哈爾濱工業大學
Harbin Institute of Technology

# 计算机网络
# 课程实验报告

| | | | | | |
|---|---|---|---|---|---|
| 实验名称 | HTTP 代理服务器的设计与实现 | | | | |
| 姓名 | 石翔宇 | | 院系 | 计算学部 | |
| 班级 | 1903103 | | 学号 | 1190200523 | |
| 任课教师 | 刘亚维 | | 指导教师 | 刘亚维 | |
| 实验地点 | 格物 207 | | 实验时间 | 2021.10.31 | |
| 实验课表现 | 出勤、表现得分(10) | | 实验报告得分(40) | 实验总分 | |
| | 操作结果得分(50) | | | | |
| 教师评语 | | | | | |
| | | | | | |

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

实验目的：

熟悉并掌握 Socket 网络编程的过程与技术；深入理解 HTTP 协议，掌握 HTTP 代理服务器的基本工作原理；掌握 HTTP 代理服务器设计与编程实现的基本技能。

实验内容：

(1) 设计并实现一个基本 HTTP 代理服务器。要求在指定端口（例如8080）接收来自客户的 HTTP 请求并且根据其中的 URL 地址访问该地址所指向的 HTTP 服务器（原服务器），接收 HTTP 服务器的响应报文，并将响应报文转发给对应的客户进行浏览。

(2) 设计并实现一个支持 Cache 功能的 HTTP 代理服务器。要求能缓存原服务器响应的对象，并能够通过修改请求报文（添加 if-modified-since头行），向原服务器确认缓存对象是否是最新版本。（选作内容，加分项目，可以当堂完成或课下完成）

(3) 扩展 HTTP 代理服务器，支持如下功能：（选作内容，加分项目，可以当堂完成或课下完成）

a) 网站过滤：允许/不允许访问某些网站；

b) 用户过滤：支持/不支持某些用户访问外部网站；

c) 网站引导：将用户对某个网站的访问引导至一个模拟网站（钓鱼）。

实验过程：

**HTTP代理服务器的实现流程：**

1. 初始化服务器端Socket并开启监听；
2. 等待客户端链接，开启新线程并创建客户端的Socket；
3. 用户过滤，若用户IP在黑名单内，则禁止访问；
4. 解析客户的HTTP请求报文获得目的Host的URL，IP地址；
5. 网站过滤，若URL在过滤列表中，则返回错误；
6. 网站引导，若URL在引导列表中，将URL和IP地址换成引导目的地址；
7. 连接目的Host获得Host端的Socket；
8. 在缓存中查找目的Host：
   a) 若没有找到，则直接将客户的HTTP请求报文发送给目的Host，将目的Host返回的报文发送给客户。
   b) 否则，在客户的HTTP请求报文中加入if-modified-since头行，发送给目的Host，若返回304则将缓存中的内容发送给客户；若返回200则将返回的报文发送给客户。
9. 按照目的Host返回的报文更新缓存。

**关键功能实现：**

解析客户的HTTP请求报文：

首先从第一行中找到HTTP请求类型，目前只支持GET和POST，同时提取请求的URL。再利用strtok_s函数按照\r\n将报文中的不同行分开，分别浏览每一行，找到有Host的行，将Host提取出来。

```
if (p[0] == 'G')
{
    memcpy(httpHeader->method, "GET", 3);
    memcpy(httpHeader->url, &p[4], strlen(p) - 13);
}
else if (p[0] == 'P')
{
    memcpy(httpHeader->method, "POST", 4);
    memcpy(httpHeader->url, &p[5], strlen(p) - 14);
}
printf("%s\n", httpHeader->url);
p = strtok_s(NULL, delim, &ptr);
while (p)
{
    switch (p[0])
    {
    case 'H': //Host
        memcpy(httpHeader->host, &p[6], strlen(p) - 6);
        break;
    default:
        break;
    }
    p = strtok_s(NULL, delim, &ptr);
}
```

**用户过滤：**

disabledUser数组中存放着被禁止的用户IP。在获得用户的socket后可以得知用户的IP地址，将其与数组中的IP进行匹配，若匹配成功则返回USER_BLOCKED表示该用户已被禁止访问。

```
ERROR_CODE UserFilter(in_addr sin_addr)
{
    for (int i = 0; i < DISABLED_MAXSIZE; i++)
    {
        if (disabledUser[i] == NULL)
            continue;
        if (strcmp(disabledUser[i], inet_ntoa(sin_addr)) == 0)
            return USER_BLOCKED;
    }
    return REQUEST_SUCCEEDED;
}
```

