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

BÁO CÁO BÀI TẬP SOCKET

Thành phố Hồ Chí Minh - 2024

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP

Đề tài:

LẬP TRÌNH SOCKET

Giáo viên hướng dẫn:

Thầy Nguyễn Thanh Quân

Thành viên nhóm:

Trần Trung Hiếu

Vũ Duy Thụ

Thành phố Hồ Chí Minh - 2024

THÔNG TIN THÀNH VIÊN

STT	MSSV	Họ và tên	Email
1	23120044	Trần Trung Hiếu	mkh.trantrunghieu@gmail.com
2	23120093	Vũ Duy Thụ	duythuvu1625@gmail.com

MỤC LỤC

PHẦN 1: ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

Mức độ hoàn thành: 93/100 %

Bài		Người thực hiện	Đánh giá	Chú thích
1	TCP	Vũ Duy Thụ	90%	Chưa xử lí được lỗi hiển thị % trên terminal, hiển thị không đạt 100% cho tất cả chunk nhưng file tải về thì vẫn đầy đủ dung lượng.
2	UDP	Trần Trung Hiếu	95%	Tốc độ ghép file bằng hàm merge_chunk vẫn đáng kể
Báo cáo		Trần Trung Hiếu	100%	

Qua đồ án này, các thành viên trong nhóm đã biết được:

+Cách hoạt động của TCP và UDP

+Ưu điểm khi xây dựng giao thức TCP

+Cách cơ bản tạo ra giao thức truyền dữ liệu tin cậy cho UDP

+Sử dụng ngôn ngữ python và thư viện socket

PHẦN 2: NỘI DUNG ĐỒ ÁN

BÀI 1: TCP

1.Ngôn ngữ lập trình: Python 3, với các thư viện

+Socket

+os: quản lý các file trên hệ điều hành

+struct: đóng gói và tạo kiểu dữ liệu packet

+threading: xử lý các tiến trình download file thành các luồng song song

2. Kịch bản giao tiếp của chương trình:

Client	**Server**
Mở socket client_socket	Mở server_socket, đọc các file vào file_list
Kết nối đến server để lấy danh sách file, sau đó hiển thị danh sách	Gửi danh sách các file đến server
Đọc những yêu cầu gửi trong file input.txt	
Kiểm tra có tồn tại file có sẵn chưa, nếu không xin	Nhận yêu cầu từ client, kiểm tra xem có tồn tại hay không
Nếu tồn tại file: Chia thành 4 threads để tải file song song	
Mỗi thread tạo socket và kết nối đến server	Kết nối đến socket của từng thread
Nhận chunk	Gửi chunk
Khi hoàn thành: join các thread lại	
Ghép các chunk đã nhận thành file hoàn chỉnh	Chờ yêu cầu tiếp theo từ client
Tiếp tục quá trình chuyển nhận file tiếp theo	

3. Hướng dẫn sử dụng

- Kiểm tra địa chỉ IPv4 mà server và client đang sử dụng, sau đó điền vào các trường sau:

Tại server:

```
def start_server(host='127.0.0.1', port=9000)
```

Tại client:

