**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**


# BÁO CÁO ĐỒ ÁN THỰC HÀNH SỐ 1

**| Đề tài |**

## LẬP TRÌNH SOCKET

**| Sinh viên thực hiện |**

| | |
|---|---|
| **Ngô Bá Sỹ Nguyên** | **23120020** |
| **Nguyễn Thái Bảo** | **23120023** |
| **Nguyễn Phú Dinh** | **23120031** |

**| Giáo viên hướng dẫn |**

**Thầy Nguyễn Thanh Quân**

**Môn học: Mạng máy tính**

**Thành phố Hồ Chí Minh - 2025**

# MỤC LỤC

# DANH MỤC BẢNG

# 1. THÔNG TIN CỦA NHÓM

**Lớp**: 23CTT1A

**Môn học**: Mạng máy tính

**Giáo viên phụ trách**: thầy Nguyễn Thanh Quân

**Danh sách thành viên**:

| STT | Họ và Tên | MSSV | Ghi chú |
|---|---|---|---|
| 1 | Ngô Bá Sỹ Nguyên | 23120020 | Thành viên |
| 2 | Nguyễn Thái Bảo | 23120023 | Nhóm trưởng |
| 3 | Nguyễn Phú Dinh | 23120031 | Thành viên |

*Bảng 1. Danh sách thành viên*

# 2. ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

| Phần | Yêu cầu | Đánh giá mức độ hoàn thành | Ghi chú |
|---|---|---|---|
| I | Nhiều clients có thể nhận được danh sách các file từ Server và ctrl-c | 100% | |
| | Hiển thị percent download các chunk của file | 100% | |
| | 5s quét file input.txt 1 lần | 100% | |
| | Client có để download chunk đầu tiên của file thành công | 100% | |
| | Nhiều client có thể tải đủ các chunk thành công từ Server.<br>Tập tin sau khi download phải đúng và đủ dung lượng | 100% | |
| II | UDP download thành công 1 chunk kèm cơ chế rdt (reliable data transfer) | 100% | |
| | UPD download thành công toàn bộ file kèm cơ chế rdt (reliable data transfer) | 100% | |
| | Báo cáo | 100% | |

*Bảng 2. Đánh giá mức độ hoàn thành*

**Đánh giá chung mức độ hoàn thành đồ án**: 100%

# 3. KỊCH BẢN GIAO TIẾP CỦA CHƯƠNG TRÌNH

## 3.1. PHẦN 1: TCP

### 3.1.1. Giao thức trao đổi giữa client và server

Chương trình sử dụng giao thức TCP để trao đổi dữ liệu giữa client và server, đảm bảo việc truyền dữ liệu đáng tin cậy và tuần tự.

**SERVER:**

- Được khởi tạo và gắn với HOST là địa chỉ của máy và PORT là cổng được chỉ định có giá trị 12345.
- Lắng nghe các kết nối đến server (nhận tối đa 10 kết nối cùng lúc).
- Hiển thị ra màn hình giá trị HOST và PORT mà server đang chạy để client kết nối đến.
- Khi một client kết nối đến, server chấp nhận kết nối và mở một tiến trình (thread) để làm việc với client. Như vậy server có thể làm việc với nhiều client cùng lúc.
- Server sẽ chờ và nhận các yêu cầu được gửi đến từ client và tiếp tục xử lý dựa trên nội dung yêu cầu:
    - **Yêu cầu kết nối:** Server sẽ hiển thị thông báo đã kết nối thành công với client và gửi lại lời chào.c
    - **Yêu cầu danh sách file**: Server đọc file "filelist.txt" chứa danh sách các file có sẵn trên server và gửi cho client.
    - **Yêu cầu kích thước file**: Server gửi cho client giá trị kích thước của file được yêu cầu.
    - **Yêu cầu tải 1 chunk của file**: Server xử lý yêu cầu, gửi tới client một chunk dữ liệu của file được yêu cầu với kích thước <chunk_size> (tính bằng byte) bắt đầu từ vị trí <offset>.
    - **Xác nhận tải được file**: Server hiển thị thông báo client đã tải thành công file được yêu cầu.
    - **Yêu cầu ngắt kết nối**: Server thông báo client đã ngắt kết nối, đóng kết nối với client và kết thúc tiến trình.
- Sau khi mọi tiến trình hoàn tất, tức là đã xử lý xong hết cho các client, server đóng kết nối, chương trình kết thúc.

