Report

CMPE156 Name: Shuli He, ID:she77@ucsc.edu
Concurrent-File-Transfer

Usage:

Use command 'make' (Makefile) to compile the source code. (Using g++)

Run:

./proxy <listen-port> <forbidden-sites-file>

example:

./proxy 12345 forbidden-sites.txt

Design:

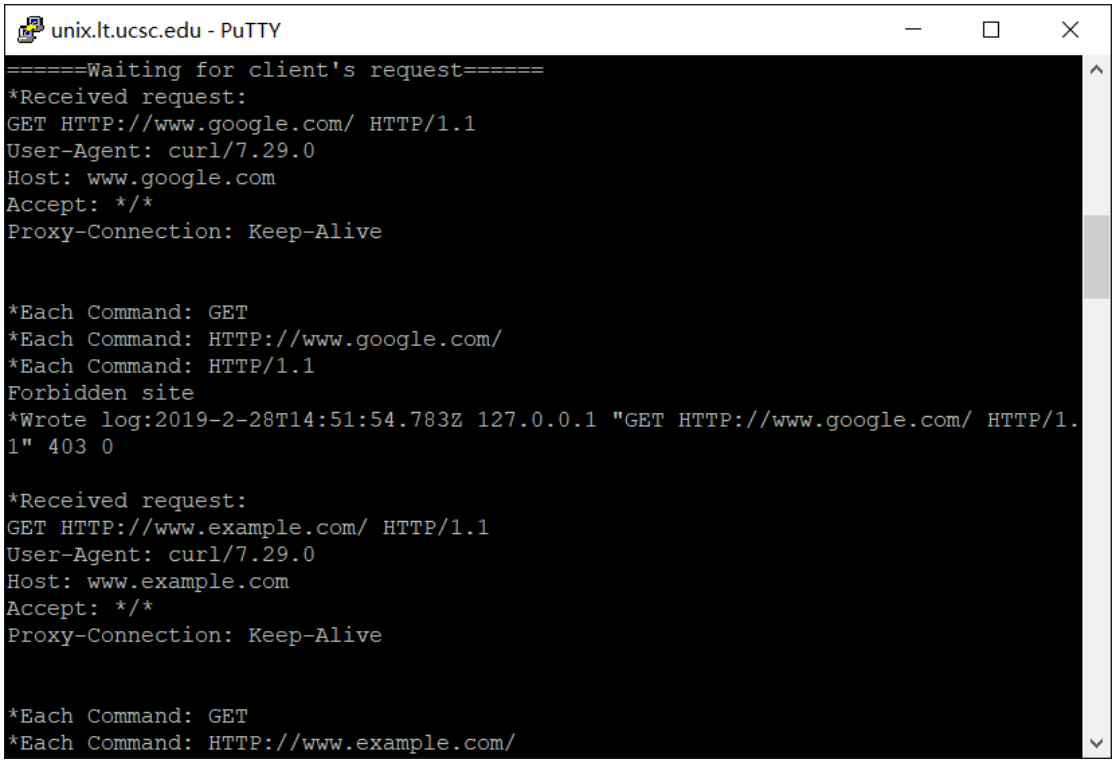
1. First, when start the proxy, create a TCP socket and bind with the local address then start listening the incoming request by listen() and accept().
2. When the proxy gets a request the accept() function will create a new TCP connection with client and return a socket file descriptor. Create a new thread to deal with the transmissions with this socket.
3. Analysis the HTTP request and verified it (If failed write to log and send back error code). Get the IP address by the url information.
4. Check the forbidden-sites-file. If the url is in the forbidden list return the 403 (Forbidden URL) code to client.

5. Create a new TCP connection with the Host by IP address. Add forward information into the request then send to the Host.
6. Receive the data from host and forward to the client.
7. Write the result to the log file. The log file is managed by a mutex lock pthread_mutex_t logLock. Only one thread could modify the file at the same time.

Potential problem:

1. Too much requests might excess the listen capacity. (is set to 10).
2. Keep alive connection might get time out.
3. Threads are not joined.
4. Do not handle the client send multiple request at once.
5. The error message send to client not contains all information in header just error code and a simple html code.
6. Set time out for host and client. The server socket might be closed and create a new one as the client send new request by the keep alive.
7. Didn't check for some kinds of invalid header.
8. Might keep-alive for error stage which is not HTTP/1.1.

Test:



Test with curl:

Log file: