# CS 408– Computer netowrking Project
# Due: June 10, 2021, 23 pm

## Web Police

This project is about the **application layer** of the network protocol stack. It involves application layer software development, proxy server, application layer protocol principles with HTTP, socket programming and multithreading. As the Internet becomes more and more popular, all kinds of websites become available and all of these are open to anyone, at any age. Some websites include unwanted content, some others violence, etc. This naturally alarmed many parents since their children could use the Internet and get exposed to such content. Access to unwanted content during working hours from offices or schools is a similar problem. One possible solution to these problems is using special programs installed by the parents or system administrators to filter unwanted websites and prevent unauthorized access.

## Description:

In this project, you will implement an application to emulate a proxy server to limit access to certain websites. For this purpose, you will design a GUI based client application that will establish a connection to web police and after that, the user can send HTTP request messages using GET commands. Web police will receive the commands and read the words forbidden (which are provided by the parents, administrators) in the URL requests from a configuration file (a simple text file that the parents can edit) and check whether a requested URL contains any of these words. If URL contains forbidden words, then the program sends a message (i.e., HTTP response) to the client indicating that the access is not allowed (you can replay“Access forbidden” by HTTP 403). Otherwise, WebPoilce sends the response of actual page content to the client GUI. Figure 1 depicts a general overview of the project.

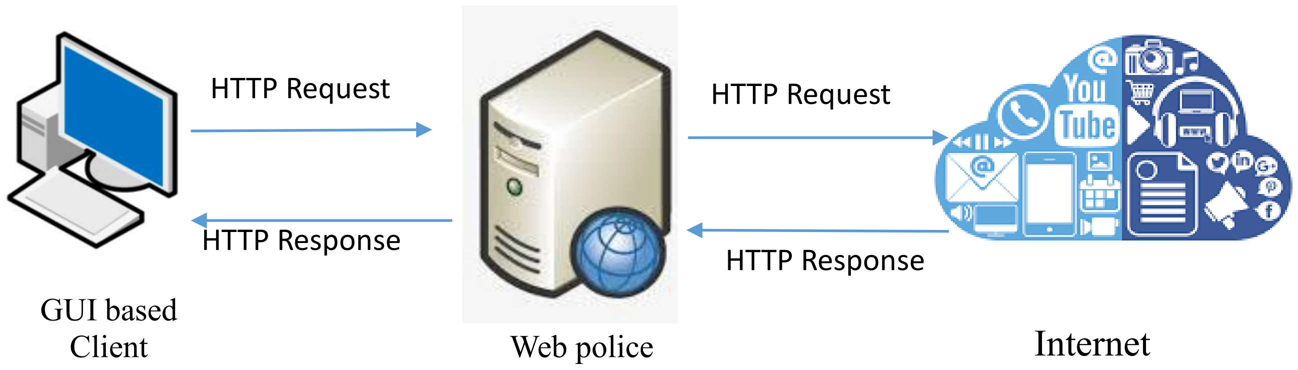

Figure 1. Overview of project

## Tips:

In the Web Police program, you will open a TCP socket at the specified port and wait for connections. After establishing a connection, if a user request from GUI arrives (i.e., the client types a GET command to access a website), the application will process it to decide whether access is allowed or not and then. The response will reply back to the client using the same socket. After the client opened its connection to WebPolice (and its request has been checked and is allowed), the real web page response needs to be sent to the client. Therefore, since the user already gave her request, now it is WebPolice's turn to forward the request so that the user can get the response. Thus, WebPolice acts as a client and requests the web page. This means you need to open a connection to the webserver (without closing the connection to the user), forward the request over this connection, get the reply and forward it back to the client. You will use threads to handle multiple connections (simultaneously and/or at different times).

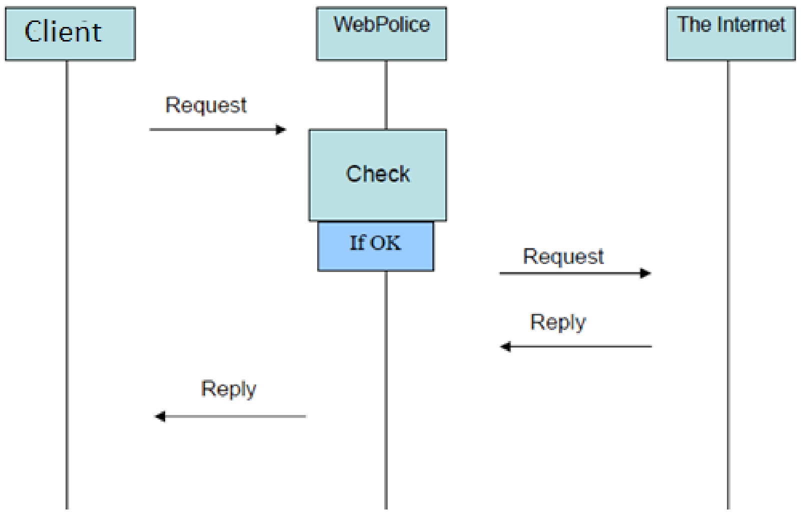

Figure 2. Client and server interactions

As shown in Figure 2, if WebPolice cannot find any of the banned words in the incoming request, the request is forwarded to the actual webserver to which they are addressed initially. The reply from the server, then, needs to be forwarded to the browser.

In this project, multithreading is a MUST. Using multithreading, more than one GUI client can be supported simultaneously by your WebPolice.
Remember, the requests generated by the browser are path-relative, meaning that a request for an image residing on a web page starts with "/" and continues with the path to that image. (e.g., the entire URL is: http://www.example.net/example.jpeg , but the request is /example.jpeg.)

To simplify your development, we provide the skeleton for the web police part of the project in the attachment (3 classes), and you need to fill the To-DO parts in the source code.

## Requirements:

1. You must use HTTP request and response messages explicitly such that your network software replies to the client and requests from the web server using HTTP.

2. Web Police must check the banned words from a BannedWords.txt file that you will create.
    - If URL contains a banned word, replay back by “Access Denied” message to clinet.
    - If word check is passed, Web Police should forward a real HTTP response to the client.

## Programming Rules

1. Preferred languages are C#, Java and Python, but C# is recommended
2. When a window is closed in your application, **all threads related to this window should be terminated.**
3. Your program should be portable. It should not require any dependencies specific to your computer. We will download, compile and run it. If it does not run, it means that your program is not running. So do test your program before submission.

## Submission

1. Submit your work to SUCourse+.
2. Delete the content of debug folders in your project directory before submission.
3. Create a folder named **WebPolice** and put your server related codes and other files here.
4. Create a folder named **Client** and put your client related codes and other files here.
5. Create a folder named **XXXX_Surname_Name**, where XXXX is your SUNet ID. Put your Server and Client folders into this folder.
    a. Compress your XXXX_Surname_Name folder **using ZIP or RAR**.
6. You will be invited for a demonstration of your work. Date and other details about the demo will be announced later.
7. Twenty-four hours late submission is possible with 10 points penalty (out of 100).