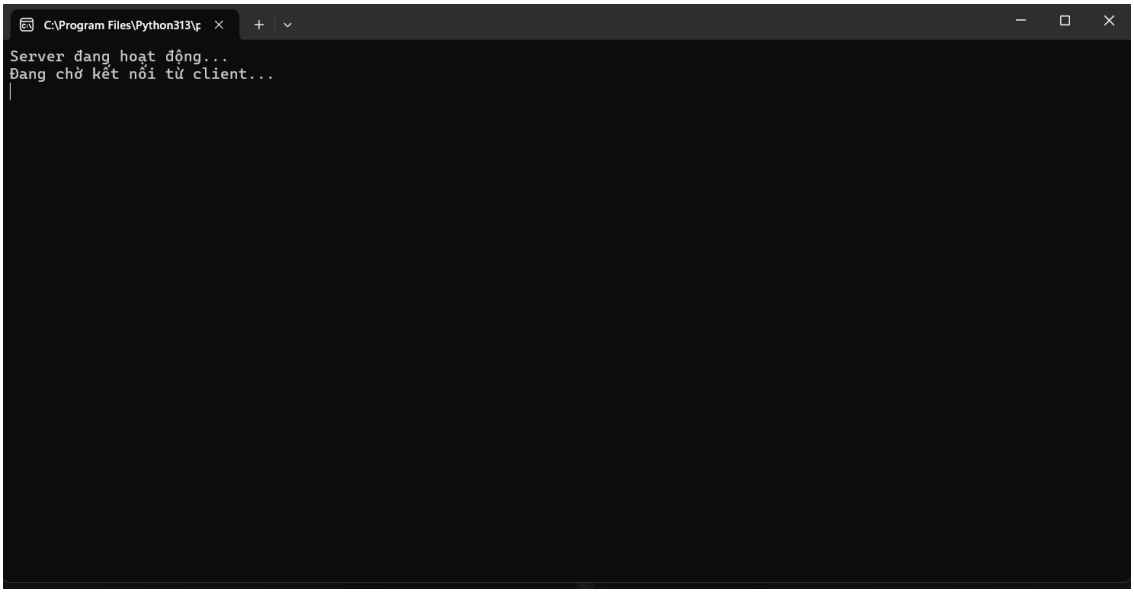
```
def main():
    downloaded_files = set()  # Tập hợp lưu các file đã tải xong

    try:
        # Kết nối tới server để lấy danh sách file
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.connect(('127.0.0.1', 9000))
```

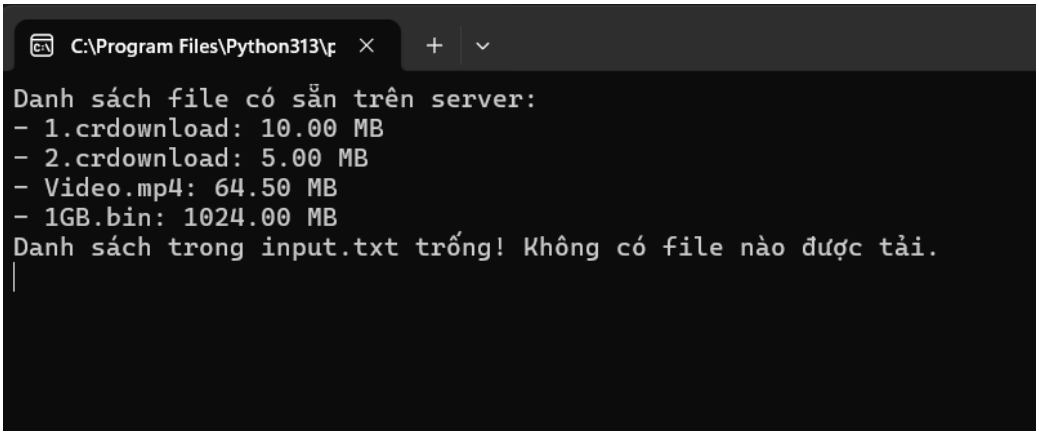
```
for file_name in input_files:
    if file_name in available_files and file_name not in downloaded_files:
        print(f"\nĐang tải {file_name}...")
        download_file('127.0.0.1', 9000, file_name, available_files[file_name])
        downloaded_files.add(file_name)
```

Hiện tại server và client đang chạy trên cùng 1 thiết bị, nên sử dụng địa chỉ loopback (127.0.0.1).

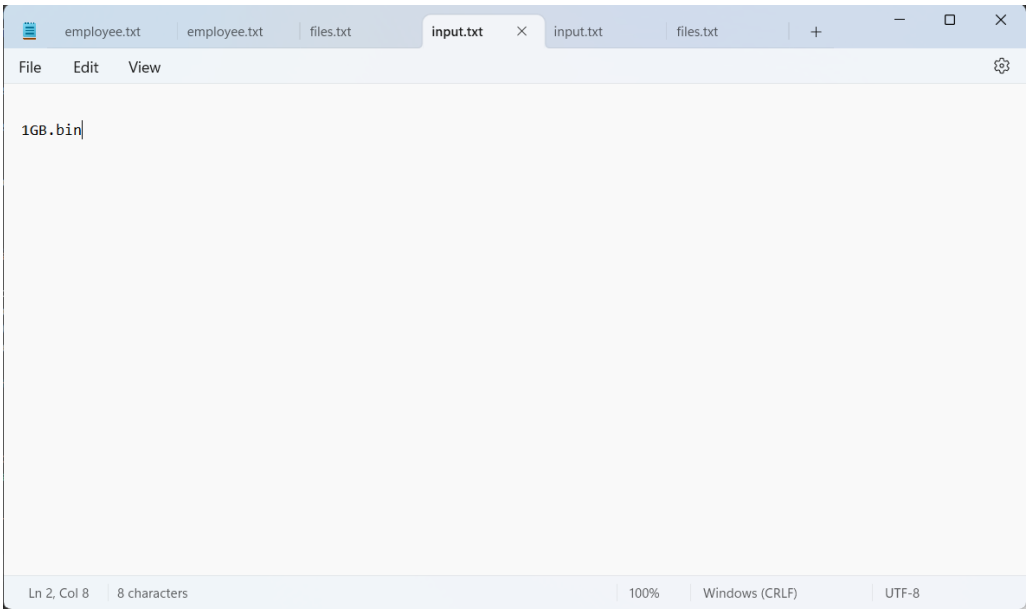
- Khởi chạy server, ta sẽ thấy kết quả như sau trên terminal:



