BÁO CÁO ĐỒ ÁN 1

MẠNG MÁY TÍNH

VIẾT ỨNG DỤNG CLIENT DOWNLOAD FILE & FOLDER

Bộ môn Mạng máy tính

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

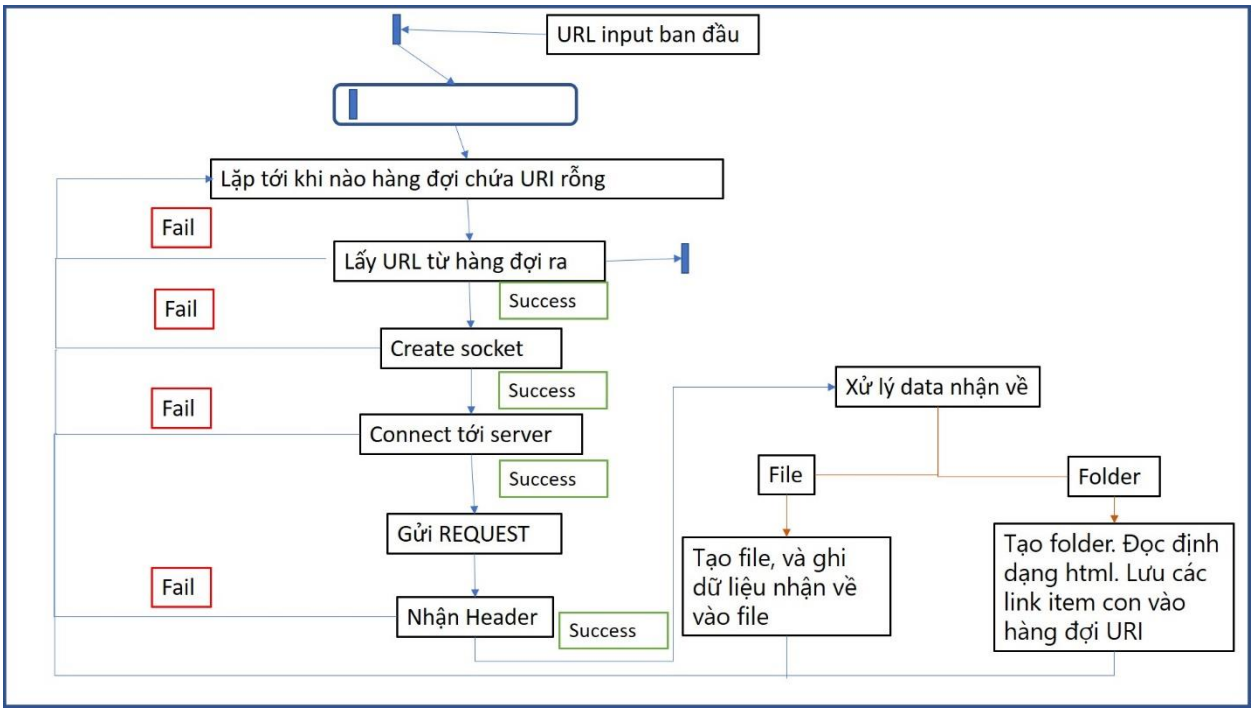
MỤC LỤC

1. PHÂN CÔNG CÔNG VIỆC

MSSV	Họ tên	Công việc
1512034	Nguyễn Đăng Bình	Tìm hiểu về giao thức HTTP, các gói header của HTTP. Viết các class NodeQueue, QueueURI, URI. Phân tích gói tin TCP với wireshark.
1512042	Nguyễn Thành Chung	Tìm hiểu về giao thức HTTP, các gói header của HTTP. Viết các class MyHTTP với công việc cài đặt giao tiếp giữa server và client bằng api socket, thực hiện lưu file, lưu folder. Viết hàm main.
1512063	Nguyễn Du Du	Tìm hiểu về giao thức HTTP, các gói header của HTTP. Tìm hiểu về chunked encoding và viết code bắt và giải mã chunked encoding. Phân tích gói tin TCP với wireshark.

