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	陈泊舟		院系	计算机科学与技术学院		
班级	1703110		学号	1173710224		
任课教师	李全龙		指导教师	李全龙		
实验地点	格物 214		实验时间	2019.10.26 上午 12 节		
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：

（注：实验报告模板中的各项内容仅供参考，可依照实际实验情况进行修改。）

本次实验的主要目的。

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。

实验环境主要是接入 internet 的主机和 windows10 操作系统，java 开发语言以及相应开发环境，jdk，eclipse 等。

套接字编程在不同语言下，表现在使用略有不同，比如在 C++中关于套接字的构造函数，仅仅是返回一个数据结构，要想建立连接，必须首先初始化环境，然后创建套接字，并配置相关协议以及 ip 端口号等信息，协议主要还是 IP 协议，TCP 协议，然后如果是 server 的话，还需要调用 listen 函数进行监听，然后是 accept 函数，来进行建立连接；另外针对于客户端虽然简单了一点，但是过程也比较繁琐，最后要使用 connect 函数来建立连接。在进行一般工作的时候，对底层的配置没有太多的变化，特别是针对像计网这样的第一次实验，这些初始化的过程可以认为都是一样的，所以使用 C++进行编程也只是凭空增加自己的工作负担而已，经过认真考虑，决定使用应用较为广泛的 java 语言实现。将繁琐的过程省略之后，也能更好的体会这个过程，从而实现这个实验更好的效果。

实验内容：

概述本次实验的主要内容，包含的实验项等。

1. 设计基本代理服务器。

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。在此次实验中我使用的端口是8080。

2. 为代理服务器添加cache功能。

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 -modif ied-since 头

行），向原服务器确认缓存对象是否是最新版本。当然也可以通过一个http请求来判断网页是否有所更新。结果是一样的。

3. 扩展 HTTP 代理服务器，支持如下功能：

3.1 网站过滤

允许/不允许访问某些网站；

3.2 用户过滤

支持/不支持某些用户访问外部网站；

3.3 网站引导

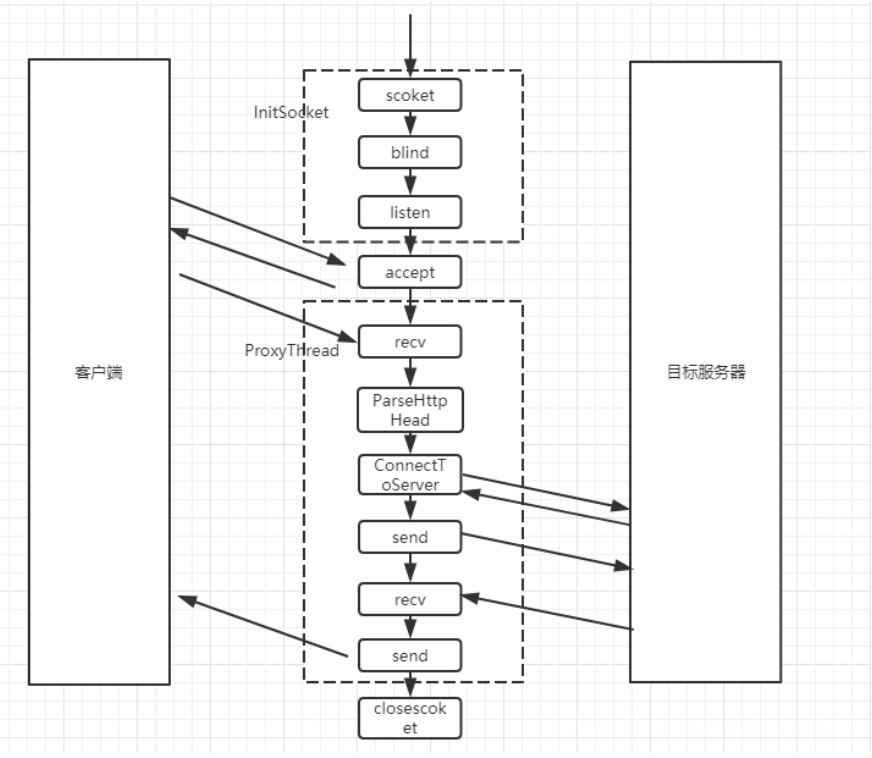
将用户对某个网站的访问引导至一个模拟网站，也就是通常所说的钓鱼。

实验过程：

以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。

1. 基本代理服务器的实现

1.1 过程逻辑



图片引自 https://blog .csdn.net/rocketeerLi/article/details/83717613

如上图所示是利用C++实现的代理服务器的基本逻辑流程。

首先，客户端和代理端都应该进行initState。在这个过程中客户端要做的事情是创建套接字，调用connect函数，等待代理服务器响应建立连接。代理一次调用船舰socket函数，bind函数绑定IP地址和端口号，调用listen函数进入监听状态，调用accept函数建立连接，并创建一个新的套接字与之通信。