**网站过滤：**

disabledHost数组中存放着被禁止的网站Host。在解析客户的HTTP请求报文后，可以获得该用户想要访问的网站Host，将其与数组中的Host进行匹配，若匹配成功则返回HOST_BLOCKED表示该网站已被禁止访问。

```
for (int i = 0; i < DISABLED_MAXSIZE; i++)
{
    if (disabledHost[i] == NULL)
        continue;
    if (strcmp(disabledHost[i], host) == 0)
        return HOST_BLOCKED;
}
```

**网站引导：**

induceSites数组中存放着被引导的网站Host，targetSites数组中存放着引导目标网站Host，两者一一对应。在解析客户的HTTP请求报文后，可以获得该用户想要访问的网站Host，

首先通过网站过滤功能，若不是被过滤的网站，则进入网站引导。将其与induceSites数组中的内容进行匹配，若匹配成功，则将其更改为induceSites数组中对应的Host。

```
for (int i = 0; i < DISABLED_MAXSIZE; i++)
{
    if (induceSites[i] == NULL)
        continue;
    if (strcmp(induceSites[i], host) == 0)
    {
        strcpy(host, targetSites[i]);
        return CHANGE_SUCCEEDED;
    }
}
```

**缓存：**

我们首先定义关于网页缓存的类，内容包括HTTP头，网页缓存，最后更新时间，网页缓存长度。

```
class WebCache
{
public:
    static WebCache *FindCache(HttpHeaderP htp);
    HttpHeaderP htP;
    char buffer[BUFFER_MAXSIZE];
    char date[DATE_LENGTH];
    int buffer_length;
    WebCache()
    {
        ZeroMemory(this->buffer, BUFFER_MAXSIZE);
        ZeroMemory(this->date, DATE_LENGTH);
        this->htP = new HttpHeader();
    }
};
```

缓存实现的第一步需要在内存中查找缓存。若当前请求的HTTP头中的方法、URL与Host都与某缓存相匹配的话，则缓存命中，直接返回该缓存。

```
WebCacheP WebCache::FindCache(HttpHeaderP htp)
{
    for (int i = 0; i < CACHE_MAXSIZE; i++)
    {
        if (Cache_storage[i] == NULL)
            continue;
        HttpHeaderP htp1 = Cache_storage[i]->htP, htp2 = htp;
        if (!strcmp(htp1->method, htp2->method) && !strcmp(htp1->url, htp2->url) && !strcmp(htp1->host, htp2->host))
            return Cache_storage[i];
    }
    return NULL;
}
```

接下来需要向服务器发送带If-Modified-Since的报文。我们实现的方法是在用户发来的请求报文中查找Host字段，在该字段前面插入If-Modified-Since字段。

```
while (true)
{
    if (Buffer[p] == 'H')
    {
        char header[10];
        ZeroMemory(header, sizeof(header));
        memcpy(header, (Buffer + p), 4);
        if (!(strcmp(header, "Host")))
        {
            char const *ife = "If-Modified-Since: ";
            memcpy(newBuffer, Buffer, p);
            // print_debug_buffer(newBuffer);
            strcat(newBuffer, ife);
            strcat(newBuffer, cacheP->date);
            strcat(newBuffer, "\r\n");
            strcat(newBuffer, Buffer + p);
            length = length + strlen(ife) + strlen(cacheP->date) + 2;
            break;
        }
    }
    p++;
}
```

若服务器返回304则说明网站内容未更新，直接将缓存中内容返回给用户；否则将服务器返回的报文返回给用户。

最后要做的就是更新缓存内容。首先将网站最后更新时间解析出来，再将最后更新时间和内容更新给缓存。若缓存中本来就没有该网站的内容，则还需要新建缓存。

```
if (isUpdate)
{
    if (cacheP == NULL)
    {
        cacheP = Cache_storage[Cache_storage_length % CACHE_MAXSIZE] = new WebCache();
        Cache_storage_length++;
        memcpy(cacheP->htP->method, httpHeader->method, strlen(httpHeader->method));
        memcpy(cacheP->htP->url, httpHeader->url, strlen(httpHeader->url));
        memcpy(cacheP->htP->host, httpHeader->host, strlen(httpHeader->host));
    }
    memcpy(cacheP->buffer, Buffer, recvLength);
    memcpy(cacheP->date, date, strlen(date));
}
```