**CLIENT:**

- Nhập IP của server muốn kết nối đến.
- Được khởi tạo và kết nối đến địa chỉ của server.
- Gửi thông điệp **CONNECT** và chờ phản hồi xác nhận kết nối. Sau khi nhận phản hồi, cũng là lời chào từ server, hiển thị ra màn hình.
- Nhận danh sách các file có sẵn ở server.
- Tạo thư mục **OUTPUT** để lưu các file đã được tải về thành công từ server. Nếu thư mục đã tồn tại thì không cần tạo mới.
- Đọc danh sách file cần tải từ file *"input.txt"* và tiến hành tải tuần tự các file trong danh sách.
    - Nếu file đã được tải rồi thì chuyển sang file kế tiếp, không tải lại.

  - Nếu file cần tải không có trong danh sách các file có sẵn trên server thì thông báo ra màn hình rồi chuyển sang file kế tiếp.
  - Khi xác định file cần tải chưa được tải thì tiến hành tải file.
  - Client lấy kích thước file từ server, chia cho 4 để được kích thước 4 chunk cần tải, nếu kích thước không chia hết cho 4 thì phần dư sẽ được cộng vào kích thước của chunk cuối cùng.
  - Client mở 4 tiến trình tiến hành tải các chunk của file. Màn hình sẽ hiển thị tỉ lệ phần trăm tiến độ tải của 4 chunk song song trên màn hình.
  - Sau khi tải thành công đủ 4 chunk, tiến hành nối các chunk lại theo đúng thứ tự để tạo thành một file hoàn chỉnh. Như vậy file cần tải đã được tải thành công, hiển thị thông báo ra màn hình.
  - Client gửi thông báo đến cho server rằng đã tải file thành công.
- Tiến hành duyệt file "input.txt" mỗi 5 giây 1 lần để lấy được danh sách file vừa được thêm mới và thực hiện download từ server. Các file đã được duyệt thì không tải lại.
- Nếu có ngoại lệ xảy ra khi tải file, client thông báo lỗi và đóng kết nối.
- Khi hoàn thành download, client bấm tổ hợp phím "Ctrl + C" để kết thúc chương trình.

### 3.1.2. Cấu trúc thông điệp

**SERVER:**

- Tin nhắn chào mừng, xác nhận client kết nối thành công:
  Nội dung: "Welcome to the server, <địa chỉ_client>!\n"
- Danh sách file: Chuỗi văn bản, mỗi dòng chứa tên file và dung lượng.
  Ví dụ: 5MB.zip 5MB
- Dung lượng file: Số nguyên (tính bằng byte)
- Chunk dữ liệu: Chuỗi các bytes.

**CLIENT:**

- **CONNECT**
  - Ý nghĩa: Kết nối với Server.
  - Phản hồi: Tin nhắn chào mừng từ Server.
- **FILELIST**
  - Ý nghĩa: Yêu cầu danh sách file có sẵn trên Server.
  - Phản hồi: Danh sách file (dạng văn bản), mỗi dòng chứa tên file và kích thước.
- **SIZE <tên_file>**
  - Ý nghĩa: Yêu cầu kích thước của một file.
  - Dữ liệu: <tên_file> là tên file cần kiểm tra.
  - Phản hồi: Dung lượng file (tính bằng byte).
- **CHUNK <chunk_id>**
  - Ý nghĩa: Thông báo rằng Client đang yêu cầu tải một phần file (chunk).
  - Dữ liệu: <chunk_id> là thứ tự của chunk.
- **REQUEST <tên_file> <offset> <chunk_size>**
  - Ý nghĩa: Yêu cầu Server gửi một phần (chunk) của file.
  - Dữ liệu:
    - <tên_file>: Tên file cần tải.

