CS 5450: Computer Network Fundamentals
Homework 2: Web Proxy

Jiaqi Tong (jt783)
Fei Li (fl392)

October 29, 2018

1 Challenges in the Web Proxy

1.1 Implementation Details

In this homework, we implemented a proxy server that provides intermediary service for requests from clients getting resources from their target server. During this procedure, we have a deep understanding of the mechanism of proxy. What's more, this lab also involved with protocols (especially HTTP), concurrency, and cache. Here is some challenges we met when implemented the proxy server or debugged.

- **HTTP request handling**
  This is the basic part of a web proxy. We simply follow the instruction to implement it. We leverage the wrappers and helper functions in csapp.h to deal with socket operations. To be more specific, the entire workflow of this part is as follows. Our proxy starts listening to a given port. When a connection from the client comes, the proxy accepts it and reads the request from the connected socket. Then the proxy uses a URI parser to parse the URI in the request to get host and query. The proxy uses these information to construct a new request and send it to the requested host, with another socket. After sending, the proxy starts to read response from the socket connected to the server. And return the entire response back to the client through client's socket. Finally, the proxy closes both sockets.

  A tricky part is that there is a "\r\n" at the end of the given user agent string. At first we did not notice that. We add an extra "\r\n" at the end of it. Then our proxy did not work for real websites, because server will recognize the end of our request early. (But it works for the tiny server, strangely). After debugging, we finally find this bug and remove the extra "\r\n". Then our proxy works for real websites.

- **Concurrency**
  For the concurrency part, we simply use the Pthread wrapper in csapp.h

to create one thread for each request, and detach it after executing it.

In addition, we used semaphores to implement multi-threading read or write requirement for caching. Refer to Readers Writers Problem[2], we implement the first modal, which may starve writers in the execution queue. To solve this problem, a solution that the operation of obtaining a lock on the shared data will always terminate in a bounded amount of time can by applied in this situation in the future. Basically, we will keep a read_count variable to record the number of readers, in order to get rid of race conditions. And the read semaphore is used to make the read_count thread-safe. If one thread holds read semaphore, the write semaphore is also blocked once there exist readers.

- **Cache**
  According to the instruction, we should combine the LFU and LRU eviction policy to develop the cache. We implemented the cache based on doubly linked list. The core part of this mechanism is updating cache after accessing a URI according to the description of LFU and LRU.

  In detail, LRU[1] (Least Recently Used) should invalidate the least recently used item before inserting a new item when the cache reached its capacity. Thus we contains a counter in the Node class which indicates the accessing times from beginning. Once the hit times for a LRU object exceed the least one in LFU, this object should displace the last one in LFU. Then we will insert new object to the head of the linked list and remove old object from the tail of the linked list. Doubly linked list can finish the insert and remove in O(1) time.

  LFU[1] (Least Frequently Used) should invalidate the least frequently used item before inserting a new item when the cache reaches its capacity. Then this new or accessed object will move from the tail to the head of the linked list just like bubble sort according to its count. Finally, we will evict the object at the end of the list and maintain its maximum cache size.

1.2 Optimization

After finish this lab, we think we can improve performance by the following three aspects:

- Add cache size
  Theoretically, if we have more cache size, we can avoid more redirect to the same server and save more time. But larger cache size also means long cache access time and low hit rate in this case, and is definitely a waste of storage. We we can balance the cache size and improve hit rate by applying better eviction strategy.

- Optimize cache structure

We choose doubly linked list to implement caching in this lab, but it is a waste of accessing time because we find a cache with linear time which is based on the numbers of cached objects. However, in real world, it's more efficient to use Hash Table or Balanced Search Tree. For example, we can implement LRU using a high-performance datatype in Python — OrderedDict, which can search items in constant time and order all the key-value pairs as well. We can also use specific BST like Red–black tree or Splay tree to optimize our cache hit mechanism.

- Starvation of writters
  Our current multithreading implementation may cause starvation of writters, so as previous description, we can solve this problem by limiting the operation time of a holding lock.

2 Security and privacy challenges

2.1 Problems

A proxy can fall into one of four categories[3]:

- Transparent proxy. It tells websites that it is a proxy server and it will pass along your IP address anyway.
- Anonymous proxy. It will identify itself as a proxy, but it won't pass your IP address to the website.
- Distorting proxy. It passes along an incorrect IP address for you, while identifying itself as a proxy.
- High Anonymity proxy. The proxy and your IP address stay a secret. The website just sees a random IP address connecting to it...that isn't yours.

Our proxy is transparent proxy, which is the least secure and does not protect personal privacy at all. When using proxy servers, your request package sent to the server is plain text which is not encrypted. A spy proxy agent may record all the data passing through especially the username, password, and other sensitive personal privacy. It means that anyone who might intercept your data in transit could easily see sensitive information, like the usernames and passwords of your online accounts, bank and credit card details, IP address, etc. So if possible, When you need to transfer the username and password or other important information through the proxy, do choose a reliable proxy that use SSL, TLS to encrypt first no matter if the destination website is secure or not.

Additionally, the anonymous function of proxy can help hacker hide their real IP to destroy other devices or engage in illegal activities, which would significantly increase the difficulty of investigation for police. In such way, the security challenges are arisen by using proxy servers.

2.2 Solutions

- Using HTTPS instead of HTTP in our proxy server.
- Hiding the IP address by whatever method, replace with an incorrect IP or a random IP.
- Applying advanced encryption algorithm and blocks access to certain websites.

2.3 Tor and other thinking

Tor[4] is free software for enabling anonymous communication. Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays to conceal a user's location and usage from anyone conducting network surveillance or traffic analysis. Using Tor makes it more difficult to trace Internet activity to the user. Tor's intended use is to protect the personal privacy of its users, as well as their freedom and ability to conduct confidential communication by keeping their Internet activities from being monitored.

Tor's benefit is similar to proxy generally. So our proxy is used to preserve privacy mainly for hiding real IP address. But we can add a proxy network like Tor to better conceal user's information.

References

[1] Cache replacement policies

[2] Readers–writers problem

[3] Hide Your Real IP Address Behind a Proxy

[4] Tor