ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐỒ ÁN THỰC HÀNH MẠNG MÁY TÍNH

Họ và Tên:

Trương Quang Huy-23CLC02-23127530

Nguyễn Bách Khoa-23CLC02-23127066

Nguyễn Nhật Nam-23CLC02-23127092

Giảng viên:

Huỳnh Thụy Bảo Trân

Nguyễn Thanh Quân

Thành phố Hồ Chí Minh, ngày 19 tháng 12 năm 2024.

BIÊN BẢN PHÂN CÔNG VIỆC NHÓM

1. Thông tin chung:

STT	MSSV	Họ và Tên	Email
1	23127530	Trương Quang Huy	tqhuy23@clc.fitus.edu.vn
2	23127066	Nguyễn Bách Khoa	nbkhoa23@clc.fitus.edu.vn
3	23127092	Nguyễn Nhật Nam	nnnam23@clc.fitus.edu.vn

2. Bảng phân công công việc:

STT	Người phụ trách	Công việc được giao	Công Việc chưa hoàn thành	Khối lượng công việc	Mức độ hoàn thành
1	Trương Quang Huy	Làm TCP (phần 3, 5 của phần thang điểm) , hỗ trợ UDP, viết báo cáo phần mình làm	0%	34%	100%
2	Nguyễn Bách Khoa	Làm TCP (phần 1, 2, 4 của phần thang điểm) , hỗ trợ UDP, viết báo cáo phần mình làm	0%	33%	100%
3	Nguyễn Nhật Nam	Làm UDP (phần 6, 7 của phần thang điểm), hỗ trợ TCP, viết báo cáo phần mình làm	0%	33%	100%

MỤC LỤC

I. PHẦN I: TCP

1. Nhiều clients có thể nhận được danh sách các file từ Server và đóng kết nối bằng Ctrl-C

Đánh giá mức độ hoàn thành: 100%
Làm được:

- Nhiều clients có thể kết nối đồng thời đến server.
- Clients gửi lệnh LIST tới server để nhận danh sách file.
- Server xử lý lệnh LIST và trả về danh sách các file kèm theo kích thước.
- Clients có thể đóng kết nối bằng Ctrl-C mà không làm server bị treo.
- Xử lý tình huống khi danh sách file rỗng hoặc file_list.txt không tồn tại. Hiện server đẩy ra danh sách rỗng mà không gửi thông báo.

Hướng dẫn sử dụng:

1. Khởi chạy server bằng cách chạy server.py.
2. Khởi chạy clients bằng cách chạy client.py.
3. Nhập lệnh LIST tại client để nhận danh sách file từ server.
4. Đóng client bằng Ctrl-C.

Giải thích chức năng:

- **Server**:
  - **Hàm**: handle_client
  - **Cách thực**: Nhận lệnh LIST, duyệt qua danh sách file từ file_list.txt và gửi danh sách file về client.
- **Client**:
  - **Hàm**: fetch_file_list
  - **Cách thực hiện**: Gửi lệnh LIST đến server và nhận danh sách file.

2. Hiển thị phần trăm download các chunk của file

Đánh giá mức độ hoàn thành: 100%
Làm được:

- Tiến trình tải mỗi chunk được hiển thị tỉ lệ phần trăm.
- Tiến trình tải được cập nhật liên tục trong thời gian thực.

Hướng dẫn sử dụng:

1. Nhập lệnh download tại client để tải file (vd: file được cung cấp trong danh sách).
2. Quan sát dòng hiển thị tiến trình tải chunk theo từng phần trăm.

Giải thích chức năng:

- Client:
    - **Hàm**: print_progress
    - **Cách thực hiện**: Cập nhật và in dòng tiến trình tải chunk theo từng phần trăm.

3. 5s quét input.txt 1 lần

Đánh giá tiến độ, mức độ hoàn thành : 100%.
Hàm liên quan: main và read_input_file

- ***Cách hoạt động:***
    - Trong hàm main, vòng lặp while True được sử dụng để liên tục kiểm tra danh sách các file cần tải trong input.txt.
    - Hàm read_input_file:
        - Đọc nội dung từ file input.txt.
        - Kiểm tra xem file nào trong danh sách chưa được tải xuống (không có trong processed_files) và thêm vào danh sách new_files.
    - Sau khi đọc xong danh sách file, main tải các file mới bằng cách gọi download_file.
    - Sau mỗi lần tải, chương trình ngủ 5 giây (time.sleep(5)) trước khi quét lại.

4. Client download được chunk đầu tiên của file thành công

Đánh giá mức độ hoàn thành: 100%
Làm được:

- Client gửi lệnh DOWNLOAD để tải chunk đầu tiên của file.
- Server xử lý đúng chunk đó, mở file, đọc chunk để trả về.
- Client nhận và lưu chunk đầu tiên vào bộ nhớ có độ chính xác cao.

