EX2 – HTTP Proxy – Phase 1

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.

You will implement the following:

An HTTP Proxy that gets requests from the user and check if the requested file appears in its DB, if yes, the file is provided from the local filesystem, otherwise, it constructs an HTTP request based on user's command line input, sends the request to a Web server, receives the reply from the server, saves the file locally and displays the reply message on screen. You should support only IPv4 connections.

Background:

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.

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.

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 proxy1.c.

The proxy requires one argument which is the <URL>.

Command line usage: proxy1 <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://

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 is 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.

In proxy1.c, you need to:

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.

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.

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: proxy1 <URL>"

Enter new line after each error message.

Examples:

1. ./proxy1 http://www.ptsv2.com/t/ex2
   Request:
   GET /t/ex2 HTTP/1.0
   Host: www.ptsv2.com

2. ./proxy1 http://blala
   This is not usage error, you will fail when trying to get the IP address for that host.

Useful function:

Strchr, Strstr, strcat

Compile the proxy:

gcc -Wall –o proxy1 proxy1.c

proxy1 is the executable file.

What to submit:

You should submit a tar file called ex2_id where id is your id number. The tar file should contain proxy1.c and README. Find README instructions in the course web-site.

Test the proxy:

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