ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

23CLC08

BÁO CÁO

SOCKET

GIẢNG VIÊN HƯỚNG DẪN

Nguyễn Thanh Quân

—o0o—

THÀNH VIÊN

23127361 - Thạch Ngọc Hân
23127170 - Phạm Thành Đạt
23127284 - Nguyễn Ngọc Mai Xuân

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 7 NĂM 2024

Contents

1 THÔNG TIN NHÓM & ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH **3**

2 KỊCH BẢN GIAO TIẾP CỦA CHƯƠNG TRÌNH **3**

2.1 PHẦN 1: . . . 3

2.1.1 Giao thức trao đổi giữa Client và Server: . . . 3

2.1.2 Cấu trúc thông điệp: . . . 4

2.1.3 Kiểu dữ liệu của thông điệp: . . . 5

2.1.4 Cách tổ chức cơ sở dữ liệu: . . . 5

2.1.5 Các chức năng chính của chương trình . . . 5

2.2 PHẦN 2: . . . 5

2.2.1 Giao thức trao đổi giữa Client và Server: . . . 5

2.2.2 Cấu trúc thông điệp: . . . 6

2.2.3 Kiểu dữ liệu của thông điệp: . . . 6

2.2.4 Cách tổ chức cơ sở dữ liệu: . . . 6

2.2.5 Các chức năng chính của chương trình . . . 7

3 Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng **7**

3.1 Ngôn ngữ lập trình: . . . 7

3.2 Thư viện và Module Python: . . . 7

3.3 Môi trường lập trình: . . . 8

3.4 Framework hỗ trợ: . . . 8

4 Hướng dẫn sử dụng các tính năng của chương trình: **8**

4.1 Chuẩn bị môi trường: . . . 8

4.2 Khởi động Server: . . . 8

4.3 Khởi động Client: . . . 9

4.4 Các tính năng chính: . . . 9

4.5 Chạy chương trình: . . . 10

5 Bảng phân công công việc chi tiết: **10**

5.1 Chi tiết công việc Phần 1: . . . 10

5.2 Chi tiết công việc Phần 2 (Server.py): . . . 11

5.2.1 Tạo và Quản lý Socket Server: . . . 11

5.2.2 Quản lý Danh Sách File (Menu): . . . 11

5.2.3 Chấp Nhận Kết Nối từ Client: 12
5.2.4 Gửi Menu File tới Client: 12
5.2.5 Xử Lý Yêu Cầu Tải File: 12
5.2.6 Quản Lý Đa Luồng: 12
5.2.7 Ghi Nhận Trạng Thái Kết Nối: 12
5.3 Chi tiết công việc Phần 2 (Client.py): 12
5.3.1 Tạo và quản lí Socket: 12
5.3.2 Khởi tạo các lớp (Classes) & các biến toàn cục: 13
5.3.3 Đọc và xử lí File (input.txt): 13
5.3.4 Xử lí tải file từ server & cập nhật danh sách yêu cầu & xử lí và hiển thị tiến độ tải file 13
5.3.5 Hàm chính của chương trình: 14

1 THÔNG TIN NHÓM & ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

Thành viên	Công việc được giao	Đánh giá
Phạm Thành Đạt	Phần 1	100% (hoàn thành)
Nguyễn Ngọc Mai Xuân	Phần 2: client.py	100 % (hoàn thành)
Thạch Ngọc Hân	Phần 2: server.py	100 % (hoàn thành)

2 KỊCH BẢN GIAO TIẾP CỦA CHƯƠNG TRÌNH

2.1 PHẦN 1:

2.1.1 Giao thức trao đổi giữa Client và Server:

Server:

- Server được khởi tạo và gắn với địa chỉ và cổng được chỉ định.
- Server lắng nghe các kết nối đến trên cổng (12345).
- Khi một client kết nối, server chấp nhận kết nối.
- Server đọc file chứa danh sách các file có sẵn và gửi cho client sau khi kết nối được thiết lập.
- Server sẽ chờ nhận các tên file cần tải từ client.
- Khi nhận tên các file, server kiểm tra nếu file được yêu cầu tồn tại trong các file có sẵn:
  - Nếu file tồn tại, server tiếp tục gửi file.
  - Nếu file không tồn tại, server sẽ gửi thông báo không tồn tại.
- Server gửi một thông báo rằng nó sắp gửi file và kích thước của file.
- Server sẽ chờ phản hồi từ client:
  - Nếu client phản hồi tiếp tục nhận file, server sẽ tiếp tục gửi file theo từng chunk. Sau khi gửi xong dữ liệu của file, server sẽ in thông báo ra màn hình.
  - Nếu client phản hồi file đã được tải, server sẽ không gửi lại.
- Nếu có ngoại lệ xảy ra, server thông báo lỗi và đóng kết nối với client. Server cũng xử lý việc client ngắt kết nối và in thông báo ra màn hình.

Client:

- Client được khởi tạo và kết nối đến server.
- Sau khi client được chấp nhận kết nối, client sẽ nhận danh sách các tệp có sẵn ở server.

- Client sẽ liên tục đọc file input để lấy những file cần tải và chưa được tải và gửi đến server.
- Client sẽ nhận phản hồi từ server:
  - Nếu file tồn tại, client sẽ nhận kích thước thực của file và tiếp tục nhận dữ liệu từ server.
  - Nếu file không tồn tại, client sẽ thông báo và tiếp tục với những file khác.
- Client sẽ kiểm tra file đó:
  - Nếu đã được tải, client sẽ phản hồi cho server rằng file đã được tải trước đó và tiếp tục với những file khác.
  - Nếu chưa được tải, client sẽ phản hồi tiếp tục nhận dữ liệu của file đó từ server.
- Client sẽ nhận từng chunk của file và ghi vào folder output. Sau khi tải xong dữ liệu của file, client sẽ in thông báo ra màn hình.
- Nếu có ngoại lệ xảy ra khi tải file, client thông báo lỗi và đóng kết nối.
- Client ngắt kết nối với server bằng tổ hợp phím "ctrl + c" và in thông báo ngắt kết nối.

2.1.2 Cấu trúc thông điệp:

- Thông điệp server gửi tới client:
  - Danh sách các file có sẵn: Thông điệp đã được pickle hóa chứa danh sách các file. Mỗi file có thể được mô tả bằng một tuple (filename (tên file), filesize (kích thước tương đối))
  - "Sending file..." kèm theo kích thước file: Khi server bắt đầu gửi file, nó gửi một thông điệp báo cho client rằng file sắp được gửi, kèm theo kích thước của file. Cú pháp thông điệp có thể là "Sending file... <size>".
  - Phần dữ liệu của tệp được gửi qua socket, chia thành nhiều khối với kích thước tối đa là size (10240 bytes trong mã nguồn).
  - Nếu file không tồn tại, server gửi thông báo "File not found".
- Thông điệp client gửi tới server:
  - "filename" được mã hóa bằng định dạng utf-8 để yêu cầu server gửi tệp cụ thể. Đây là tên của tệp mà client muốn tải về.
  - "Client is receiving file..." sau khi client nhận thông báo từ server rằng tệp sẽ được gửi. Đây là cách thông báo cho server rằng client đã sẵn sàng để nhận tệp.
  - Nếu file đã được tải, client sẽ gửi thông điệp "File is already downloaded." cho server để server không dữ liệu của file đó.

2.1.3 Kiểu dữ liệu của thông điệp:

- Chuỗi (string): các thông điệp phản hồi chủ yếu là chuỗi văn bản và tên các file cần tải cũng là chuỗi văn bản.
- Dữ liệu nhị phân (binary data): Dữ liệu của các file được gửi và nhận dưới dạng nhị phân.
- Dữ liệu tuần tự hóa (serialized data): danh sách các file có sẵn ở server được tuần tự hóa bằng pickle khi gửi.