Hướng dẫn sử dụng:

1. Nhập tên file muốn tải từ danh sách.
2. Client gửi lệnh DOWNLOAD đến server với tham số offset bằng 0 và chunk_size là kích thước chunk mong muốn.

Giải thích chức năng:

- Server:
  - **Hàm**: handle_client
  - **Cách thực hiện**: Xử lý lệnh DOWNLOAD, mở file đọc chunk dựa trên offset được cung cấp và trả chunk về client.
- Client:
  - **Hàm**: download_part

    Cách thực hiện: Gửi lệnh DOWNLOAD đến server để nhận chunk đầu tiên của file và lưu lại.

5. Nhiều client có thể tải đủ các chunk thành công từ Server

Đánh giá mức độ hoàn thành: 100%

Hàm liên quan:

1. **Trên Client:** download_file và các download_part (trong luồng con - threading.Thread)
2. **Trên Server:** handle_client.

- ***Cách hoạt động:***
  - **Phía Client:**
    - Hàm download_file chia file cần tải thành 4 phần bằng nhau (dựa trên kích thước file_size // 4).
    - Với mỗi phần, tạo một luồng con (threading.Thread) gọi hàm download_part.
    - Hàm download_part:
      - Gửi yêu cầu DOWNLOAD kèm theo tên file, offset (vị trí bắt đầu), và kích thước chunk đến server.
      - Nhận dữ liệu từ server từng phần (chunk) qua socket, cập nhật tiến trình vào danh sách progress.

- **Phía Server:**
    - Hàm handle_client lắng nghe yêu cầu DOWNLOAD từ Client.
    - Dựa trên thông tin offset và chunk size, server đọc dữ liệu từ file và gửi về client.
    - Các kết nối client được xử lý độc lập thông qua luồng (threading.Thread).

- **Đảm bảo nhiều client tải đủ chunk:**
    - Sử dụng đa luồng (multi-threading) trên cả Client và Server.
    - Từng phần dữ liệu của file được tải song song, và server xử lý nhiều client cùng lúc**.**

6. Tập tin sau khi download phải đúng và đủ dung lượng

Đánh giá mức độ hoàn thành: 100%

Hàm liên quan:

1. **Client:** download_file.

- **Cách hoạt động:**
    - Hàm download_file thực hiện tải từng phần của file (chunk) và lưu vào danh sách parts.
    - Sau khi tất cả các phần được tải, hàm ghép chúng lại thành file hoàn chỉnh:
        - Sử dụng vòng lặp để ghi tuần tự từng phần từ parts vào file đầu ra (output_path).
        - Đảm bảo thứ tự và kích thước của từng phần khớp với dữ liệu gốc.
    - Kích thước file sau khi tải xong sẽ bằng kích thước gốc do server cung cấp (file_size).
    - Nếu có lỗi trong quá trình nhận hoặc ghép dữ liệu, tiến trình tải sẽ thất bại.

- **Đảm bảo tính toàn vẹn:**
    - Mỗi chunk được tải dựa trên thông số offset và kích thước chính xác từ server.

Các phần dữ liệu được lưu và ghép lại theo thứ tự để đảm bảo file tải về không bị lỗi

7. Kịch bản giao tiếp

1. Giao thức giao tiếp

- **Giao thức:** TCP (Transmission Control Protocol)
- **Minh chứng trong mã nguồn:**
  - ❖ **Trong file server.py:**
    - s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    - s.bind((SERVER, SERVER_PORT))
    - s.listen()
    - socket.AF_INET: Sử dụng IPv4.
    - socket.SOCK_STREAM: Sử dụng giao thức TCP.
  - ❖ **Trong file client.py:**
    - client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect((SERVER_IP, SERVER_PORT))

- socket.SOCK_STREAM: Xác nhận rằng client cũng sử dụng TCP.

2. Cấu trúc thông điệp và kiểu dữ liệu

- **Thông điệp từ client tới server:**
  - Dạng chuỗi (string) UTF-8.
  - Các lệnh hợp lệ gồm:
    - LIST: Yêu cầu danh sách file.
    - SIZE <Tên file>: Yêu cầu kích thước file.
    - DOWNLOAD <Tên file> <Offset> <Chunk size>: Yêu cầu tải một phần file.
- **Thông điệp từ server tới client:**
  - Dạng chuỗi (string) UTF-8 cho phản hồi lỗi hoặc thông tin (danh sách file, kích thước).
  - Dạng byte (binary) cho dữ liệu file.

3. Tổ chức cơ sở dữ liệu

- **File file_list.txt:**
  - Lưu danh sách file và kích thước tương ứng trên server.

