4/22/2015

Kodjo Komedja, Durko Justin | Computer Networks | April 22 2015

**UNIVERSITY OF NORTH TEXAS**

**PROJET**

Multithreaded web proxy

**Table of Contents**

1. Problem being solved
2. Approach
3. Implementation

**Problem**

In this project, you will develop a web proxy. When your proxy receives an HTTP request for an object from a browser, it generates a new HTTP request for the same object and sends it to a remote server that is hosting the requested object. When the proxy receives the corresponding HTTP response with the object from the remote server, it creates a new HTTP response, including the object, and sends it to the client. The proxy will be muti-threaded, so it will be able to handle multiple requests at the same time. Your proxy should have the following features: 1. Blocking web sites if they are found in a black list (www.facebook.com, www.youtube.com, www.hulu.com, www.virus.com) 2. Filtering out inappropriate language (http://www.hyperhero.com/en/insults.htm) from a requested site 3. Caching sites that have already been visited.

**Approach to solve the problem**

The goal of TCP proxy server is to act as an intermediary forwarding TCP based connection from external client applications onto remote server. The communication architecture diagram start from from the client to proxy which treat the request according to built-in parameters then contact the remote HTTP server and get the response back from server, then read the content to verify the matching of set of requirements before rendering a view (response) to client browsers.

// Up flow communication: Client ---> Proxy---> HTTP server +---------------+

// +---->------> |

// +-----------+ | | Remote Server |

// +---------> [x]--->----+ +---<---[x] |

// | | TCP Proxy | | +---------------+

// +-----------+ | +--<--[x] Server <-----<------+

// | [x]--->--+ | +-----------+

// | Client | |

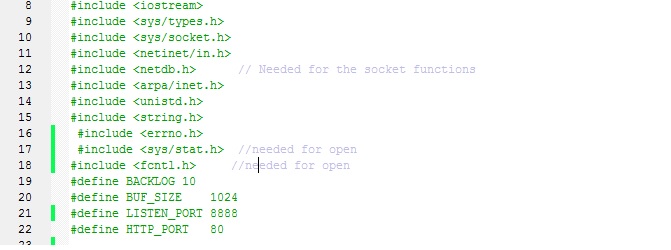
// | <-----<----+

// +-----------+

// Down flow communication: Client <--- Proxy <--- HTTP server

**Understanding socket programing**

On the proxy we start by defining the port number (8888) we wish to run our application on. We include built-in library headers which are important for defining an end point to socket, and open ,read, write files, multithreading concept, hashmap (saving blocking specific websites).



They are followed by defined variables *int sock-send, sockaddr\_in, addr\_send* which are required to initialize a socket end point connection and to make the binding. The socket is created with *sock\_send=socket(PF\_INET,SOCK\_STREAM,…)*. We connect to server socket with *connect(sock\_send, &addr\_send,….)*

**Understanding multithread in C++**

On the proxy server, required variables are *pthread\_t , thread\_result*. We use a socket binding for clients browser apps, and we listen to port number from client application and launch a thread to handle the client operation. A thread count is initialized to 10 and decrement each time a new client browser is connected (which means our server can only accept 10 users) which is enough for multithreading with a clients - proxy server setup.

TO BE COMPLETED BY KODJO

Another example of multithreading (two clients connected server)

To bE COMPLETED BY KODJO

**Understanding writing data to file as *ofstream* and reading from file as *ifstream , and both fstream***

This allows us the capability to save our data in file which is very useful for caching request. Because the proxy server application is executed on (UNT computer machines), we can only save our data in file and send the content to client browsers if they ask for a web content that already existed . We will clarify step by step how our code work in next following lines.

**Understanding reading from and writing to sockets**

TO BE COMPLETED BY KODJO

**Understanding Hashmap data structure:** *Defines a data structure for clients records in file*

The solution involved the handling of blocking website contents which were handled with STL Map. The blocking content within that map class contain relevant information to clients to take place. In addition, using hashmap, make the parsing bad word faster, than using a regular read() method, or loop.

**Implementation**

TO BE COMPLETED BY JUSTIN AND KODJO