

-----\*



KHOA CÔNG NGHỆ THÔNG TIN  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# BÁO CÁO ĐỒ ÁN MẠNG MÁY TÍNH

ĐỀ TÀI: Chương trình Proxy Server sử dụng SOCKET API

Lớp : CQ2017/1  
Môn học : Mạng máy tính  
Giáo viên hướng dẫn : Lê Hà Minh

THÀNH PHỐ HỒ CHÍ MINH – 2019

## **I. Danh sách thành viên thực hiện đồ án:**

<b>MSSV</b>	<b>Tên</b>
1712270	Đào Đức Anh
1712457	Nguyễn Khánh Hoàng
1712631	Nguyễn Thành Nhân
1712798	Trần Trung Thọ

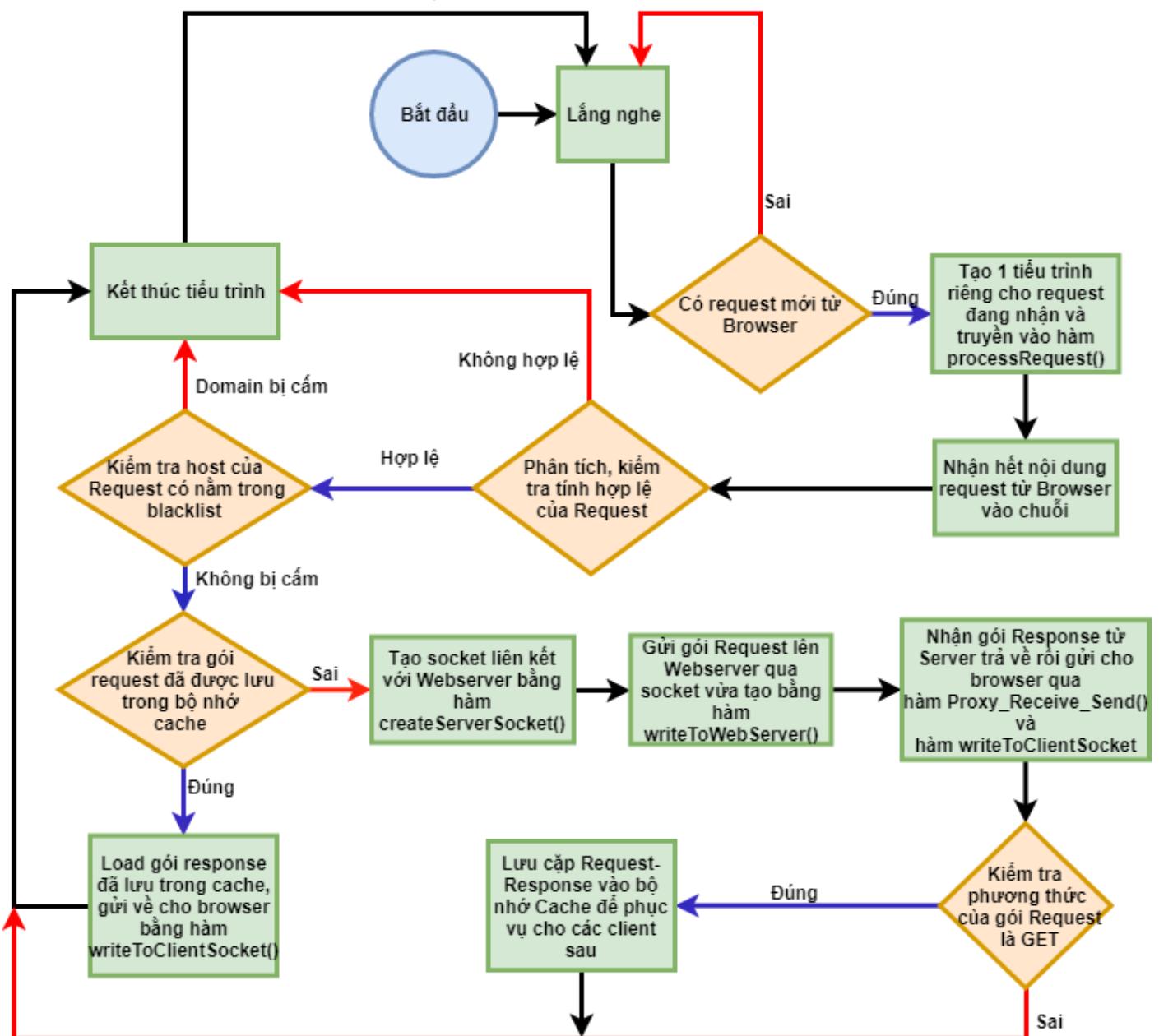
## **II. Báo cáo nội dung đồ án**

### **1. Bảng phân công công việc:**

Yêu cầu	Người đảm nhận	Mức độ hoàn thành
<b>Các yêu cầu chính</b>		
Xác định hướng làm, khung chương trình, tìm hiểu, phổ biến cho các thành viên.	Đức Anh	100%
Hàm xử lí chính main() và processRequest(), nhóm hàm phân tích và kiểm tra gói Request	Đức Anh, Trung Thọ	100%
Nhóm hàm gửi Request lên Server và nhận về Browser ( writeToWebServer(), Proxy_Receive_Send(), writeToClientSocket())	Đức Anh, Khánh Hoàng, Thành Nhân	100%
Proxy Server hỗ trợ phương thức GET/POST	Đức Anh	100%
Xử lí đa luồng multithread cho nhiều client truy cập cùng lúc	Khánh Hoàng	100%
Chặn truy cập với các domain bị cấm trong file blacklist.conf	Khánh Hoàng	100%
Cơ chế caching phục vụ các client khác nhanh chóng hơn	Trung Thọ	100%
Kiểm thử chương trình, báo lỗi (nếu có)	Thành Nhân	100%
Bắt gói tin bằng Wireshark và mô tả quá trình nhận – gửi dữ liệu giữa các bên.	Thành Nhân	100%
Viết báo cáo, ghép code, chú thích chương trình.	Đức Anh, Khánh Hoàng, Trung Thọ	100%
Các hàm phát sinh, ý tưởng, các mã lỗi hệ thống, debug chương trình.	Đức Anh, Khánh Hoàng, Thành Nhân, Trung Thọ	100%
<b>Mức độ hoàn thành project</b>		<b>Hoàn thành 100% yêu cầu đề ra</b>
<b>Mức độ đóng góp của các thành viên</b>	Đức Anh	100%
	Khánh Hoàng	100%
	Thành Nhân	100%
	Trung Thọ	100%

## 2. Những hàm thực hiện những chức năng chính của chương trình:

### a) Sơ đồ xử lí chính của chương trình



### b) HANDLE WINAPI

CreateThread([LPSECURITY\\_ATTRIBUTES](#) lpThreadAttributes,

[SIZE\\_T](#) dwStackSize,

[LPTHREAD\\_START\\_ROUTINE](#) lpStartAddress,

[LPVOID](#) lpParameter,

[DWORD](#) dwCreationFlags,

[LPDWORD](#) lpThreadId);

- Ta chỉ cần quan tâm tới tham số thứ 3 *lpStartAddress* và tham số thứ 4 *lpParameter*. Tham số thứ 3 là tên của hàm mà ta sẽ tạo một Thread mới thực hiện các lệnh trong hàm đó. Tham số thứ 4 là con trỏ kiểu void, có giá trị là địa chỉ của biến đóng vai trò là tham số được truyền vào hàm ở tham số thứ 3.

Gọi hàm: CreateThread([NULL](#), 0, processRequest, sockClient, 0, &threadID);

Hàm *processRequest*, tham số truyền vào hàm là biến kiểu con trỏ *SOCKET*

- Chức năng của hàm là tạo một Thread mới, Thread này thực hiện các tác vụ được mô tả trong thân hàm của hàm *processRequest*.
- Kết quả sau khi gọi hàm: Một Thread mới được tạo để xử lý request mà Proxy vừa nhận được từ browser

c) **DWORD WINAPI processRequest(LPVOID arg)**

- arg là con trỏ kiểu void có giá trị là địa chỉ của biến mà đóng vai trò là tham số của hàm processRequest.  
Gọi hàm bằng cách tạo Thread mới bằng hàm CreateThread ở trên:  
CreateThread(NULL, 0, processRequest, sockClient, 0, &threadID);  
Biến con trỏ kiểu SOCKET sockClient là tham số mà ta cần truyền vào hàm processRequest
- Chức năng của hàm là xử lý thông điệp yêu cầu (request message) mà proxy server nhận được từ browser, gửi request message này tới Web Server, sau đó nhận lại thông điệp phản hồi (response) từ Web Server, rồi lại gửi thông điệp này về cho browser tại Client.
- Kết quả sau khi gọi hàm: một thông điệp yêu cầu từ Client được proxy server xử lý, gửi tới cho Web Server, và một thông điệp phản hồi từ Web Server được proxy server chuyển về lại cho Browser.

d) **struct ParsedRequest\* ParsedRequest\_create()**

- Chức năng của hàm: Hàm trả về một con trỏ kiểu ParsedRequest đã được cấp phát một vùng nhớ mới trên bộ nhớ Heap.  
Cấu trúc ParsedRequest chứa các thông số của một thông điệp yêu cầu (request message) như method (phương thức HTTP), protocol (giao thức), host (tên miền của Web Server), port (cổng tại Web Server), v.v...
- Gọi hàm: **struct ParsedRequest \*req; req = ParsedRequest\_create();**
- Kết quả sau khi gọi hàm: Con trỏ req được cấp phát một vùng nhớ ứng một đối tượng thuộc kiểu cấu trúc ParsedRequest

e) **int ParsedRequest\_parse(struct ParsedRequest \* parse, const char \*buf, int buflen)**

