**Code description and design decisions**

Since this is a multi-threaded program, let’s break down how the main thread / child threads will behave in the program.

Main Thread

The main thread is responsible for:

1. Parsing inputs: port\_number, TELEMETRY ENABLED and blacklist file name.
2. Initializing blacklist into a vector.
3. Listening on the socket with its port specified by user.
4. Allocating connections/forwarding data to child threads using OpenMP and epoll.

Child Threads

Child threads are initialized by the Main thread into a thread pool. They’re split into 1) making new connections, 2) having an existing connection.

When the Main thread checks that it’s a new connection, the child thread will be allocated with the connection and the child thread will call the function handle\_new\_client. Handle\_new\_client will be responsible for

1. Reading data from the client\_sock\_fd passed by Main thread.
2. Parsing the http header read from client\_sock\_fd.
3. Checking if the connection made is in the blacklist.
   1. if it is, prevent the connection from being made.
4. Make a DNS query to map the input URL to its respective IP address and port.
5. Make the connection and send a SSL greeting.
6. Then persist the connection for a while via epoll to ensure all the data has been sent.

Blacklist

The blacklist file is parsed into a vector (aka array) and prior to each connection there’ll be a check to see if the connection’s domain is in the blacklist.

How the check works is it’ll loop through every element in the vector and do a substring search on whether the input’s domain contains any of the blacklist’s domain. if it does the thread will be blocked from making the connection and an error message will be printed out, else it’ll allow the connection to be made.

**While running the telemetry, observe and explain the difference between HTTP/1.0**

**and HTTP/1.1.**

|  |  |
| --- | --- |
| http1\_logs.txt (HTTP/1.0) | http1.1\_logs.txt (HTTP/1.1) |
| Hostname: sg.yahoo.com, Size: 212014 bytes, Time: 0.683 sec | Hostname: sg.yahoo.com, Size: 302062 bytes, Time: 55.480 sec |
| Hostname: sg.yahoo.com, Size: 7638 bytes, Time: 0.045 sec |
| Hostname: sg.yahoo.com, Size: 89044 bytes, Time: 0.127 sec |

Table 1: sg.yahoo.com HTTP/1.0 VS HTTP/1.1

There are several differences between HTTP/1.0 and HTTP/1.1 as observed in Table 1. Namely:

1. There are 3 connections made when using HTTP/1.0 while there’s only 1 when using HTTP/1.1.
2. The time of each connection is short when using HTTP/1.0 (< 1 sec) while the duration of the connection for HTTP/1.1 is significantly longer (55.48sec).

Reason for both observation:

HTTP/1.1 maintains persistence connection by keeping the connection alive, and hence bundles requests/responses under the same connection while its within TTL (time to live) or until the connection is terminated.

Whereas HTTP/1.0 does not maintain persistent connection and hence it’ll terminate connections after each requests/response. This also means that HTTP/1.0 requires a brand-new connection to be made for every request/response.