- Định dạng: <Tên file> <Kích thước (MB)>

- **File hệ thống:**
  - Server đọc dữ liệu file trực tiếp từ hệ thống tệp (filesystem).
  - Không có cơ sở dữ liệu phức tạp.

4. Chứng minh từ mã nguồn

- **Cấu trúc thông điệp:**
  - handle_client (server): Phân tích cú pháp thông điệp từ client bằng split().
  - send và recv (socket): Truyền và nhận thông điệp giữa client và server.
- **Tổ chức cơ sở dữ liệu:**
  - read_file_list (server): Đọc danh sách file từ file_list.txt.
  - with open(file_name, "rb") (server): Đọc dữ liệu file từ hệ thống tệp.

II. PHẦN II: UDP

1. Nhiều clients có thể nhận được danh sách các file từ Server

Đánh giá mức độ hoàn thành: 100%

Làm được:

- Server xử lý yêu cầu LIST và trả về danh sách file cùng kích thước.
- Client gửi lệnh LIST để nhận danh sách file.
- Server xử lý tình huống khi file file_list.txt không tồn tại, trả về thông báo lỗi.

Hướng dẫn sử dụng:

1. Chạy udp_server.py để khởi động server.
2. Chạy udp_client.py để khởi động client.
3. Nhập lệnh LIST từ client để nhận danh sách file từ server.

Giải thích chức năng:

- Server:
  - Hàm: read_file_list và xử lý lệnh trong main.
  - Cách thực: Đọc file file_list.txt, gửi danh sách file về client.
- Client:

- Hàm: list_files.
- Cách thực: Gửi lệnh LIST tới server và nhận danh sách file, in ra màn hình.

2. Hiển thị phần trăm download các chunk của file

Đánh giá mức độ hoàn thành: 100%

Làm được:

- Client hiển thị tiến trình tải của từng chunk dưới dạng phần trăm hoàn thành.
- Tiến trình được cập nhật liên tục trên màn hình console.

Hướng dẫn sử dụng:

1. Nhập lệnh DOWNLOAD <file_name> <offset> từ client để tải chunk.
2. Quan sát phần trăm tiến trình hiển thị trên màn hình**.**

Giải thích chức năng:

- Server:
  - Hàm: send_chunk.
  - Cách thực: Gửi chunk dữ liệu cho client, kèm seq_num và checksum để kiểm tra tính toàn vẹn.
- Client:
  - Hàm: print_progress.
  - Cách thực: Tính toán và hiển thị phần trăm tải chunk dựa trên số chunk đã nhận.

3. Client tải chunk đầu tiên của file thành công

Đánh giá mức độ hoàn thành: 100%

Làm được:

- Client gửi yêu cầu tải chunk đầu tiên của file với offset = 0.
- Server xử lý, đọc chunk từ file và gửi về client.
- Client lưu chunk đầu tiên và kiểm tra checksum thành công.

Hướng dẫn sử dụng:

1. Gửi lệnh DOWNLOAD <file_name> 0 từ client.
2. Chunk đầu tiên sẽ được tải về và lưu tại thư mục đầu ra (output).

Giải thích chức năng:

- Server:
  - Hàm: handle_download_request.
  - Cách thực: Mở file, đọc chunk từ vị trí offset, gửi chunk kèm seq_num và checksum về client.
- Client:
  - Hàm: download_chunk.
  - Cách thực: Nhận chunk, kiểm tra checksum và lưu vào file.

4. UDP Reliable Data Request (RDT)

Đánh giá mức độ hoàn thành: 100%

Làm được:

- Server gửi gói tin kèm checksum để kiểm tra tính toàn vẹn dữ liệu.
- Client gửi ACK để xác nhận đã nhận đúng chunk.
- Server tự động retry nếu không nhận được ACK hoặc gặp timeout.

Hướng dẫn sửa dụng:

1. Chạy server và client như trên.
2. Gửi lệnh DOWNLOAD <file_name> <offset> từ client để kiểm tra cơ chế gửi lại khi gặp lỗi.

Giải thích chức năng:

- Server:
  - Hàm: send_chunk.
  - Cách thực: Gửi gói tin, chờ ACK từ client, retry tối đa 5 lần nếu timeout.
- Client:
  - Hàm: download_chunk.
  - Cách thực: Nhận gói tin, kiểm tra checksum, gửi ACK để xác nhận.

5. Tải song song nhiều chunk của file

Đánh giá mức độ hoàn thành: 100%

Làm được:

- Client chia file thành nhiều chunk dựa trên kích thước CHUNK_SIZE.
- Sử dụng nhiều luồng (threads) để tải song song các chunk từ server.
- Giới hạn số luồng tải đồng thời bằng Semaphore để tránh quá tải.