        - <offset>: Vị trí byte bắt đầu của phần cần tải.
        - <chunk_size>: Kích thước của phần cần tải (tính bằng byte).
- **ACK <tên_file>**
    - Ý nghĩa: Thông báo rằng Client đã tải xong một file.
    - Dữ liệu: <tên_file> là tên file đã tải xong.
- **EXIT**
    - Ý nghĩa: Đóng kết nối.

### 3.1.3. Kiểu dữ liệu của thông điệp

- Dữ liệu văn bản: Tất cả các yêu cầu và phản hồi (trừ chunk dữ liệu) đều là chuỗi văn bản được mã hóa theo UTF-8.
- Dữ liệu nhị phân: Chunk dữ liệu được gửi dưới dạng byte.

### 3.1.4. Tổ chức cơ sở dữ liệu

**SERVER**:

- Thư mục lưu file:
    - Tất cả file được lưu trong thư mục files trên Server.
    - Thư mục này được quét mỗi lần Server khởi động để tạo danh sách file trong tập tin "filelist.txt".
- Danh sách file (filelist.txt):
    - File văn bản lưu tên và kích thước của từng file trong thư mục files.
    - Mỗi dòng trong filelist.txt có định dạng: *<tên_file> <dung_lượng_MB>*.

**CLIENT**:

- Danh sách file yêu cầu: Đọc từ tệp input.txt.
- Thư mục tải xuống: Các file tải về được lưu trong thư mục output.

## 3.2. Phần 2: UDP

### 3.2.1. Giao thức trao đổi giữa client và server

Chương trình sử dụng giao thức UDP (User Datagram Protocol), kết hợp cơ chế đảm bảo độ tin cậy (RDT - Reliable Data Transfer) thông qua việc sử dụng:

- ACK (Acknowledgment): Thông báo xác nhận gói tin được nhận đúng.
- NAK (Negative Acknowledgment): Thông báo xác nhận gói tin lỗi.
- Timeout: Định thời để xử lý gói tin mất mát hoặc không được xác nhận.
- Checksum: Đảm bảo tính toàn vẹn dữ liệu thông qua kiểm tra checksum.

**SERVER:**

- Được khởi tạo và gắn với HOST là địa chỉ của máy và PORT là cổng được chỉ định có giá trị 12345.
- Lắng nghe kết nối đến server (chỉ phục vụ duy nhất 1 client download).
- Hiển thị ra màn hình giá trị HOST và PORT mà server đang chạy để client kết nối đến.

- Thực hiện kết nối ban đầu (handshake): chờ thông điệp **CONNECT** từ client.
    - Nếu đúng: Hiển thị yêu cầu kết nối từ server ra màn hình → Gửi thông điệp chào mừng / xác nhận kết nối thành công cho client → chờ **ACK**.
    - Nếu sai: Yêu cầu kết nối không hợp lệ, kết thúc chương trình.
- Server sẽ chờ và nhận các yêu cầu được gửi đến từ client và tiếp tục xử lý dựa trên nội dung yêu cầu:
    - **Yêu cầu danh sách file**: Server đọc file "filelist.txt" chứa danh sách các file có sẵn trên server và gửi cho client.
    - **Yêu cầu kích thước file**: Server gửi cho client giá trị kích thước của file được yêu cầu.
    - **Yêu cầu tải file**: Server xử lý yêu cầu, gửi tới client file cần tải theo cơ chế gửi dữ liệu tin cậy, ở đây là "Stop and Wait".
    - **Xác nhận tải được file**: Server hiển thị thông báo client đã tải thành công file được yêu cầu.
    - **Yêu cầu ngắt kết nối**: Server thông báo client đã ngắt kết nối, đóng kết nối với client và kết thúc tiến trình.
- Chương trình kết thúc khi client đã tải xong và ngắt kết nối.

