哈尔滨工业大学

<<计算机网络>>

实验报告

(2019 年度秋季学期)

姓　　名	张景润
学　　号	**1172510217**
学　　院	计算机科学与技术学院
教　　师	刘亚维
实验名称	**HTTP** 代理服务器的设计与实现

目录

一、 实验目的 1
二、 实验环境 1
三、 实验内容 1
四、 实验原理 1
1. 浏览器使用代理 1
2. 设计并实现一个基本HTTP代理服务器 2
3. 实现代理服务器缓存功能（选作内容，加分项） 4
4. 实现用户、网页过滤、钓鱼（选作内容，加分项） 6
五、 实验过程 7
1. 程序总流程图：包括附加功能 8
2. 设计并实现一个基本HTTP代理服务器 8
3. 实现代理服务器缓存功能（选作内容，加分项） 10
4. 实现用户、网页过滤、钓鱼（选作内容，加分项） 11
六、 实验结果 12
1. 设计并实现一个基本HTTP代理服务器 12
2. 实现代理服务器缓存功能（选作内容，加分项） 14
3. 实现用户、网页过滤、钓鱼（选作内容，加分项） 16
七、 实验心得 19
1. 知识收获 19
2. 代码体会 19

一、实验目的

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

二、实验环境

(1) 接入 Internet 的实验主机；
(2) Windows xp 或 Windows 7/8 或 Windows 10；
(3) 开发语言：C/C++/Java 等

三、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）。

四、实验原理

1. 浏览器使用代理

为了使电脑访问网址时通过代理服务器，我进行了如截图的操作。操作过程是：使用快捷键 Windows + I 打开 Windows 10 设置中心、打开网络和 Internet、

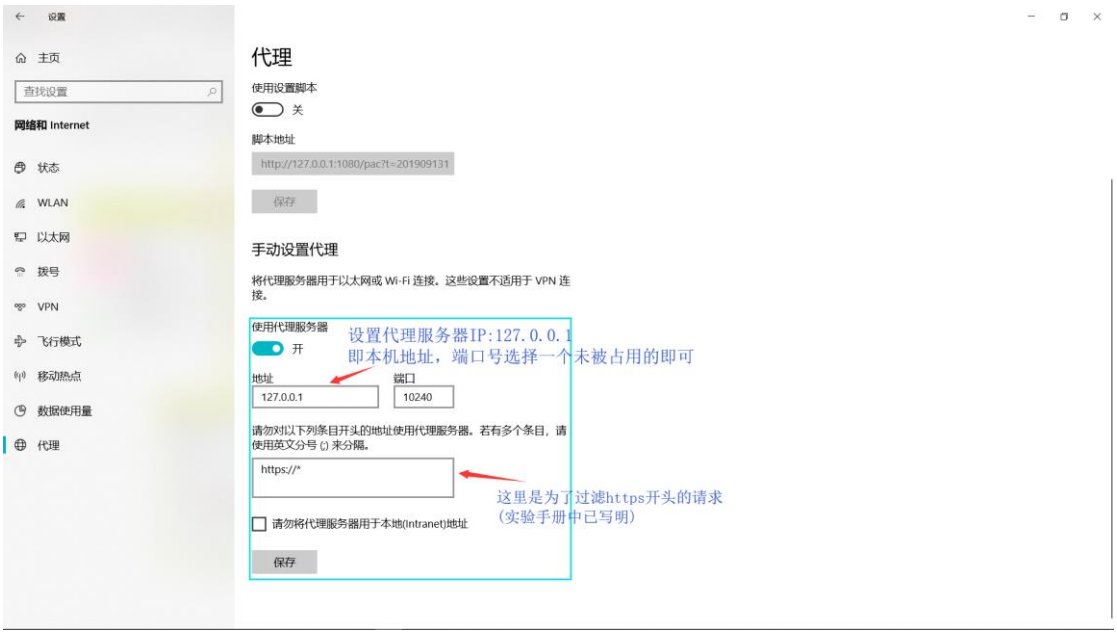
转到代理、使用代理服务器、输入地址为 127.0.0.1，端口号为 10240、过滤来自 https 开头的请求、保存。

几点说明

(1) 设置的目的：确保本机的 HTTP 服务请求都是通过设置的代理服务器进行相应的处理的。同时该设置不依赖于具体的浏览器，设置完成后，通过不同的浏览器访问不同的 URL，都会通过该代理服务器的实现；

(2) 设置代理服务器的地址为 127.0.0.1：127.X.X.X 是本地环回地址，用于本地软件环回测试，因此这里设置 127.0.0.1 即可表示本机将所有的请求发往 127.0.0.1 代理服务器即本机；

(3) 设置监听端口号为 10240：可以设置为公认端口号之外的任何未被使用的端口号，用于不停监听来自本机的网络请求，这里设置为 10240 是为了和实验参考书保持一致；



2. 设计并实现一个基本 HTTP 代理服务器

(1) 实现一个多用户代理服务器，即实现为一个多线程的并发服务器。首先代理服务器创建 HTTP 代理服务的 TCP 主套接字，通过该主套接字不断监听等待客户端的连接请求。当客户端连接之后，创建一个子线程，由子线程执行上述一对一的代理过程，服务结束之后子线程终止。与此同时，主线程不断接受下一个客户的代理服务。

(2) 代理服务器从功能上说就是一个和客户端建立连接的服务器以及和客户请求的服务器建立连接的客户端。对于客户端来说，它的功能是接收来自客户的 HTTP 请求，并对该报文进行相应的处理（解析头部，头部信息修改），并将其

转发给相应的服务器端；同时接收来自服务器端的响应报文，并对其进行相应的处理（解析头部，缓存响应），并将其转发给客户端。

使用代理的B/S

(3) 代理服务器作为一个并发的面向连接的服务器的基本流程如下：主线程创建一个主套接字，并绑定熟知端口号，不断监听来自客户端（本机）的请求；接受来自客户端的请求并为其建立一个新的线程，在该线程中，建立一个与之通信的套接字，用于接收客户端的请求以及转发来自服务器端的响应报文到客户端。