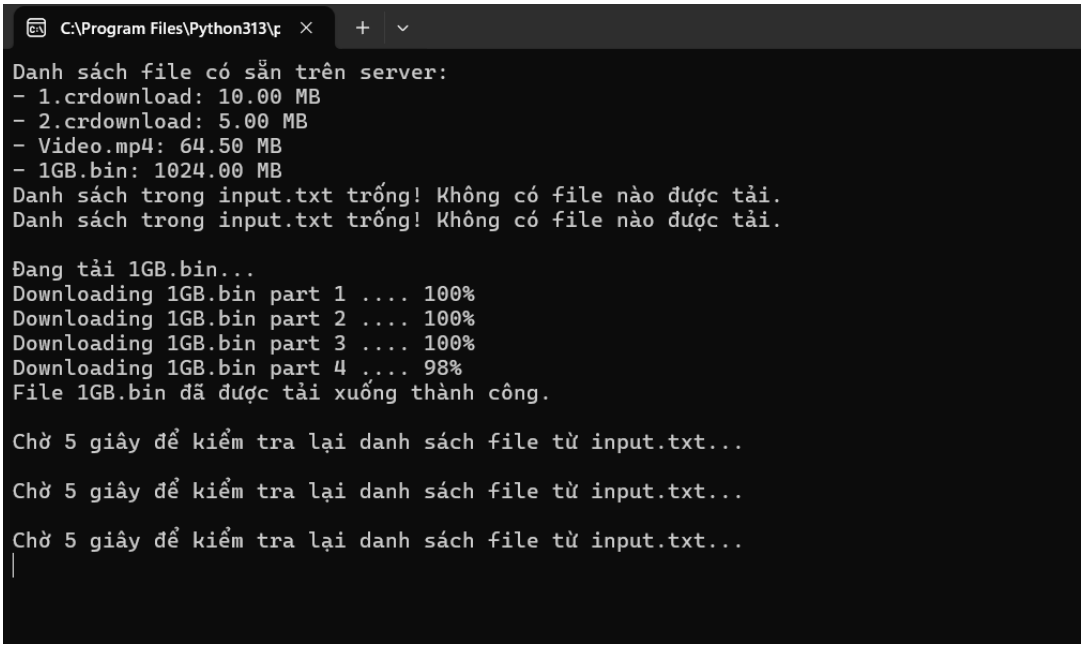
- Khởi chạy client, đảm bảo rằng 2 thiết bị đã được gán chung 1 địa chỉ đường mạng vào mã nguồn, sẽ được kết quả sau:



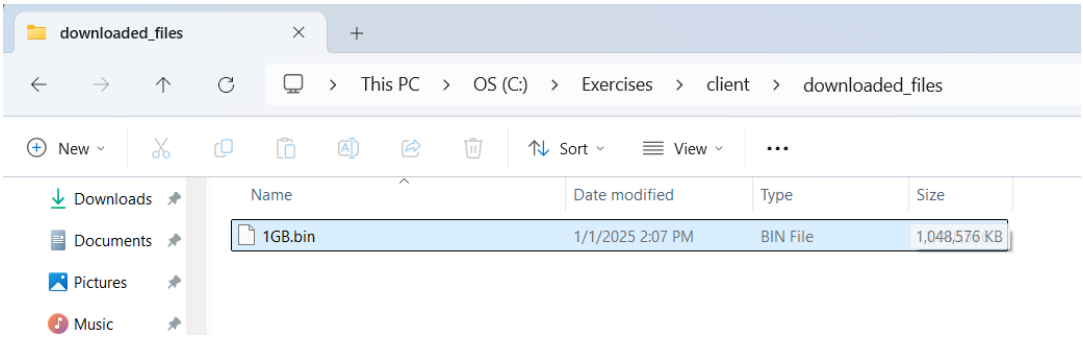
- Mở file input.txt tại client, điền tên file muốn tải và lưu lại.



