Programming Assignment 2

Assigned: 10/01/2018

Due: 10/15/2018

This assignment is modified based on Kai Shen’s networking class in Rochester [[link](http://www.cs.rochester.edu/~kshen/csc257-fall2014/)].

# Description

In this assignment you are asked to build a multi-threaded Web proxy server that is capable of delivering Web content on behalf of a remote Web server. When a user's browser is configured to connect via a proxy server, her browser establishes a connection to the proxy server's machine and forwards its complete request to the proxy server rather than the “real” Web server. The proxy server accepts user's HTTP requests and forwards them to their final destinations---essentially it introduces an extra “hop” between the user's browser (or the client) and the Web server. There are several reasons why people want to use a proxy server instead of connecting to the remote Web server directly: 1) users want to anonymize themselves from the Web server; 2) a corporation might want to monitor or restrict employees' Web surfing; 3) a proxy can be used to locally cache data in order to reduce the amount of global traffic. Multi-threading is not crucial to the functionality of a Web proxy server, but it is important to allow the server to process multiple simultaneous requests in parallel, a must-have feature for a practical server.

# Requirements in Detail:

The proxy server acts as an HTTP server to client browsers while it acts as an HTTP client to the “real” Web server. The protocol specification for version 1.1 of HTTP is defined in [RFC 2616](http://www.w3.org/Protocols/rfc2616/rfc2616.html). You will find that this document is quite long and you may not be able to read it through. For your relief, you are asked to implement a very small part of the complete protocol, which will be explained below. We do try to be as specific as possible in this assignment description in order to reduce your need to refer to the lengthy protocol specification. But you should be prepared to do so occasionally.

An HTTP-compliant Web server supports HEAD, POST, and GET methods. Your proxy server only needs to support GET method. To serve each request, we first need to parse the request line and headers sent by the client. Since we will only support GET method, we only care about Web page name in request line. A request for the proxy server may look like this:

GET http://www.foo.com/mumbo.html HTTP/1.1  
 Connection: close  
 ... …

The requested Web page name contains the Web server name www.foo.com and the requested file on that server /mumbo.html. In this case, your proxy server should make a TCP connection to Web server www.foo.com at the default port 80 and ask for file /mumbo.html by sending the following request:

GET /mumbo.html HTTP/1.1\r\n

Connection: close\r\n\r\n

After sending a request to the “real” Web server, an HTTP response including the requested file will be received at the proxy server. The proxy server should then forward the content to the client. There are three parts in an HTTP response message: the status line, the response headers, and the entity body. The status line and response headers are terminated by an empty line (or an extra “\r\n”). In short, an HTTP response message may look like the following:

HTTP/1.1 200 OK  
 Date: Thu, 24 Jan 2013 15:29:52 GMT  
 Content-Type: text/html  
 Server: Joe's Web proxy server  
 Connection: close  
   
 ... ... <content of the main.html>

The status line and most of the header fields should be forwarded to the client without modification.

The server should be able to handle multiple simultaneous service requests in parallel. This means that the Web proxy server is multi-threaded. In the main thread, the server listens at a fixed port. When it receives a TCP connection request, it sets up a TCP connection socket and services the request in a separate thread.

The “Connection: close” setting in the above examples serves an important role --- disabling the persistent connection feature in HTTP 1.1. A persistent connection allows multiple Web requests to flow through a single TCP connection. However, the support for persistent connections is somewhat challenging and it is not required for this assignment. Note that when you test your proxy server through a standard Web browser, the “Connection: close” setting will likely not be in the HTTP request. Your proxy server can insert this setting into the request before forwarding it to the Web server so that the persistent connection is turned off.

# Program Skeleton

Here we provide a hint on what the program skeleton may look like. You are, however, not required to design your program this way.

Main routine {  
 Parse the command line input (server port number);  
 Create a server socket listening on the specified port;  
 For each incoming client socket connection {  
 Spawn a worker thread to handle the connection;  
 Loop back to wait for/handle the next connection;  
 }  
 }  
   
 Worker thread routine {  
 Read the request line and header fields until two consecutive new lines;  
 (Note that a new line can be a single "\n" or a character pair "\r\n".)  
 Examine the first line (request line);  
 If the request method is not "GET" {  
 Return an error HTTP response with the status code "HTTP\_BAD\_METHOD";  
 }  
 Make TCP connection to the "real" Web server;   
 Send over an HTTP request;  
 Receive the server's response;  
 Close the TCP connection to the server;  
 Send the server's response back to the client;  
 Close the connection socket to the client.  
 }

# Testing

For testing, you can configure a Web browser to use your proxy server. For Firefox 17.0.1, you can configure it in the following way: 1) Choose “Edit” in the Menu; 2) Click “Preferences”'; 3) Choose “Advanced” Tab; 4) Choose “Network” Sub-Tab; 5) Click “Settings” for Connection; 6) Click “Manual proxy configuration” and fill the server name as well as its port number for “HTTP Proxy”. Make sure your proxy server is running and now all HTTP requests from your browser are served via the proxy server!

# Turn-in

You are asked to turn in your source code files, and a PDF document with screenshots of how you tested your proxy. Please note, your proxy server should log information about what object is being requested each time.

# Note

* Only use the socket library in Python, you are not allowed to use libraries like httplib, urllib, etc.
* Remember to test your proxy with website begin with http (do \*not\* test with webiste begin with https)
* Make sure to make your proxy multi-threaded.