Computer Networks Project 2

Proxy

2014037901 나윤환

Part1. 프록시 기본 구현

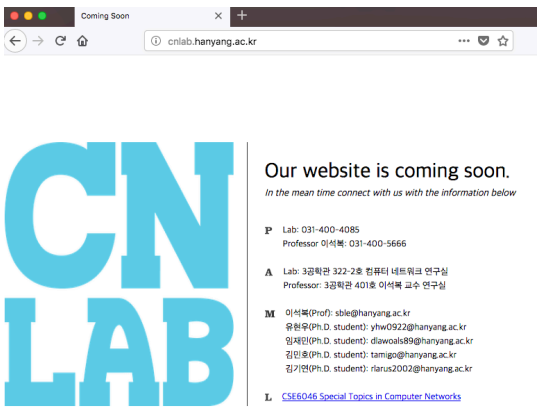
```
Origin Request message
GET http://cnlab.hanyang.ac.kr/ HTTP/1.1
Host: cnlab.hanyang.ac.kr
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:60.0) Gecko/2010010
1 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

Request Method: GET
URL: http://cnlab.hanyang.ac.kr/
Host: cnlab.hanyang.ac.kr
```

Socket 을 열어서 기본적으로 client 에서 오는 http request 메시지를 출력하고, HTTP Request header 를 파싱해서 Method 와 URL 그리고 Host 값을 추출합니다.

```
Request Header
GET / HTTP/1.0
Host: cnlab.hanyang.ac.kr
Connection: close
```

파싱한 정보를 기반으로하여 Custom HTTP Request Header 를 만듭니다. 이때 HTTP version 은 1.0 이고, connection 은 close 로 설정해줍니다. 해당 Custom HTTP Request 로 가져온 값을

```
while ((n = read(host_fd, read_buff, READ_BUFF_SIZE)) > 0)
{
  data_size += write(client_fd, read_buff, n);
  if (data_size <= MAX_OBJECT_SIZE)
    strcat(object, read_buff);
  memset(read_buff, '\0', READ_BUFF_SIZE);
}
```



Read Buffer 로 읽어들이고, 바로 client socket 을 통해 write 해서 받아온 HTTP response 를 전달하여 웹 브라우저에 화면이 정상적으로 출력되로록 합니다.

Part2. LRU Cache 구현

LRU Cache 는 기본적으로 Double Linked List 를 이용하여 구현하였습니다. Header 가 가장 오래된 값이고, tail 로 갈수록 가장 최근 값이며, Host 에 HTTP Request 를 날리기이전에 HTTP Request 헤더에서 파싱된 URL 을 기반으로 LRU Linked List 에서 url 을 갖고 있는지를 검사합니다. 만약에 url 이 없다면, Host 에 HTTP Request 를 날려서 값을 받아오고, LRU Cache 가 갖고있다면, Cache 에서 값을 가져옵니다.

```
-------- Cached All Pages List --------
Cache Size: 111498 bytes
no. 1:  http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js 57709 by
tes
no. 2:  http://cnlab.hanyang.ac.kr/static/images/CNLAB_basic_compressed.png    1
354 bytes
no. 3:  http://cnlab.hanyang.ac.kr/     2220 bytes
no. 4:  http://cnlab.hanyang.ac.kr/static/tools/style.css       4099 bytes
no. 5:  http://cnlab.hanyang.ac.kr/static/tools/960.css 5582 bytes
no. 6:  http://cnlab.hanyang.ac.kr/static/js/Clarendon_LT_Std_700.font.js      2
1984 bytes
no. 7:  http://cnlab.hanyang.ac.kr/static/js/cufon-yui.js       18550 bytes
--------------------------------------------------
```

LRU Cache 에서 HIT 가 된다면 해당 URL 을 다시 가장 최근으로 옮겨서 오래된 데이터가 지워지도록 합니다. (no.1 의 jquery 가 HIT 되기 이전)

```
-------- HIT in Cache --------
Cached URL: http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js


-------- Cached All Pages List --------
Cache Size: 111498 bytes
no. 1:  http://cnlab.hanyang.ac.kr/static/images/CNLAB_basic_compressed.png    1
354 bytes
no. 2:  http://cnlab.hanyang.ac.kr/     2220 bytes
no. 3:  http://cnlab.hanyang.ac.kr/static/tools/style.css       4099 bytes
no. 4:  http://cnlab.hanyang.ac.kr/static/tools/960.css 5582 bytes
no. 5:  http://cnlab.hanyang.ac.kr/static/js/Clarendon_LT_Std_700.font.js      2
1984 bytes
no. 6:  http://cnlab.hanyang.ac.kr/static/js/cufon-yui.js       18550 bytes
no. 7:  http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js 57709 by
tes
--------------------------------------------------
```

(no.1 의 jquery 가 HIT 후 no.7 으로 가장 최신으로 업데이트 된 것을 확인할 수 있다.)

```
LOG:: Fri Jun  8 16:20:37 2018 172.217.161.42 http://ajax.googleapis.com/ajax/li
bs/jquery/1.3.2/jquery.min.js 57709
```

LOG 도 구현하였습니다.

Part3. Thread 사용

```
socklen_t client_len = sizeof(client_addr);
client_fd = accept(proxy_fd, (struct sockaddr *)&client_addr, &client_len);
if (client_fd < 0)
  error("Failed to accept");

getpeername(client_fd, (struct sockaddr *)&client_addr, &client_len);
thread_fd thread_socket;
thread_socket.client_fd = client_fd;
thread_socket.thread_no = thread_cnt;

if (pthread_create(&thread[thread_cnt], NULL, &proxy, (void *)&thread_socket) != 0)
  error("Failed to create Thread");
pthread_detach(thread[thread_cnt]);
thread_cnt++;
```

```
// Read HTTP Request from client
memset(proxy_buff, '\0', READ_BUFF_SIZE);
pthread_mutex_lock(&mutex_cache);
if (read(client_fd, proxy_buff, READ_BUFF_SIZE) < 0)
{
  close(client_fd);
  error("Failed to read");
}
pthread_mutex_unlock(&mutex_cache);
```

Thread 와 Mutex 를 이용하여 공유된 자원에 접근할 때 동시에 접근하는 것을 방지하였습니다.