2.1.4 Cách tổ chức cơ sở dữ liệu:

- Server:
  - Các file có sẵn trong server sẽ được lưu trữ vào folder "server_folder" ở server.
  - File tên "file.txt" chứa danh sách tên các file có sẵn và kích thước tương đối của chúng sẽ được lưu trữ ở server.
- Client:
  - File tên "input.txt" được lưu trữ ở client chứa danh sách các file cần tải.
  - Folder "output" là nơi các file cần tải sẽ được lưu trữ vào.

2.1.5 Các chức năng chính của chương trình

- Server:
  - Kết nối tuần tự với từng client.
  - Gửi danh sách các file có sẵn cho client.
  - Cho phép client tải tuần tự những file có sẵn trên server
- Client:
  - Kết nối với server
  - Gửi tên những file cần tải tới server
  - Nhận dữ liệu các file đã yêu cầu và lưu trữ vào folder "output"
  - Ngắt kết nối với server bằng tổ hợp phím "ctrl + c"

2.2 PHẦN 2:

2.2.1 Giao thức trao đổi giữa Client và Server:

- **Kết nối**
  Client sử dụng giao thức TCP/IP để kết nối đến server. Các bước cơ bản bao gồm:
  - **Khởi tạo kết nối:** Client tạo một socket và kết nối đến địa chỉ IP và cổng của server.

  - **Chấp nhận kết nối:** Server lắng nghe các kết nối đến và chấp nhận kết nối từ các client.

- **Gửi và Nhận Dữ liệu**

  - **Client:** Gửi yêu cầu tải file tới server, bao gồm tên file và mức độ ưu tiên.
  - **Server:** Gửi danh sách các file có sẵn (menu) và kích thước từng file đến client sau khi kết nối được thiết lập. Sau đó, server nhận và xử lý các yêu cầu tải file từ client, gửi dữ liệu file tương ứng.

2.2.2 Cấu trúc thông điệp:

- **Thông điệp từ Client tới Server:**
  Thông điệp yêu cầu tải file:

  - Tên file (name): Chuỗi ký tự (string).
  - Mức độ ưu tiên (priority): HIGH, CRITICAL, hoặc NORMAL (chuỗi ký tự).

- **Thông điệp từ Server tới Client:**
  Thông điệp từ Server tới Client

  - Số lượng file (NumberOfFile_Menu): Định dạng 4 byte.
  - Tên file: Độ dài tên file (4 byte) + Tên file (chuỗi ký tự).
  - Kích thước file (size): Định dạng 4 byte.

2.2.3 Kiểu dữ liệu của thông điệp:

- **Tên file:** Chuỗi ký tự (string).
- **Mức độ ưu tiên:** Chuỗi ký tự (string).
- **Số lượng file:** Số nguyên 4 byte (integer).
- **Kích thước file:** Số nguyên 4 byte (integer).

2.2.4 Cách tổ chức cơ sở dữ liệu:

- File tên "file.txt" chứa danh sách tên các file có sẵn và kích thước tương đối của chúng sẽ được lưu trữ ở server.
- File tên "input.txt" được lưu trữ ở client chứa danh sách các file cần tải.
- Folder "output" là nơi các file cần tải sẽ được lưu trữ vào.

- **Dữ liệu Client:**
  Client lưu trữ danh sách các yêu cầu tải file (preRequest) dưới dạng danh sách (list) các đối tượng Request. Mỗi đối tượng Request chứa:

  - Tên file (name).

  - Mức độ ưu tiên (priority).
  - Kích thước file (size).
  - Tiến độ tải file (progress).

- **Dữ liệu Server:**
  Server lưu trữ thông tin về các file có sẵn dưới dạng danh sách (list) các đối tượng menu_file (file.txt). Mỗi đối tượng menu_file chứa:

  - Tên file (name).
  - Kích thước file (size).

2.2.5 Các chức năng chính của chương trình

- **Kiểm tra Yêu cầu Tải File Tồn tại**
  Hàm check_request_downloadFile kiểm tra xem yêu cầu tải file với tên file cụ thể đã tồn tại trong danh sách preRequest hay chưa.