**CLIENT:**

- Nhập IP của server muốn kết nối đến.
- Được khởi tạo và kết nối đến địa chỉ của server.
- Gửi thông điệp **CONNECT** và chờ phản hồi xác nhận kết nối. Sau khi nhận phản hồi, cũng là lời chào từ server, hiển thị ra màn hình.
- Nhận danh sách các file có sẵn ở server.
- Tạo thư mục **OUTPUT** để lưu các file đã được tải về thành công từ server. Nếu thư mục đã tồn tại thì không cần tạo mới.
- Đọc danh sách file cần tải từ file *"input.txt"* và tiến hành tải tuần tự các file trong danh sách.
    - Nếu file đã được tải rồi thì chuyển sang file kế tiếp, không tải lại.
    - Nếu file cần tải không có trong danh sách các file có sẵn trên server thì thông báo ra màn hình rồi chuyển sang file kế tiếp.
    - Khi xác định file cần tải chưa được tải thì tiến hành tải file.
    - Client lấy kích thước file từ server, gửi yêu cầu và tiến hành tải file. Quá trình tải file diễn ra theo cơ chế gửi dữ liệu tin cậy **"Stop and Wait"**, có ghi lại sequence number và ack number.
    - Màn hình sẽ hiển thị tỉ lệ phần trăm tiến độ tải file trên màn hình.
    - Sau khi tải thành công, hiển thị thông báo ra màn hình.
- Tiến hành duyệt file "input.txt" mỗi 5 giây 1 lần để lấy được danh sách file vừa được thêm mới và thực hiện download từ server. Các file đã được duyệt thì không tải lại.
- Khi hoàn thành download, client bấm tổ hợp phím "Ctrl + C" để kết thúc chương trình.

### 3.1.2. Cấu trúc thông điệp

Mỗi gói tin có cấu trúc (struct) bao gồm:

- Checksum (32 byte đầu): Chuỗi hash MD5 dùng để kiểm tra tính toàn vẹn dữ liệu.

- Header (4 byte tiếp theo): Số thứ tự gói tin (seq_num).
- Payload (phần còn lại): Dữ liệu chính (phần nội dung yêu cầu hoặc phản hồi).

**SERVER:**

Thông điệp bên trong mỗi gói tin:

- Tin nhắn chào mừng, xác nhận client kết nối thành công:
  Nội dung: "Welcome to the server, <địa chỉ_client>!\n"
- Danh sách file: Chuỗi văn bản, mỗi dòng chứa tên file và dung lượng.
  Ví dụ: 5MB.zip 5MB
- Dung lượng file: Số nguyên (tính bằng byte)
- Chunk dữ liệu: Chuỗi các bytes.

**CLIENT:**

Thông điệp bên trong mỗi gói tin:

- **CONNECT**
  - Ý nghĩa: Kết nối với Server.
  - Phản hồi: Tin nhắn chào mừng từ Server.
- **FILELIST**
  - Ý nghĩa: Yêu cầu danh sách file có sẵn trên Server.
  - Phản hồi: Danh sách file (dạng văn bản), mỗi dòng chứa tên file và kích thước.
- **SIZE <tên_file>**
  - Ý nghĩa: Yêu cầu kích thước của một file.
  - Dữ liệu: <tên_file> là tên file cần kiểm tra.
  - Phản hồi: Dung lượng file (tính bằng byte).
  - Dữ liệu: <chunk_id> là thứ tự của chunk.
- **REQUEST <filename> <offset> <chunk_size> <seq_num>**
  - Ý nghĩa: Yêu cầu Server gửi một phần (chunk) của file.
  - Dữ liệu:
    - <tên_file>: Tên file cần tải.
    - <offset>: Vị trí byte bắt đầu của phần cần tải.
    - <chunk_size>: Kích thước của phần cần tải (tính bằng byte).
    - <seq_num>: số thứ tự gói tin
- **EXIT**
  - Ý nghĩa: Đóng kết nối.