- Ta sẽ thấy client quét thành công và bắt đầu tải file.

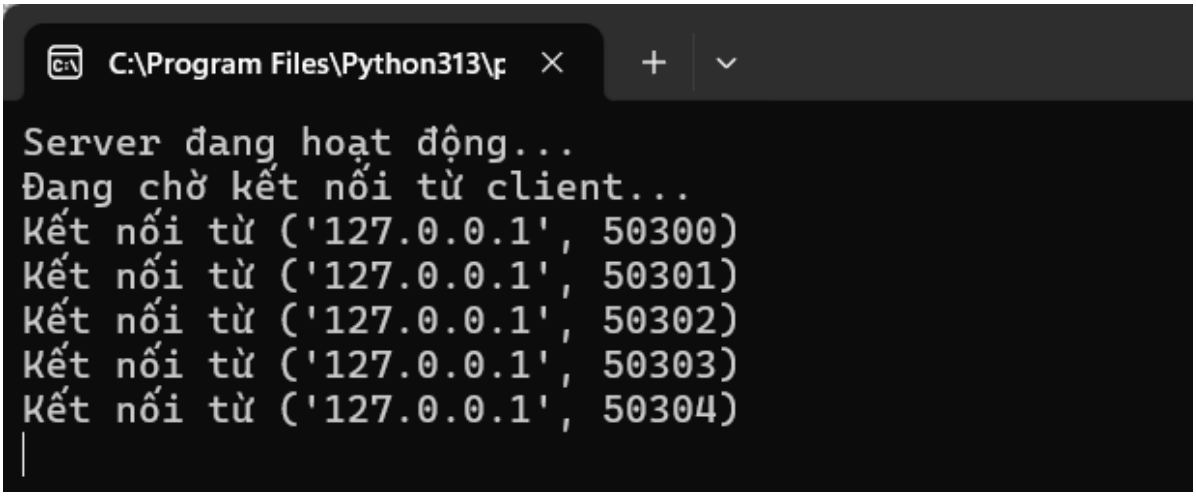


- Sau khi tải xong, các chunk được ghép lại với nhau và được lưu tại file: downloaded files tại client.



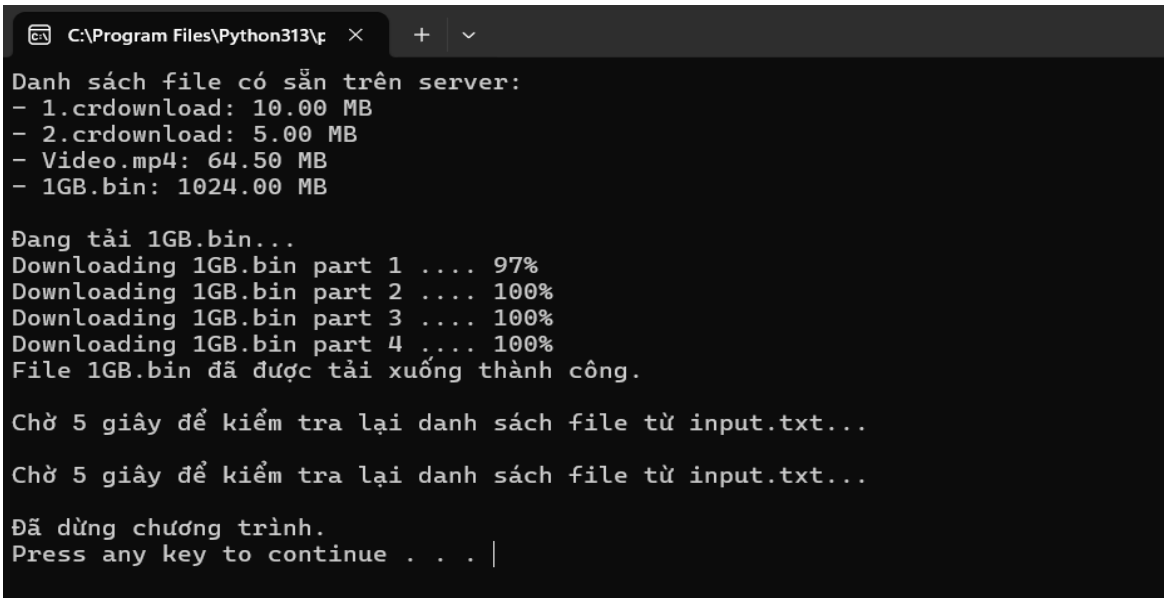
File được gửi đầy đủ nhưng chỉ hiển thị kết quả 98% ở chunk 4 là vì cách xử lí mã nguồn để chạy % gửi file, việc ghi đè lên các dòng trước đó sẽ khiến việc gửi file và ghép các chunk thực hiện xong nhưng % vẫn chưa chạy hết, do đó đây chỉ là lỗi hiển thị và không ảnh hưởng gì đến dung lượng file client nhận được.

- Cùng lúc đó, server cho kết quả như sau:



Server nhận kết nối từ client, mở them 4 kết nối song song để tải 4 chunk nên ta có kết quả như trên.

- Nếu muốn tiếp tục tải, chỉ cần ghi tiếp vào file input.txt ở client và chương trình sẽ lặp lại quá trình này cho đến khi nhấn Ctrl-C.



Bài 02: UDP

1.Ngôn ngữ lập trình: Python 3, với các thư viện

+Socket

+os: quản lý các file trên hệ điều hành

+struct: đóng gói và tạo kiểu dữ liệu packet

2.Cấu trúc gói tin:

Gói packet được định nghĩa với các thông số :

+Payload: giữ thông tin nhị phân của gói tin

+Length: kích cỡ của gói tin

+Seq: giá trị byte bắt đầu của gói tin

+Ack: giá trị byte đã đọc được sau khi client nhận được gói tin

+Chunk index: Thứ tự của gói tin

Gói packet có các hàm:

+Calculate checksum: đóng gói các thông số seq,ack,length,chunk index và payload sau đó tính checksum bằng toán tử bitwise &, giá trị thu được sẽ là 16bits

+Serialize: đóng gói và chuyển gói tin sang dạng nhị phân để có thể gửi qua socket (tương tự lệnh encode, nhưng dùng cho cả gói packet)

+Deserialize: phân giải ngược lại gói tin đã serialize thành một packet

```
class packet():
    #Init a package with payload, seq and ack numbers
    def __init__(self,payload,length,seq,ack,chunk_index):
        self.payload= payload
        self.length = length

        self.seq = seq;
        self.ack = ack;
        self.chunk_index = chunk_index
        self.checksum = 0

    def calculate_checksum(self)->int:
        # Combine header fields and payload into a single byte sequence for checksum
        header_data = struct.pack('!IIII', self.seq, self.ack, self.length,self.chunk_index)
        data = header_data + self.payload
        checksum = sum(data) & 0xFFFF  # Calculate checksum (16-bit)
        return checksum
```

3.Kịch bản giao tiếp của chương trình:

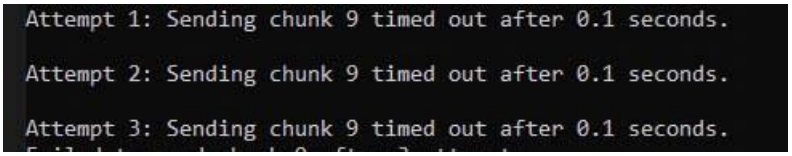
Client	**Server**
Mở socket client_socket	Mở server_socket, bind {IP_ADDRESS,PORT}
Gửi CONNECT_MSG cho server để server biết về địa chỉ client	Nhận CONNECT_MSG, gửi ACK để xác lập kết nối giữa hai bên, ghi nhận địa chỉ client
Nhận và hiển thị danh sách files	Gửi danh sách các file(filesList)
Client kiểm tra file input.txt và cập nhật yêu cầu nhận file ở dòng mới nhất	
	Server nhận được và kiểm tra xem file có tồn tại trên server hay không
Nhận xác nhận tồn tại file và kích cỡ file (nếu có từ server)	**Nếu tìm thấy file:** gửi kích cỡ file cho client
Tính số lượng chunk phải gửi/nhận	
Chờ file nhận và ACK tương ứng, có hiển thị phần trăm đã nhận. Nhận và phân tích gói packet đã nhận	Server lần lượt gửi từng chunk dựa theo kích cỡ được tính, có hiển thị phần trăm đã gửi. Với mỗi chunk được tạo thành một packet
+Nếu file nhận được hợp lệ, checksum đúng →ghi nhận và viết file vào folder download	
	+Nếu server không nhận được ACK hợp lệ, sẽ tiến hành gửi lại sau timeout và số lần max_retries đã thiết lập
Hoàn tất quá trình gửi nhận file và kiểm tra có thiếu chunk không, nếu có sẽ thông báo	
Tiến hành ghép các chunk đã nhận thành file hoàn chỉnh	Vào trạng thái đợi yêu cầu tiếp theo của client
Tiếp tục quá trình chuyển nhận file	

4.Các tính năng:

+Đảm bảo truyền dữ liệu:

-Thứ tự: seq number và chunk_index kiểm tra xem cả về thứ tự và điểm bắt đầu của file nhị phân của gói tin

-Mất dữ liệu: giải quyết bằng đặt socket.settimeout và max_retries



(quá trình gửi lại dữ liệu được thông báo ở phía server)

-Dữ liệu trùng lặp: ở phía client, nếu có gói tin hợp lệ, client sẽ gửi lại ack và chỉ ghi lại dữ liệu nếu như chưa tồn tại (dùng os.path.exists())

+Truyền file bằng các chunk:

-Tùy theo kích cỡ của dữ liệu, sẽ đưa chia thành số lượng chunk_count lớn nhỏ khác nhau. Nếu gói tin đủ nhỏ, sẽ không phải chia

+Đảm bảo nhiều kiểu dữ liệu: .txt, .pdf, .mp4 ,… do đều gửi dữ liệu theo dạng nhị phân

5.Hương dẫn sử dụng

+Trong source code server và client bạn có thể chỉnh sửa những thông số trước khi chạy :

```
INPUT_PATH   = "input.txt"
SERVER_IP    = "127.0.0.1"
SERVER_PORT = 12345
BUFFER_SIZE = 1024
SMALLEST_SIZE = 600002
PKT_SIZE = 30000
HEADER_SIZE = 18
```

-Với input path là tên file nhận yêu cầu

-server_ip và server_port là địa chỉ server

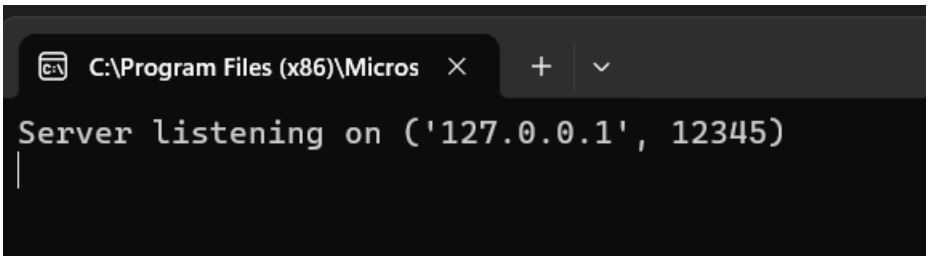
-buffer size: kích cỡ buffer cho các tin nhắn thông thường

-smallest_size: kích cỡ nhỏ nhất mà sẽ file sẽ không chia chunk (tính bằng byte)

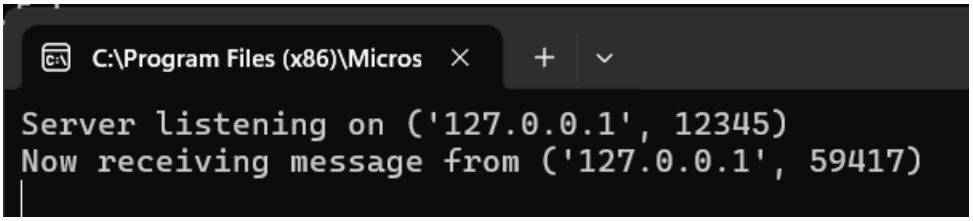
-pkt_size: kích cỡ một packet

-header_size: kích cỡ header gói tin

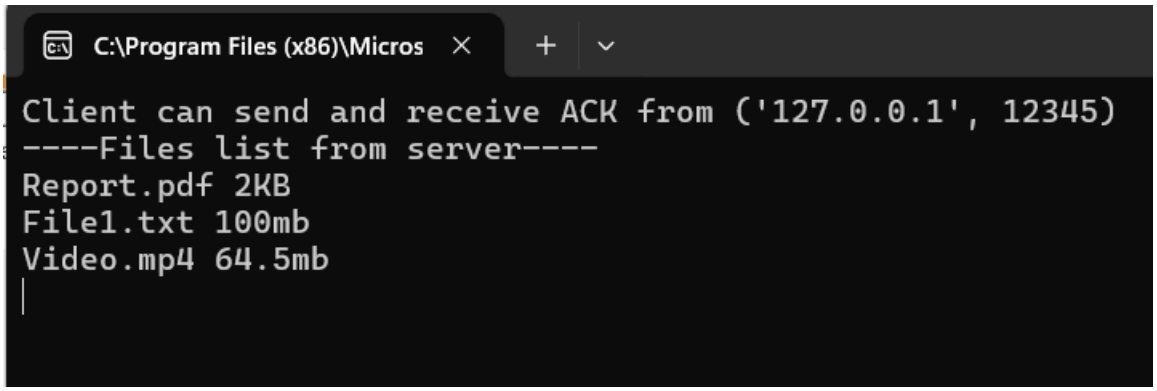
+Chạy source code server, bạn sẽ thấy địa chỉ hiện tại của server



+Sau đó chạy source code client, khi đó :

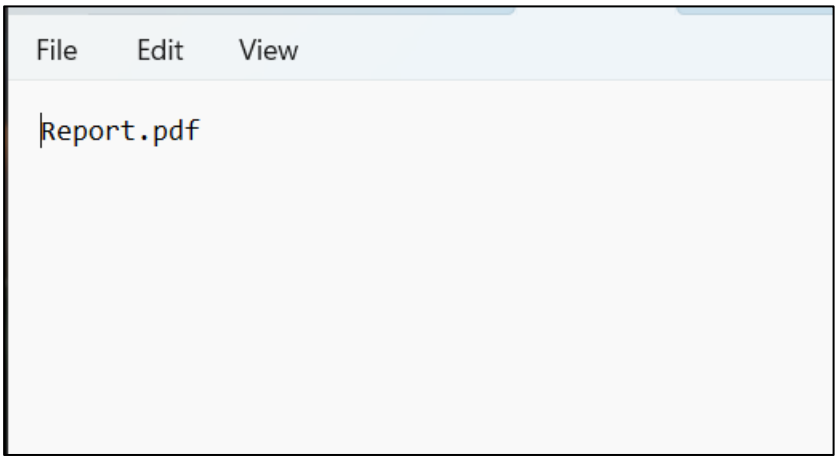


(server xác nhận kết nối từ client)

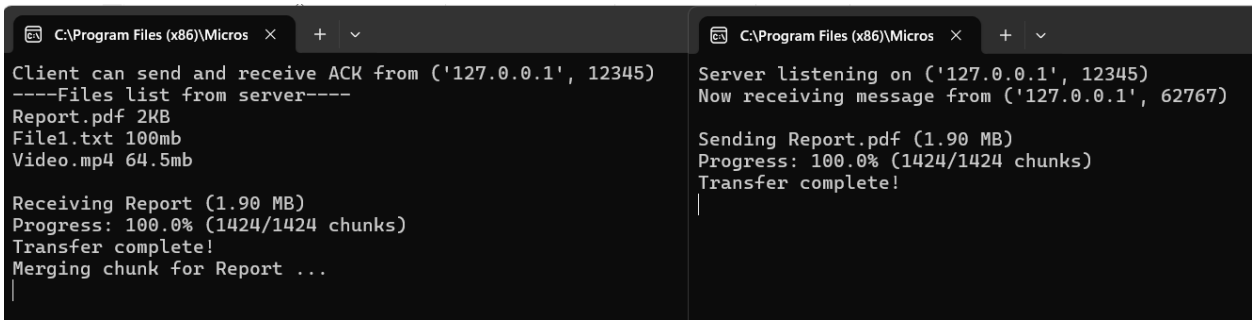


(Client xác nhận có thể liên lạc đến server và hiển thị các file đã ghi trong danh sách file)

