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

Lập Trình Socket

Thành viên gồm :
Chiều Hòa Thuận - 21127698

Contents

I. Thông tin các thành viên

Họ tên: Chiều Hòa Thuận
MSSV: 21127698

II. Mức độ hoàn thành 90%

III. Kịch bản giao tiếp

1. Giao thức trao đổi giữa Client và Server

- **Giao thức tầng vận chuyển**: UDP (User Datagram Protocol)
- **Giao thức tầng ứng dụng tùy chỉnh (Application protocol)** do sinh viên định nghĩa.

Các loại thông điệp trao đổi:

Loại thông điệp	Gửi từ	Ý nghĩa
LIST_FILES	Client → Server	Yêu cầu danh sách file có sẵn
GET_FILE_SIZE:<filename>	Client → Server	Yêu cầu kích thước file
REQUEST_FILE:<filename>:<offset>:<size>	Client → Server	Yêu cầu một phần (chunk) của file
ACK:<filename>:<offset>	Client → Server	Gửi xác nhận (ACK) chunk đã nhận
FILE_LIST:\n<name (size)>\n...	Server → Client	Phản hồi danh sách file
FILE_SIZE:<bytes>	Server → Client	Phản hồi kích thước file
FILE_CHUNK:<file>:<offset>:<size>:[DATA]	Server → Client	Gửi chunk của file
ERROR:...	Server → Client	Thông báo lỗi (file không tồn tại, cú pháp sai...)

2. Cấu trúc thông điệp

2.1. Client → Server

2.2. Server → Client

Thành phần	Dạng dữ liệu	Ghi chú
FILE_LIST	String	Danh sách các file từ input.txt trên Server
FILE_SIZE	String	Số byte của file yêu cầu
FILE_CHUNK	Binary	Một phần dữ liệu nhị phân (chunk) với tiêu đề định dạng
ERROR	String	Thông báo lỗi nếu file không tồn tại hoặc sai cú pháp

3. Kiểu dữ liệu của thông điệp

Trường	Kiểu dữ liệu	Mô tả
filename	String	Tên file (có thể có định dạng .zip, .bin)
offset, size	Integer	Byte offset và kích thước chunk yêu cầu
data	Bytes (binary)	Dữ liệu của chunk trong FILE_CHUNK
ack	String	ACK chunk (ACK:<filename>:<offset>)

4. Cách tổ chức dữ liệu

4.1. Trên Server

- File input.txt: chứa danh sách tên file và dung lượng (VD: File1.zip 20MB)
- Các file thực tế được đặt cùng thư mục với server.py

4.2. Trên Client

- Chạy request_file_list() để lấy danh sách file trước
- Gọi get_file_size() → nhận FILE_SIZE:<bytes>
- Tải dữ liệu bằng REQUEST_FILE:<...> từng chunk, mỗi chunk gửi ACK

5. Cơ chế đảm bảo truyền tin cậy (Reliable Data Transfer)

Cơ chế	Mô tả
ACK	Client gửi ACK:<filename>:<offset> để xác nhận đã nhận được chunk
Tái truyền	Server gửi lại tối đa 5 lần nếu không nhận được ACK từ Client
Kiểm tra hợp lệ dữ liệu	Chunk phải bắt đầu bằng "FILE_CHUNK", nếu không thì bỏ qua
Timeout	Cả Server và Client đều có timeout để tránh treo chương trình

6. Môi trường lập trình

- **Ngôn ngữ**: Python 3.x
- **Socket API**: Thư viện socket chuẩn trong Python (không dùng thư viện ngoài)
- **Môi trường phát triển**: Tự do (Windows, Linux, WSL, VM...)

IV. Hướng Dẫn Chạy Chương Trình

1. Khởi động Server

- Mở terminal (hoặc CMD) tại thư mục chứa server.py.
- Đảm bảo có file input.txt liệt kê danh sách file cho phép download, ví dụ:
  python
- Chạy lệnh:
  Python3 server.py
- Server sẽ khởi động tại địa chỉ 127.0.0.1:12345 và tự động cập nhật danh sách file mỗi 5 giây nếu input.txt thay đổi.

2. Khởi động Client

- Mở terminal thứ hai và chạy chương trình client.py:
  Python3 client.py
- Màn hình sẽ hiển thị các lựa chọn:
  - Lấy danh sách file từ Server
  - Ghi nhận tên file muốn tải từ input.txt
  - Tự động kiểm tra và tải file nếu có tên mới thêm vào

3. Cách thêm file cần tải trên Client

- Mở file input.txt trên máy Client bằng Notepad hoặc trình soạn thảo bất kỳ.
- Không xóa các dòng đã có trong input.txt, vì chương trình sẽ chỉ xử lý những dòng mới được thêm.
- Client sẽ quét lại file này mỗi 5 giây để kiểm tra tên mới cần tải.

4. Quá trình download file

- Với mỗi file cần tải, Client sẽ:
  - Gửi yêu cầu nhận kích thước file (GET_FILE_SIZE)
  - Tự chia thành 4 phần (chunk) đều nhau
  - Tạo 4 socket để tải song song các phần
  - Gửi ACK cho mỗi chunk nhận thành công
  - Hiển thị tiến độ tải theo phần trăm từng chunk
- Sau khi tải đủ 4 phần, Client sẽ tự động ghép lại thành file hoàn chỉnh và lưu về thư mục hiện tại.

5. Kết thúc chương trình

- Có thể dừng Client bất kỳ lúc nào bằng tổ hợp phím:
  Ctrl + C
- Server có thể tiếp tục chạy để phục vụ các client khác (nếu nâng cấp cho đa client), hoặc nhấn Ctrl + C để tắt Server.

V. Phân Công Công Việc

Họ tên	Mã số sinh viên	Nhiệm vụ
Chiều Hòa Thuận	21127698	- Thiết kế và lập trình cả hai phía: Server và Client - Xây dựng giao thức tầng ứng dụng cho truyền file qua UDP - Hiện thực cơ chế chia file thành 4 chunk và truyền song song - Cài đặt và xử lý logic xác nhận (ACK), resend nếu mất gói - Hiển thị tiến trình download từng chunk trên màn hình Client - Tổ chức dữ liệu và đọc file input.txt định kỳ - Viết báo cáo, kiểm thử chương trình và chuẩn bị phần vấn đáp

- Đây là đồ án thực hiện cá nhân (1 người), toàn bộ mã nguồn, thiết kế và báo cáo được thực hiện bởi sinh viên.
- Trong quá trình làm việc, sinh viên đã kiểm thử chương trình bằng mô hình: 1 máy thật chạy Server và 1 máy ảo chạy Client để mô phỏng kết nối mạng UDP và tình huống mất gói, đảm bảo đáp ứng tiêu chí của đề bài.

VI. Tài Liệu Tham Khảo

Tài liệu môn học "Lập trình mạng" – Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên, ĐHQG TP.HCM

Python Official Documentation – socket module
https://docs.python.org/3/library/socket.html

Real Python - Working with UDP sockets in Python
https://realpython.com/python-sockets/#udp-sockets

Tài liệu hướng dẫn sử dụng UDP Reliable Transfer từ giảng viên

ThinkBroadband & Hetzner – Nguồn cung cấp file mẫu để kiểm thử tốc độ tải:

a. https://www.thinkbroadband.com/download
b. https://ash-speed.hetzner.com/

Stack Overflow – Thảo luận và xử lý lỗi liên quan đến socket timeout và multithreading

TutorialsPoint – Hướng dẫn xử lý sự kiện Ctrl+C trong Python
https://www.tutorialspoint.com/how-do-i-catch-a-ctrlplusc-event-in-python