### 3.1.3. Kiểu dữ liệu của thông điệp

- Checksum: Chuỗi MD5 dạng byte (32 byte).
- Số thứ tự gói tin (seq_num): Số nguyên 4 byte (unsigned int, big-endian).
- Payload:
  - Chuỗi UTF-8 (khi gửi thông tin như danh sách file, kích thước file).
  - Dữ liệu nhị phân (khi truyền tải nội dung file).

### 3.1.4. Tổ chức cơ sở dữ liệu

**SERVER**:

- Thư mục lưu file:
  - Tất cả file được lưu trong thư mục files trên Server.
  - Thư mục này được quét mỗi lần Server khởi động để tạo danh sách file trong tập tin "filelist.txt".
- Danh sách file (filelist.txt):
  - File văn bản lưu tên và kích thước của từng file trong thư mục files.
  - Mỗi dòng trong filelist.txt có định dạng: *<tên_file> <dung_lượng_MB>*.

**CLIENT**:

- Danh sách file yêu cầu: Đọc từ tệp "input.txt".
- Thư mục tải xuống: Các file tải về được lưu trong thư mục output.

# 4. MÔI TRƯỜNG LẬP TRÌNH VÀ FRAMEWORK HỖ TRỢ

## 4.1. Ngôn ngữ lập trình

Python: Toàn bộ mã nguồn của ứng dụng được viết bằng ngôn ngữ lập trình Python – một ngôn ngữ lập trình mạnh mẽ, dễ sử dụng và có nhiều thư viện hỗ trợ cho việc lập trình mạng.

## 4.2. Thư viện và module

- **socket**: Thư viện chuẩn của Python để lập trình mạng, hỗ trợ cả giao thức TCP và UDP.
- **threading**: Thư viện chuẩn của Python để tạo và quản lý các luồng (threads), giúp thực hiện các tác vụ đồng thời.
- **os**: Thư viện chuẩn của Python để tương tác với hệ điều hành, bao gồm các thao tác với file và thư mục.
- **sys**: Thư viện chuẩn của Python để tương tác với hệ thống, bao gồm việc xử lý các tham số dòng lệnh và thoát chương trình.
- **time**: Thư viện chuẩn của Python để làm việc với thời gian, bao gồm việc đo thời gian và tạo độ trễ.
- **signal**: Thư viện chuẩn của Python để xử lý các tín hiệu từ hệ điều hành, giúp thực hiện việc tắt chương trình một cách an toàn (sử dụng cho Ctrl + C).
- **struct**: Thư viện chuẩn của Python để làm việc với dữ liệu nhị phân, hỗ trợ đóng gói và giải nén dữ liệu.
- **hashlib**: Thư viện chuẩn của Python để tính toán các giá trị băm (hash), hỗ trợ việc kiểm tra tính toàn vẹn của dữ liệu.

## 4.3. Môi trường lập trình

- Visual Studio Code (VS Code): Trình soạn thảo mã nguồn mạnh mẽ và phổ biến, có nhiều tiện ích mở rộng hữu ích cho việc lập trình Python.
- Python 3.12.3

# 5. HƯỚNG DẪN SỬ DỤNG CÁC TÍNH NĂNG

## 5.1. Chuẩn bị môi trường

Cài đặt phiên bản Python phù hợp và install các thư viện / module cần thiết.

## 5.2. Khởi động

***Lưu ý***: Khởi động server trước khi khởi động client.

**Cách 1**: Điều hướng đến file server.py và client.py trong Visual Studio Code, chọn "Run Python File in Dedicated Terminal".

**Cách 2**:

- Mở một terminal và di chuyển đến thư mục chứa file server.py. Chạy lệnh python server.py để khởi động server.
- Mở một terminal khác và di chuyển đến thư mục chứa file client.py. Chạy lệnh python client.py để khởi động client.

## 5.3. Hướng dẫn tải file từ server

Sau khi client kết nối đến server, trên màn hình client sẽ hiện thị danh sách các file có sẵn trên server. Đó là danh sách các file mà client có thể tải.

Trước tiên, cần tạo một file "input.txt" trong cùng thư mục với client.py

Thêm vào cuối danh sách file cần download trong tập tin "input.txt" tên những file cần download bất cứ khi nào muốn (có thể thêm một hoặc nhiều tên file mỗi lần, có thể sử dụng chương trình khác để mở file và thêm vào thủ công).

Lúc này, client sẽ đọc danh sách file từ input.txt mỗi 5 giây và tải tuần tự từng file (không tải các file đã được duyệt) theo các bước sau:

- **Phần 1 - TCP**
  1. Mở 4 kết nối song song đến server để tải 4 phần của file cùng lúc.
  2. Chia file thành 4 phần và yêu cầu server gửi từng phần.
  3. Sau khi tải xong các phần, nối các phần lại thành file hoàn chỉnh và lưu vào thư mục output cùng thư mục với "client.py".
- **Phần 2 - UDP**
  1. Gửi yêu cầu tải file cần tải tới server.
  2. Sau khi tải xong, file được lưu vào thư mục output cùng thư mục với "client.py".

Đối với phần 1 – TCP, nhiều client có thể cùng lúc kết nối và tải file từ server, nhưng tối đa là 10 clients.

Đối với phần 2 – UDP, Server chỉ phục vụ duy nhất 1 Client download (1 socket duy nhất tại server).

Khi tải chunk / file, màn hình có thanh phần trăm hiển thị tiến độ download để tiện theo dõi.

## 5.4. Hướng dẫn kết thúc chương trình cho client

Sau khi hoàn thành tải các file yêu cầu, client có thể dừng chương trình bằng tổ hợp phím “Ctrl + C”.

# 6. BẢNG PHÂN CÔNG CÔNG VIỆC

| MSSV | Thành viên | Công việc đảm nhiệm | Mức độ hoàn thành |
|---|---|---|---|
| **23120020** | Ngô Bá Sỹ Nguyên | - Triển khai ý tưởng và cài đặt mã nguồn cho **Server** sử dụng **TCP** tại tầng Transport.<br>- Kiểm thử và đánh giá chương trình client TCP và server + client UDP.<br>- Kiểm tra báo cáo | 100% |
| **23120023** | Nguyễn Thái Bảo | - Triển khai ý tưởng và cài đặt mã nguồn cho **Server và Client** sử dụng **UDP** tại tầng Transport.<br>- Kiểm thử và đánh giá chương trình server + client TCP.<br>- Viết báo cáo | 100% |
| **23120031** | Nguyễn Phú Dinh | - Triển khai ý tưởng và cài đặt mã nguồn cho **Client** sử dụng **TCP** tại tầng Transport.<br>- Kiểm thử và đánh giá chương trình server TCP và server + client UDP.<br>- Kiểm tra báo cáo | 100% |

*Bảng 3. Bảng phân công công việc*

# 7. TÀI LIỆU THAM KHẢO

1. Giáo trình Mạng máy tính – Đại học Khoa học Tự nhiên – Đại học Quốc gia Thành phố Hồ Chí Minh
2. Computer Networking: A Top – Down Approach ($8^{th}$ edition) - J.F Kurose and K.W. Ross
3. Tài liệu môn học (Slides bài giảng tiếng Việt và tiếng Anh)
4. Tài liệu tham khảo Đồ án Socket (Moodle môn học)