- Các tham số của hàm: parse là biến kiểu cấu trúc ParsedRequest, buf là chuỗi chứa nội dung của request, buflen là chiều dài của chuỗi buf
- Chức năng của hàm: Phân tích cú pháp, tách các thông tin chính yếu của chuỗi buf kiểu char\* chứa thông điệp yêu cầu (request message) và gán cho các trường thuộc tính của con trỏ parse kiểu ParsedRequest.
- Kết quả sau khi gọi hàm: Nếu phân tách và gán thành công, hàm trả về giá trị 0, ngược lại, nếu có lỗi, hàm trả về -1, và Thread được kết thúc.

f) **char\* convert\_Request\_to\_String(struct ParsedRequest \*req)**

- Tham số của hàm: req là con trỏ kiểu ParsedRequest
- Chức năng của hàm: hàm thực hiện chức năng ghép lại các trường thuộc tính của struct ParsedRequest thành 1 chuỗi kiểu char\* và trả về chuỗi này
- Kết quả sau khi gọi hàm: thực hiện thành công, hàm sẽ trả về chuỗi kiểu char\* là 1 thông điệp yêu cầu, ngược lại, nếu lỗi, hàm sẽ thông báo lỗi và trả về NULL

g) **int createServerSocket(char \*Host, char \*Port)**

- Các tham số của hàm: Host là chuỗi chứa tên miền của Web Server mà ta muốn kết nối đến, Port là chuỗi chứa số hiệu của port tại Web Server
- Chức năng của hàm: tạo một Socket kết nối với Web Server tương ứng với Host và Port
- Kết quả sau khi gọi hàm: Nếu thành công, hàm trả về SOCKET mới được tạo, ngược lại, nếu có lỗi, báo lỗi và trả về -1.

h) **int writeToWebServer(const char\* buff\_to\_server, SOCKET sockfd, int buff\_length)**

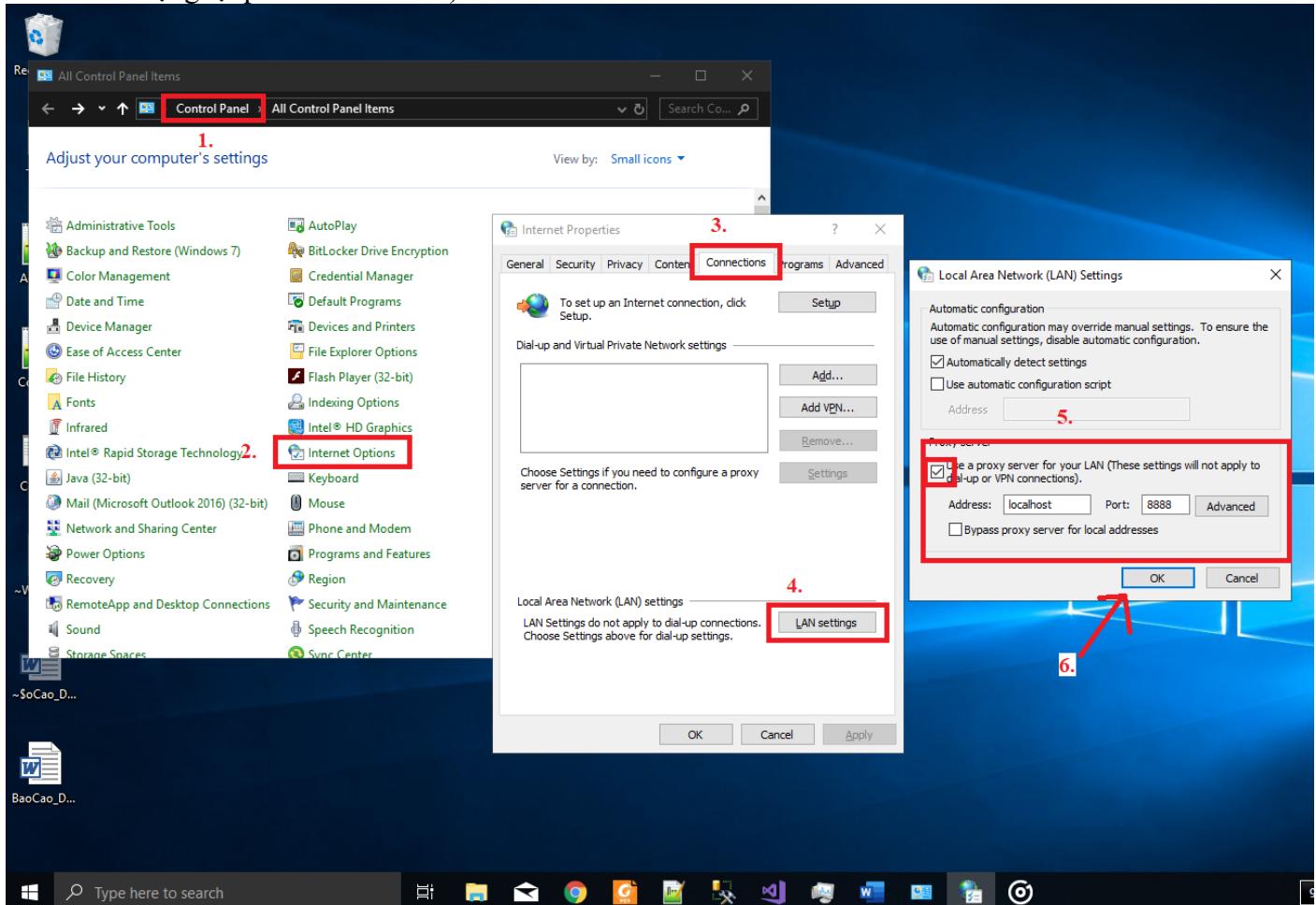
- Các tham số của hàm: buff\_to\_server là chuỗi kiểu const char\* chứa thông điệp yêu cầu mà ta cần gửi tới Web Server, sockfd là biến SOCKET mô tả SOCKET kết nối với Web Server, buff\_length là độ dài của chuỗi buff\_to\_server.

- Chức năng của hàm: Gửi thông điệp yêu cầu (request message) mà proxy server vừa nhận được từ browser đến máy chủ Web Server. Thông điệp này được chứa trong chuỗi buff\_to\_server kiểu const char\*
  - Kết quả sau khi gọi hàm: Nếu thực hiện thành công, hàm trả về 0, ngược lại nếu có lỗi, thông báo lỗi và trả về -1
- i) `int writeToClientSocket(const char* buff_to_client, SOCKET sockfd, int buff_length)`
- Các tham số của hàm: buff\_to\_client là chuỗi kiểu const char\* chứa thông điệp phản hồi mà ta cần gửi về browser tại Client, sockfd là biến SOCKET mô tả SOCKET đại diện cho Client, buff\_length là độ dài của chuỗi buff\_to\_client kiểu const char\*
  - Chức năng của hàm: Gửi thông điệp phản hồi (response) mà proxy server nhận được từ Web Server đến Browser trên máy khách Client. Thông điệp này được chứa trong chuỗi buff\_to\_client kiểu const char\*
  - Kết quả sau khi gọi hàm: Nếu thực hiện thành công, hàm trả về 0, ngược lại nếu có lỗi, thông báo lỗi và trả về -1
- j) `string Proxy_Receive_Send(SOCKET Clientfd, SOCKET Serverfd)`
- Các tham số của hàm: Clientfd là SOCKET mà proxy server dùng để liên lạc với browser, Serverfd là SOCKET mà proxy server dùng để liên lạc với web server
  - Chức năng của hàm: Nhận response message từ Web Server và gửi về cho browser
  - Kết quả sau khi gọi hàm: Trả về thông điệp response mà proxy server nhận được từ Web Server.
- k) `int LoadBlackList(vector<string>& arr)`
- Tham số của hàm: 1 vector gồm các phần tử kiểu string, dùng để lưu các tên miền được đưa vào danh sách đen (bị chặn)
  - Chức năng của hàm: Đọc danh sách đen các tên miền bị chặn từ file blacklist.conf rồi đưa vào vector arr, mỗi phần tử là một tên miền
  - Kết quả sau khi gọi hàm: Nếu thực hiện đọc file và lưu vào vector thành công, hàm trả về 0, ngược lại trả về -1.
- l) `bool CheckDomain(string hostname)`
- Tham số của hàm: một chuỗi kiểu string là tên miền cần kiểm tra
  - Chức năng của hàm: kiểm tra xem tên miền chứa trong chuỗi kiểu string hostname có nằm trong blacklist hay không
  - Kết quả sau khi gọi hàm: Nếu tên miền hostname bị cấm, trả về 1, ngược lại, trả về 0.
- m) Ngoài các hàm trên, chương trình còn sử dụng các hàm hệ thống của thư viện MFC và Winsock2.h như:
- bind(`SOCKET s, const sockaddr* addr, int namelen`): kết nối địa chỉ trong addr với socket s.
  - listen(`SOCKET s, int backlog`): chuyển trạng thái thành lắng nghe kết nối đang đến tại socket s.
  - accept(`SOCKET s, const sockaddr* addr, int* addrlen`): cho phép kết nối vào socket s đang ở trạng thái listen sẵn.
  - send(`SOCKET s, const char* buf, int len, int flag`): gửi đến socket s chuỗi buf.
  - recv(`SOCKET s, const char* buf, int len, int flag`): nhận từ socket s chuỗi buf.
  - getaddrinfo(`PTCSTR pNodeName, PTCSTR pServiceName, const ADDRINFOA *pHints, PADDTINFOA * ppResult`): chuyển đổi từ ANSI host name sang 1 địa chỉ (Ipv4, domain,...).
  - socket(`int af, int type, int protocol`): tạo 1 socket theo các tham số được truyền vào.

