EX1 – HTTP Proxy – Client Side

Goals:

The purpose of this project is two-fold:

1. Give students hand-on experience with socket programming
2. Help students better understand application-level protocols by implementing a well-known protocol, HTTP.

In this programming assignment you will write simple HTTP Proxy. Students are not required to implement the full HTTP specification, but only a very limited subset of it.

What is the Meaning of Proxy?

Proxy refers to someone authorized to carry out an action on behalf of someone else, and proxy servers deliver this in the online world. A proxy server acts as a gateway between users and the internet and prevents access to anyone outside the network. Regular internet access via a web browser enables users to connect directly with websites. But a proxy acts as an intermediary, which communicates with webpages on users' behalf.

Why An HTTP Proxy Can Be Useful?

One reason is it examines Web traffic to identify suspicious content that can be a virus or other type of intrusion.

In this exercise you will program the client side which implements the following:

- **A client side which gets a request, an input from the user, of a certain URL.**
- **Then the client checks whether the request already sent to an external http server.**
- **If the request was never sent**
    - **It constructs an HTTP request based on the user's input.**
    - **Sends the request to an external HTTP Web server.**
    - **Receives the response from the server.**
    - **Saves the file locally.**
    - **Displays the response message on the screen.**
- **Else**
    - **It constructs an HTTP response and displays it on the screen.**

You should support only IPv4 connections.

What is HTTP?

HTTP stands for Hyper Text Transfer Protocol and is used for communication among web clients and servers. HTTP has a simple stateless client/server paradigm. The web client initiates a conversation by opening a connection to the server. Once a connection is set up, the client sends an HTTP request to server. Upon receiving the HTTP request from the client, the server sends an HTTP response back to the client.

HTTP Proxy server – Program Flow

In case there is an HTTP proxy, the proxy gets the requests from the client, if it has the requested file, it sends it back and otherwise, it sends a request to the server to retrieve the file and send it back to the client after saving a copy of the file locally.

HTTP Request Structure

An HTTP request consists of two parts: a header and a body. In this project, the basic HTTP request from a client doesn't contain a body.

The first line of any request header should be:

Method Request-URI Version. An example HTTP1.0 request is:

GET /index.html HTTP/1.0

Host: www.jce.ac.il

The request header and body are separated by two sets of carriage return and line feed (\r\n). Since we do not need the body, the end of a header marks the end of a request. Using a C char string, the example request above should be: **"GET /index.html HTTP/1.0\r\nHost: www.jce.ac.il\r\n\r\n"**.

What is a URL?

Uniform Resource Locators (URLs) are formatted strings that identify resources in the web: documents, images, downloadable files, electronic mailboxes, etc. It generally has the format: Protocol://Host[:port]/Filepath.

In this project, when a port is not specified, the default HTTP port number of 80 is used. For example, a file called "foo.html" on HTTP server "www.yoyo.com" in directory "/pub/files" corresponds to this URL: http://www.yoyo.com/pub/files/foo.html. The default HTTP network port is 80; if an HTTP server resides on a different network port (say, port 1234), then the URL becomes: http://www.yoyo.com:1234/pub/files/foo.html.

Program Description and What You Need to Do:

You will write the program cproxy.c.

The proxy requires one mandatory argument which is the <URL> and one optional parameter wich is the flag '-s'.

Command line usage: cproxy <URL>. <URL> specifies the URL of the object that the client is requesting from server. The URL format is http://hostname[:port]/filepath.

You can assume that the url has to start with http://

If the flag -s exists, your program should open a browser to present the page, more later.

In this exercise, we will implement only the GET request.

If the url has no path, the path in the request should be "/".

Each file the proxy gets from the server, it saves it on its local filesystem in the following format. If, for example, the url is http://www.yoyo.com:1234/pub/files/foo.html. The proxy should have a directory called www.yoyo.com, under www.yoyo.com there should be a directory called pub, under pub there should be a directory called files, and under files foo.html should be saved. For each directory that does not appear in the filesystem, the proxy should create it.

In a similar way, when the proxy gets the above url, it has to check if the file appears in its filesystem under the specified path. If the url has no path, the proxy should look for index.html under the domain directory.

If the file is found, the proxy creates http response in the following format:
HTTP/1.0 200 OK\r\n

Content-Length: N\r\n\r\n

Where N is the file size in bytes.

Implement the following logic in cproxy.c

1. Parse the <URL> given in the command line.
2. Check if the file appears in the local filesystem (under current directory).
    a. If yes
        i. construct HTTP response with the requested file
        ii. print the following msg:
        printf("File is given from local filesystem\n");
    b. Otherwise:
        i. construct an HTTP request based on the options specified in the command line
        ii. print the request to stdout in the following format:
        printf("HTTP request =\n%s\nLEN = %d\n", request, strlen(request));
        where request holds your constructed request.
        iii. Connect to the server

iv. Send the HTTP request to the server
v. Receive an HTTP response
vi. Save the file locally

c. Display the response on the screen
d. Print the length of the response in the following format:
printf("\n Total response bytes: %d\n",size);
where size is the number of characters in the response.
e. If the flag -s is specified, use 'system(path);' to present the file using the browser, **where path is the local path to your saved file**. This will help you verify that the file is saved correctly.

Running Example:

1. If the file is given from local filesystem:
File is given from local filesystem
HTTP/1.0 200 OK
Content-Length: 405

<the file itself – 405 bytes>
Total response bytes: 442

2. If the file is given from the server:
HTTP request =
<http request>
LEN = 143
HTTP/1.0 200 OK
Content-Length: 405

<the file itself – 405 bytes>
Total response bytes: 442

Your proxy should close connection after getting the file. You should use HTTP/1.0 but add the header Host to your request.

Error handling:

1. In any case of a failure in one of the system calls, use perror(<sys_call>) and exit the program (for errors on gethostbyname call herror instead).
2. In any case of wrong command usage, print "Usage: cproxy <URL> [-s]"

Enter a new line after each error message.

Examples:

The website www.josephwcarrillo.com is an example for a website that can be reached by http (and not https), so you can use it for your testing.

1. ./cproxy http://www.josephwcarrillo.com/JosephWhitfieldCarrillo.jpg
   Request:
   GET /JosephWhitfieldCarrillo.jpg HTTP/1.0
   Host: www.josephwcarrillo.com

2. ./cproxy http://blala
   This is not usage error, you will fail when trying to get the IP address for that host.

Useful functions:

- Strchr
- Strstr
- Strcat
- Google some more useful functions.

Compile the proxy:

gcc -Wall –o cproxy cproxy.c

cproxy is the executable file.

What to submit:

You should submit a tar file called ex1_id where id is your id number. The tar file should contain cproxy.c and README. Find README instructions on the course web-site.

Test Case:

You can use the client to connect to any **HTTP** server. You should try different URLs to make sure that the client works correctly.