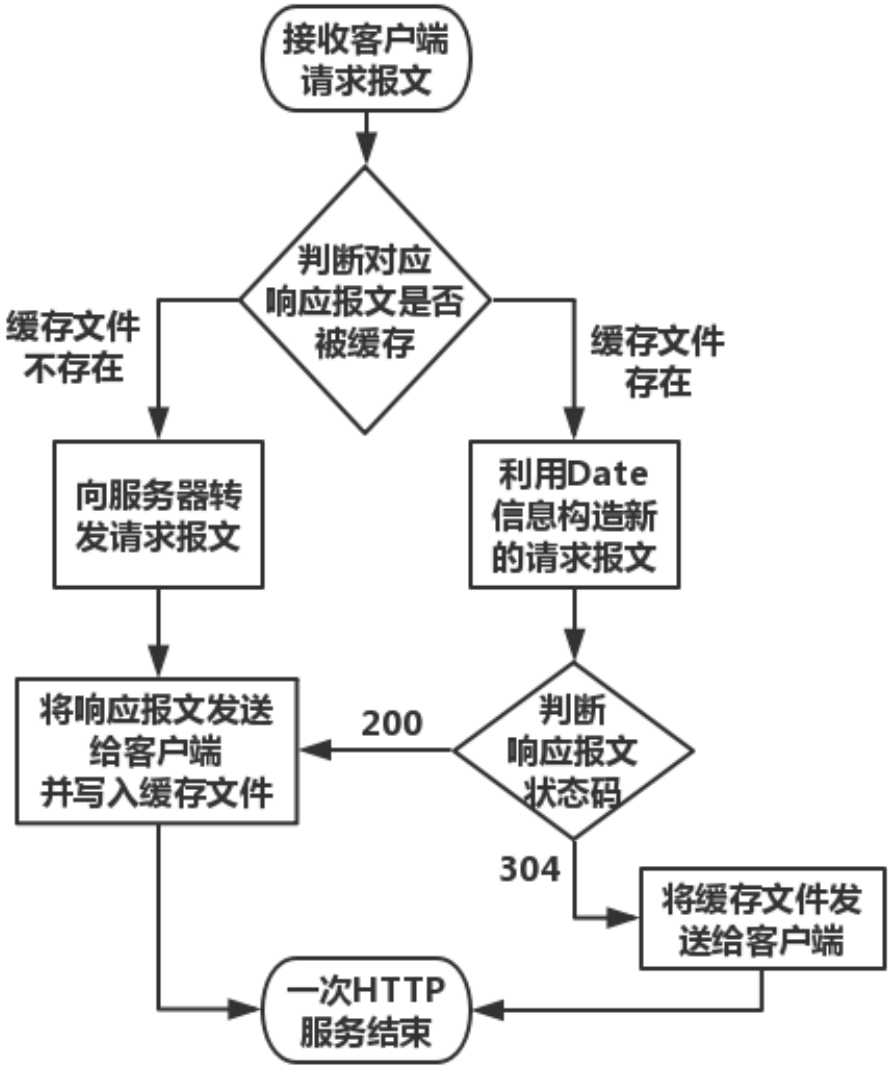
主线程1: 创建（主）套接字，并绑定熟知端口号；
主线程2: 设置（主）套接字为被动监听模式，准备用于服务器；
主线程3: 反复调用***accept()***函数接收下一个连接请求（通过主套接字），并创建一个新的子线程处理该客户响应；
子线程1: 接收一个客户的服务请求（通过新创建的套接字）；
子线程2: 遵循应用层协议与特定客户进行交互；
子线程3: 关闭/释放连接并退出（线程终止）.

(4) 代理服务器作为一个面向连接的客户端的基本流程如下：通过上一步建立的新的线程与服务器端进行通信。代理服务器在这时，作为一个客户机与目标服务器建立连接请求，并转发来自客户端的 HTTP 请求，同时接收来自服务器的响应报文，将其通过上一步建立的与客户端通信的 socket 与客户端进行转发，完成客户端和服务器的一次通信。

3. 实现代理服务器缓存功能（选作内容，加分项）

(1) 代理服务器进行本地缓存：代理服务器要实现缓存功能，就需要将服务器的响应报文缓存在本地（代理服务器）文件中，并根据客户端的需要进行查询历史文件的缓存信息，并判断是否直接返回缓存文件中的信息作为响应报文。

(2) 基本判断逻辑如下：代理服务器收到客户端的请求报文，客户端根据请求报文的头部信息，在代理服务器的缓存文件中进行检索；若找不到该文件，说明未被缓存，则直接向服务器递交请求；若找到该文件，则提取该缓存文件中的 Date 信息，并重新构造请求报文，在请求报文头部结尾添加行：if-modified-since Date（这里的 Date 指的是提取的响应报文中的 Date 后面的日期）。注意：不可以直接将该行添加在头部第一行，否则会请求报文错误，因为头部行第一行是请求行。发送的修改过的请求报文被服务器接收后，服务器会发送响应报文。若响应报文头部行中含有 304 Not Modified，则说明我们的缓存信息是可用的，未被更新的，所以我们直接将缓存文件中的响应报文发送给客户端；若头部行中含有 200 OK，则说明请求信息已被更新，我们需要将服务器发送的响应报文转发给客户端（因为这时候响应报文包含相应的请求响应数据），同时将该响应缓存更新到本地缓存文件中。功能实现逻辑如下图。



缓存功能逻辑流程图

```
HTTP/1.1 200 OK
Date: Sun, 27 Oct 2019 07:30:53 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Expires: Sun, 27 Oct 2019 06:46:09 GMT
Server: waf/4.17.1-2.el6
Cache-Control: max-age=3600
Cache-Control: private
Content-Encoding: gzip
Age: 6284
X-Via: 1.1 nxian149:7 (Cdn Cache Server V2.0), 1.1 shuangx148:2
```

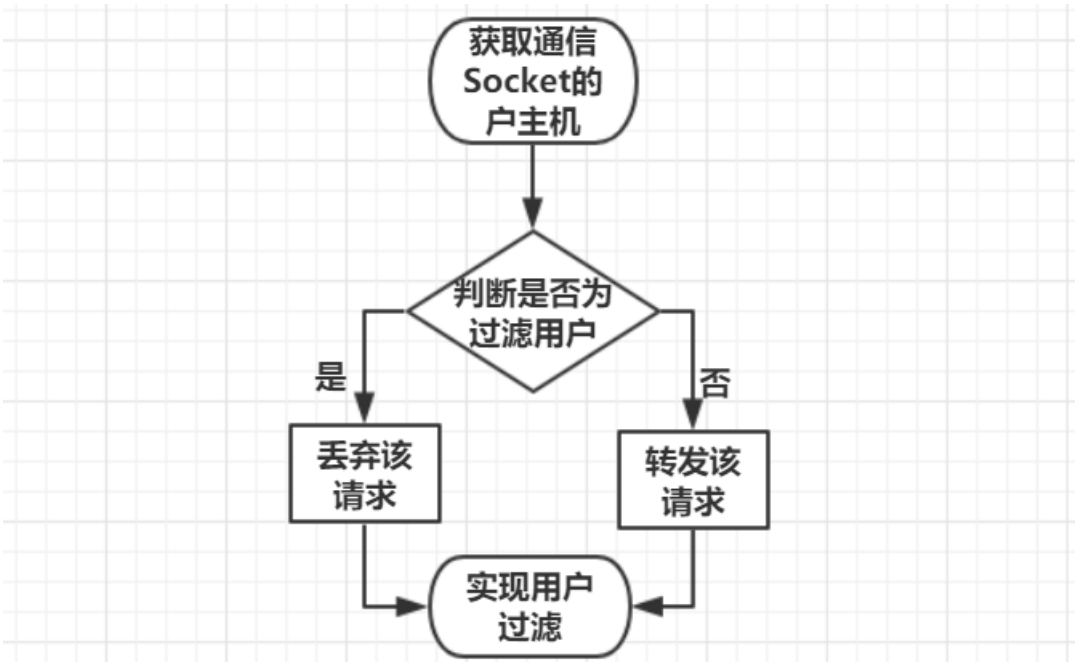
www.7k7k.com 响应报文缓存文件

状态码	含义
100～199	表示成功接收请求，要求客户端继续提交下一次请求才能完成整个处理过程
200～299	表示成功接收请求并已完成整个处理过程
300～399	为完成请求，客户需进一步细化请求。例如，请求的资源已经移动一个新地址
400～499	客户端的请求有错误
500～599	服务器端出现错误

