ĐỒ ÁN THỰC HÀNH 1 – LẬP TRÌNH SOCKET

MÔN MẠNG MÁY TÍNH

1. Quy định chung

- Đồ án được làm theo nhóm: mỗi nhóm tối đa **3** sinh viên, tối thiểu **2** sinh viên (trong trường hợp sĩ số lớp lẻ), sinh viên tự chọn nhóm (sử dụng nhóm thực hành đã đăng ký nếu có).

- **Các bài làm giống nhau sẽ đều bị điểm 0 toàn bộ phần thực hành tất cả các nhóm liên quan (dù có điểm các bài tập, đồ án thực hành khác)**.

- Môi trường lập trình: Tự do lựa chọn ngôn ngữ lập trình, tự do lựa chọn môi trường hệ điều hành: Windows, Unix/Linux, macOS

- Ngôn ngữ lập trình GV có thể hỗ trợ: C/C++, C#, Java, Python

- Thư viện hỗ trợ lập trình socket cho phép sử dụng như: Socket, CSocket, winsock. Tức là chỉ sử dụng các thư viện Socket do ngôn ngữ lập trình cung cấp, không dùng các thư viện bên ngoài liên quan đến Socket

2. Cách thức nộp bài

- **Nộp bài trực tiếp trên Website môn học, không chấp nhận nộp bài qua email hay hình thức khác.**
- Tên file: **MSSV1_MSSV2_MSSV3.zip** (Với MSSV1 < MSSV2 < MSSV3)

Ví dụ: Nhóm gồm 3 sinh viên: 2212001, 2212002, và 2212003, tên file nộp: **2212001_2212002_2212003.zip**

Cấu trúc file nộp gồm thư mục MSSV1_MSSV2_MSSV3 chứa các file:

1. **Report.pdf:** chứa báo cáo về bài làm
2. **Release**: thư mục chứa file thực thi của chương trình, **nếu có** (*.exe, nếu Python thì không cần)
3. **Source**: thư mục chứa source code của chương trình , yêu cầu nộp cả project đã xóa bỏ thư mục Debug và các file không cần thiết khác.. *Nhóm nào chỉ nộp file *.cpp và *.h và không biên dịch được thì bị 0 điểm.*

Lưu ý: Cần thực hiện đúng các yêu cầu trên, nếu không, bài làm sẽ không được chấm.

3. Hình thức chấm bài

Chấm vấn đáp vào thời điểm kết thúc phần thực hành.

4. Tiêu chí đánh giá

Về chương trình:

- Mục tiêu của đồ án này tập trung chủ yếu vào 2 vấn đề: lập trình socket, tự đưa ra được giao thức truyền file giữa client và server. Do đó các tiêu chí đánh giá dựa vào các chức năng chính được liệt kê trong yêu cầu của chương trình (có ghi chú thang điểm cho từng chức năng)

Về báo cáo:

- Thông tin của nhóm.
- Đánh giá mức độ hoàn thành từ 0 – 100% (Chú thích rõ những mục làm được,chưa làm được và còn bị lỗi)
- Kịch bản giao tiếp của chương trình: Giao thức trao đổi giữa client và server, cấu trúc thông điệp, kiểu dữ liệu của thông điệp, cách tổ chức cơ sở dữ liệu (nếu có).
- Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng.
- Hướng dẫn sử dụng các tính năng chương trình.
- Bảng phân công công việc và cho biết rõ ràng ai làm việc gì cách rõ ràng. Không ghi chung chung như chia đều công việc hay cùng làm mọi việc.
- Các nguồn tài liệu tham khảo.

Lưu ý: Trong báo cáo không dán các đoạn source code của chương trình. Mã chương trình chỉ trình bày nếu thật sự cần thiết và nếu cần minh họa cho các mô hình cài đặt hay các cơ chế đồng bộ (minh họa dạng mã giả, prototype hàm).