实验结果：

**基础功能：**

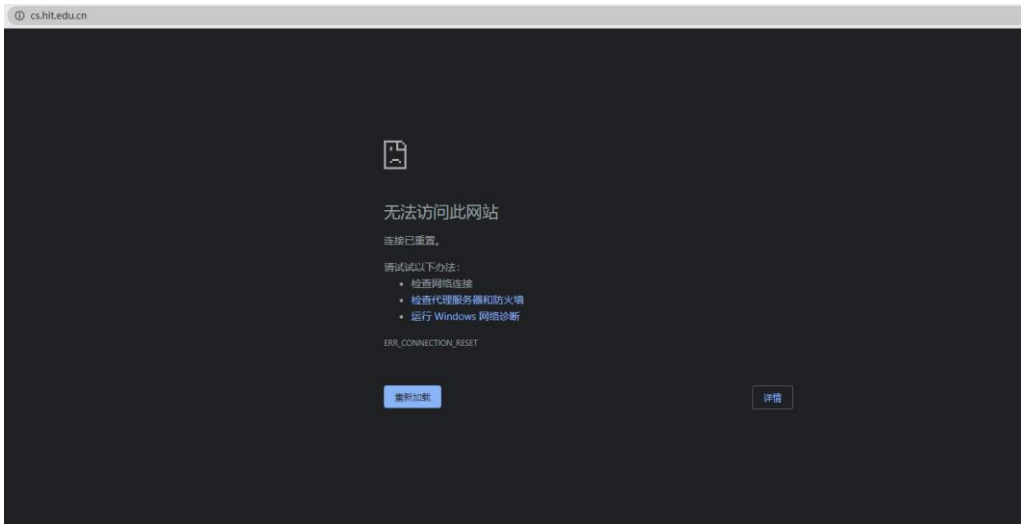
首先是代理服务器最基础的功能实现。我们访问http://cs.hit.edu.cn/结果如下：


代理服务器端显示日志如下：

```
Waiting for connection...
Recieve a connection from 127.0.0.1:40203
GET http://cs.hit.edu.cn/ HTTP/1.1
http://cs.hit.edu.cn/
Successfully connected host cs.hit.edu.cn 0
Cache storage length is 0
Closing socket...
Waiting for connection...
```

**用户过滤：**

我们将本地IP地址127.0.0.1限制访问，结果如下：

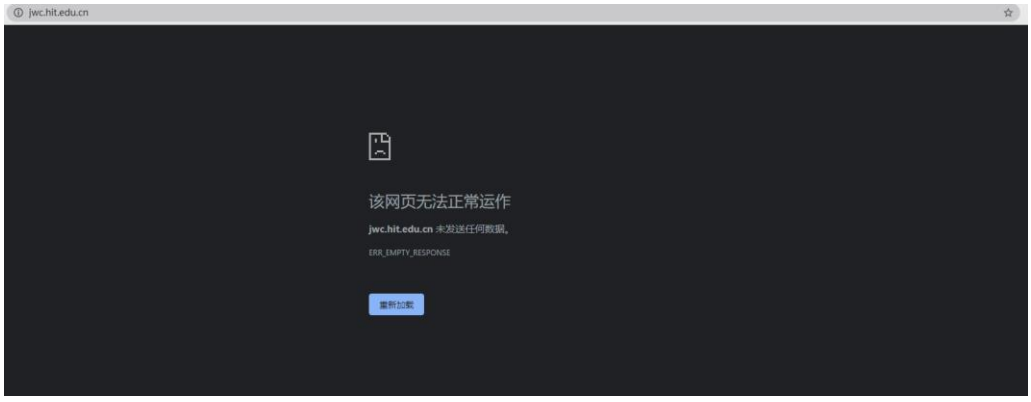



可以看到当前用户已经无法访问任何网站。代理服务器端显示日志如下：

```
Waiting for connection...
Recieve a connection from 127.0.0.1:4365
## The user is not allowed to connect
Closing socket...
```