2. NHỮNG HÀM CHỨC NĂNG CHÍNH

Kịch bản chính của chương trình



Hình 1. Kịch bản giao tiếp chính của chương trình

Các hàm chính

```
void get_FileAndFolder(QueueURI& uris, int port, int versionHTTP)
```

Hàm thực hiện chức năng chính của chương trình là khởi tạo, lấy dữ liệu (file, folder) từ server, chạy trong vòng lặp khi hàng đợi cho tới khi nào không còn URI nào nữa. Hàm truyền vào 3 tham số là **uris** (tham chiếu, hàng đợi chứa các uri cần get); **port** (số hiệu cổng port sử dụng); **versionHTTP** (version HTTP để GET, 0: HTTP 1.0 và 1: HTTP 1.1).

```
int Create(int domain = AF_INET, int type = SOCK_STREAM, int protocol = 0)
```

(MyHTTP.h)

Hàm thực hiện việc khởi tạo socket và các bước khởi tạo dữ liệu cần thiết để kết nối đến server. Hàm trả về socketID nếu thành công và -1 nếu thất bại. Hàm truyền vào 3 tham số là domain (loại định dang cho URI), type(Loại Strem nào sẽ sử dụng để truyền và nhận dữ liệu), protocol (Loại giao thức nào được sử dụng để truyền và nhận dữ liệu).

int Connect() (MyHTTP.h)
Hàm kết nối đến server với các giá trị đã khởi tạo sẵn cho đối tượng MyHTTP. Nếu thành công thì trả về 1, thất bại trả về 0.

void MyHTTP::makeRequest(char request[1024], const int& type) (MyHTTP.h)
Hàm thực hiện việc format chuỗi request tới server dựa vào tham số type là version của HTTP đang sử dụng. Chuỗi request này dùng để gửi yêu cầu tới server.

Ví dụ với một URI: http://students.iitk.ac.in/programmingclub/course/lectures/
request http 1.0: "GET /programmingclub/course/lectures/ HTTP/1.0\r\nHost: students.iitk.ac.in\r\n\r\n"
request http 1.1: "GET /programmingclub/course/lectures/ HTTP/1.1\r\nHost: students.iitk.ac.in \r\n""Connection: close\r\n""\r\n".

void Send(char request[1024]); (MyHTTP.h)
Hàm thực hiện việc gửi chuỗi request tới server.

int Receive(char* pData, int nBytes); (MyHTTP.h)
Hàm thực hiện việc nhận dữ liệu từ server gửi về. Hàm trả về số byte nhận được. Tham số pData dùng để lưu dữ liệu nhận về và nBytes với ý nghĩa là số bytes muốn nhận về.

int GetHeader(char* header, int pSize); (MyHTTP.h)
Hàm thực việc lấy header từ server. Hàm trả về số byte của header. Tham số header dùng để lưu dữ liệu của header từ server; pSize là số bytes muốn lấy về.

bool isValidResponse(char* data, const int& versionHTTP); (MyHTTP.h)
Hàm kiểm tra xem gói header có thông báo thành công hay không (200 OK). Nếu thành công thì trả về true, thất bại trả về false. Hàm truyền vào data (dữ liệu header); versionHTTP là loại version của HTTP.

bool hasChunkedEncoding(char *data); (MyHTTP.h)
Hàm kiểm tra xem gói header có trường nào thông báo gói dữ liệu này sẽ bị mã hóa Chunk hay không? Nếu có trả về true, nếu không trả về false. Tham số data là dữ liệu của header.

void decodeChunk(char* data, char* src, int totalSize); (MyHTTP.h)
Hàm thực hiện việc giải mã Chunk. Tham số data (chưa dữ liệu được giải mã); src (chứa dữ liệu bị mã hóa chunk); totalSize (chứa kích thước theo bytes của chuỗi bị mã hóa).

void writeData(QueueURI& uris, const int& versionHTTP, const bool& hasChunked) (MyHTTP.h)
Hàm thực hiện việc nhận dữ liệu và lưu vào máy tính. Nếu dữ liệu nhận về là file thì tạo và lưu file vào máy. Ngược lại nếu là file .html (cấu trúc các subitem), thì thực hiện việc tạo folder và nạp các link vào hàng đợi uris để tiếp tục kết nối và tải về. Tham số

versionHTTP (version của HTTP đang dùng); hasChunked (cờ kiểm tra gói dữ liệu nhận về có bị Chunked Encoding hay không?).