Về vấn đáp:

- Chuẩn bị thiết bị để chạy Client và Server trên 2 máy khác nhau (có thể dùng một máy thật và 1 máy ảo, ...), chương trình, báo cáo đầy đủ (không cần in).
- Trả lời các câu hỏi từ GV
- Trường hợp trả lời sai hoặc không trả lời được sẽ trừ trực tiếp điểm vào tổng điểm đồ án.

Lưu ý: Tất cả thành viên của nhóm phải tham gia buổi vấn đáp. Thành viên vắng mặt sẽ xử lý theo quy định sau:

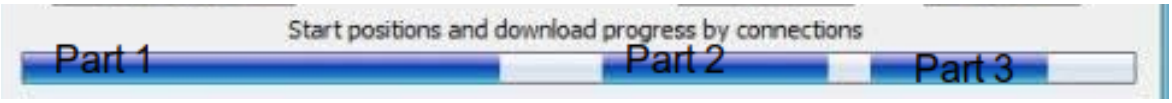
- Có phép (gửi email xin phép trước buổi vấn đáp): trừ điểm vấn đáp trực tiếp
- Không phép: 0 điểm đồ án.

5. Nội dung:

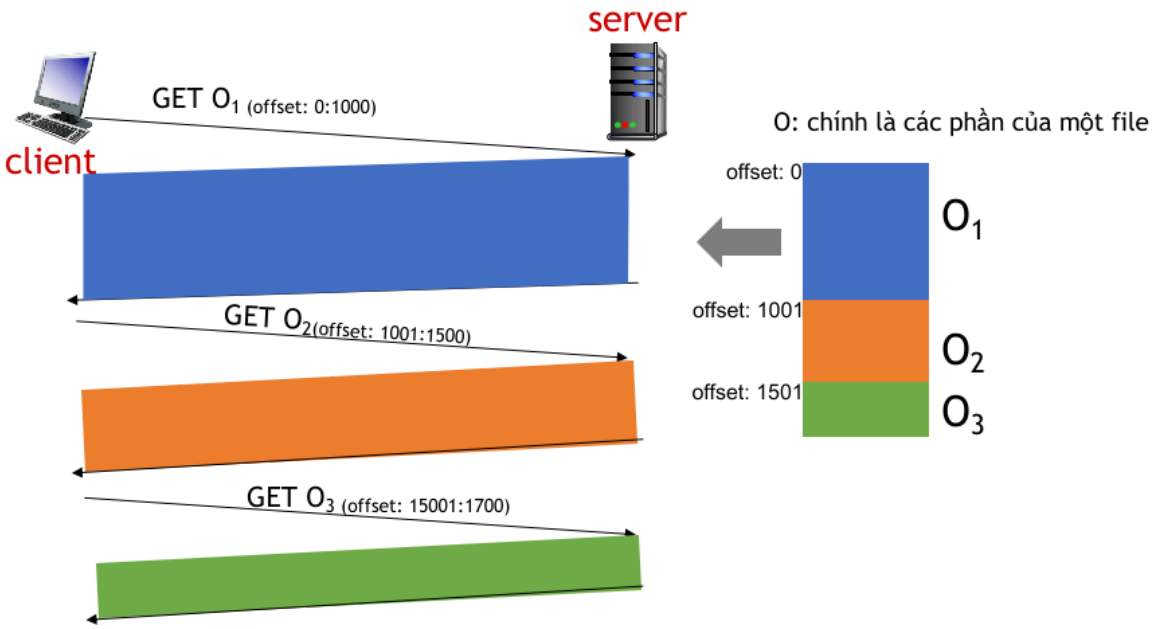
Lưu ý nội dung đồ án gồm có 2 phần cần được thực hiện.