**网站过滤：**

我们将网站jwc.hit.edu.cn限制访问，结果如下：




可以看到已经无法访问jwc.hit.edu.cn。代理服务器端显示日志如下：

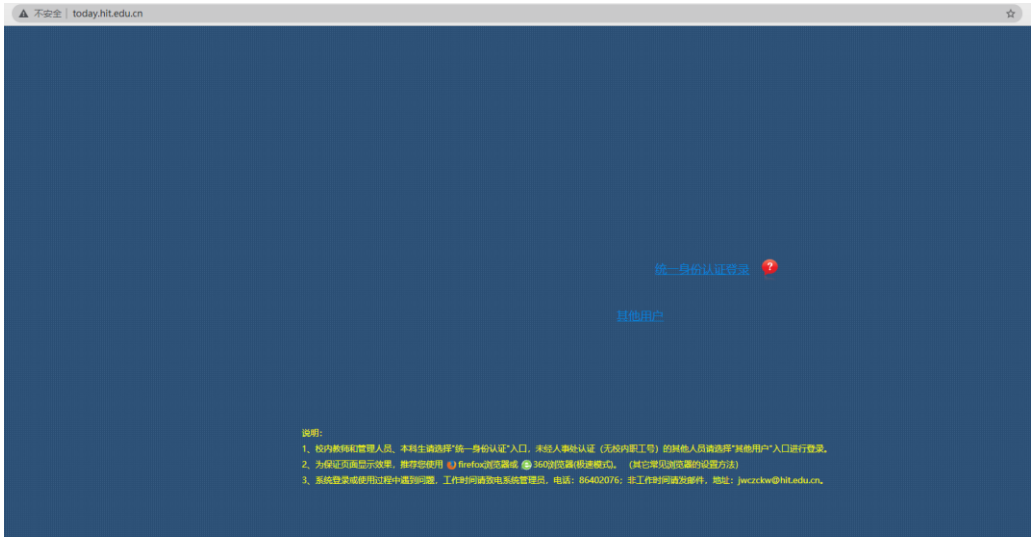
```
Waiting for connection...
Recieve a connection from 127.0.0.1:17934
GET http://jwc.hit.edu.cn/ HTTP/1.1
http://jwc.hit.edu.cn/
## An error occurred: 5
Closing socket...
```

从日志可以看到发生了错误，编号为5，这正是网站被过滤时提示的错误编号。

```
#define HOST_BLOCKED 5
```

**网站引导：**

我们将对网站today.hit.edu.cn的所有访问都转到jwts.hit.edu.cn中，结果如下：




可以看到地址栏中写的是today.hit.edu.cn但显示的却是jwts.hit.edu.cn的内容。代理服务器端显示日志如下：

```
Waiting for connection...
Recieve a connection from 127.0.0.1:45855
GET http://today.hit.edu.cn/ HTTP/1.1
http://today.hit.edu.cn/
Successfully redirected host to jwts.hit.edu.cn
Successfully connected host jwts.hit.edu.cn 0
Cache storage length is 0
Closing socket...
```

**缓存：**

我们将两次访问cs.hit.edu.cn来测试缓存功能，代理服务器端显示日志如下：

这是第一次访问cs.hit.edu.cn中某个jpg图片资源时的日志：

```
Waiting for connection...
Recieve a connection from 127.0.0.1:44044
GET http://cs.hit.edu.cn/_upload/article/images/aa/d2/e115aa08451d89d465c768f9d899/f8d195a9
-db5a-4cf2-a3be-311e7ad2c4d1.jpg HTTP/1.1
http://cs.hit.edu.cn/_upload/article/images/aa/d2/e115aa08451d89d465c768f9d899/f8d195a9-db5
a-4cf2-a3be-311e7ad2c4d1.jpg
Successfully connected host cs.hit.edu.cn 3
Cache storage length is 3
Closing socket...
```

这是第二次访问的日志：

```
Waiting for connection...
Recieve a connection from 127.0.0.1:56844
GET http://cs.hit.edu.cn/_upload/article/images/aa/d2/e115aa08451d89d465c768f9d899/f8d195a9
-db5a-4cf2-a3be-311e7ad2c4d1.jpg HTTP/1.1
http://cs.hit.edu.cn/_upload/article/images/aa/d2/e115aa08451d89d465c768f9d899/f8d195a9-db5
a-4cf2-a3be-311e7ad2c4d1.jpg
Successfully connected host cs.hit.edu.cn 9
Cache storage length is 9
Using cache
Closing socket...
```

可以看到缓存的长度增加了（还有别的资源也被缓存下来），并且在第二次访问时代理服务器端给出

了日志Using cache，这表明代理服务器判断缓存无更新，将缓存里的内容返回给了用户。

问题讨论：

在实现网站诱导时，发现不更改HTTP报文中的Host和URL字段也能返回目标诱导网站的页面，但只能是主页面，其下的URL是无法正常访问的。

心得体会：

通过此次实验，我更加深刻地理解HTTP协议的原理以及工作方式，掌握了Socket网络编程技术，了解了HTTP代理服务器的基本原理，实现了一个小的HTTP代理服务器，激发了我对计算机网络的兴趣。