3. MỨC ĐỘ HOÀN THÀNH

STT	Chức năng	Mức độ hoàn thành
1	Thực hiện download các file và sub folder bậc 1 trong 1 trang web (hỗ trợ http 1.0 và http 1.1)	100%
2	Thực hiện download một file (hỗ trợ http 1.0 và http 1.1)	100%
3	Thực hiện download tất các file và folder con trong 1 trang web (hỗ trợ http 1.0 và http 1.1)	100%

4. CÁCH CHẠY VÀ KẾT QUẢ CỦA CHƯƠNG TRÌNH

Chương trình được chạy bằng command line (Window)

$ <tên chương trình> <URL> **<version HTTP>**

$ 1512034_1512042_1512063 http://hinhnendep.vn/wp-content/uploads/2016/08/hinh-nen-doremon-dep-nhat-1.jpg --http1.0

Or

$ 1512034_1512042_1512063 http://hinhnendep.vn/wp-content/uploads/2016/08/hinh-nen-doremon-dep-nhat-1.jpg --http1.1

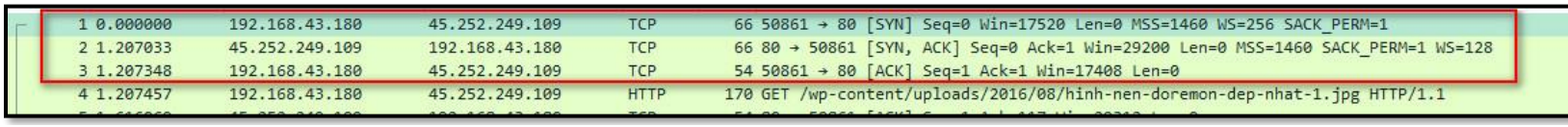
Sau khi chạy chương trình thì chương trình sẽ tự động tải file về thư mục hiện hành nếu như link hợp lệ.

5. REPORT QUÁ TRÌNH GỬI NHẬN TCP SEGMENT DATA – ACK BẰNG WIRESHARK

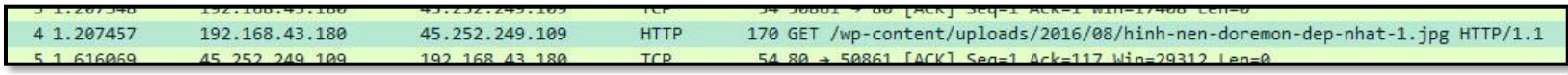
Địa chỉ máy client: 192.168.43.180; sử dụng port 50861.

Địa chỉ server: 45.252.249.109; sử dụng port 80.

Ban đầu khi thực hiện kết nối thì thực hiện three ways handshake.



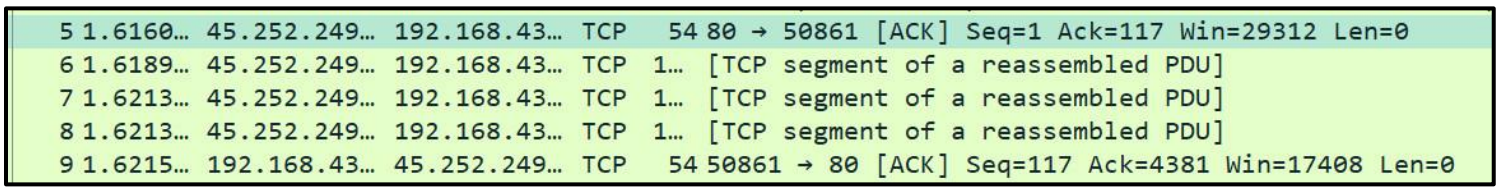
Hình 2.Three ways HandShake



Hình 3. Gói HTTP Request

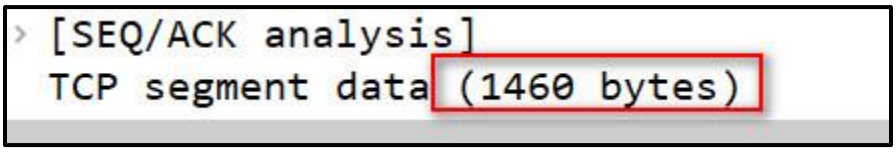
Ở client gửi gói tin HTTP đến server (lệnh GET để yêu cầu lấy dữ liệu từ server).

Server gửi gói tin (5) với byte đầu tiên là thứ nhất và ACK = 117 (số mong muốn nhận về từ client) đến client. Qúa trình gửi nhận dữ liệu thật sự giữa server và client chính thức bắt đầu.



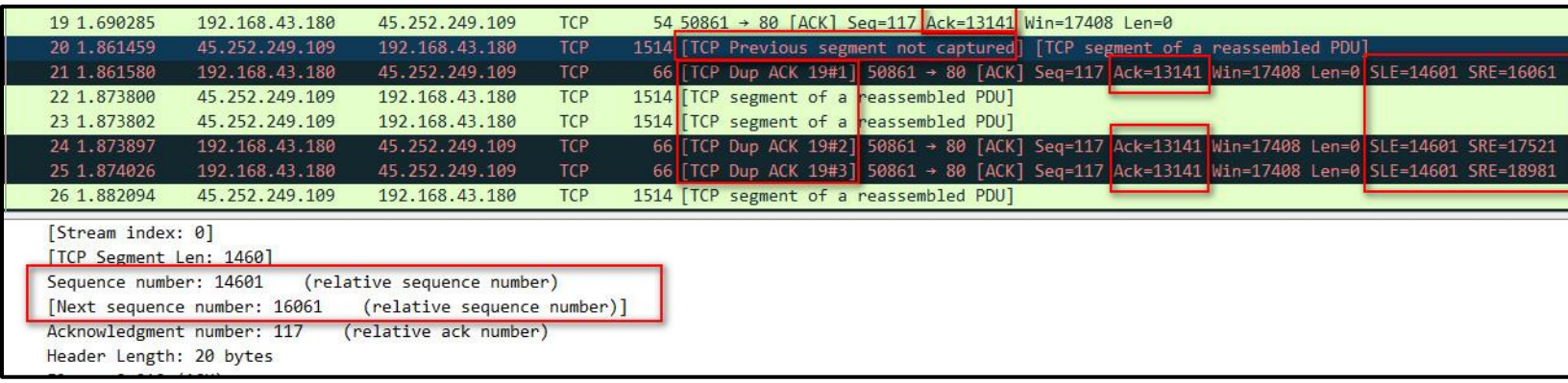
Hình 4. Gửi và nhận dữ liệu

Tại dòng 6, 7, 8: Client nhận được các gói TCP từ Server. Với trường dữ liệu (data) có kích thước 1460 ở mỗi gói.



Hình 5. Kích thước của trường data trong gói TCP

Tại dòng 9: Client gửi về server gói báo nhận. (Seq = 117; ACK = 4381). Client đã nhận được 4380 bytes, và muốn server gửi tiếp gói dữ liệu khác bắt đầu với byte thứ 4381. Qúa trình truyền và nhận dữ liệu cứ tiếp tục xảy ra như vậy.



