README.md

Programming Assignment 3 - HTTP Web Proxy

Zachary Atkins

CSCI 5273

Build Instructions

Navigate to the root directory and run `make`. The webserver is written in C++11, so it should work with any C++ compiler.

Run instructions

Run the HTTP proxy with the command:

```
./bin/webproxy {PORT_NUMBER} {CACHE_TIMEOUT_SECONDS}
```

Functionality

The proxy opens a listening socket on the user-provided port. Then, it accepts each new incoming TCP connection, creates a `ProxyConnection` functor with the accepted socket, and spawns a new thread to run the created functor.

Within the main event loop, each `ProxyConnection` waits for an HTTP request from the client, parses the message into a HTTP request. Then depending on the request type and requested source, the proxy forwards the request to the destination server.

The proxy supports two request types: `GET` and `CONNECT`.

GET Request

1. The proxy attempts to resolve the IP address for the host. The IP address is added to the IP cache. If the host or its IP address are in the blacklist, the proxy sends a `403 Forbidden` response to the client.
2. The proxy establishes a connection with the server using the resolved IP address. If the connection attempt errors, the proxy sends a `404 Not Found` response.
3. The request is forwarded to the server.
4. The proxy waits for a response from the server. If the request times out, the proxy sends a `504 Gateway Timeout` response to the client.
5. The response from the server is parsed and then forwarded to the client. If the response has a code of `200`, then the response is cached.
6. If the response has a content type of `text/html`, then the webpage is parsed for any links. The links are then prefetched in a separate thread using the `Prefetcher` class.

CONNECT Request

The `CONNECT` requests are similar, but even simpler. Steps 1 and 2 are the same, and if the connection attempt is successful, the proxy sends a `200 OK` response to the client. Then, all data is forwarded directly between the two sockets until one is closed. `CONNECT` requests allow for encrypted communication, such as HTTPS.

Blacklist

The blacklist is loaded from `blacklist.txt`. Each line of the file should contain one host or IP address.

Caching

Caching is performed by the `Cache` object, which wraps an `std::unordered_map` in a thread-safe fashion using C++ `std::mutex` and `std::scoped_lock` objects. The entire program has a single copy of the IP and page caches, which each thread is given access to via a C++ `std::shared_ptr`, which ensures reference counting of the shared memory.

Prefetching

The `Prefetcher` class is used to parse HTML pages and pre-cache links in a separate thread. A new thread is created to parse the page, then a separate C++ `std::task` is created to send a `GET` request for each link. Links beginning with `https://` are ignored, since they would require a `CONNECT` request and cannot be cached.