Programming Assignment #2

CSCE 3530 – Introduction to Computer Networks

Fall 2020

100 Points
Due: 09/29/2020, 11:55 PM

Instructions: Compile the C programs in one of the CSE (cse01 – cse06) servers and make sure it's working. Comment your code. Create a Makefile to compile the source code and to clean the binaries. Create a readme file that describes how to compile, execute, and test the code. Create an assignment folder with folder name as *euid_PA2* (example: xyz0202_PA2) and add all the source files, Makefile, and the readme file to the folder. Please create a zip archive of your assignment folder and upload the zip file to Canvas. **Not following the above instructions could result up to 50% deduction from your assignment score. Late submissions are not allowed.**

Objective:

Create a web proxy server that can be connected by a single client and would only allow http requests. The proxy server should be able to cache up to five recent websites.

Requirements:

1. Create a C-based proxy server and a client using TCP sockets
2. The proxy server should be able to accept and service single client's http requests
3. The proxy server should be able to process the client's request and forward the response to the client
4. The proxy server should run on cse01.cse.unt.edu machine and the client should run on cse02.cse.unt.edu machine
5. The proxy server should be able to cache at least five recent requested webpages, if available
6. When a http request is made to a website that is cached, the proxy server should return the cached page.

Procedure:

1. Create a C-based server that can accept single client's request using TCP sockets
2. The created proxy server should also be able to process the client HTTP request
3. Make sure the proxy server runs on cse01.cse.unt.edu and the format to start the proxy server as follows

```
./pyserver <port_number>
```
where pyserver is the proxy server executable and port_number is the port number on which the proxy server listens

4. Create a C-based client that can connect to the proxy server using TCP sockets
5. Make sure the client runs on cse02.cse.unt.edu and connects to the proxy server. The user can request the desired web page using the below format
```
./client <port_number>
url: <url>
```
where client is the client executable, port_number is the port number on which the client connects the server and url is the requested url starting with www
6. Once the proxy server gets a request from the client, it checks the cache for the requested page. If the page is not found in the cache, then it forwards the request to the web server. Figure 1 shows the overall architecture
7. The proxy server checks for the response from the web server
8. If the HTTP response is 200, the returned web page from the web server is cached in the proxy server. The proxy server stores the webpage in a file and assigns a filename based on the time of visit. The filename format is YYYYMMDDhhmmss. Where YYYY is the year, MM is the month, DD is the day, hh is the hour in 24-hour format, mm is the minutes, and ss is the seconds when the website was visited
9. A list file (list.txt) is created which stores the URL of the webpage and the associated cached web page filename
10. The list file stores five recent URLs. The cached websites that are not listed in the list file should be deleted
11. Once the returned web page is cached, the web page is forwarded to the client. Verify to see if the returned page is same as the browser returned page.
12. If the HTTP response is not 200, do not cache the web page instead forward the HTTP response to the client
13. When the client requests a webpage that is in the list.txt file (cached) the stored page is returned
14. Test web caching by accessing multiple websites. Most websites are not http websites so only select http websites. Some http websites are given here
15. A sample list.txt is available on Canvas for reference.

Deliverables:

1. Commented server and client C code
2. A Makefile to compile (*make*) all the source code and to clean (*make clean*) all the executables
3. A readme file that describes how to compile, execute, and test the code.

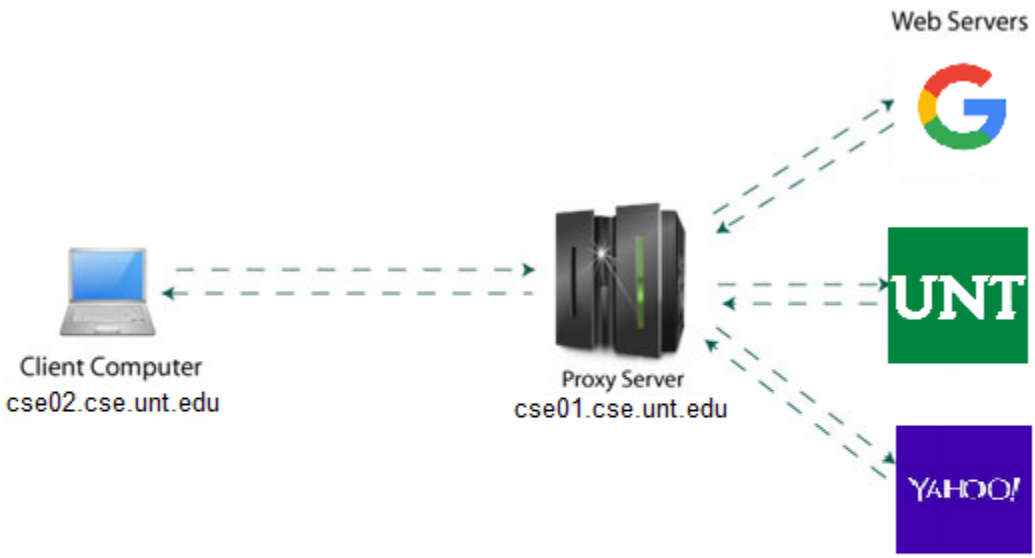

Figure 1. Overall Architecture