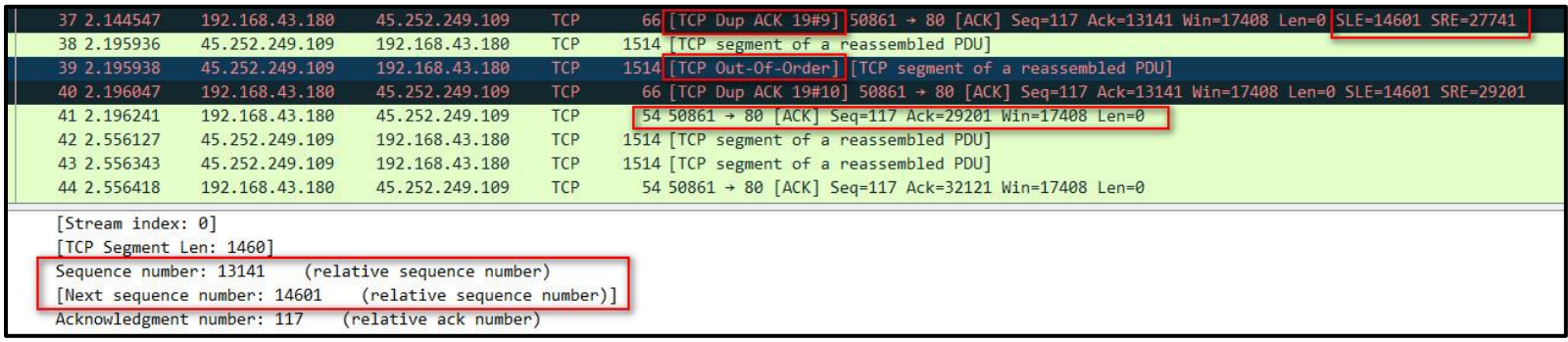
Hình 6. Hiện tượng mất gói

Tại dòng 19: Bên client muốn nhận byte tiếp theo từ server là byte thứ 13141. Tuy nhiên, tại dòng 20 thì lại nhận được gói tin với byte thứ 14601, và hiện dòng "TCP Previous segment not captured". Nghĩa là gói tin trước đó không bắt được (gói bị mất).

Tại dòng 21: Client lại tiếp tục gửi đến server ACK = 13141, mong muốn nhận gói có byte đầu tiên thứ 13141.

Tại dòng 22, 23: Server lại tiếp tục gửi các gói khác đến client.

Tại dòng 24, 25: Client vẫn chưa nhận được gói với byte đầu tiên thứ 13141 nên liên tục gửi về server ACK = 13141. Đồng thời cũng gửi theo thông tin các byte đã nhận được (SLE = 14601; SRE = 18981 nghĩa là đã nhận được từ byte thứ 14601 đến byte thứ 18980). Qúa trình tiếp tục như vậy cho tới dòng 39. Client đã gửi lại 9 gói tin ACK = 13141 đến server. Lúc này Client đã nhận được byte thứ 14601 đến byte 29200.

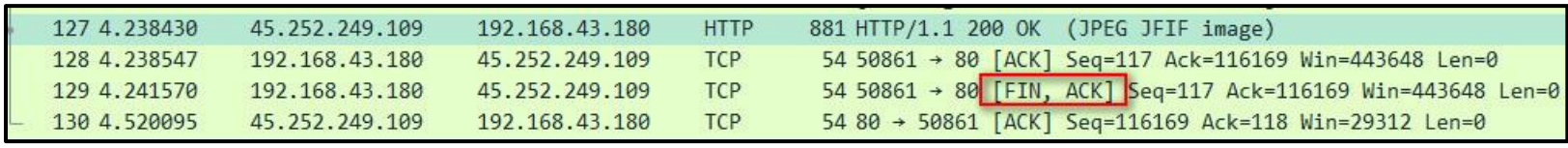


Hình 7. Gói đến không đúng thứ tự

Tại dòng 39: Client nhận được 1 gói "TCP Out-Of-Order", với byte đầu tiên thứ 13141. Đây là gói server gửi lại do bị mất, nhưng không đúng thứ tự nên bên client phải tổ chức sắp xếp lại các gói đã nhận cho đúng thứ thự.

Tại dòng 40: Do phía client chưa sắp xếp xong, nên nó vẫn gửi lại một gói với ACK = 13141.

Tại dòng 41: Client đã sắp xếp xong, và báo về server thành công ACK = 29201, muốn nhận gói tiếp theo với byte đầu tiên thứ 29201. Qúa trình gửi và nhận dữ liệu lại tiếp tục diễn ra bình thường.



Hình 8. Đóng kết nối

Tại dòng 127: Server gửi những byte cuối cùng và thông báo đã kết thúc hết byte bằng gói tin HTTP.

Tại dòng 129: Client yêu cầu đóng kết nối đến server bằng cờ FIN.

Tại dòng 130: Server báo về client, kết nối được đóng. Kết thúc giao tiếp giữa client và server.

6. TÀI LIỆU THAM KHẢO

sockets Programming in C++.pdf
https://www.tutorialspoint.com/http/