, you can design all the implementation details by yourself, such as how to split and assemble a file before and after a transmission, the header format of different packets, how PClient decides which PClient to request files from, etc.

What you need to do

- Tracker.py: you need to design and implement the function of Tracker
  - Tracker is used to store information for communication between PClients, such as the address of each node, the downloadable file shared in the network, the holders of a file or file block, etc.
  - You need to design and implement the communication rules between PClient and Tracker, the specific information stored on the Tracker **(the examples given above are for reference only, please design the detail according to your own design requirements)** and the API provided to PClient
  - We provide a very simple implementation of Tracker in SimpleTracker.py, if you are confused with what to do, you can try to simply add more functions in it.
  - **Note: Tracker does not store the actual file content in our settings! ! ! In our test case, the tracker will be allocated a small upload bandwidth, so it is not recommended to save the file on the tracker! ! !**
  - **Note: You can only send and receive packets by calling the provided __send__() and __recv__() functions;** `import socket` **is forbidden**
- PClient.py (Need to implement four functions mentioned above，**all these functions should be blocking functions, i.e. return only after finishing everything in it**)

    - register()
        - Function: PClient registers to Tracker, informing Tracker that he owns a file and is willing to provide it for other PClients to download
        - Input: a file path, such as "./alice.txt" for the alice.txt file in the current folder
        - Output: fid, a specific identifier of a file which is designed by yourself; you can directly use the hash value of the file
    - download()
        - Function: PClient will first ask the Tracker about which nodes own a certain file, and then requests the files from those nodes
        - Input: fid, specify the file to be downloaded, the type should be the same as the return value of register()
        - Output: bytes data, which is the content of the whole file
    - cancel()
        - Function: Cancel the registration of the specific file on the Tracker. After this function returns, other PClients will not be able to get that file from this PClient
        - Input: fid, specify the file to be downloaded, the type should be the same as the return value of register()
        - Output: Anything if required in your design; can be no return value
    - close()
        - Function: Cancel all the registration of files on Tracker, exit the P2P network, and no longer be able to send or receive any packets
        - Input: Nothing
        - Output: Nothing
    - Please design and implement these four functions without violating the given requirements
    - You are free to implement a more complex constructor without modifying provided code in it and add other functions for PClient if needed
    - **Note: You can only send and receive packets by calling the provided __send__() and __recv__() functions; `import socket` is forbidden**

Other Codes Provided

- Proxy.py **(You are forbidden to modify it)**
    - Tracker and each PClient will have a unique proxy, in which bandwidth will be set to simulate transmission delay
- SimpleTracker.py
    - A simple implementation of Tracker as a reference for you
- SC_Model/: traditional Server-Client model
- P2P_test/: some test cases for the P2P file sharing system

Other Details

- In our P2P network, there will be only one Tracker and several PClients, and they will communicate with each other through their own proxy.
    - **Remind again: Tracker is not exactly the same as the traditional Server. Tracker only stores control information, such as the information of each PClient in the current network (such as address) and the information of each file (such as which PClient holding a certain file). There will be no actual file on it**
- Proxy

    - The proxy will create a socket on a port and transfer the packet through it (if the port number is not specified, the port will be bound to a random port instead)
    - There will be a simulation of transmission delay in the proxy, and the upload and download bandwidth can be specified in during initialization
- PClient
    - The parameter lest of the constructor `(tracker_addr,proxy=None,port=None,upload_rate=0,download_rate=0)`
        - The address of the Tracker in the P2P network must be specified
        - If a proxy is specified, the PClient will directly use it as his own proxy
        - If no proxy is specified
            - If the port is specified, the PClient will try to bind their proxy to the given port
            - If the port is not specified, the PClient will try to bind their proxy to a random port
- Description of a basic scene: PClient *A* join the network and want to download file *F*
    - *A* is initialized and given a Tracker address
    - *A* request the relevant information of the file *F* from the Tracker
        - You are free to design and implement how PClient and Tracker interact with each other; there is no detailed requirements of your implementation
    - *A* requests the file from PClients who own *F* and completes the download
        - You are free to design and implement how PClients transfer files with others; there is no detailed requirements of your implementation
    - *A* exits the network by calling close()

Testing Scenarios

Our test will include, but is not limited to the following scenarios, more scenarios can be found in the P2P_test folder：

- (SimpleTest.py) PClient A, B join the P2P network; A registers files *F1* on the Tracker; B downloads *F1*; A cancels the registration of *F1*; PClient C joins the network and downloads *F1*
    - Result: B downloads the file from A; C downloads the file from B
- PClient A, B, C, D join the network (A has low upload bandwidth, B, C, D have high upload and download bandwidth); A registers a file *F1*; B, C, D download *F1*
    - Result: Due to A's low upload bandwidth while B, C, and D all have high upload and download bandwidths, B, C, and D finish finish the download almost at the same time

Grading

- The P2P file sharing can be finished successfully in a simple network environment (30)
    - There will be no join of new nodes or exit of old nodes during the file transfering process
- The P2P file sharing can be finished successfully in a complex network environment (20)
    - There may be join of new nodes and exit of old nodes during the file transfering process
- The efficiency of your P2P implementation compared to tradition Server-Client model in different networks (20)
- Presentation (30)
    - PPT is required and must at least include the following points (you can change the order if needed)

        - System design(Tracker, PClient, communication rules and file transfer protocol)
        - Implementation (what technique has been used and how you use them)
        - Highlight of your project (the points about the design or implementation that you most want to show)
        - Description of test scenarios (you are suggested to add your own test cases except those provided)
        - Test result
        - Summary (review + future work)
- BONUS(20)：tit-for-tat(You can find this in the lecture slide Chapter2-3, page 10-11)
    - Realize the preference of nodes with high upload bandwidth when downloading files(10)
        - When the upload bandwidth of a PClient in sharing suddenly drops, the PClient will actively choke that node and select another high-speed PClients
    - Realize the re-selection of the choked high-speed PClient(10)

Suggestion for Design

- You can follow this order to design and implement your project
    - Design the specific information stored on the Tracker (such as storing the entire file/ the file blocks of each file)
    - Design the API provided by Tracker to PClient
    - Design the communication rules and packet format between Tracker and PClient
    - Design the communication rules, including the format of data packet and control packet, among PClients in a simple network environment
    - Design strategies to make it usable under complex network environments (such as using timeout mechanism to detect whether the requested PClient has left the network)

TIPS

- If you don't know how to use **__send__()** and **__recv__()** functions, please refer to **sendto()** and **recvfrom()** functions in udp socket, or the usage of these two functions in the given Server-Client model
- If you use the hash function to generate fid, it is recommended to use hashlib. The origin hash function provided by python depends on the environment variable PYTHONHASHSEED, which may result in different hash code when the same content is hashed
- **You are strongly recommended to learn and use python's multi-threading knowledge, which will greatly improve the speed of file sharing**
- If you have any question about this document or the project, feel free to communicate with us or leave a comment in the link below
    - 【腾讯文档】P2P文件传输系统 https://docs.qq.com/sheet/DRVl1SUh5T1JYZ0ZN