Hướng dẫn sử dụng:

1. Chạy lệnh tải file trong client.
2. Quan sát các chunk được tải song song trên màn hình.

Giải thích chức năng:

- Client:
    - Hàm: download_file.
    - Cách thực:
        1. Tạo nhiều luồng để tải song song các chunk.
        2. Sử dụng Semaphore để giới hạn số luồng tải đồng thời.
        3. Chờ tất cả các luồng hoàn thành (đồng bộ).

6. Kịch bản giao tiếp

Giao thức trao đổi giữa client và server:

- Lệnh LIST:
    - Client gửi: Chuỗi ký tự LIST.
    - Server phản hồi: Danh sách file có sẵn, mỗi file một dòng (dạng: <file_name> <file_size>).
- Lệnh SIZE <file_name>:
    - Client gửi: Chuỗi ký tự SIZE <file_name>.
    - Server phản hồi: Kích thước file dưới dạng số nguyên (byte). Nếu file không tồn tại, server trả về chuỗi ERROR.
- Lệnh DOWNLOAD <file_name> <offset>:
    - Client gửi: Chuỗi ký tự DOWNLOAD <file_name> <offset>.

- Server phản hồi:
    1. Chunk dữ liệu kèm seq_num và checksum.
    2. Nếu chunk cuối cùng, server gửi chuỗi END.

  Cấu trúc thông điệp:

- Dữ liệu gửi từ server:
    - Header: seq_num checksum (dạng chuỗi, cách nhau bởi dấu cách).
    - Body: Dữ liệu chunk.
    - Tổng gói tin: Header | Body (ngăn cách bởi ký tự |).
- Dữ liệu gửi từ client:
    - ACK: Chuỗi ký tự ACK <seq_num> để xác nhận nhận đúng chunk.

  Kiểu dữ liệu:

- Thông điệp trao đổi: Chuỗi ký tự (str) được mã hóa thành byte (bytes).
- Dữ liệu chunk: Dạng nhị phân (bytes).

Cách tổ chức cơ sở dữ liệu:

- Server lưu trữ danh sách file trong file_list.txt, mỗi dòng chứa:

  <file_name> <file_size>
- Các file thực tế được lưu trong thư mục cùng cấp với chương trình server.

7. Tổng kết

Mức độ hoàn thành: 100%

Chương trình đã thực hiện được tất cả các yêu cầu chính, bao gồm:

1. Tải danh sách file từ server.
2. Tải các chunk với tiến trình hiển thị.
3. Kiểm tra tính toàn vẹn dữ liệu và retry khi lỗi.
4. Tải song song nhiều chunk.

III. MÔI TRƯỜNG LẬP TRÌNH VÀ FRAMEWORK

Môi trường lập trình:

- Ngôn ngữ lập trình: Python (mã nguồn viết bằng Python).
- Phiên bản Python: Phiên bản Python 3.7 trở lên được khuyến nghị để đảm bảo hỗ trợ đầy đủ các tính năng như threading và các phương thức của thư viện socket.
- Hệ điều hành: Linux, macOS hoặc Windows đều hỗ trợ thực thi các tệp Python này.
- Một số lệnh liên quan đến đường dẫn file và quyền truy cập có thể cần điều chỉnh trên từng hệ điều hành.

Framework và thư viện:

- Thư viện tiêu chuẩn Python (không cần cài thêm):

- socket: Được sử dụng để tạo và quản lý các kết nối TCP giữa client và server.
- os: Quản lý hệ thống tệp, như kiểm tra và tạo thư mục (output), đọc file (file_list.txt, input.txt).
- threading: Hỗ trợ chạy đa luồng, giúp xử lý nhiều client đồng thời (trong tcp_server.py) và tải nhiều phần file song song (trong tcp_client.py).
- time: Được sử dụng để quản lý thời gian nghỉ (sleep) và tính toán tiến độ tải file.

Công cụ hỗ trợ:

- Trình thông dịch Python: Yêu cầu cài đặt Python trên máy để thực thi mã. Trình soạn thảo hoặc IDE:

- Visual Studio Code (VS Code): Có hỗ trợ mạnh mẽ cho Python, bao gồm debug, quản lý thư viện, và terminal tích hợp.
- PyCharm: Hỗ trợ gợi ý mã, quản lý dự án và debug.
- Notepad++ hoặc Sublime Text: Các trình soạn thảo nhẹ nếu không cần IDE phức tạp.

IV. CÁC NGUỒN TÀI LIỆU THAM KHẢO

1. GeeksforGeeks

https://www.geeksforgeeks.org/differences-between-tcp-and-udp/

2. TutorialsPoint

https://www.tutorialspoint.com/data_communication_computer_network/transmission_control_protocol.htm

https://www.tutorialspoint.com/data_communication_computer_network/user_datagram_protocol.htm

3. **Stack Overflow**

   https://stackoverflow.com/questions/5970383/difference-between-tcp-and-udp

4. **ChatGPT**

   https://chatgpt.com