BÁO CÁO ĐỒ ÁN

MÔN HỌC: MẠNG MÁY TÍNH
BÀI TẬP SOCKET

Tên: Đỗ Thành Nhơn – 1512387
Nguyễn Thành Tân – 1512491
Hà Tấn Linh – 1512284
Lớp: 15CNTN

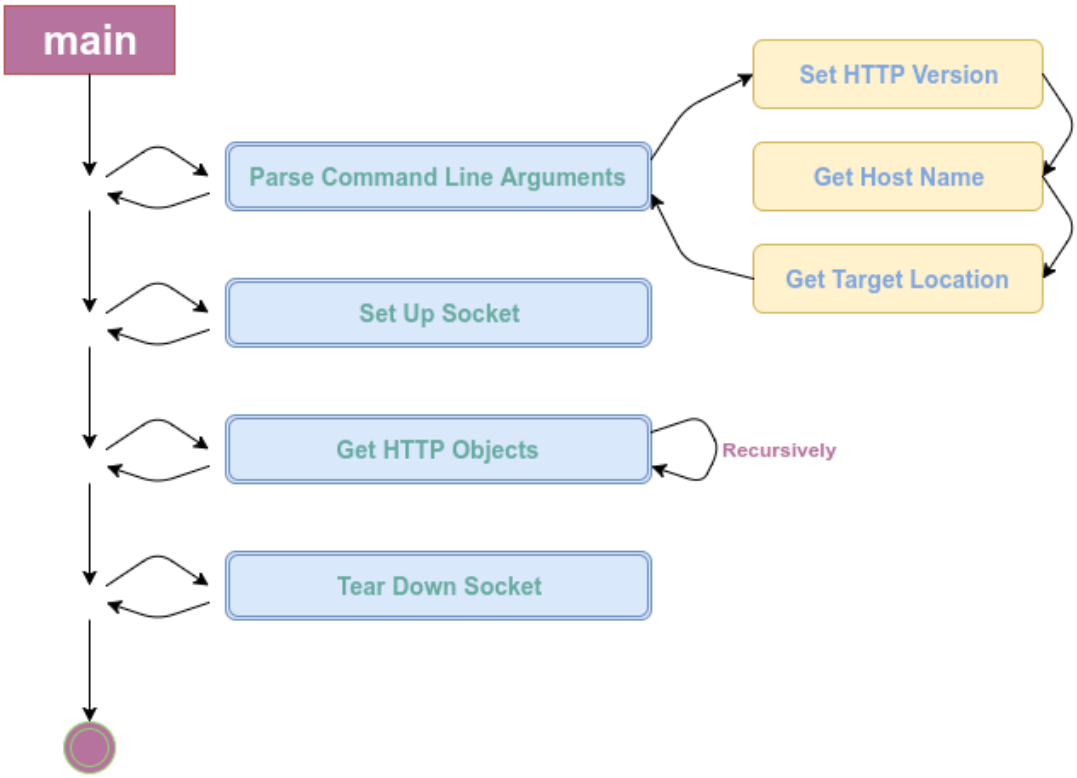
PHỤ LỤC

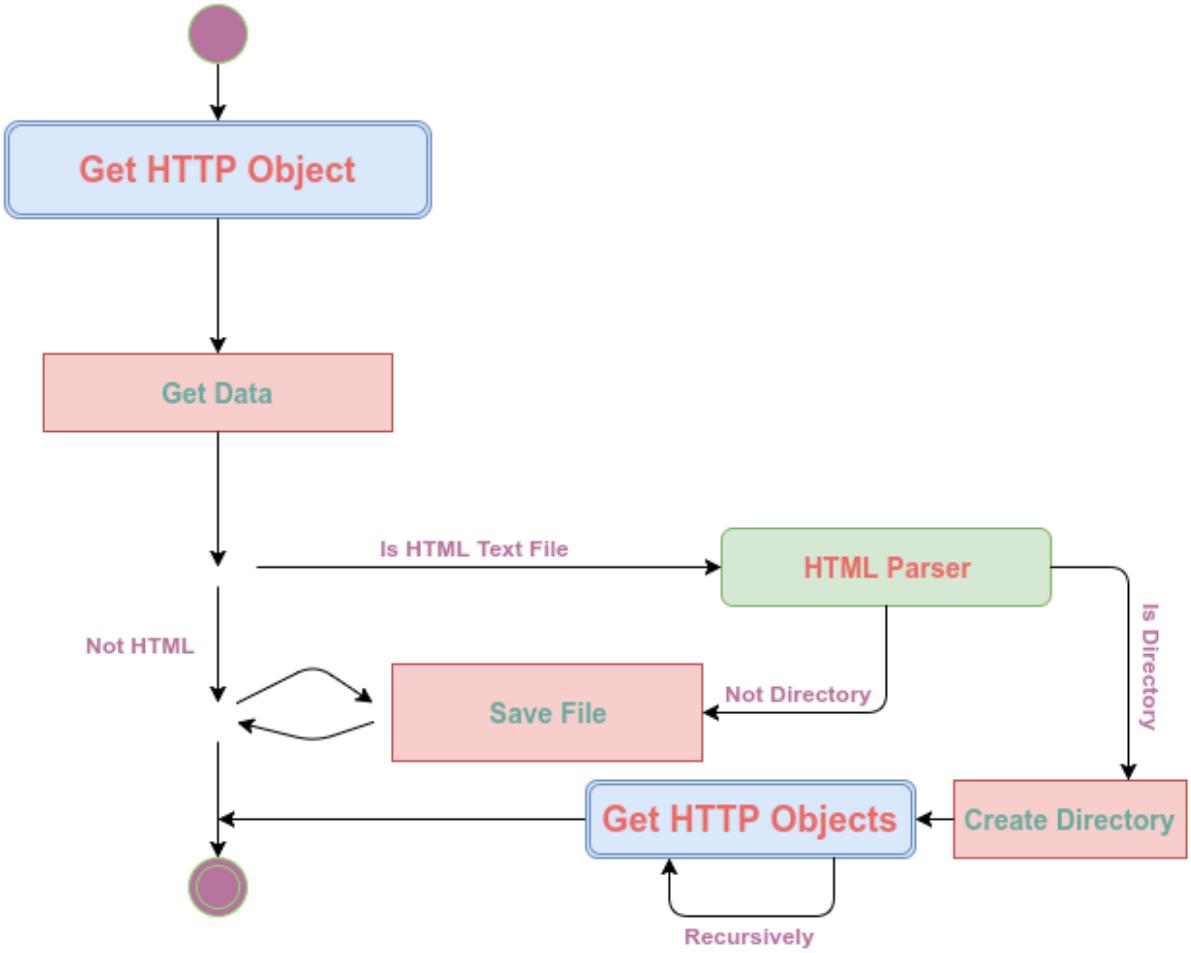
I. Phân công công việc.

STT	Họ tên	Công việc
1	Hà Tấn Linh	Tìm hiểu giải pháp. Thiết kế chương trình. Vẽ Flowchart, phân công. Cài đặt Module GetObject.
2	Đỗ Thành Nhơn	Tìm hiểu giải pháp. Thiết kế chương trình. Cài đặt Modulo Parse HTML. Viết báo cáo phần 1, 2, và 3. Tổng hợp kết quả.
3	Nguyễn Thành Tân	Tìm hiểu giải pháp. Thiết kế chương trình. Cài đặt Modulo Parse URL. Kiểm thử. Viết báo cáo phần 4, 5.

Source code được quản lí bởi Version Control và được upload ở:
https://github.com/hatanlinh13/project-1-computer-network

II. Những hàm chức năng chính.





1. Parse URL.

   Hàm int cmd_parser(int argc, char **argv, char **host_name, char **target_location);

   - Chức năng: xử lí chuỗi URL đầu vào thành host name và đường dẫn đến thư mục cần download ở server, xác định HTTP version.
   - Tham số: hàm nhận vào 4 tham số lần lượt là tham số từ CMD khi gọi chương trình và 2 tham số còn lại dùng để ghi kết quả là host name và đường dẫn đến thư mục download.

2. Parse HTML.

   Hàm html_parser(char *data, char ***object_list);

   - Chức năng: từ nội dung file HTML, hàm sẽ xử lí và trả vễ danh sách tên của các file và folder.
   - Tham số: hàm nhận vào 2 tham số, tham số thứ nhất là nội dung của file HTML và tham số thứ 2 được dùng để ghi kết quả là danh sách chuỗi tên.

3. Get Object.

   Hàm Get_http_object:

- Chức năng: Kết nối đến HTTP server, yêu cầu tập tin, tải về và xử lý phù hợp, gọi đệ quy cho các tập tin và thư mục con trong trường hợp target_location là thư mục.
- Tham số: địa chỉ tên miền của HTTP server, đường dẫn tới đối tượng cần download, thư mục để lưu đối tượng đã download về.
- Cài đặt: Gọi hàm create_message để tạo HTTP GET message dựa vào tham số 1 và 2. Sau đó gọi hàm get_data để lấy dữ liệu, sau đó gọi hàm html_parser từ module htmlp để phân tích dữ liệu, sau đó gọi create_name để tách tên tập tin/thư mục từ target_location, cuối cùng tuỳ vào kết quả phân tích mà save_file hoặc create_dir rồi gọi đệ quy.

Hàm Create_name:

- Chức năng: tách tên từ target_location, thay tất cả kí tự %20 bằng khoảng trắng, thêm tiền tố MSSV nếu cần thiết.
- Tham số: target_location - đường dẫn tới đối tượng trên server.
- Trị trả về: địa chỉ của chuỗi tên được cấp phát động.

Hàm Create_message:

- Tham số: host_name: địa chỉ http server, target_location: đường dẫn.
- Chức năng: tạo ra câu truy vấn HTTP GET để gửi đến server.
- Trị trả về: chuỗi truy vấn được cấp phát động.

Hàm Free_objects:

- Chức năng: giải phóng bộ nhớ của danh sách đã được cấp phát bởi html_parser khi không dùng đến, tránh memory leak.
- Tham số: char **object_list: danh sách tên con được trả về bởi html_parser.

Hàm Get_data:

- Chức năng: dựa vào giá trị http_version để gọi hàm get_http10_data hoặc get_http11_data tương ứng để thực hiện download file dc yêu cầu.
- Tham số: câu truy vấn gửi đến server, địa chỉ của http server, buffer để lưu dữ liệu, số byte đã nhận được.
- Trị trả về: 1: file nhận được là html, 2 file nhận dc là file thường, 0 lỗi.

Hàm Get_http10_data:

- Chức năng: Download file bằng HTTP 1.0.
- Tham số: tương tự get_data.
- Cài đặt: Đầu tiên gọi hàm set_up_socket của module sockmngt để tạo kết nối đến host_name và tạo một socket. Sau đó gọi hàm get_http_header để lấy thông tin header của HTTP respond. Xử lý thông tin header ở các trường Status-Code để xem tập tin có tải được hay không, Content_Type để phân biệt loại file và cuối cùng là Content_length để biết độ dài của phần body respond hoặc server trả về tập tin theo dạng connection-closed để có cách đọc dữ liệu phù hợp.
- Trị trả về: 1: tập tin html, 2: tập tin thường, 0: lỗi.

Hàm Get_http11_data:

- Chức năng: Download file bằng HTTP 1.1

- Tham số: tương tự get_data.
- Trị trả về: 1: tập tin html, 2: tập tin thường, 0: lỗi.

Hàm Get_http_header:

- Chức năng: nhận từ socket header của HTTP respond và lưu vào buffer hdr.
- Tham số: char *hdr: buffer để lưu trữ header.
- Trị trả về: 1 nếu thành công, 0: lỗi.

III. Mức độ hoàn thành.

1. Các chức năng đã hoàn thành.

   - Hỗ trợ HTTP 1.0.
   - Hỗ trợ HTTP 1.1.
   - Nâng cao: download tất cả file/folder liên quan.