响应报文状态码解析

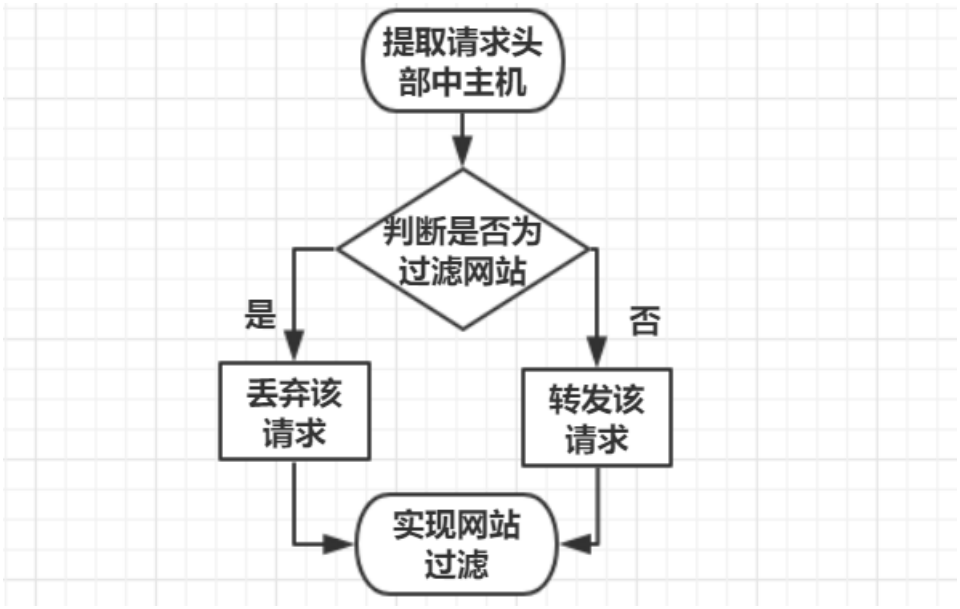
4. 实现用户、网页过滤、钓鱼（选作内容，加分项）

(1) 用户过滤：通过配置我的代理服务器的过滤文件，代理服务器根据与客户端通信的套接字获取客户端的主机，若是过滤文件中指明的要被过滤掉的用户，则直接丢弃客户的请求报文，不再向服务器发送请求，以此达到过滤用户的目的。处理逻辑流程图如下。



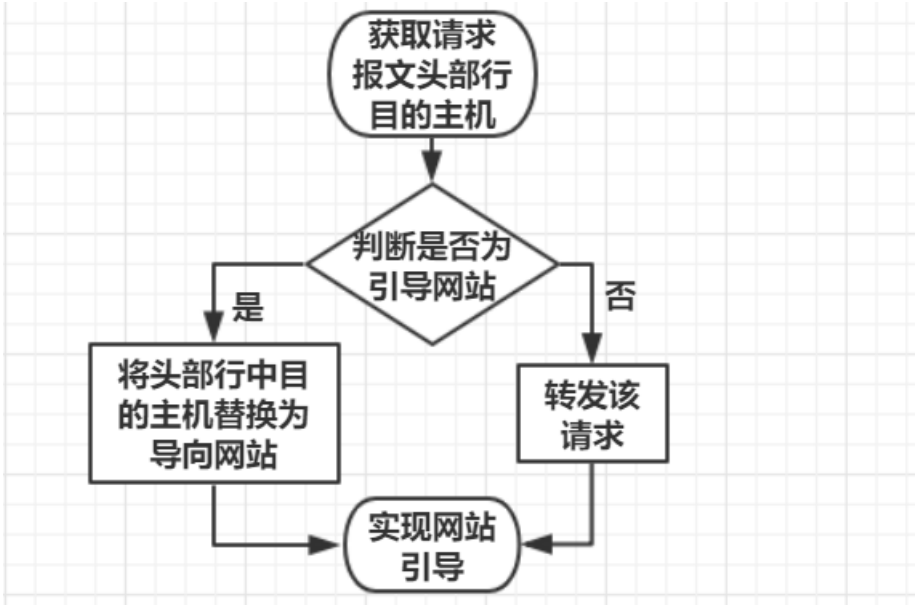
用户过滤逻辑流程图

(2) 网页过滤：通过配置我的代理服务器过滤文件，代理服务器提取请求报文中的头部行获取目的主机，若是过滤文件中指明的要被过滤掉的主机名，则直接丢弃客户的请求报文，不再向服务器发送请求，以此达到网页过滤的目的。处理逻辑流程图如下。



网站过滤逻辑流程图

(3) 网站引导（钓鱼）：通过配置我的代理服务器的过滤文件，代理服务器提取客户端请求报文中的目的主机，若是过滤文件中指明的要被引导的网站，则将该请求报文中的头部行中的 URL 进行替换，替换为要引导向的 URL 地址，以此达到网页引导的目的。处理逻辑流程图如下。