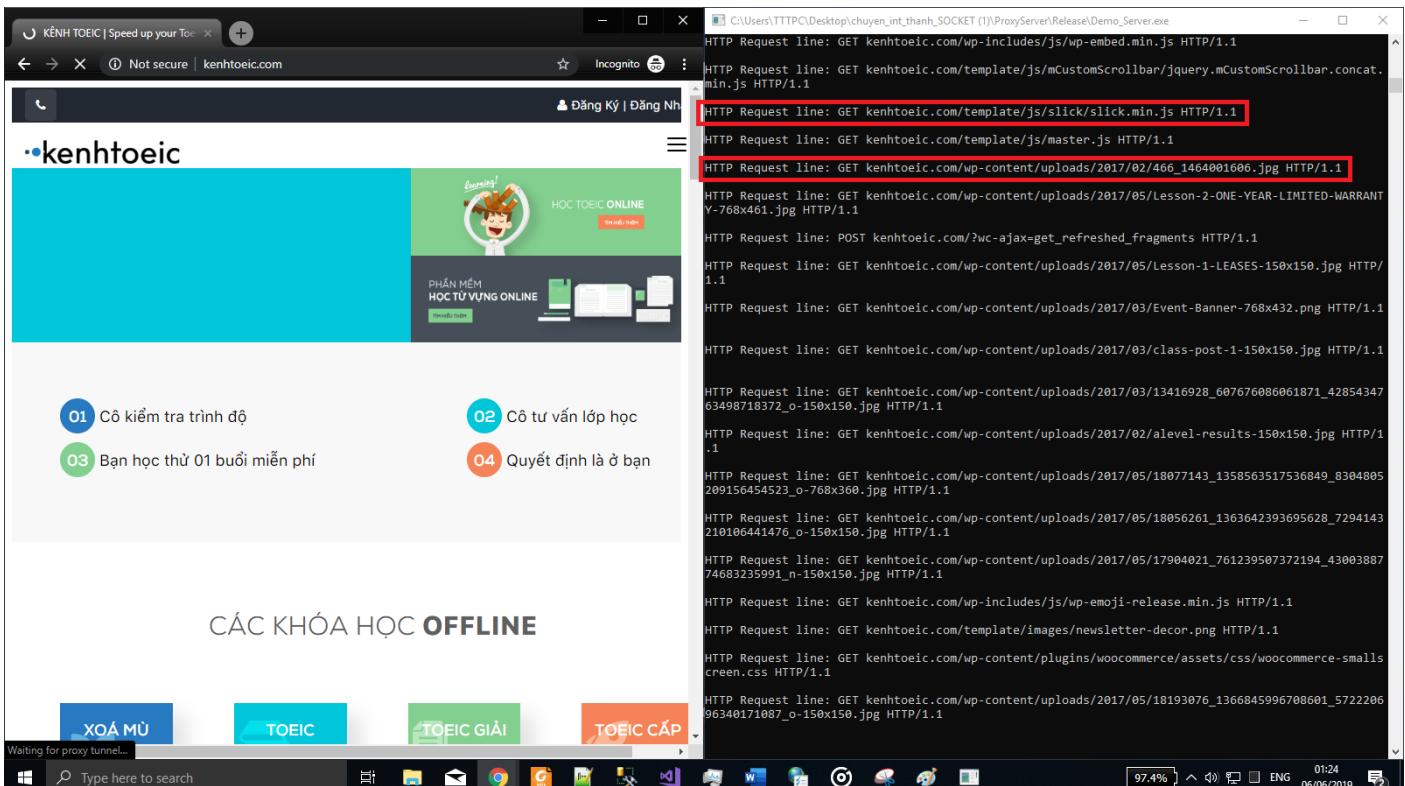
### **3. *Chạy thử chương trình, kiểm tra kết quả chạy được.***

### a) Cách chạy chương trình

- Bước 1: Thiết lập cho mọi browser kết nối đến Proxy: Mở control panel → Internet option → tab Connections → Chọn LAN settings → tick chọn sử dụng proxy và thông số như hình. (address: localhost – client và proxy server cùng nằm trên 1 máy tính vật lý, port 8888 – proxy server hoạt động tại port 8888 như đề)



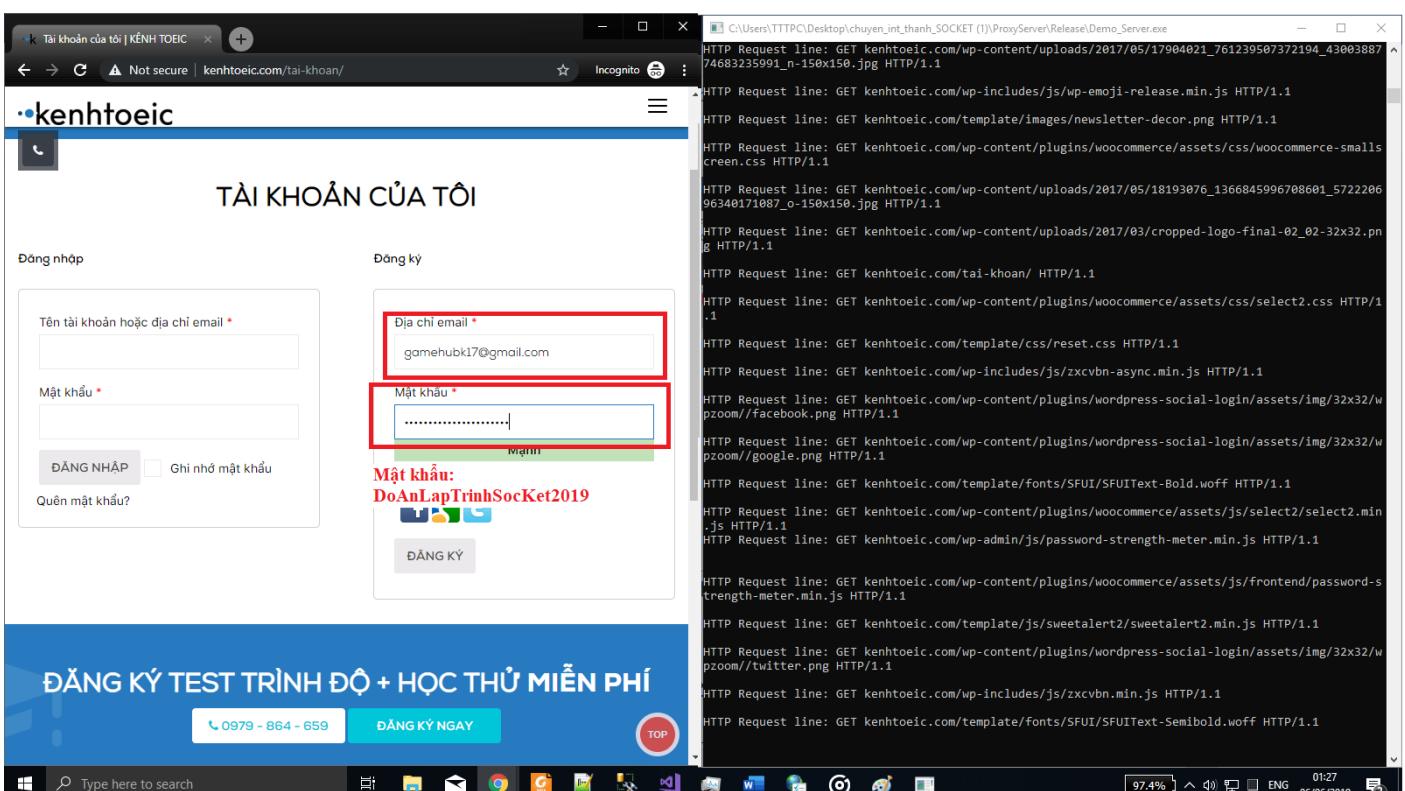
- Bước 2: Chạy file .exe đã build ở chế độ release
- Bước 3: Mở trình duyệt và truy cập vào các trang web hỗ trợ http.



Màn hình console cho ta biết đã có những request nào từ browser được gửi đến Proxy Server và được Proxy Server chấp nhận xử lí bằng cách in ra những request line.

- Đối với các Request không hợp lệ:
  - Thiếu thông tin ở header, không thuộc phương thức GET/POST, sử dụng giao thức HTTPS, .... Proxy Server sẽ bỏ qua không xử lí và không in ra.
  - Đối với những Request có host bị chặn bởi blacklist: Proxy Server sẽ không gửi gói request lên WebServer, gửi gói response báo lỗi 403 về cho browser, in trạng thái ra màn hình và kết thúc tiêu trình.

### b) Phương thức POST



Điền địa chỉ mail đã chuẩn bị từ trước và mật khẩu, sau khi nhấp vào đăng ký, ta thấy gói request POST được gửi đi từ browser và trang web được load thành công vào tài khoản vừa đăng ký.

The screenshot illustrates a browser session on the website [kenhtoeic.com/tai-khoan](http://kenhtoeic.com/tai-khoan). A red box highlights the 'Đăng Xuất | Tài Khoản' button. Below it, another red box highlights the 'ĐĂNG KÝ NGAY' button. On the left, there's a sidebar with 'Bảng điều khiển', 'Địa chỉ', and 'Đăng xuất'. The main content area features a 'CHINH PHỤC TOEIC PART 7' section with a 'PART 7' icon and a 'BẮT ĐẦU HỌC' button. To the right, there's a 'GRAMMAR TOEIC' section with a 'NGỮ PHÁP TOEIC' and a 'BẮT ĐẦU HỌC' button. The proxy server log on the right shows several HTTP requests. One specific POST request is highlighted in red, containing the registration email and password. Red arrows point from the highlighted fields in the browser to the corresponding entries in the proxy log.