2. Các chức năng chưa hoàn thành.

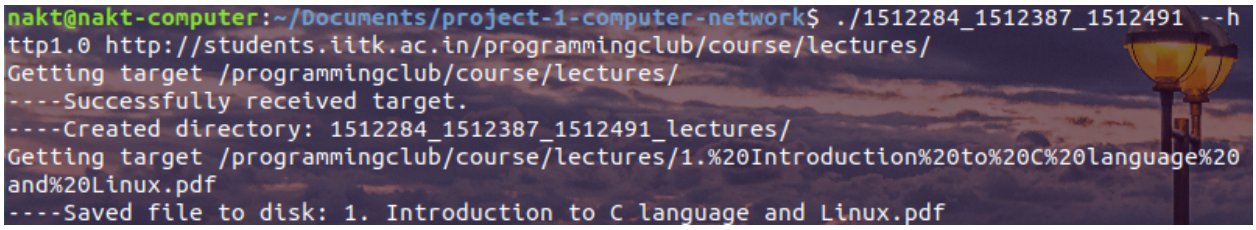
   <không có>

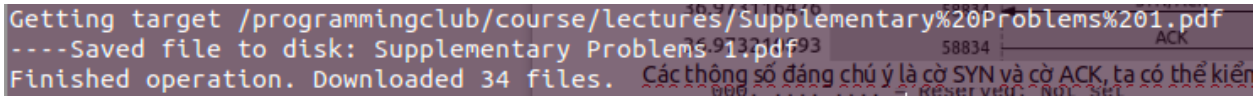
3. Mức độ hoàn thành đồ án.

   Đã hoàn thành 100% yêu cầu của đồ án.

IV. Cách chạy và kết quả chương trình.

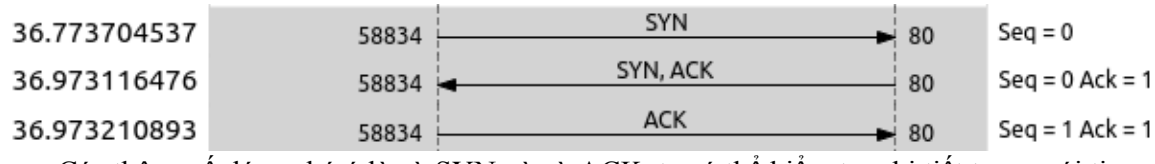
- Vào thư mục gốc chương trình chạy lênh "make" để biên dịch.
- Sau khi biên dịch thành công sẽ có file thực thi tên "1512284_1512387_1512491". Chạy bằng command line như sau:
- ./1512284_1512387_1512491 –http1.0 http://students.iitk.ac.in/programmingclub/course/lectures/
- Để tải file với http 1.0, nếu muốn có thể sửa 1.0 thành 1.1. Nếu không định nghĩa thì chương trình mặc định http1.0.
- Kết quả hiện thị:





V. Dùng Wireshare bắt gói tin.

- Đầu tiên là "threeway handshake" giữa client và server

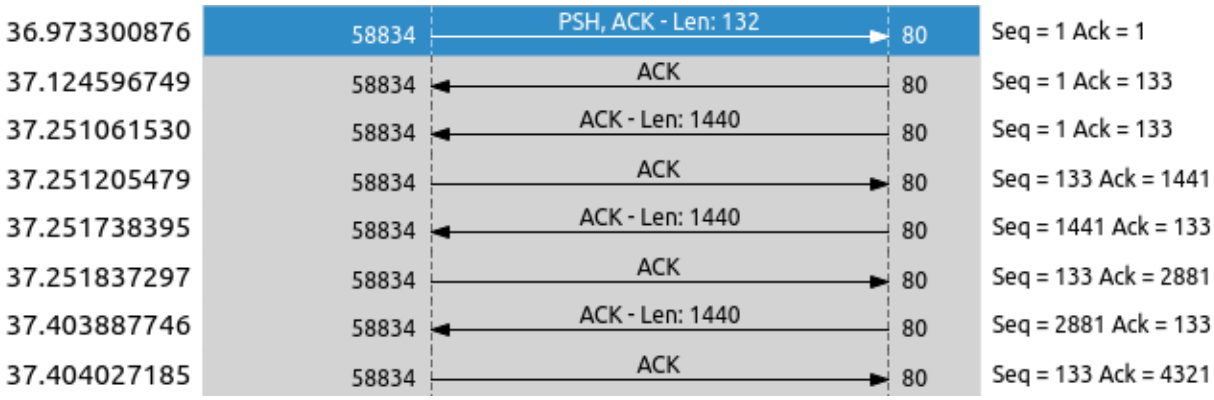


Các thông số đáng chú ý là cờ SYN và cờ ACK, ta có thể kiểm tra chi tiết trong gói tin:

```
000. .... .... = Reserved: Not set
...0 .... .... = Nonce: Not set
.... 0... .... = Congestion Window Reduced (CWR): Not set
.... .0.. .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
.... ...0 .... = Acknowledgment: Not set
.... .... 0... = Push: Not set
.... .... .0.. = Reset: Not set
.... .... ..1. = Syn: Set
.... .... ...0 = Fin: Not set
```

như ta thấy trong gói tin thứ nhất, cờ SYN được bật, tương tự cho 2 gói tin thứ 2 và thứ 3.

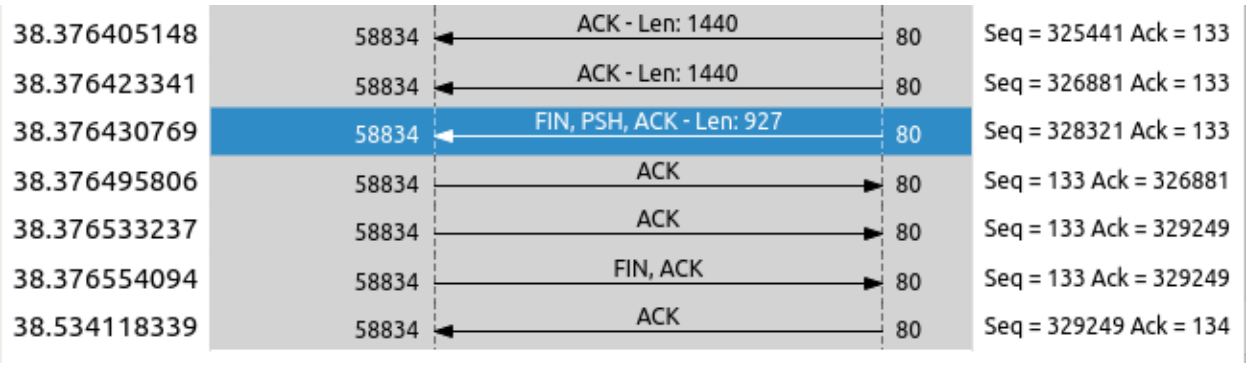
- Quá trình diễn ra như sau:
  - Client gửi thông điệp muốn tạo kết nối với sever (cờ SYN được bật)
  - Server gửi lại thông điệp đã nhận (cờ ACK được bật) và thông điệp muốn tạo kết nối với client (cờ SYN).
  - Client gửi lại thông điệp đã nhận cho server, kết thúc quá trình "threeway handshake".
- Thông số khác cần lưu ý là Seq và Ack với ý nghĩa như sau:
  - Seq: số nguyên 32 biết cho biết lượng data đã gửi.
  - Ack: lượng dữ liệu liên tục dài nhất đã nhận được.
- Tiếp đến là quá trình truyền nhận dữ liệu giữa client và server:



- Client gửi gói tin request HTTP đến server và đề nghị server hãy dùng thông điệp này cho tầng application (cờ PSH được bật), lúc này Seq là 1 vì trước đó đến giờ client mới chỉ gửi 1 byte dữ liệu (trong quá trình handshake) và Ack là 1 xác nhận chỉ nhận được 1 byte dữ liệu từ server (trong quá trình handshake). Payload của request này là 132 bytes.
- Server gửi thông điệp xác nhận đã nhận được request, Ack lúc này là 133 (132 của payload + 1 handshake). Seq vẫn là 1 vì tới giờ thì server vẫn chỉ mới gửi dữ liệu cho client trong quá trình handshake.
- Sau khi xử lý xong request (download trong trường hợp này), một quá trình lặp lại gửi nhận gói tin bắt đầu:
  - Server gửi gói tin chứa dữ liệu mà client request. Seq và Ack vẫn bằng gói tin trước (giải thích tương tự). Payload của gói tin này là 1440

- Client gửi thông điệp xác nhận với Ack = 1441(1440 của payload + 1 handshake) cho server, Seq = 133 (132 của payload lúc trước + 1 handshake).

- Cuối cùng là kết thúc phiên kết nối sau khi dữ liệu đã tải xong:



- Gói tin thứ 3 từ server là có cờ FIN được bật ra hiệu cho client rằng không còn dữ liệu nữa, cờ PSH ra hiệu cho client có thể chuyển dữ liệu lên tầng application được rồi, cờ ACK ý nghĩa như từ đầu đến giờ.
- Gói tin thứ 6 từ client có cờ FIN ra hiệu đóng kết nối.
- Gói thứ 7 gửi xác nhận từ server, Ack = 134 (1 handshake + 132 payload + 1 finish). Kết nối đóng.