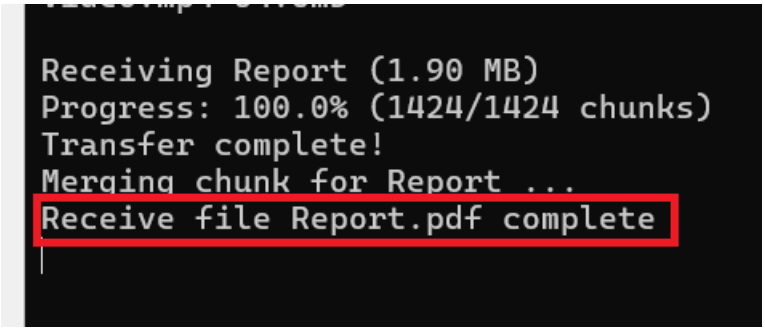
+Mở file input.txt phía client và nhập tên file muốn nhận



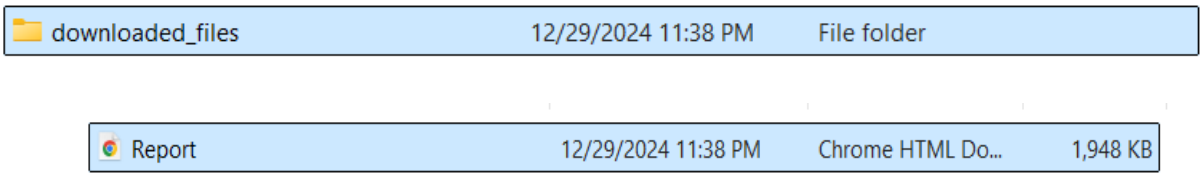
+Quá trình gửi sẽ hiện thỉ ở cả 2 phía



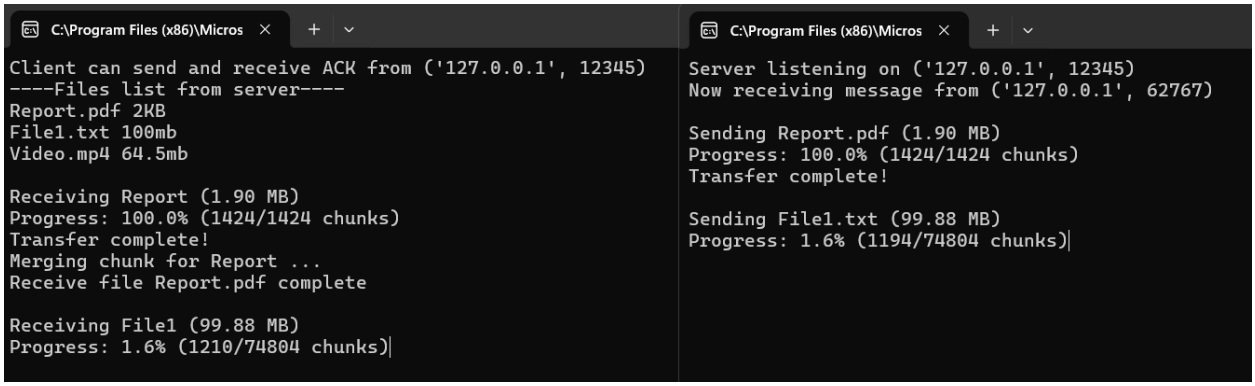
+Nếu nhận thành công sẽ có thông báo :



+Kiểm tra phía client ta sẽ thấy có thư mục download_files chứa file đã tải



+Sau đó tiếp tục nhập file cần tải tiếp theo (có thể nhập trong quá trình tải file trước)



+Trong thư mục download file ta có thể thấy các chunk riêng lẻ đang được tải

TÀI LIỆU THAM KHẢO

Mô hình reliable-data-transfer (rdt) cho UDP :

https://gaia.cs.umass.edu/kurose_ross/interactive/index.php

Thư viện socket :

https://docs.python.org/3/library/socket.html

Thư viện struct (tạo gói tin):

https://docs.python.org/3/library/struct.html