### c) Đa luồng phục vụ nhiều client

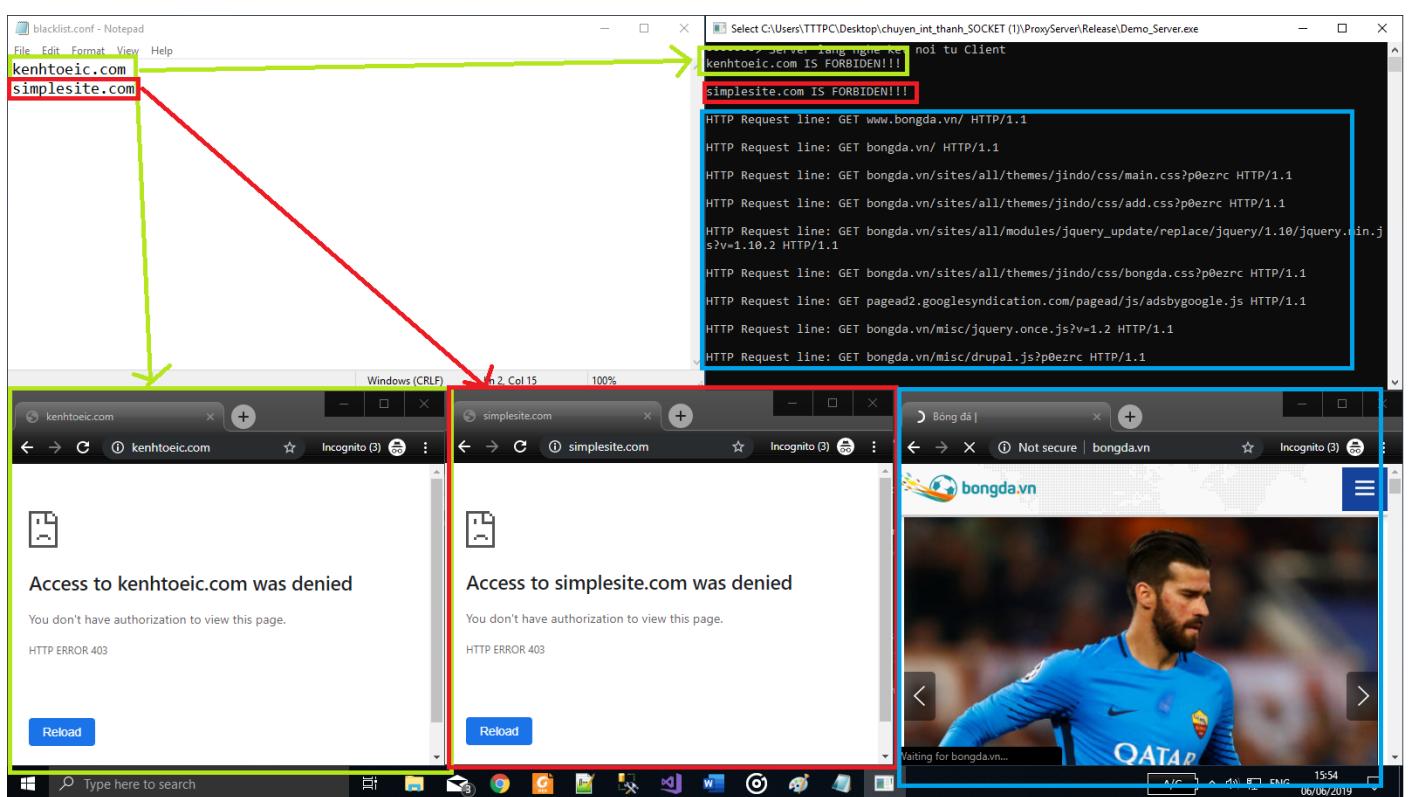
Ta truy cập cùng lúc vào các trang web bằng 1 (hoặc nhiều trình duyệt) với mỗi trình duyệt 1 (hoặc nhiều tab)

The screenshot shows three browser tabs open simultaneously: 'SimpleSite.com', 'Bongda.vn', and 'kenhtoeic.com'. Each tab has multiple requests being processed by a proxy server, as shown in the log window on the right. Red arrows point from the browser tabs to the corresponding requests in the proxy log, illustrating how each client (tab) is handled concurrently by the proxy server.

Ta load đồng thời 3 trang [www.kenhtoeic.com](http://www.kenhtoeic.com), [www.simplesite.com](http://www.simplesite.com), [www.bongda.vn](http://www.bongda.vn) và thấy rằng request của từng trang được gửi song song lên Proxy và được xử lý đồng thời.

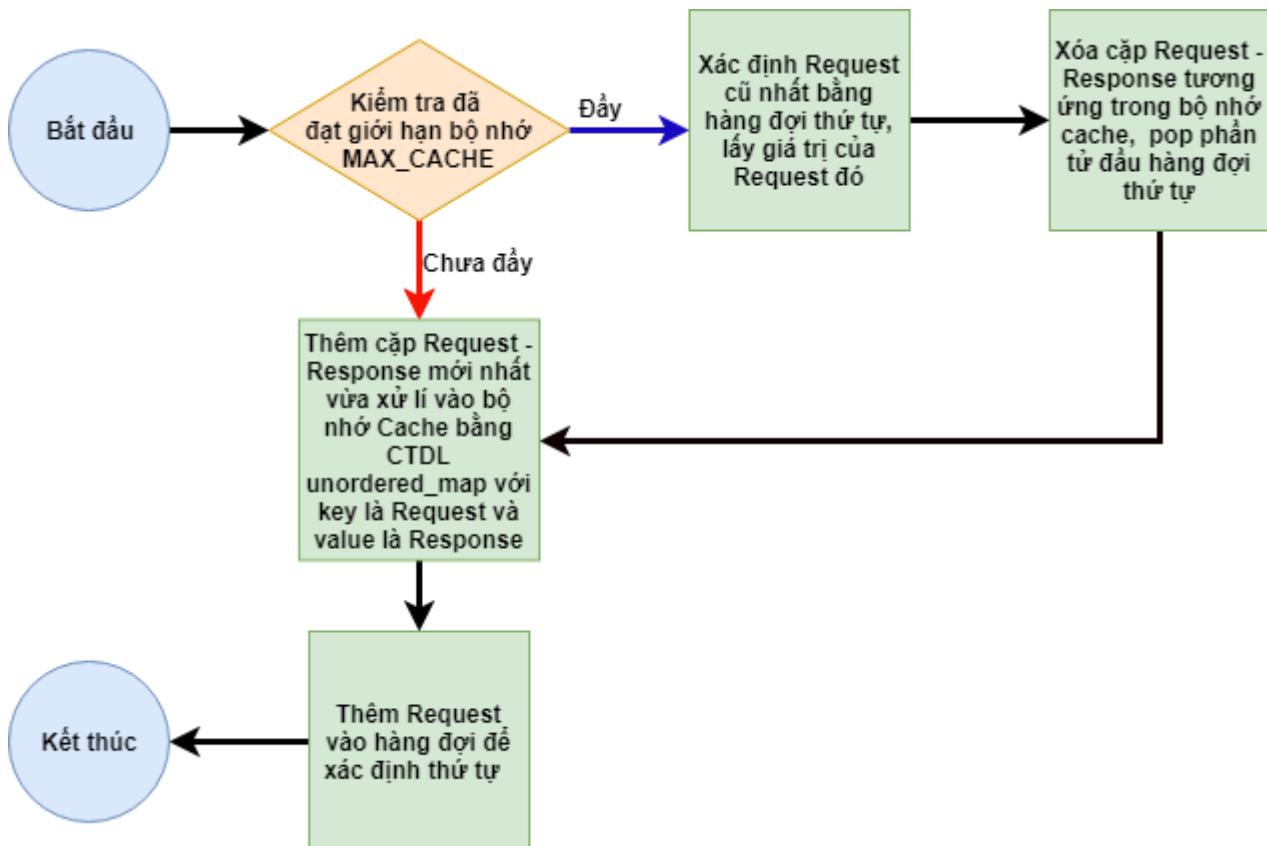
### d) Blacklist

- Bước 1: Ta chuẩn bị sẵn file blacklist.conf, đặt vào chung thư mục với file thực thi .exe.
- Bước 2: Ta tiến hành thử nghiệm trên 2 trang web: [www.kenhtoeic.com](http://www.kenhtoeic.com), [www.simplesite.com](http://www.simplesite.com). Ta thêm domain name của 2 trang web này vào file blacklist.conf, mỗi domain nằm trên 1 dòng, lưu file.
- Bước 3: Khởi động Proxy server
- Bước 4: Mở trình duyệt và truy vấn đến 2 trang trên, ta nhận được kết quả 403 Forbidden, đồng thời, màn hình console báo đã chặn 2 trang web này và không có request nào từ 2 trang này được Proxy Server gửi lên Webserver
- Để chắc chắn Proxy Server vẫn hoạt động bình thường, ta mở thêm 1 trình duyệt và truy cập trang [www.bongda.vn](http://www.bongda.vn) hoặc bất kì trang nào khác hỗ trợ giao thức http mà không có domain thuộc blacklist, ta thấy request được gửi đi bình thường và trang load được nội dung.