网站引导处理流程图

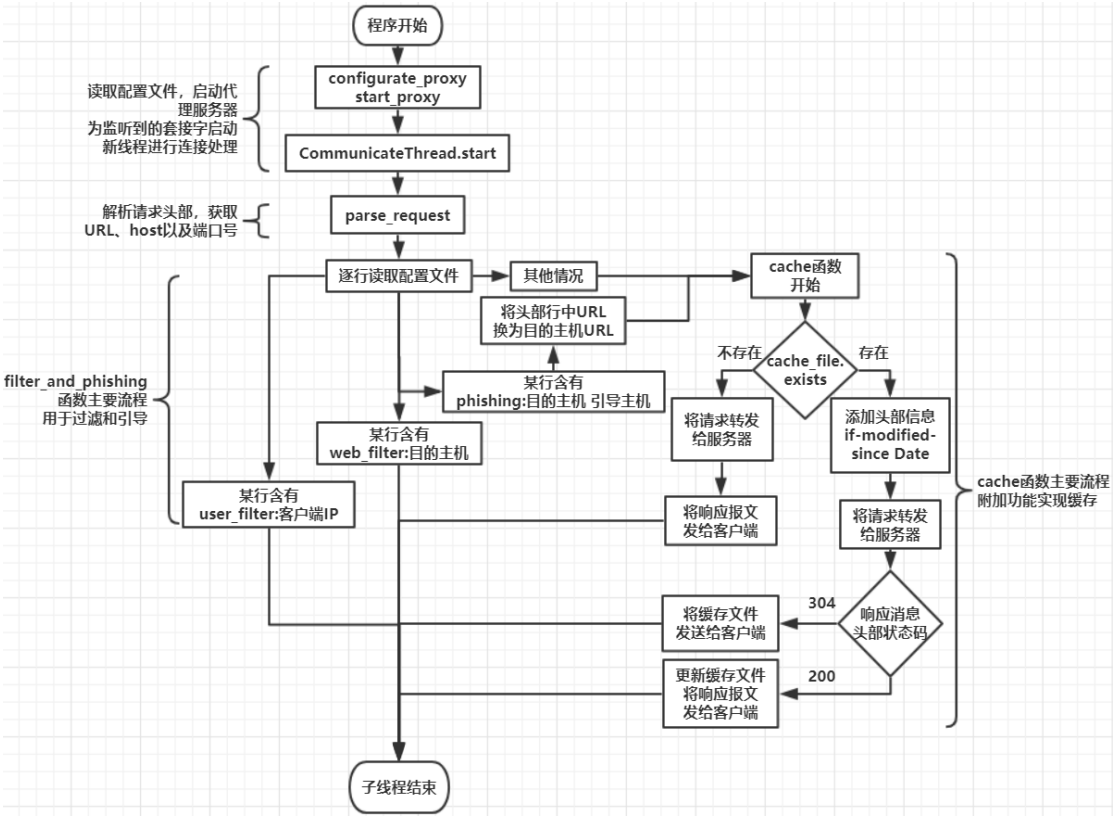
五、实验过程

关于实验代码的几点说明：

我的代码包括两个类 `HTTP_Proxy` 和 `CommunicateThread` 以及一个文件夹 file 及其中的一个配置文件 `configuration_file.txt`。

`HTTP_Proxy` 为可运行主类，用于不断监听客户端的 HTTP 请求，`CommunicateThread` 用于接收来自客户端的 HTTP 请求报文并将其转发给服务器端，并做相应的处理，该类继承自 Thread 类，用于多线程并发处理 HTTP 请求。

1. 程序总流程图：包括附加功能



2. 设计并实现一个基本 HTTP 代理服务器

(1) 注意点：在从 client_socket 与 server_socket 的流中读取数据时，需要设置超时时间，因为如果不设置超时时间，流的末尾只有到套接字关闭的时候才会出现，因此程序会陷入阻塞，无法继续执行。我这里设置阻塞时间为 1000ms；

```
while (true) {
  try {
    server_socket.setSoTimeout(this.socket_time_out);// 设置超时时间用于跳出流阻塞
    int b = proxy_server_in.read();// 读取响应报文
    if (b == -1) {
      break;
    } else {
      out_bytes.add((byte) (b));
      server_socket.setSoTimeout(0);
    }
  } catch (SocketTimeoutException e) {
    break;
  }
}
```

(2) 新建一个 ServerSocket 对象，并输入参数 server_port=10240，表示该 socket 用于不断监听端口 10240 的 HTTP 请求；

(3) 启动函数 `HTTP_Proxy.configurate_proxy(null)`，用于读取配置文件 `configuration_file.txt`，初始化类属性：是否开启用户过滤、网站过滤与引导，程序默认打开所有功能。

(4) 启动函数 `HTTP_Proxy.start_proxy()`，设置 while 循环条件为 true，让 server_socket 不断监听来自客户端的 HTTP 请求，为接收的没一个请求新建一个 `CommunicateThread` 线程用处理该请求的连接；

```
while (true) {// 不断监听来自客户端的请求
  new CommunicateThread(this.server_socket.accept()).start();// 新建子线程处理连接请求
}
```

(5) 代理服务器设置 client_socket 用于与客户端建立连接，首先通过流处理获取客户端 HTTP 请求，将请求首行调用 `this.parse_request(proxy_line)` 按空格进行解析处理：获取请求 URL、获取请求 host；

```
public void parse_request(String head_line) {
  this.URL = head_line.split("[ ]")[1];// 获取请求的目的URL
  int index = -1;
  this.host = this.URL;// 下面用于获取请求的主机名
  if ((index = this.host.indexOf("http://")) != -1) {// 去掉URL中的http://
    this.host = this.host.substring(index + 7);
  }
  if ((index = this.host.indexOf("https://")) != -1) {// 去掉URL中的https://
    this.host = this.host.substring(index + 8);
  }
  if ((index = this.host.indexOf("/")) != -1) {// 去掉URL中的/
    this.host = this.host.substring(0, index);
  }
  if ((index = this.host.indexOf(":")) != -1) {// 去掉URL中的:
    this.port = Integer.valueOf(this.host.substring(index + 1));
    this.host = this.host.substring(0, index);
  }
}
```

(6) 一个正常的首次请求，代理服务器将请求转发给目的主机：需要建立一个与目的主机通信的套接字 `server_socket = new Socket(this.host, this.port)` ，并将请求报文写入到这个套接字的输出流，即可将请求发送到目的主机；

(7) 接着需要代理服务器作为客户端接收来自服务器的响应报文（并将响应报文缓存到对应的文件中），即将与服务器通信的套接字 server_socket 的输入流中取到的信息写到与客户端通信的套接字 client_socket 的输出流中，即完成代理服务器的转发响应报文的功能。

```
while (true) {
  try {
    server_socket.setSoTimeout(this.socket_time_out);// 设置超时时间用于跳出阻塞状态
    int b = proxy_server_in.read();// 字节流读取响应报文
    if (b == -1) {
      break;
    } else {
      cache_file_out.write(b);// 写入到缓存文件中
      proxy_client_out.write(b);// 写入到向客户端发送响应的流
      server_socket.setSoTimeout(0);
    }
  } catch (SocketTimeoutException e) {
    break;
  }
```

3. 实现代理服务器缓存功能（选作内容，加分项）

(1) 注意点：

a) 我们要将一个 URL 对应的响应报文保存在本地缓存中，文件的目录组织方式是`"src/file/" + this.host + "/" + this.URL.hashCode() + ".txt"`，其中每个 host 为一个目录，下面存放的是所有的该目的主机的响应报文，文件名为 URL 调用 hashCode 得到的数字；

b) 在构造新的请求报文时，我们不可以将加入的新行插入到原请求报文的头部，因为请求报文头部为请求行，用于请求报文信息确定。若加在首行，服务器端会发送 400 Bad Request 响应表示请求报文格式错误。因此我在这里将 if-modified-since Date 语句加在请求报文的倒数第二句（倒数第一句为\r\n）。

(2) 缓存功能基本逻辑：首次请求某个 URL，要将该请求的响应报文保存在代理服务器的本地缓存文件中，并像基本功能实现中的操作一样，转发客户端的请求报文和服务器端的响应报文。当再次请求某个 URL 时，根据 hashCode 函数得到该响应报文对应的文件路径，并获取文件的最后修改时间，构造新的请求报文（即在原请求报文的倒数第二行加上语句 if-modified-since Date）。然后接收到服务器发送的响应报文，若头部行中的状态码为 304 说明缓存文件可以使用，若为 200，我们则需要将该响应报文转发给客户端并将其更新写入到对应的缓存文件中。

(3) 具体由函数 `in_cache_new()`实现：首先确定文件名 `this.URL.hashCode()+".txt"`，判断该文件是否存在：若不存在，则正常执行本报告中的第 2 项的说明；若存在，则说明该 URL 被访问过，则需要构造新的请求报文，调用 cache_file 的 `lastModified()`函数，并将该文件的最后修改时间（long 数据类型）根据 `SimpleDateFormat("EEE, d MMM yyyy HH:mm:ss z", Locale.ENGLISH)`改为 HTTP 的响应格式 GMT 时间，将该报文通过 server_socket 的输出流转发给服务器端；

(4) 代理服务器通过 server_socket 的输入流接收服务器的响应报文。如果头部行中含有状态码 304，则说明本地缓存文件可用，则将本地缓存直接作为响应报文通过 client_socket 套接字的输出流转发给客户端；若含有状态码 200，说明缓存文件不可用，响应报文有更新，我们需要将响应报文直接通过 client_socket 套接字的输出流转发给客户端，并将该报文写入到本地缓存文件中。

```
if (this.respose_gram.split("\r\n")[0].contains("304")) {// 响应报文头含304，则缓存可用
  System.out.println("缓存命中数:" + (++HTTP_Proxy.cache_hit) + "\t命中:" + this.URL
      + "\t文件名:" + this.URL.hashCode());
  FileInputStream cache_file_read = new FileInputStream(cache_file);
  int b;// 直接将缓存报文发送给客户端
  while ((b = cache_file_read.read()) != -1) {
    proxy_client_out.write(b);// 写入客户端的流
  }
  cache_file_read.close();
```

状态码含有 304

```
} else if (this.respose_gram.split("\r\n")[0].contains("200")) {// 响应报文头含200，更新
  FileOutputStream cache_file_out = new FileOutputStream(cache_file);// 写缓存文件的流
  proxy_client_out.write(this.respose_byte);// 将从服务器读取到的转发给客户端
  cache_file_out.write(this.respose_byte);// 更新本地缓存
  cache_file_out.close();
}
```

状态码含有 200

4. 实现用户、网页过滤、钓鱼（选作内容，加分项）

(1) 注意点：

a) 是否打开这三个功能需要在配置文件中进行相应的设置（格式要正确），默认这三个功能全部打开；

(2) 是否开启用户、网页过滤和钓鱼功能可以在配置文件中进行配置 `configuration_file.txt`。默认开启，子线程通过调函数 `filter_and_phishing()` 实现该附加功能。

(3) 用户过滤：client_socket 套接字调用 `getInetAddress().getHostAddress()` 获取客户端的主机 IP，并读取配置文件的每一行，判断含有 user_filter 的数据行是否含有该 IP，若含有该 IP，说明客户端为被过滤用户，这里设置为 127.0.0.1 代表过滤本机地址，过滤后，将无法进行 HTTP 访问；

```
if (HTTP_Proxy.user_filter
    && line_filter.contains(client_socket.getInetAddress().getHostAddress())
    && line_filter.contains("user_filter")) {
  System.out.println(
      "你不能访问该网站，因为你是被限制用户:" + client_socket.getInetAddress().getHostAddress
  bfr_filter.close();
  return false;
```

(4) 网站过滤：从 client_socket 的输入流获取客户的请求报文首行信息，提取出目的主机 host，并读取配置文件的每一行，判断含有 web_filter 的数据行是否含有该 Host，若含有该 Host，说明目的主机已被过滤，我这里设置哈工大教务处网站为被过滤网站；

```
} else if (HTTP_Proxy.web_filter && line_filter.contains(this.host)
    && line_filter.contains("web_filter")) {// 若开启了网站过滤且目的主机为被过滤的主机，
  System.out.println("网站受限:\t" + this.host);
  bfr_filter.close();
  return false;
```

(5) 网站引导（钓鱼）：类似于网站过滤，从 client_socket 的输入流获取客户的请求报文首行信息，提取出目的主机 host，并读取配置文件的每一行，判断含有 phishing:的数据行是否含有该 Host，若含有该 Host，说明目的主机已被引导，并提取出该行中 host 后面的信息，为要导向的目的网站。并构造新的请求报文，即将头部行中的 URL 替换为引导主机对应的 URL，并将其转发给服务器端。

```
} else if (HTTP_Proxy.phishing && line_filter.contains(this.host + " ")
    && line_filter.contains("phishing")) {// 若开启了网站引导且目的主机为被引导的主机则将
  this.host = line_filter.split(" ")[1];
  String old_URL = this.URL;
  this.URL = "http://" + this.host + "/";
  this.port = 80;
  request_gram = request_gram.replace(old_URL, this.URL);// 将请求报文中的头部URL替换为
  System.out.println("网站引导:\t" + this.host);
  bfr_filter.close();
  return true;
```

六、实验结果

1. 设计并实现一个基本 HTTP 代理服务器

(1) 访问哈工大网站没问题（以今日哈工大为例）：**today.hit.edu.cn**

网站截图

```
HTTP_Proxy [Java Application] D:\java\bin\javaw.exe (2019年10月27日 下午11:15:11)
代理服务器:         运行
监听端口:           10240
用户过滤:           开启
网页过滤:           开启
服务器缓存:         开启
**************************************************
缓存文件不存在，需要向服务器转发请求
GET http://today.hit.edu.cn/ HTTP/1.1
Host: today.hit.edu.cn
Proxy-Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.362
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: _ga=GA1.3.1561201084.1572074005; _gid=GA1.3.1624053936.1572074005

接收响应报文成功，并将其存入缓存文件中，文件名:    333368466.txt
```

(2) 访问百度视频网站没问题（可以看视频）：**v.baidu.com**

百度视频主页，可以看视频，但代理压力较大

```
接收响应报文成功，并将其存入缓存文件中，文件名:    1063847798.txt
缓存命中数: 50    命中: http://asimgs.pplive.cn/imgs/materials/2019/10/24/1aed4a3b094ef55f22a9cddf4fbe014f
缓存命中数: 51    命中: http://asimgs.pplive.cn/imgs/materials/2019/10/24/72c2aa3afd652a6f2f39e06edb25be7c
接收响应报文成功，并将其存入缓存文件中，文件名:    -749898981.txt
缓存命中数: 52    命中: http://asimgs.pplive.cn/imgs/materials/2019/10/24/1aed4a3b094ef55f22a9cddf4fbe014f
缓存命中数: 53    命中: http://asimgs.pplive.cn/imgs/materials/2019/10/24/72c2aa3afd652a6f2f39e06edb25be7c
```