Phần I (6.5đ): Sử dụng TCP tại tầng Transport

Viết chương trình Client/Server cho phép client download song song nhiều phần khác nhau(chunk) của một file từ Server.
Ví dụ phần mềm IDM bên dưới



Mô tả - ví dụ:



Server:

- Sử dụng 1 file Text hoặc JSON (hoặc sử dụng file có cấu trúc khác) để lưu danh sách các file cho phép client download gồm *tên file, và dung lượng*. Ví dụ file text (.txt):

```
File1.zip 5MB
File2.zip 10MB
File3.zip 20MB
File4.zip 50MB
File5.zip 100MB
File6.zip 200MB
File7.zip 512MB
File8.zip 1GB
```

- Server sẽ nhận yêu cầu từ Client download file nào, từ offset nào để gửi đến Client.
- File chứa trên Server, **không** được cắt nhỏ sẵn thành các file nhỏ hơn.

- Server có thể phục vụ đồng thời yêu cầu download file từ nhiều client.

Client:

- Client kết nối đến Server, Client sẽ nhận được thông tin danh sách các file từ server và hiển thị trên màn hình.
- Để thực hiện việc download file từ server thì client có một file để ghi nhận danh sách các tên file sẽ được download, file này là **input.txt**. Người dùng sẽ thêm vào cuối danh sách file cần download trong tập tin **input.txt** (không xóa file ghi lại) tên những file cần download bất cứ khi nào muốn mà không làm ảnh hưởng việc thực thi của chương trình (Lưu ý: 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).
- Client chịu trách nhiệm duyệt file input.txt **mỗi 5s một lần**, để lấy được danh sách file vừa được thêm mới và thực hiện download từ server. (không thực hiện download những file đã duyệt trước đó).

Ví dụ file input.txt

```
File1.zip
File2.zip
(Ví dụ File2.zip được thêm vào file input.txt sau khoảng 20 giây và lưu file này lại, client sẽ phát hiện so với lần đọc trước đó thì File2.zip là file mới cần được download sau khi hoàn thành download xong File1.zip)
```

- **Mỗi Client download tuần tự từng file** theo danh sách trong tập tin input.txt.
- Với mỗi một file cần download, client sẽ mở **4 kết nối song song** đến Server để bắt đầu download các phần của 1 file. Có thể dựa vào dung lượng của file và chia 4 để yêu cầu server gửi từng chunk cho mỗi kết nối. Trên màn hình client cho phép hiển thị tỉ lệ % (từ 0-xx%) dựa trên tiến độ download các phần của file đang thực hiện download (Lưu ý: Một client chỉ được mở 4 kết nối song song đến Server để download 1 file).
- Sau đó, client cần nối các phần đã download của một file thành file hoàn chỉnh. (kiểm tra bằng cách kiểm tra tổng dung lượng và mở file thành công)
- Khi Client hoàn thành download, thì có thể bấm "**Ctrl + C**" để kết thúc chương trình. (https://www.tutorialspoint.com/how-do-i-catch-a-ctrlplusc-event-in-cplusplus)

Ví dụ màn hình console của Client:

```
Downloading File5.zip part 1 ....  45%
Downloading File5.zip part 2 ....  15%
Downloading File5.zip part 3 ....  25%
Downloading File5.zip part 4 ....  85%
```

- **Gợi ý:** Sinh viên cần tạo protocol cho chương trình của mình (Application protocol) để đánh dấu chunk bắt đầu-đang-kết thúc download thuộc tập tin nào và khi nhận thì ghi các chunk đúng và đủ vào file để nhận đủ 100% dung lượng tập tin cần download.

Phần II (3đ): Sử dụng UDP tại tầng Transport (1 Client - 1 Server)

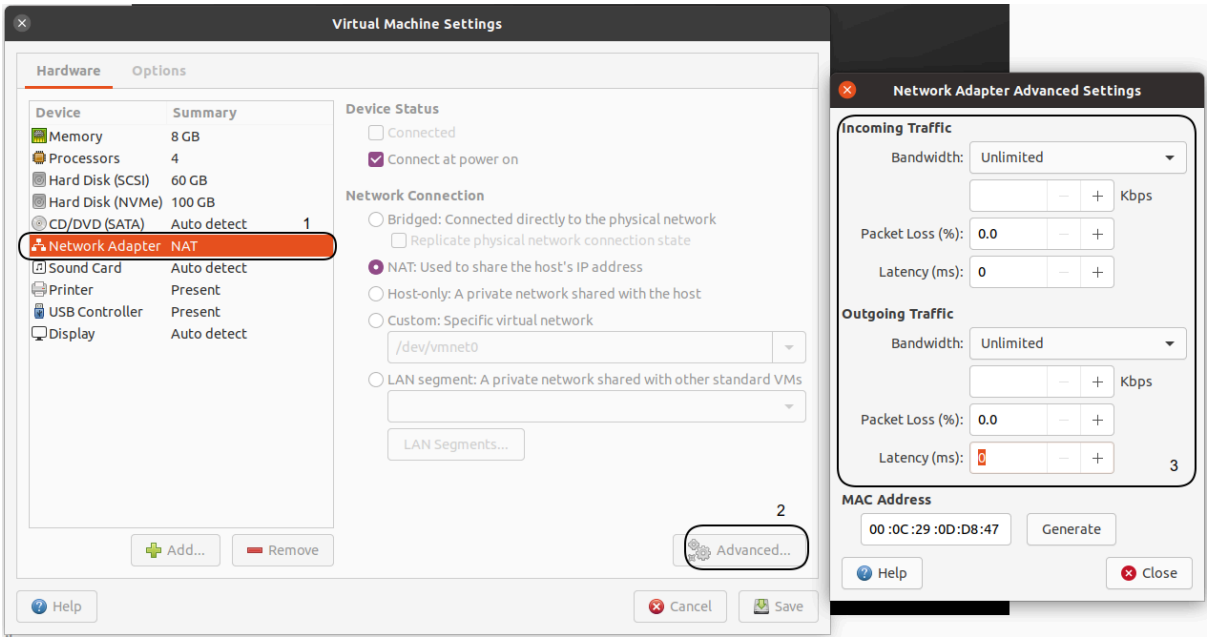
- Tương tự bài phần 1, các bạn sẽ sử dụng UDP. Các bạn cần tự đưa ra cơ chế để gửi dữ liệu tin cậy (rdt - reliable data transfer) ở tầng Application.
  Gợi ý: đưa ra phương thức kiểm tra thứ tự gói tin, kiểm tra mất mát gói tin, lỗi gói tin,...
- Server chỉ phục vụ duy nhất 1 Client download (1 socket duy nhất tại Server). Client có thể download nhiều chunk (nhiều socket) cùng lúc từ Server.
- Không cần xây dựng cơ chế congestion control.
- Chương trình chỉ test 1 client-1 server trong phần II.

6. Tài nguyên

Các bạn có thể lấy các file test có dung lượng khác nhau ở đây (5MB, 10MB, 20 MB, 50 MB, ...):

- https://www.thinkbroadband.com/download
- https://ash-speed.hetzner.com/
- http://xcal1.vodafone.co.uk/

Trong trường hợp bạn muốn test bandwidth chậm với máy ảo để test (chạy Server hoặc Client), bạn có thể chỉnh bandwidth ở đây để giới hạn tốc độ gửi chậm lại



Hoặc bạn có thể dùng tools bên dưới để giới hạn tốc độ bandwidth giữa Client-Server:

- NetLimiter
- TMeter
- Clumsy

Hoặc có thể dùng sleep để test chương trình của mình.

7. Thang điểm

STT	Yêu cầu	Điểm
Phần I	Nhiều clients có thể nhận được danh sách các file từ Server và ctrl-c	1
	Hiển thị percent download các chunk của file	1
	5s quét file input.txt 1 lần	0.5
	Client có để download chunk đầu tiên của file thành công	1
	Nhiều client có thể tải đủ các chunk thành công từ Server. Tập tin sau khi download phải đúng và đủ dung lượng	3
Phần II	UDP download thành công 1 chunk kèm cơ chế rdt (reliable data transfer)	1
	UPD download thành công toàn bộ file kèm cơ chế rdt (reliable data transfer)	2
	Báo cáo	0,5