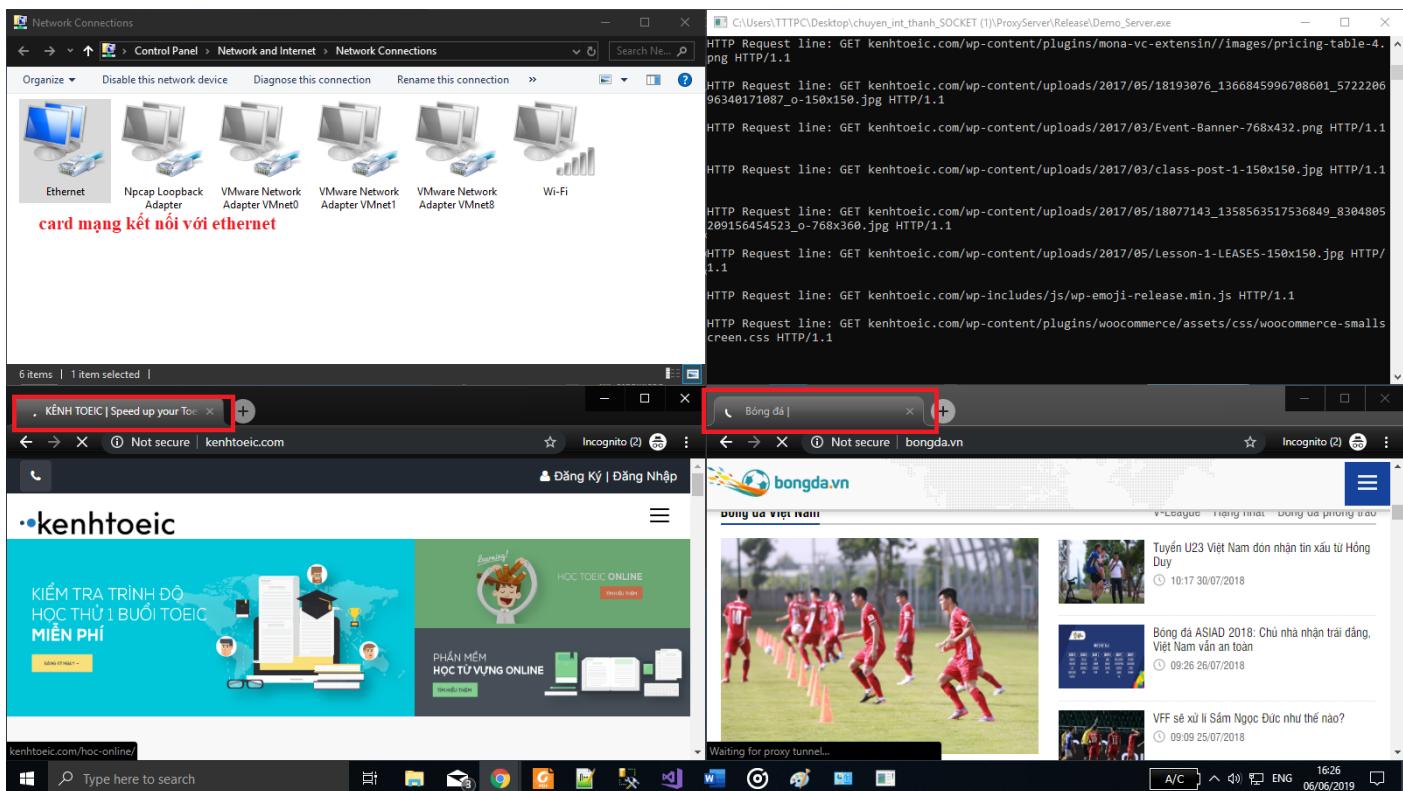


### e) Cache

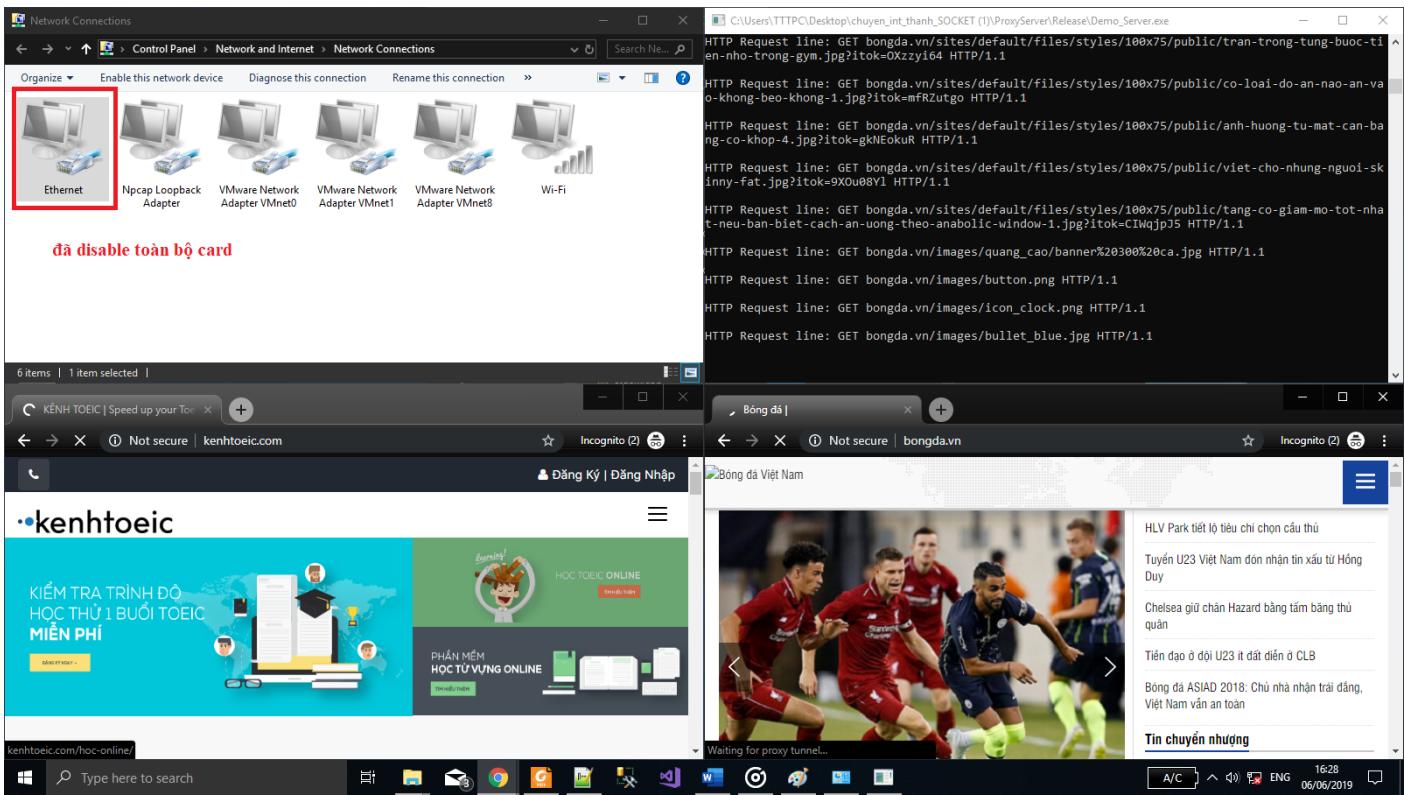
- Cơ chế caching do nhóm đưa ra:
  - o Bộ nhớ cache tồn tại song song với Proxy Server, nếu Proxy Server tắt => bộ nhớ được reset.
  - o Bộ nhớ được lưu trữ trực tiếp lên RAM bằng CTDL unordered\_map để thuận tiện cho việc truy xuất cũng như lưu trữ, kiểm tra, đổi chiều mà không mất thời gian đọc, ghi file gây lỗi.
  - o Việc lưu lại toàn bộ dữ liệu từ khi khởi động Proxy Server là dư thừa và dễ gây bát cập trong quá trình update, hao phí tài nguyên bộ nhớ. Nhóm quyết định sẽ cache 500 request được Proxy Server gửi đi mới nhất (khoảng 3~4 trang web thông thường không quá nhiều hiệu ứng, hình ảnh nặng), nếu có nhu cầu mở rộng bộ nhớ khi đủ tài nguyên, có thể dễ dàng chỉnh sửa tại define MAX\_CACHE trong source code mà không làm ảnh hưởng đến các chức năng.
  - o Cơ chế caching, update và thay thế request cũ được thể hiện bằng sơ đồ sau (MAX\_CACHE=500):



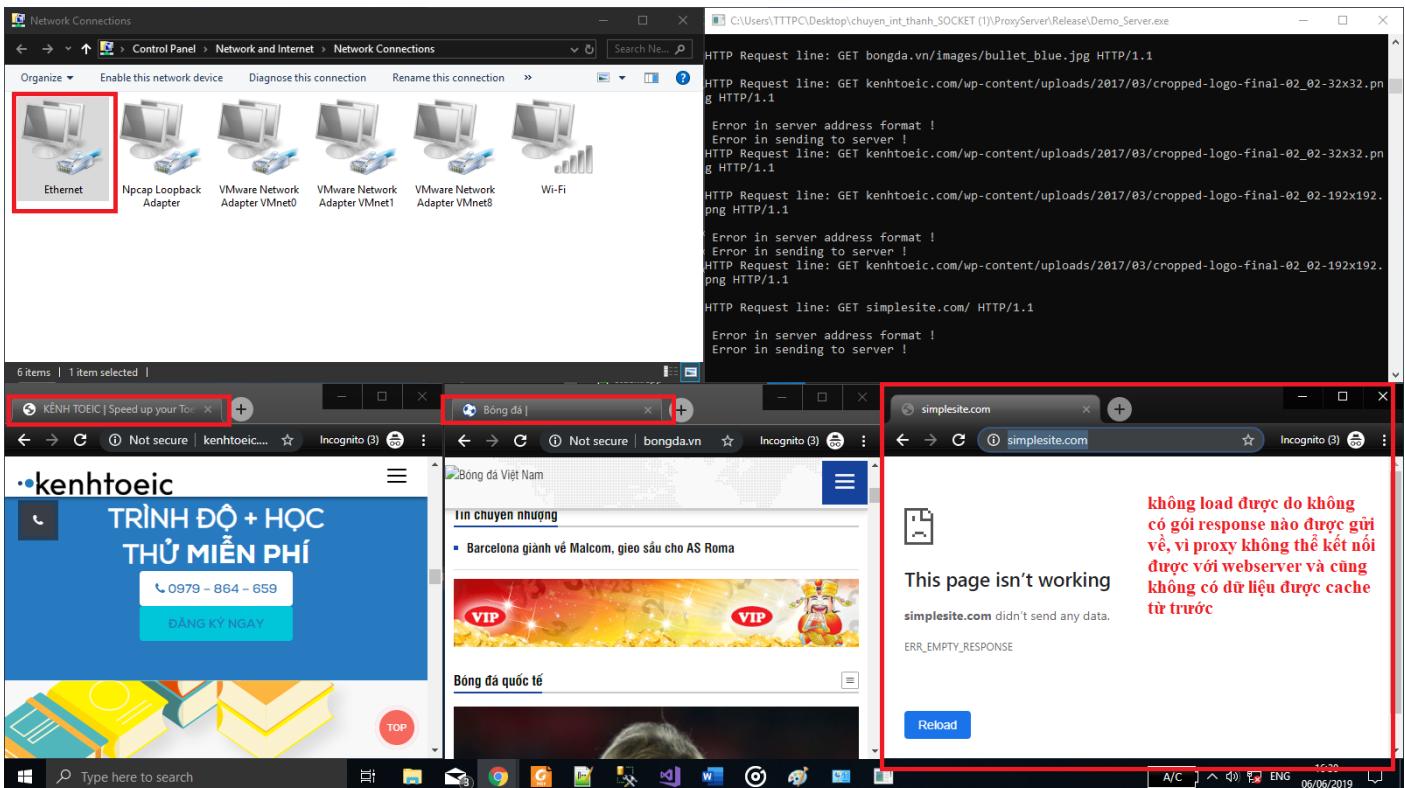
- Bước 1: ta khởi động Proxy, mở trình duyệt, truy cập 3 trang web: [www.kenhtoeic.com](http://www.kenhtoeic.com), [www.bongda.vn](http://www.bongda.vn) (đã bỏ chặn trong blacklist) cùng lúc và đợi đến khi 3 trang được load hoàn toàn.