(3) 访问著名小游戏网站没问题：www.7k7k.com

2. 实现代理服务器缓存功能（选作内容，加分项）

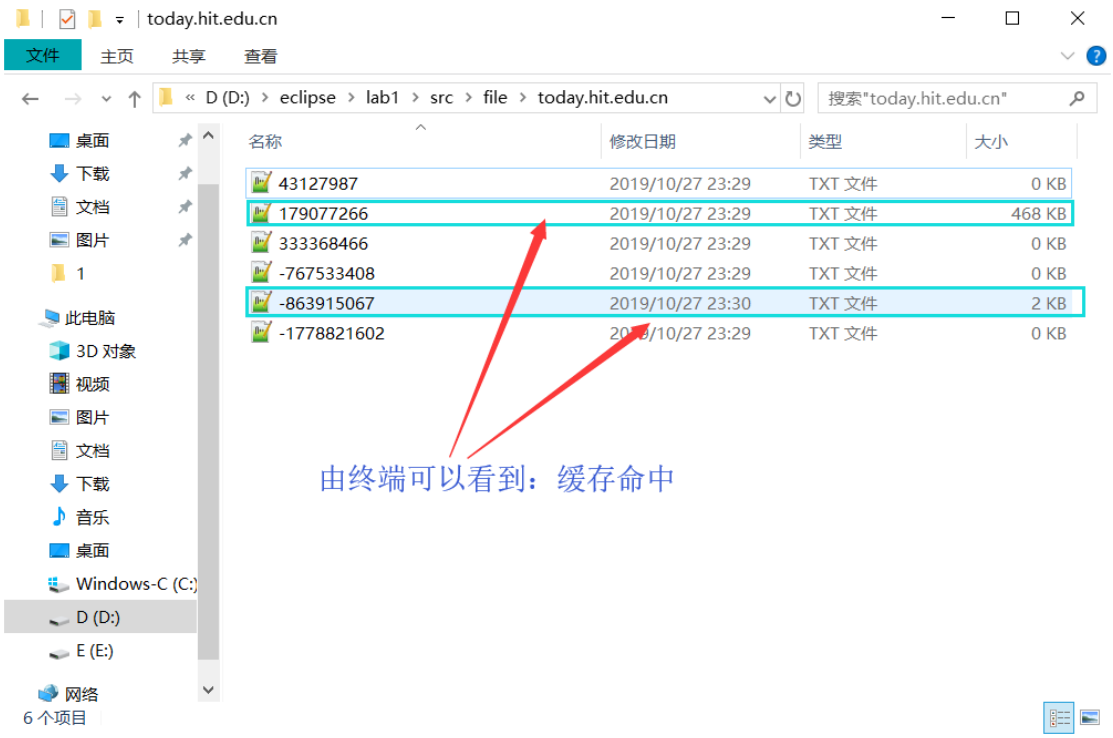
(1) 第一次访问今日哈工大时，缓存文件不存在，一定是缓存不命中的；

```
HTTP_Proxy [Java Application] D:\java\bin\javaw.exe (2019年10月27日 下午11:28:46)
代理服务器:        运行
监听端口:          10240
用户过滤:          开启
网页过滤:          开启
服务器缓存:        开启
**************************************************
缓存文件不存在，需要向服务器转发请求
GET http://today.hit.edu.cn/ HTTP/1.1
Host: today.hit.edu.cn
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.362
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: _ga=GA1.3.1561201084.1572074005; _gid=GA1.3.1624053936.1572074005
```

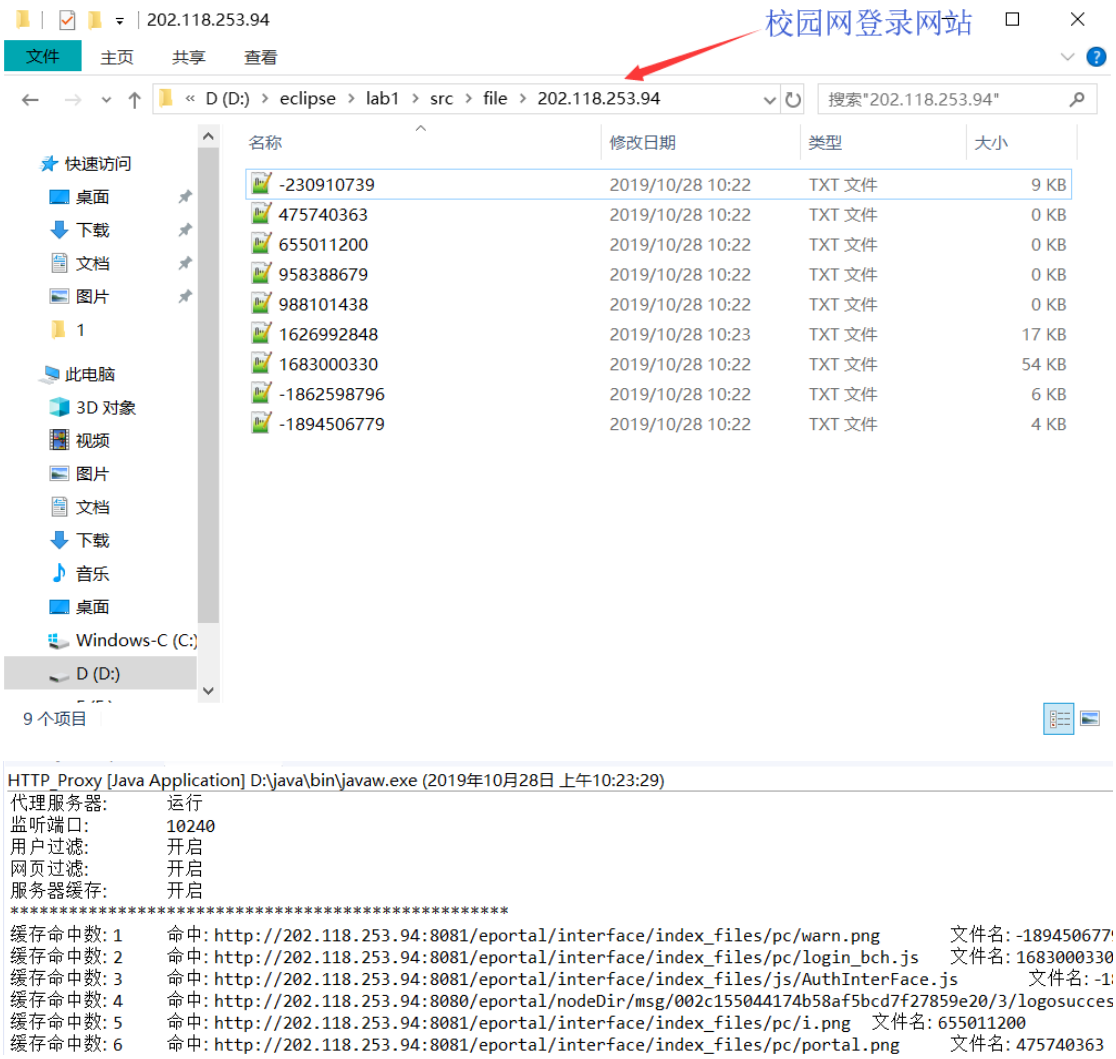
(2) 第二次访问今日哈工大时，存在缓存命中情况，同时可以看到缓存文件多了今日哈工大这一项。如截图；

```
缓存文件不存在，需要向服务器转发请求
GET http://today.hit.edu.cn/sites/today1.prod1.dpweb1.hit.edu.cn/themes/hit_today/favicon.ico HTTP/1.1
Host: today.hit.edu.cn
Proxy-Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.362
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://today.hit.edu.cn/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: _ga=GA1.3.1561201084.1572074005; _gid=GA1.3.1624053936.1572074005; _gat=1

接收响应报文成功，并将其存入缓存文件中，文件名:     -863915067.txt
缓存命中数: 2     命中: http://today.hit.edu.cn/     文件名: 179077266
缓存命中数: 3     命中: http://today.hit.edu.cn/sites/today1.prod1.dpweb1.hit.edu.cn/themes/hit_today/favi
```



