Design Document for P2 HTTP Proxy

Partner A: Jiayi Cao

Partner B: Emiliano Villarreal

1. Socket Management

Describe 1) how many socket objects are used, 2) the variable name for each socket, and 3) the purpose of each socket object.

1) 1 listening socket, up to 50 STREAM socket connections
2) client_socket and proxy_socket are the variable names used for the STREAM and listening sockets respectively
3) proxy_socket listens for connection requests from the browser, client_socket are created whenever a request is received and these sockets will

2. Multi-threading

Overall Design

Describe how multithreading is used in your program. You must include 1) the number of threads you use including the main thread, 2) what each thread is for and doing, 3) any loops you have for each thread.

1. 1 listening socket, up to 50 stream socket connections
2. We made two threads, one for the server socket to communicate with client and one for client to communicate with server
3. While loop in main for multithreading, references proxy function to allow for multiple clients

Justification

Justify why your design is efficient and show evidence it can handle multiple client requests efficiently. Specify your testing scenario (how many requests were made, which websites were being used, etc

We tested by running multiple clients for different websites (cnn.com, neverssl.com, wikipedia.com, twitter.com, and youtube.com). Each of these websites and clients ran in a reasonable amount of time, indicating the efficiency of the design.

3. Streaming

Describe how streaming is implemented in your Proxy and the parameter (i.e. num of bytes) used for streaming. Justify the choice of your parameter.

We stream data from client to proxy then stream that data from proxy to server. The parameter is given as BUFFER_SIZE and is 2048 bytes because it is an adequate amount of bytes (in segments) for a socket to handle and we don't expect to stream more than that amount of bytes at once for this project.

4. Data structures

In the cell below, list any notable data structures you used and justify the use of the data structure. Also specify how you handle synchronization for the data structure if there were any need for synchronization. If none, you can say "None".

Some data structures we used were tuples and strings. Strings for instance were used for messages that were put into logs.

None.

5. How shutdown is handled

Describe how you handled the shutdown gracefully.

Socket will use error handling to print "Keyboard Interrupt: Closing down proxy" whenever the socket stops receiving information and user uses "Ctrl-C" to exit.

6. Error handling

Describe how you handle unexpected errors by specifying 1) what kind of errors you may encounter 2) what you do for each error.

One error we encountered was when data received from the client was empty. We used try except to have all sockets close and exit out of the proxy function whenever there's empty data encountered.

Another error handling was for if the server failed to receive data in the nonconnect function. This was also handled by try/except.

Another error handled was for if the server_socket failed to make a TCP connection with the server. Using try/except, we had the proxy send "200 OK" in the case of successful connection and "502 Bad Gateway" in case of failed connection. In case of failed connection, we closed all sockets and returned out of the function

7. Any libraries used

List any libraries that you used for the implementation and justify the usage.

import socket- create and use sockets

import sys- read in stuff from terminal

import re- used to help modify headers

import os- read in files/path

import threading- multithreading

import errno- reports and retrieves error conditions

import json- for logging

import uuid - to name a file

8. Reflection

What was the most challenging part working on this project? Most fun part?

The most challenging part of the project was error handling. The most fun was when we successfully created the first logs.

If you are to do this all over again, how would you do it differently?

If I could do this differently I would've attended one more session of office hours and save myself some stressful hours.

Reflection on pair programming

Log of the amount of time spent driving and the amount of time spent working individually for each part (e.g., X drives 1 hour; Y drives 45 minutes; X works alone for 1 hour, etc.)

- For Check point: Emiliano drives 1.5 hours, Jiayi drives 1.5 hours, Emiliano works alone 1 hour, Jiayi works alone 1 hour
- For Final submission: Emiliano drives 2 hours, Jiayi drives 2 hours, Emiliano works alone 1 hour, Jiayi works alone 1 hour

What went well/or not-so-well doing pair programming? What was your take away in this process?

Pair programming went well. We both collaborated, put forth ideas, and caught errors that the other couldn't catch. Our takeaway is that pair programming is an efficient way to complete software development tasks

Submission Instruction

Remember to export to pdf and include the pdf under the project root. Make sure to push the pdf and all of your latest code to the team repository.