- Bước 2: Vẫn giữ Proxy Server hoạt động, ta tắt 2 trình duyệt để xóa đi caching của Browser cho những trang web này, vì ta đang muốn kiểm tra cơ chế caching của Proxy Server.
- Bước 3: Ta disable toàn bộ card mạng của máy tính để chắc chắn rằng không có Request nào được gửi lên Webserver cũng nhưng không có response được nhận thêm từ Webserver, ta mở trình duyệt và truy cập lại 2 trang web với tên miền giống hệt lúc nãy



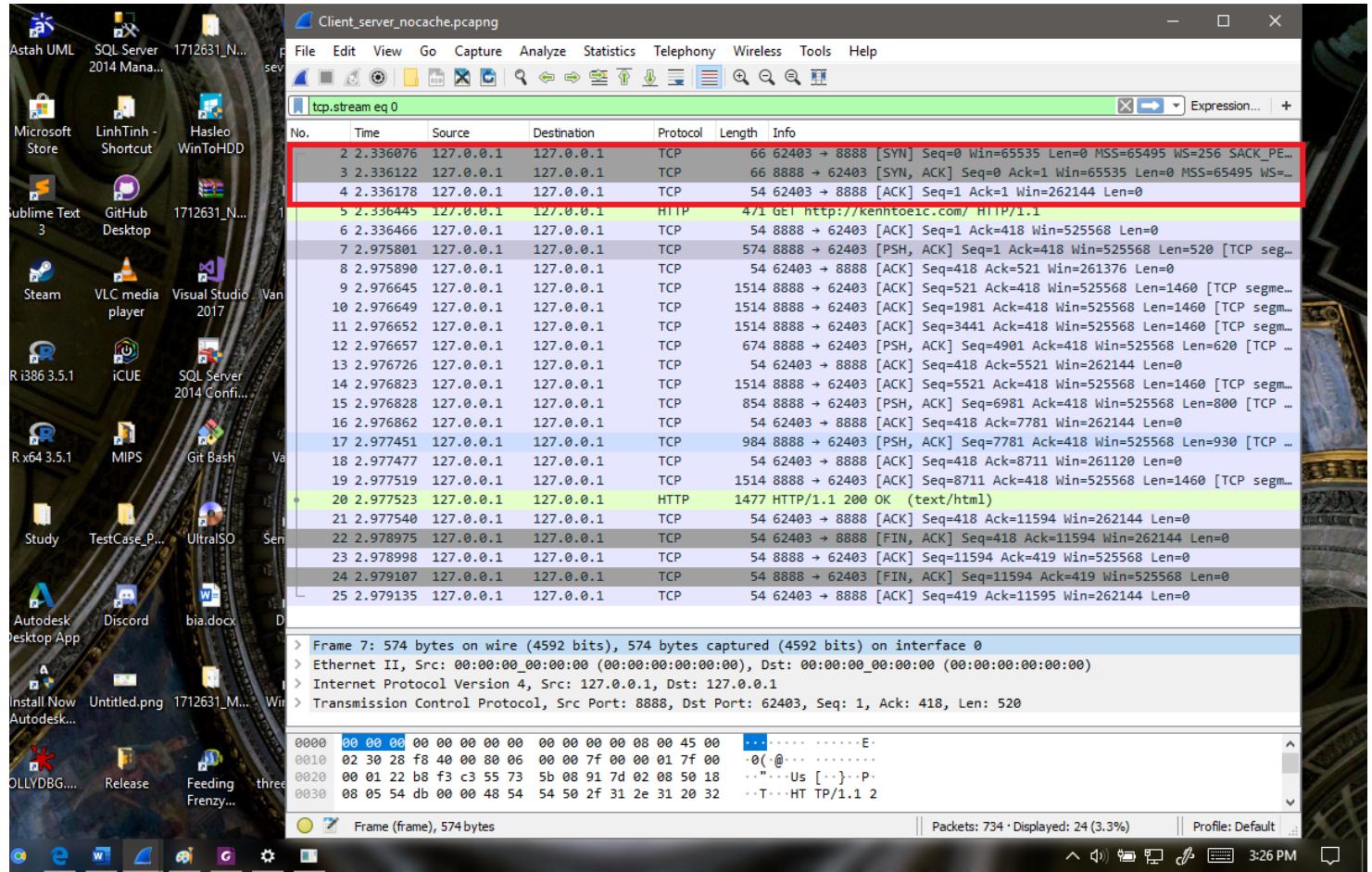
- Kết quả: ta load được 2 trang web nhờ bộ nhớ cache đã lưu trữ các gói request và response với nhau, ta load thứ 1 trang thứ 3 là [www.simplesite.com](http://www.simplesite.com) mà lúc trước chưa cache, kết quả là không có nội dung được thể hiện lên vì không hề có data nào được gửi về browser.



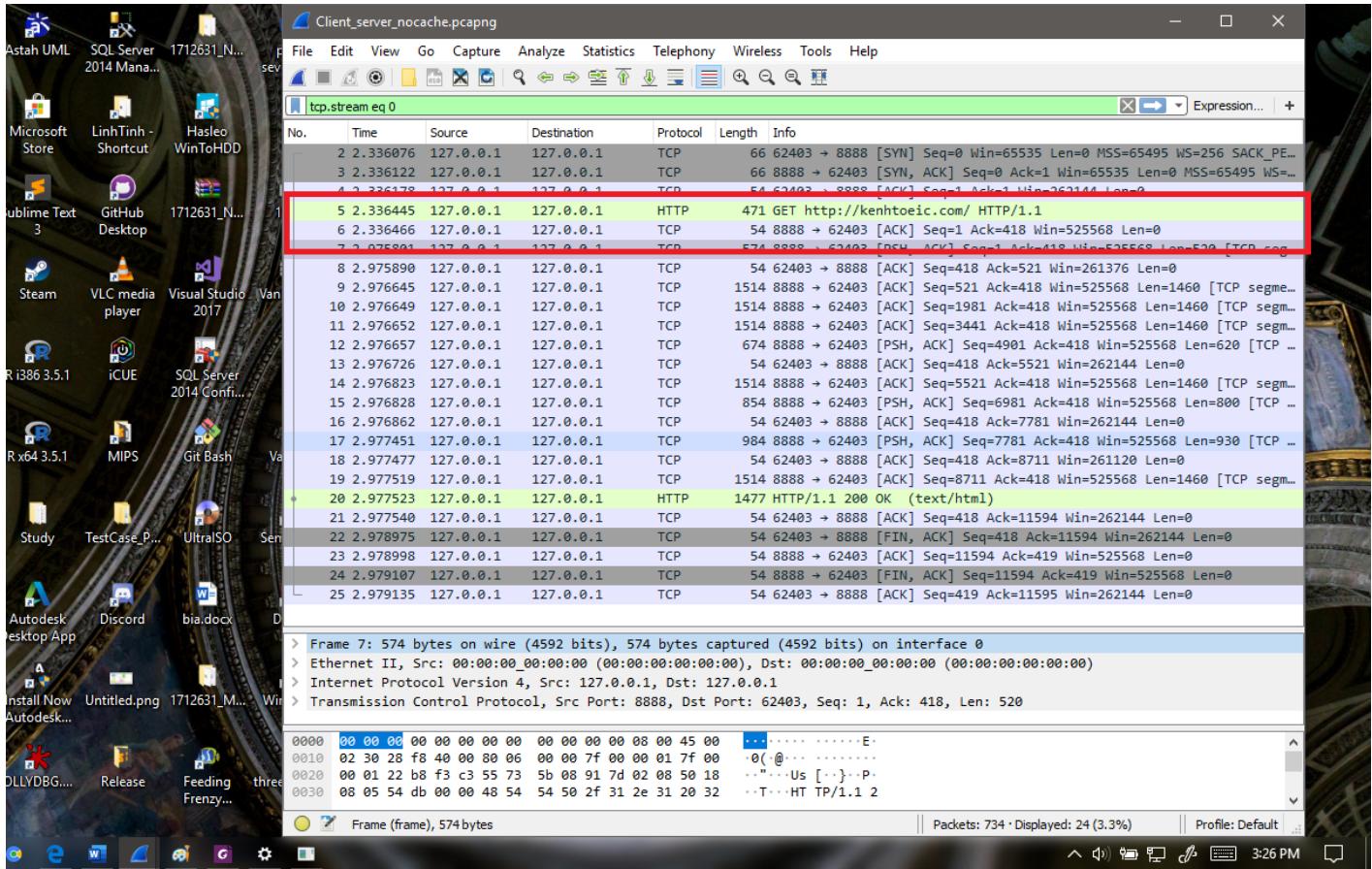
#### 4. Dùng Wireshark bắt gói tin tại Proxy Server

Mô tả quá trình gửi nhận dữ liệu giữa Client – Proxy Server:

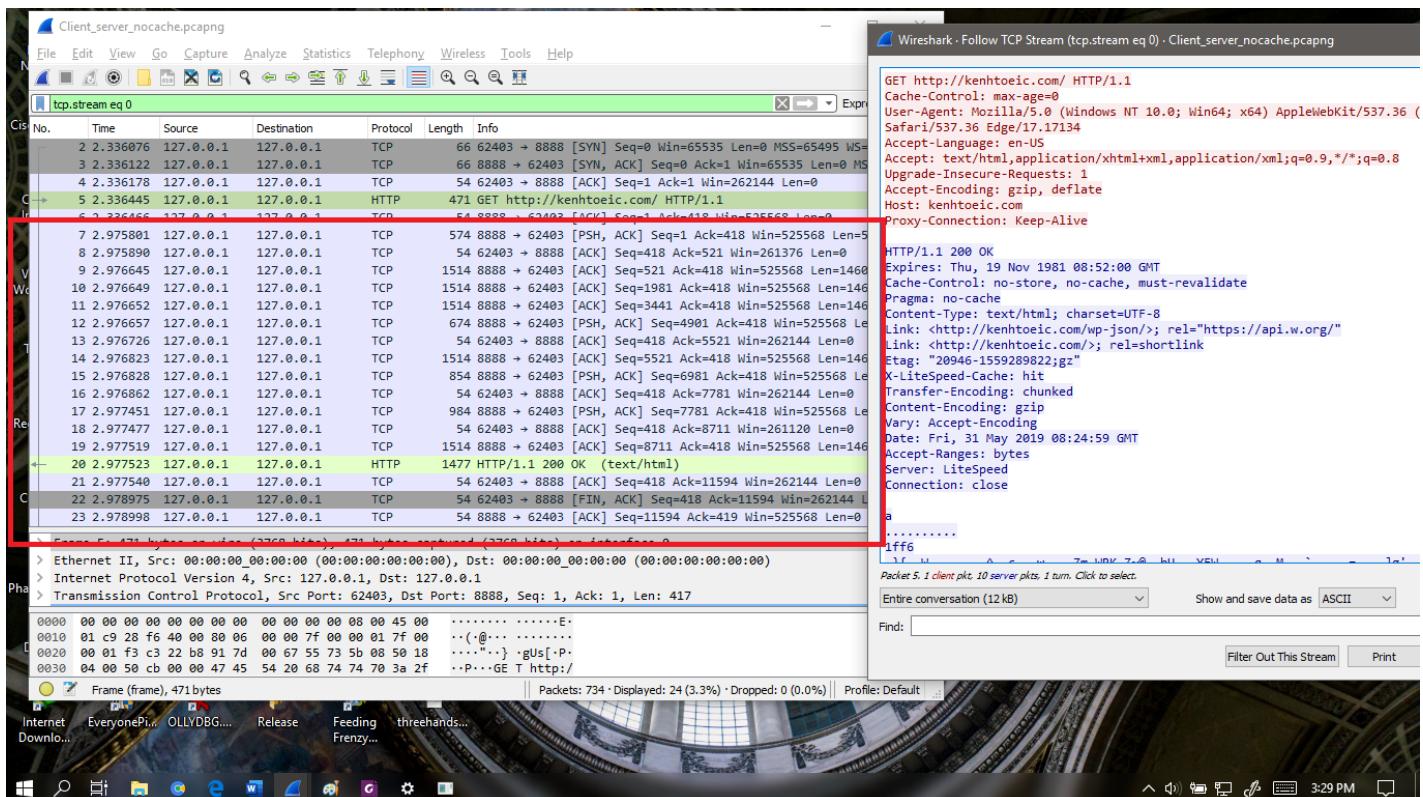
- Đầu tiên, client tạo kết nối đối với proxy server tại port 8888 thông qua quá trình bắt tay ba bước. Sau khi đã thiết lập kết nối, proxy server chờ yêu cầu từ client.



- Proxy server nhận gói tin có chứa request và gửi trả gói tin ACK cho Client xác nhận đã nhận được yêu cầu.



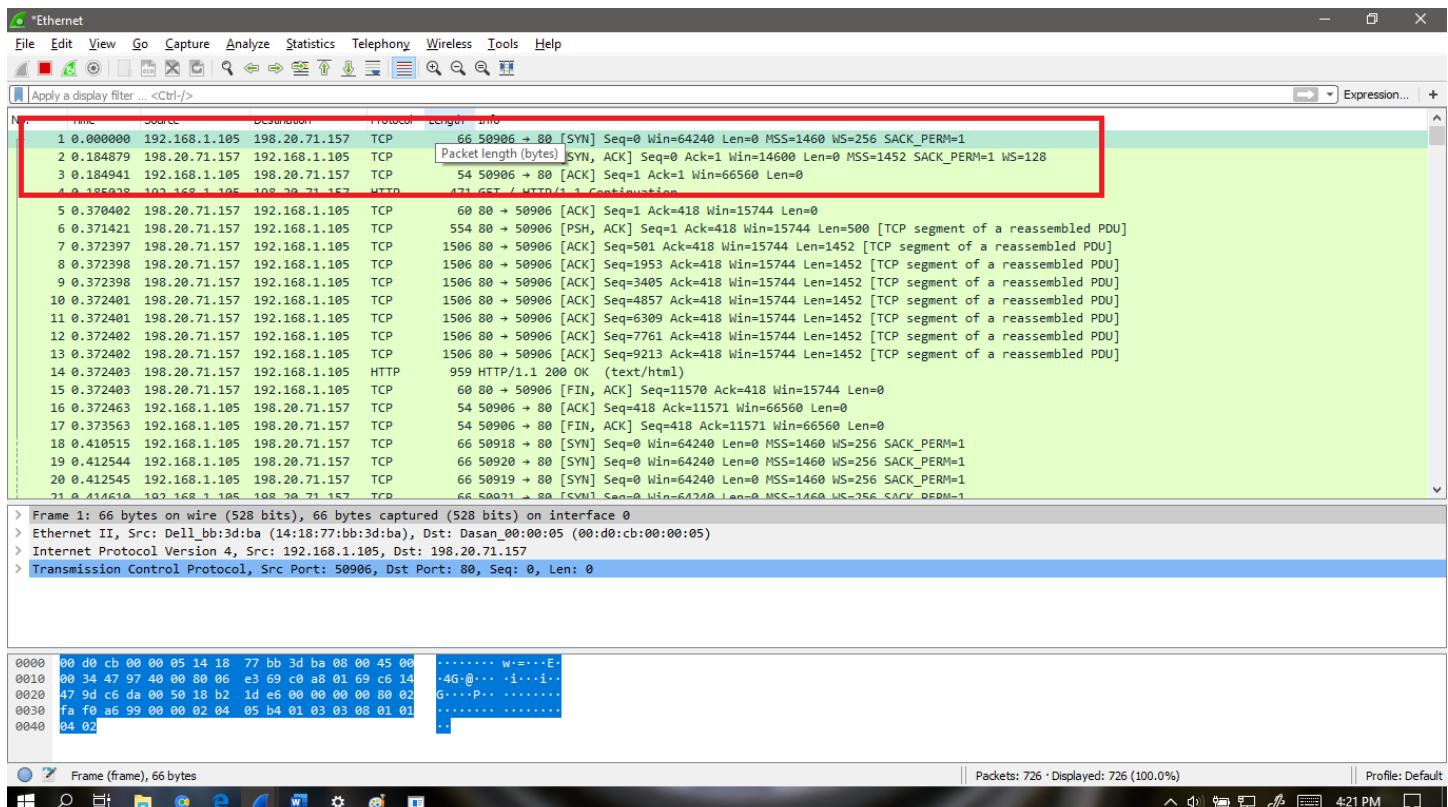
- Proxy sẽ kiểm tra xem có bản sao nào của đối tượng được yêu cầu đã lưu trước đó hay chưa, nếu chưa proxy sẽ kết nối và nhận dữ liệu từ server, nếu đã có proxy sẽ gửi nó về cho client thông qua thông điệp phản hồi.
- Proxy server gửi các gói tin phản hồi TCP, HTTP chứa dữ liệu cho client và sau đó yêu cầu đóng kết nối khi đã hoàn tất.