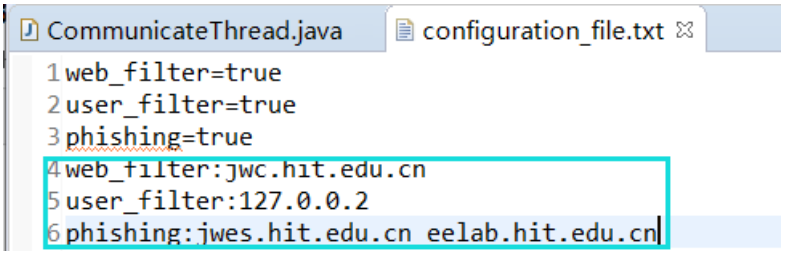
(3) 当访问较大的数据流网站时，命中次数较高（因为传递 HTTP 报文较多）



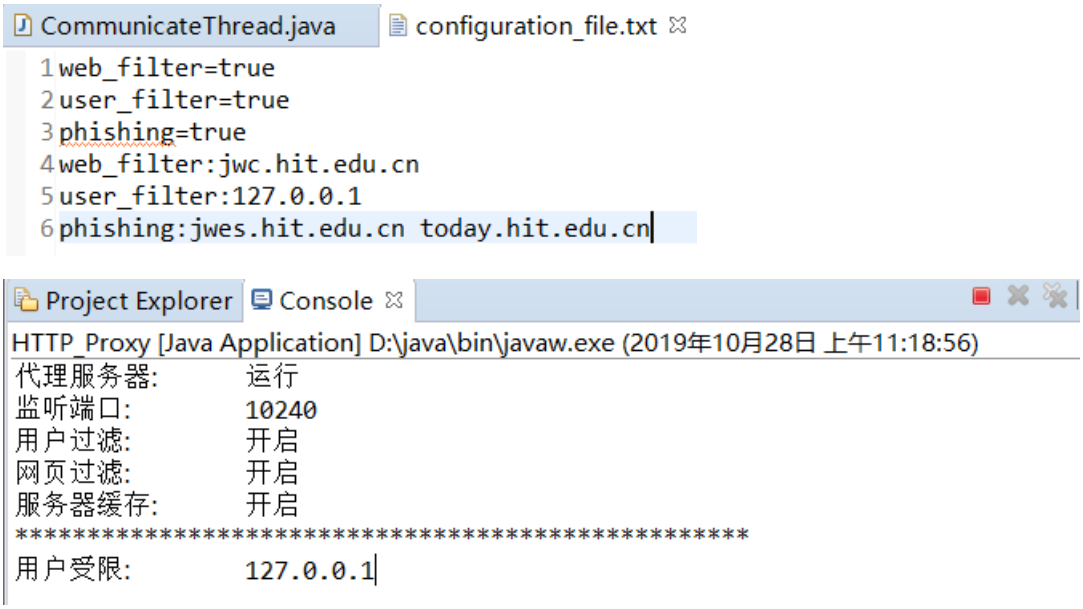
第二次访问时命中了 6 项

3. 实现用户、网页过滤、钓鱼（选作内容，加分项）

(1) 首先我的配置文件内容如下：前三行用于配置附加功能打开，后三行用于具体的过滤信息和引导信息，这里网页过滤对象为 cs.hit.edu.cn、用户过滤对象为 127.0.0.2（后期会修改为 127.0.0.1 以过滤本机用户）、网站引导对象为 jwes.hit.edu.cn 将其导向 today.hit.edu.cn 网站，若要修改配置信息，请按照 ReadMe 文件要求的格式进行配置文件内容。

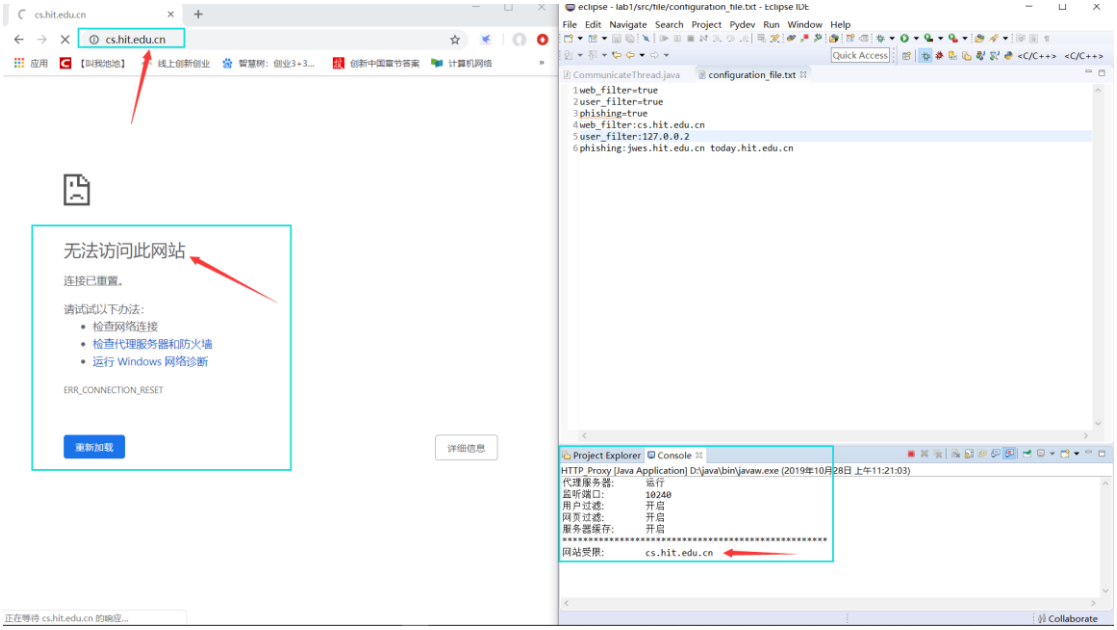


(2) 用户过滤实现截图：用户过滤对象首先是 127.0.0.2，本机作为客户端仍然可以访问服务器获得 HTTP 服务；但是当改为 127.0.0.1 时，本机无法访问服务器，两部分如截图；

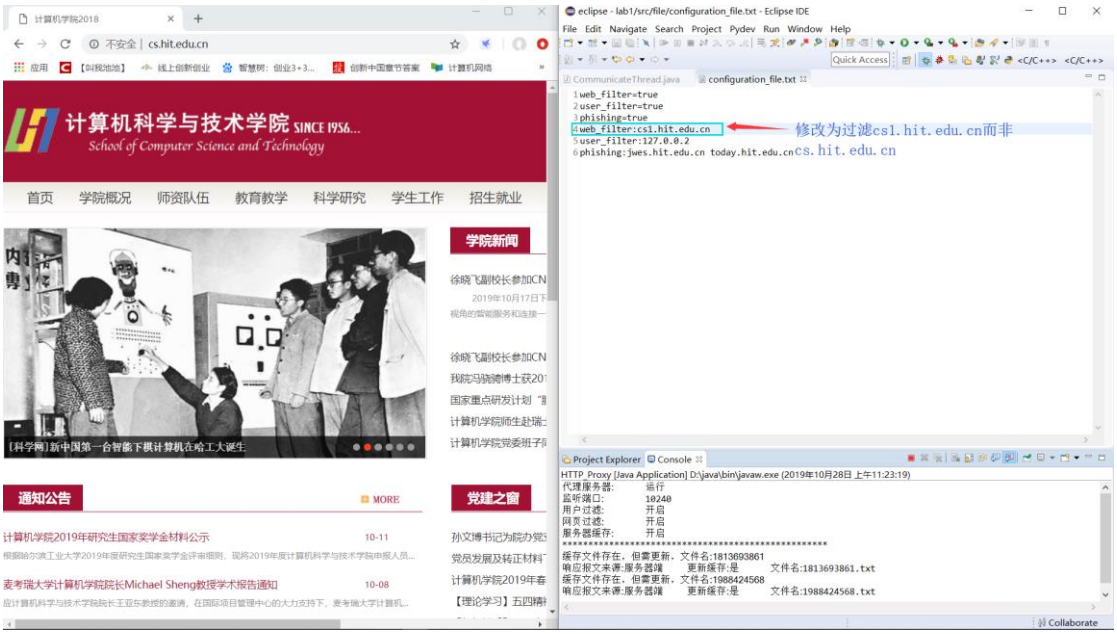


用户受限

(3) 网页过滤实现截图：网页过滤对象 cs.hit.edu.cn，无法访问该网站如截图；当删去时，可以正常访问



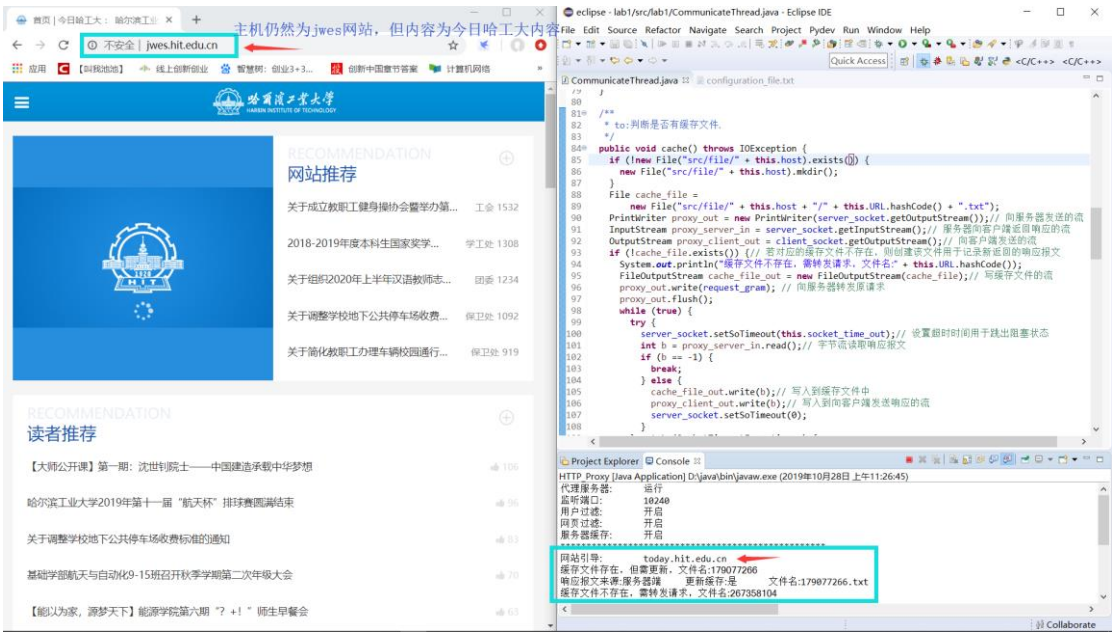
修改前无法访问



修改后可以访问

(4) 网站引导实现截图：我的网站引导对象为从 jwes.hit.edu.cn 将其导向 today.hit.edu.cn 网站，如截图（已配置钓鱼功能打开：true），我们可以看到网站主机名仍为 jwes.hit.edu.cn，但是加载的时今日哈工大的网页；（经测试，可以引导向绝大多数网站）

```
3 phishing=true
4 web_filter:cs.hit.edu.cn
5 user_filter:127.0.0.2
6 phishing:jwes.hit.edu.cn today.hit.edu.cn
```



七、实验心得

1. 知识收获

掌握了 Socket 编程有关的知识结构，较为深入的理解了应用层的 HTTP 协议的相关内容。熟悉并掌握 Socket 网络编程的过程与技术；同时深入的理解了 HTTP 协议，掌握了 HTTP 代理服务器的基本工作原理，并实现了简单的的缓存功能以及过滤和引导功能，同时掌握了 HTTP 代理服务器设计与编程实现的基本技能。对这一章节的知识有了一个系统全面的了解和体会。

2. 代码体会

(1) 通过编程训练，掌握了 MOOC 中学习不到的知识，比如：代理服务器的工作原理及其实现；

(2) 同时也掌握了重点知识的细节内容，比如：HTTP 请求报文的详细格式内容和携带的对应的内容以及响应报文的详细格式和携带的对应的内容；

(3) 通过 java 编程，对 Java 的流处理机制印象深刻，我在实现代理服务器的功能时，用了多次套接字的流处理获取和发送相关的报文信息，加深了我对 java 代码的认识。