- **Đọc File Yêu Cầu Đầu Vào**
  Hàm Read_InputFile đọc file input.txt mỗi 2 giây để cập nhật các yêu cầu tải **file mới**. Nếu file yêu cầu tồn tại trong menu của server (file.txt), nó sẽ được thêm vào danh sách preRequest với các thông tin chi tiết.

- **Xử lý Yêu cầu Tải File**
  Hàm mainProcess thực hiện việc tải file từ server dựa trên danh sách yêu cầu preRequest.

3 Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng

3.1 Ngôn ngữ lập trình:

Python: Tận dụng các thư viện và module có sẵn của Python để xử lý mạng và đa luồng.

3.2 Thư viện và Module Python:

- Socket: Dùng để tạo kết nối mạng (TCP/IP) giữa client và server.
- Threading: Dùng để tạo và quản lý các thread, giúp xử lý đồng thời nhiều kết nối client.
- Collections: Cung cấp các cấu trúc dữ liệu nâng cao, sử dụng để lưu trữ thông tin về yêu cầu tải file.
- Time: Thư viện chuẩn của Python được sử dụng để làm việc với các thao tác liên quan đến thời gian, như tạo độ trễ (delay) hoặc tính toán thời gian. Cụ thể, trong mã nguồn của bạn, time.sleep() có thể được sử dụng để kiểm tra và đọc yêu cầu tải file từ input.txt mỗi 2 giây.

- OS và SYS: Dùng để tương tác với hệ điều hành và xử lý các thao tác liên quan đến hệ thống file.

3.3 Môi trường lập trình:

Python 3.12 (64 - bit): Môi trường lập trình chính để thực thi ứng dụng.

3.4 Framework hỗ trợ:

Chương trình tự xây dựng một framework đơn giản để xử lý giao tiếp giữa server và client, quản lý kết nối, xử lý yêu cầu và gửi/nhận dữ liệu. Framework này được xây dựng bằng cách sử dụng các thư viện chuẩn của Python như socket, time, threading, time,...

- Framework trong server:
  - Mô hình Server-Client: Chương trình server được thiết kế để lắng nghe các kết nối từ client qua socket TCP/IP. Mỗi khi có một client kết nối, server sẽ tạo một luồng (thread) để xử lý kết nối đó.
  - Xử lý đa luồng (Multi-threading): Server sử dụng luồng để xử lý nhiều kết nối client đồng thời, giúp server có thể xử lý nhiều yêu cầu từ các client khác nhau mà không bị gián đoạn.
  - Giao thức xử lý yêu cầu: Server nhận yêu cầu từ client, kiểm tra xem file có tồn tại trong menu hay không, sau đó gửi dữ liệu file tương ứng.
- Framework trong client:
  - Mô hình Client-Server: Client kết nối đến server qua socket TCP/IP và gửi các yêu cầu tải file đến server.
  - Kiểm tra và xử lý yêu cầu: Client đọc các yêu cầu từ một file (input.txt) và gửi chúng đến server, sau đó nhận dữ liệu file từ server và xử lý nó.
  - Kiểm tra định kỳ: Client sử dụng vòng lặp và time.sleep() để kiểm tra định kỳ các yêu cầu tải file từ file input.txt.

4 Hướng dẫn sử dụng các tính năng của chương trình:

4.1 Chuẩn bị môi trường:

Cài đặt Python 3.12 (64 - bit).

4.2 Khởi động Server:

- **Chạy Mã Nguồn Server:**
  - Mở một terminal và di chuyển đến thư mục chứa file sever_cmt.py.

  - Chạy lệnh sau để khởi động server: python sever_cmt.py

- **Kiểm Tra Server:**
  - Server sẽ lắng nghe các kết nối từ client và hiển thị địa chỉ IP của server.
  - Khi server nhận được kết nối từ client, nó sẽ gửi danh sách các file có sẵn (menu) đến client.

4.3 Khởi động Client:

- **Chạy Mã Nguồn Client:**
  - Mở một terminal khác và di chuyển đến thư mục chứa file client_cmt.py.
  - Chạy lệnh sau để khởi động client: python client_cmt.py
- **Nhập Địa Chỉ IP của Server:**
  - Khi chạy client, bạn sẽ được yêu cầu nhập địa chỉ IP của server. Nhập địa chỉ IP mà server đã hiển thị khi khởi động.
- **Gửi Yêu Cầu Tải File:**
  - Client sẽ đọc file input.txt mỗi 2 giây để cập nhật các yêu cầu tải file mới. Các yêu cầu này phải có định dạng: <tên_file> <mức_độ_ưu_tiên>
  - Ví dụ:
    * file1.txt HIGH
    * file2.txt NORMAL
- **Nhận và Lưu File:**
  - Client sẽ nhận dữ liệu file từ server và lưu lại trên máy local. Tiến độ tải file sẽ được cập nhật liên tục.

4.4 Các tính năng chính:

- **Kiểm Tra Yêu Cầu Tải File Tồn Tại:**
  - Sử dụng hàm check_request_downloadFile để kiểm tra xem yêu cầu tải file đã tồn tại trong danh sách preRequest hay chưa.
- **Đọc File Yêu Cầu Đầu Vào:**
  - Hàm Read_InputFile sẽ đọc file input.txt mỗi 2 giây để cập nhật các yêu cầu tải file mới.
- **Xử Lý Yêu Cầu Tải File:**
  - Hàm mainProcess thực hiện việc tải file từ server dựa trên danh sách yêu cầu pre-Request.

- **Xử Lý Kết Nối Từ Client:**
  - Server chấp nhận kết nối từ client và tạo một luồng (thread) để xử lý từng kết nối riêng biệt.
  - Gửi danh sách menu tới client khi kết nối được thiết lập.
  - Nhận yêu cầu từ client, kiểm tra yêu cầu có hợp lệ không và gửi dữ liệu tệp tin tương ứng.

4.5 Chạy chương trình:

- **Khởi Động Server:**
  python sever_cmt.py
- **Khởi Động Client:**
  python client_cmt.py

5 Bảng phân công công việc chi tiết:

Phần thực hiện	Người thực hiện	Công việc chi tiết
Phần 1	Phạm Thành Đạt	- Khởi tạo và quản lí server và client. - Đọc và xử lý các file. - Nhận và lưu trữ dữ liệu file. - Xử lý các ngoại lệ trong quá trình gửi và nhận file. - Hiển thị tiến độ tải file. - Điều khiển việc kết nối ở client.
Phần 2: Client.py	Nguyễn Ngọc Mai Xuân	- Tạo và quản lý socket kết nối tới server. - Khởi tạo các lớp (Classes) và các biến toàn cục. - Đọc và xử lí File (input.txt) - Xử lí tải file từ server - Cập nhật danh sách yêu cầu - Xử lí và hiển thị tiến độ tải file
Phần 2: Server.py	Thạch Ngọc Hân	- Khởi tạo và thiết lập server. - Quản lý Menu. - Xử lý kết nối từ Client. - Truyền tải tệp dựa trên mức độ ưu tiên. - Xử lý ngoại lệ và dọn dẹp. - Quản lý đa luồng. - Ghi nhận trạng thái kết nối.

5.1 Chi tiết công việc Phần 1:

- **Khởi tạo và quản lí server và client.**

  - Khởi tạo server để lắng nghe tuần tự kết nối từ client.
  - Khởi tạo client kết nối với server thông qua giao thức TCP/IP.

- **Đọc và xử lý các file.**
  - Đọc và gửi danh sách file ở server.
  - Đọc và gửi tên các file cần tải ở client.
  - Gửi dữ liệu file theo yêu cầu.

- **Nhận và lưu trữ dữ liệu file.**
  - Nhận dữ liệu file ở client.
  - Lưu trữ dữ liệu vào folder output.

- **Xử lý các ngoại lệ trong quá trình gửi và nhận file.**
  - Xử lý việc file không tồn tại.
  - Xử lý việc file đã được tải trước đó.

- **Hiển thị tiến độ tải file.**
  - Hiển thị tiến độ tải file ở client.
  - Thông báo file đã được tải thành công.

- **Điều khiển việc kết nối ở client và server.**
  - Hiển thị trạng thái kết nối ở server và client.
  - Xử lý các sự cố kết nối nếu có.
  - Xử lý việc ngắt kết nối ở client.

5.2 Chi tiết công việc Phần 2 (Server.py):

5.2.1 Tạo và Quản lý Socket Server:

- Tạo socket bằng cách sử dụng socket.socket().
- Lấy địa chỉ IP của máy chủ và cổng để cấu hình socket.
- Gán socket với địa chỉ và cổng bằng bind().
- Bắt đầu lắng nghe kết nối với listen().
- Hiển thị địa chỉ IP và cổng để các client có thể kết nối.

5.2.2 Quản lý Danh Sách File (Menu):

- Đọc danh sách file từ một file cấu hình (file.txt).
- Phân tích nội dung để tạo danh sách các file và kích thước của chúng.
- Sắp xếp danh sách file theo tên để dễ quản lý và gửi cho client.

5.2.3 Chấp Nhận Kết Nối từ Client:

- Chấp nhận kết nối
  - Sử dụng accept() để nhận kết nối từ client.
  - Tạo một luồng riêng để xử lý từng kết nối client, cho phép nhiều client kết nối đồng thời mà không ảnh hưởng đến nhau.
- Xử lý kết nối
  - Gửi danh sách file có sẵn tới client khi kết nối thiết lập.
  - Nhận yêu cầu tải file từ client và xử lý yêu cầu đó.
  - Cập nhật trạng thái kết nối của client (kết nối mới, ngắt kết nối, lỗi kết nối).

5.2.4 Gửi Menu File tới Client:

- Chuyển đổi số lượng file và thông tin về từng file thành các byte và gửi qua socket.
- Đảm bảo dữ liệu được gửi đầy đủ và chính xác.

5.2.5 Xử Lý Yêu Cầu Tải File:

- Nhận tên file, độ ưu tiên và kích thước từ client.
- Kiểm tra xem file có tồn tại không và đọc file từ vị trí yêu cầu.
- Gửi dữ liệu file cho client theo mức độ ưu tiên và kích thước yêu cầu.

5.2.6 Quản Lý Đa Luồng:

- Tạo một luồng mới cho mỗi kết nối client để xử lý độc lập.
- Quản lý và theo dõi trạng thái của các luồng để đảm bảo hiệu quả hoạt động.

5.2.7 Ghi Nhận Trạng Thái Kết Nối:

- Cập nhật và in ra thông tin về trạng thái kết nối của client.
- Xử lý các sự kiện kết nối cố gắng, lỗi kết nối hoặc ngắt kết nối.

5.3 Chi tiết công việc Phần 2 (Client.py):

5.3.1 Tạo và quản lí Socket:

- Khởi tạo một socket để kết nối tới server.
- Sử dụng địa chỉ IP và cổng để thiết lập kết nối (thư viện socket) thông qua giao thức TCP/IP

5.3.2 Khởi tạo các lớp (Classes) & các biến toàn cục:

Classes:

- Request: Lưu trữ thông tin về các yêu cầu tải file, bao gồm tên file, độ ưu tiên, kích thước file, và tiến độ tải.
- menu_file: Lưu trữ thông tin về các file có sẵn trên server, bao gồm tên file và kích thước của nó.

Biến toàn cục:

- preRequest[]: Danh sách lưu trữ các yêu cầu tải file đã được ghi nhận từ file đầu vào (input.txt).
- NumberOfFile_Menu: Biến lưu trữ số lượng file có sẵn trong menu của server.

5.3.3 Đọc và xử lí File (input.txt):

Định kỳ đọc file input.txt để cập nhật các yêu cầu tải file mới và kiểm tra tính hợp lệ của các yêu cầu trước khi gửi tới server

- check_request_downloadFile(preRequest, request_name): Kiểm tra xem yêu cầu tải file với tên file cụ thể đã tồn tại trong danh sách preRequest hay chưa. Nếu có, trả về True; nếu không, trả về False.
- Read_InputFile(s, Menu): Chạy liên tục để đọc file input.txt mỗi 2 giây. Nếu phát hiện yêu cầu mới mà chưa có trong danh sách preRequest, và file này có trong menu của server, yêu cầu sẽ được thêm vào danh sách preRequest.

5.3.4 Xử lí tải file từ server & cập nhật danh sách yêu cầu & xử lí và hiển thị tiến độ tải file

Xử lí việc tải file từ server đồng thời nó liên tục kiểm tra danh sách preRequest và gửi yêu cầu tải file (các file này nằm trong input.txt) với tên + độ ưu tiên tới server.

- Chờ trước khi bắt đầu xử lí để đảm bảo rằng các thiết lập ban đầu đã hoàn tất trước khi xử lý các yêu cầu tải xuống.
- Xử lí các yêu cầu bằng cách lấy số lượng file yêu cầu tải xuống có trong danh sách preRequest
- Kiểm tra và cập nhật trạng thái: Nếu kích thước của tệp cần tải xuống bằng 0 hoặc tệp đã được tải xuống hoàn toàn (progress == 1), hàm sẽ cập nhật trạng thái bằng cách gọi Status(preRequest,n) (là hàm hiển thị trạng thái khi file tải xong hoặc không tìm thấy file để tải) và tiếp tục sang yêu cầu tiếp theo.

- Xử lí tải xuống: Tạo đường dẫn đầy đủ cho tệp sẽ được tải xuống. Sau đó, đọc file theo dạng đọc binary: mở tệp với chế độ append binary (ghi thêm vào cuối tệp), nếu tệp chưa tồn tại, nó sẽ được tạo mới rồi lấy kích thước hiện tại của tệp (vị trí con trỏ file)
- Kiểm tra tiến độ tải xuống: Nếu kích thước hiện tại của tệp đã bằng hoặc lớn hơn kích thước dự kiến (preRequest[i].size), thì tệp đã được tải xuống hoàn toàn và tiến trình được đánh dấu hoàn thành (progress = 1).
- Nếu file chưa tải thì nó sẽ gửi thông tin yêu cầu theo thứ tự tới server bao gồm: Độ dài tên tệp, tên tệp, độ dài độ ưu tiên, độ ưu tiên, và kích thước hiện tại của tệp.

Nhận dữ liệu file (file mà client yêu cầu) và ghi vào ổ đĩa cục bộ, cập nhật tiến độ, in ra trạng thái tải và xử lí ngắt chương trình.

- Nhận dữ liệu từ server: Nhận 4 byte đầu tiên từ server, chứa thông tin về kích thước dữ liệu sẽ nhận. Nếu không nhận được dữ liệu vòng lặp sẽ dừng.
- Nếu nhận được bình thường thì sẽ tiếp tục nhận gói dữ liệu với kích thước tối đa của một gói dữ liệu mà client muốn nhận trong một lần gọi recv, giá trị này là 10 MB (1024 bytes x 10000) (10000 là số người lập trình có thể quy định) và số byte tối thiểu mà client có thể nhận được data_length - len(data)
- Kiểm tra gói dữ liệu nhận được: Nếu packet rỗng (không chứa dữ liệu), điều đó có nghĩa là kết nối có thể đã bị đóng hoặc server đã ngừng gửi dữ liệu. Trong trường hợp này, vòng lặp sẽ dừng với lệnh break.
- Xử lí ngắt chương trình: Bắt sự kiện KeyboardInterrupt để xử lý khi người dùng dừng chương trình bằng cách nhấn Ctrl+C.

5.3.5 Hàm chính của chương trình:

- Kết nối đến server
- In ra danh sách các file có sẵn trên server để người dùng có thể biết được các file nào có thể tải xuống.
- Khởi tạo các luồng (Threads): Một luồng để đọc file input.txt và cập nhật danh sách yêu cầu tải file và một luồng khác để xử lý việc tải file từ server và ghi dữ liệu vào máy client.
- Chương trình sẽ chạy liên tục, chờ nhận và xử lý các yêu cầu từ input.txt và thực hiện tải file từ server.