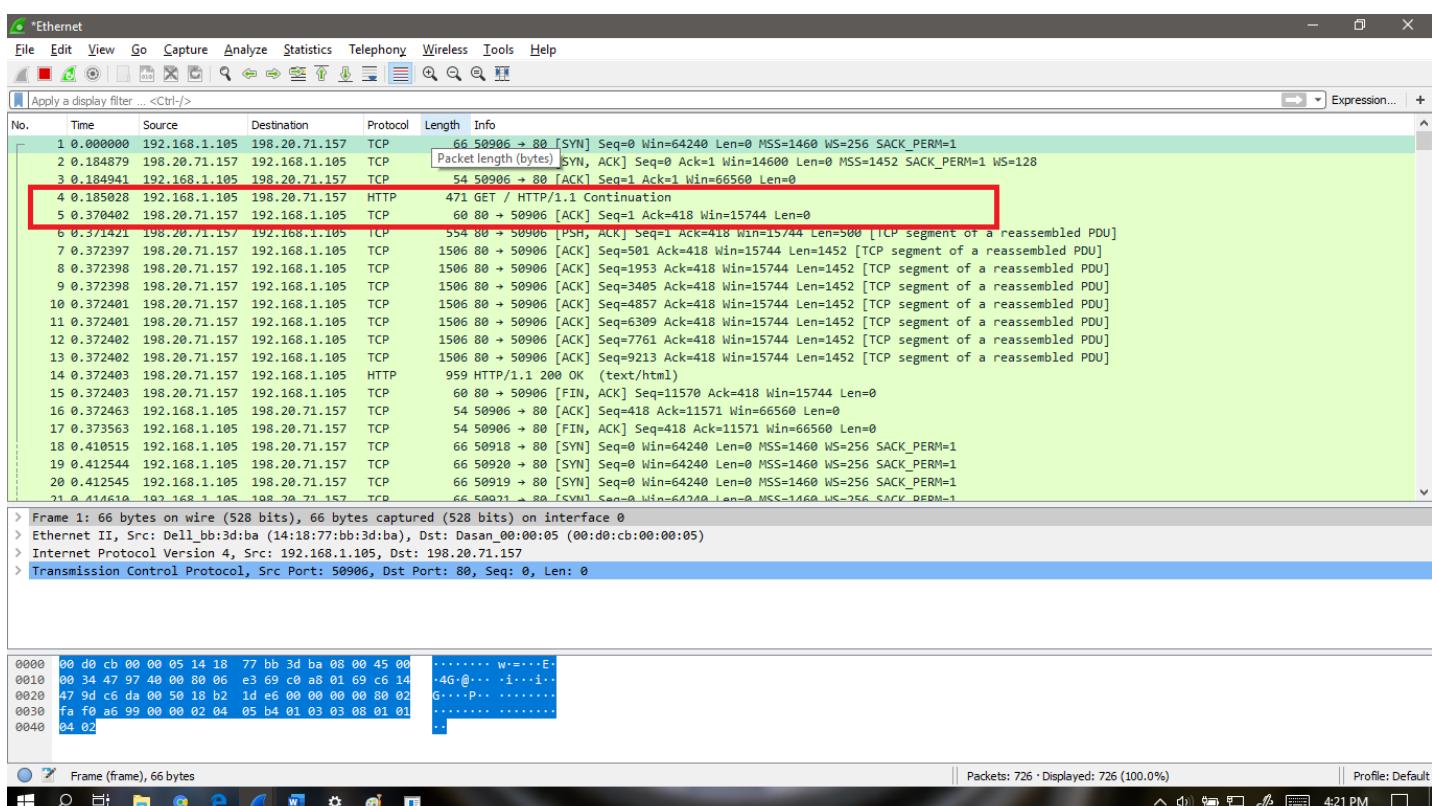
- Quá trình này được lặp đi lặp lại đến khi trang web được load xong (Ở đây chúng ta hiểu là khi tải xong trang web thì browser sẽ không gửi gói tin request đến proxy server)

### Mô tả quá trình gửi nhận dữ liệu giữa Proxy Server – Web Server:

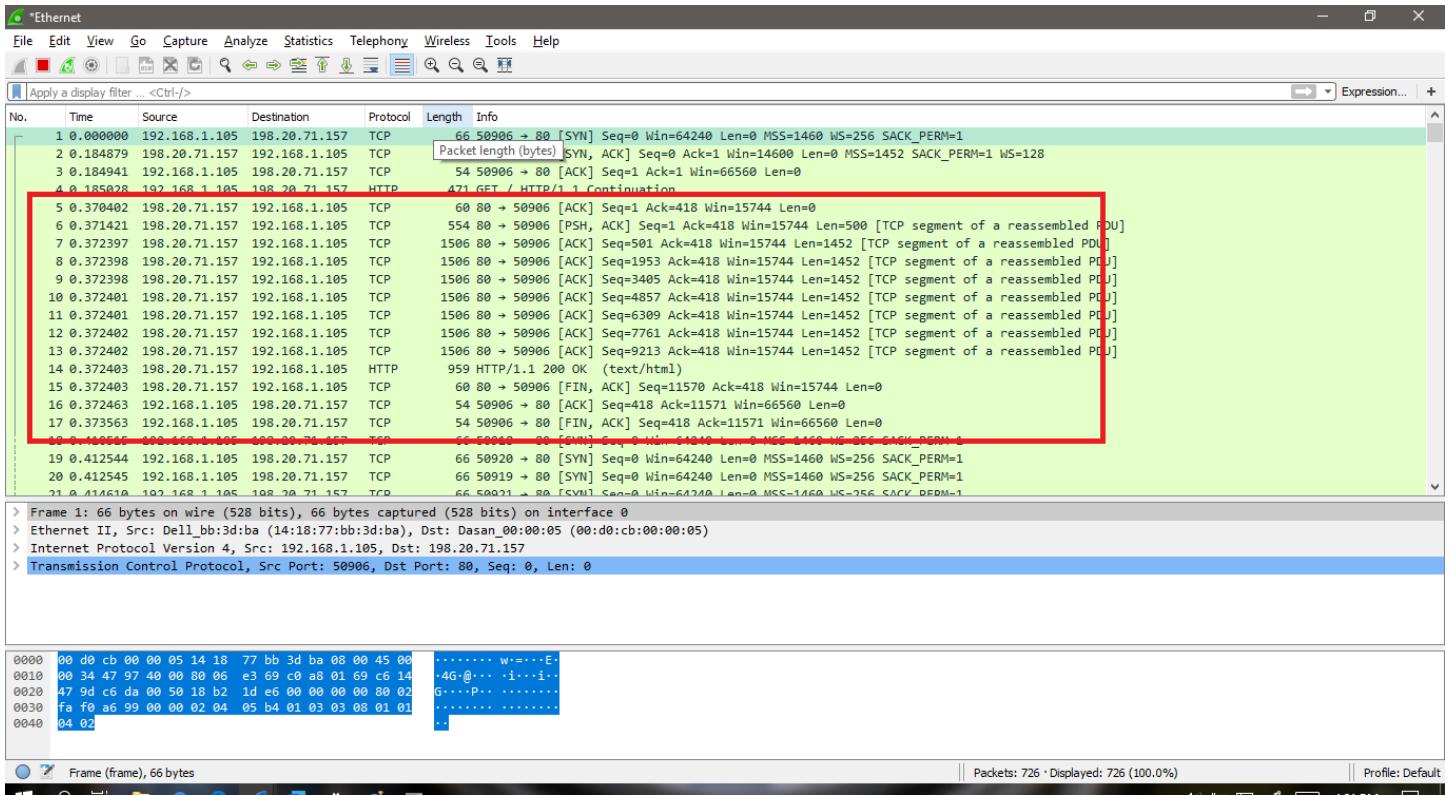
- Tại proxy server, sau khi nhận request từ browser, proxy thực hiện thiết lập kết nối với web server thông qua quá trình bắt tay ba bước



- Sau đó, proxy gửi gói tin request đó đến web server nếu chưa có bản sao lưu, sau khi nhận được web server phản hồi bằng gói tin ACK xác nhận đã nhận được request.



- Web server sẽ gửi các gói tin TCP, HTTP chúa đến proxy, sau đó yêu cầu đóng kết nối khi đã gửi hoàn tất.



- Quá trình này lặp lại đến khi proxy không còn request gửi đến web server nữa, ở đây ta hiểu web được load xong
- Proxy server sẽ gửi những gói tin này đến cho client thông qua kết nối được thiết lập với client. Và lưu 1 bản sao để phục vụ cho các client tiếp theo truy cập.

## 5. Giải thích tại sao chúng ta lại cần Proxy Server

Proxy Server trong hệ thống mạng máy tính là một máy tính đóng vai trò như một người đại diện trung gian giữa những máy khách (clients) và những máy chủ (servers) mà những máy khách này muốn truy cập đến, thực hiện nhiều chức năng cho nhiều mục đích khác nhau như bảo mật, kiểm duyệt nội dung, caching (bộ nhớ đệm) ...

- Cụ thể, đối với mục đích bảo mật, proxy server nắm một danh sách đen các địa chỉ IP hay các tên miền (domain), các dịch vụ trên Internet mà các client bị cấm truy cập vào. Khi proxy server nhận được yêu cầu truy cập tới một địa chỉ IP nào đó từ một client, nếu nó nằm trong danh sách đen trên proxy server, proxy server ngay lập tức chặn và gửi lại thông báo từ chối truy cập (Access Denied) cho client đó. Danh sách đen có thể là các trang web nhiễm mã độc, các trang web có nội dung bị cấm ở một vùng lãnh thổ hay quốc gia nào đó.
- Tương tự với mục đích kiểm duyệt nội dung, proxy server sẽ bắt và hủy những yêu cầu đến từ client mà proxy server cho rằng nội dung trong đó là không hợp lệ, không được phép.
- Đối với mục đích Caching, những Proxy Server được thiết kế với một trong các chức năng của nó là HTTP caching, là những máy chủ trung gian, duy trì một bộ nhớ đệm lưu trữ các loại đối tượng tập tin Internet như html, hình ảnh, âm thanh, video, văn bản, gif, ...v.v... mà chúng được gửi và nhận thông qua giao thức HTTP. Khi một máy khách (client) gửi một yêu cầu truy cập tới một tài nguyên nào đó, proxy server kiểm tra xem nó có đang giữ một bản sao của tài nguyên đó không. Nếu có rồi thì tài nguyên này sẽ được chuyển phát tới ngay cho client. Nếu không, yêu cầu của client sẽ được proxy đại diện và gửi tới Web Server, sau đó proxy nhận được tài nguyên từ Web Server mà client yêu cầu, proxy sẽ cùng lúc thực hiện lưu trữ vào bộ nhớ đệm và chuyển tài nguyên này về lại cho client.

## III. Nguồn tài liệu tham khảo:

- Slide lý thuyết trên lớp.
- Slide thực hành trên moodle
- Giáo trình mạng máy tính
- Youtube
- GitHub