接下来代理服务器接受来自客户端的请求，并从中提取将要访问的目的服务器网址，并于相应服务器建立连接，这个连接的建立过程与上面相同，在与服务器建立连接之后，将相应的请求发送给目标服务器，然后等待服务器的响应。并将相应的响应报文发送给客户端。至此完成一次代理工作，接下来就是循环过程，不断的接受新的客户请求。

上面说的是C++实现的方式，而使用java进行实现，建立连接的过程已经被封装起来了，在使用的时候，只需要调用socket函数或者serverSocket函数，就可以分别完成客户端和服务器端的环境初始化。但是逻辑过程是一样的。

1.2 细节问题

1.2.1 Java包的选择

Java语言提供了很多进行网络编程的库，还有一些第三方的库可以进行调用，在考量了实验的内容以及难度之后，决定使用java.net.Socket;和java.net.ServerSocket;进行套接字的创建，以及连接的建立和服务器监听，这两个类，支持底层操作，也提供默认的封装操作。具体使用如下，分为服务器端和客户端：

```
int port = 8080;
ServerSocket proxyServerSocket = new ServerSocket(port);
while (true) {
    System.out.println(1);
    Socket socket = proxyServerSocket.accept();
      Socket serverSocket = new Socket(host, 80);
```

服务器端和客户端使用

1.2.2 处理客户请求

在处理客户端请求时，为了实现多用户的代理服务器，应该新开一个线程进行处理，也就是每一个线程处理一个客户的一个请求。代码使用如下：

```
Socket socket = proxyServerSocket.accept();
(new processClient(socket)).start();
```

连接建立方法

展示部分客户端请求处理的核心代码，关于每一步的作用见注释：

```
public void run(){
    byte[] bytes = new byte[4096];//将每次读取的数据存储到bytes中
    try {
        inputStream = clientSocket.g etInputStream();//获取请求输入流
        outputStream= clientSocket.g etOutputStream();//获取响应输出流
        inputStream.read(bytes);//读取请求
        String    messag e = new String (bytes);
        String    patternString    = "Host: [0-9a-zA-Z.]+";
        Pattern pattern = Pattern.compile(patternString );
        Matcher matcher=pattern.matcher(messag e);//获取访问网站名，
提取Host关键字
        if (matcher.find()) {
            String    host = matcher.g roup().replace("Host: ", "");
            if (haveThisF ile(host) == f alse) {
                Socket serverSocket = new Socket(host, 80);
                    //与服务器建立连接
                serverOutputStream = serverSocket.g etOutputStream();
                serverOutputStream.write(bytes);//将请求传给目标服务器
```

基本流程详解

1.2.3 解决超时问题

在进行客户请求处理，转发请求，接受响应，转发响应的过程中，经常会出现超时的问题，抛出connect timed out异常。首先可以肯定是中间处理时间太长了，导致访问超时。有两种解决方案，一个是将timeout设置为一个较大值，另一个就是设计一个更加高效的策略，我选择的是第二种。为了实现高效，需要找到能够节省时间的过程。经过分析，发现我的从服务器接收数据和向客户端传输数据的过程是串行的，但是这个过程完全可以通过两个线程并行执行解决，也就是一个线程读取数据的同时，另一个数据进行数据的

输出。直观上来说，这一过程的时间最终表现为之前的方法的一半。

核心代码展示如下，并附带相关注释：

```
/**
 * transmit messag e to server
 * @author 陈泊舟
 */
class thread2 extends Thread {
    private InputStream input;//定义输入流
    private OutputStream output;//定义输出流
    public thread2(InputStream input, OutputStream output) {//构造函数
        this.input = input;
        this.output = output;
    }
    public void run() {
        try {//每次读一个byte传递
            while (true) {
                int next = input.read();
                if (next == -1) break;
                output.write(next);
            }
        } catch (IOException e) {//异常处理
            System.out.println("Maybe something  wrong  happened!");
        }
    }
}
```

信息传输详解

1.2.4 解决写错误问题

在程序执行的过程中，通常连接已经建立了，但是总是不能完整地打开网页，同时会抛出异常write time out。为了解决这样的问题，尝试了多种读写

方法。具体如下：

```
inputStream.read(bytes);
String    messag e = new String (bytes);
serverOutputStream.write(bytes);
```

一种是将数据读取到byte类型的变量中，然后通过输出流的write函数将其输出。注意在这个过程中不能使用string 类型的变量进行传递。会出现静态类型错误。

```
int next = serverInputStream.read();
if (next == -1) break;
outputStream.write(next);
tmp.add(new Integ er(next));
```

这一种方法是将数据作为int类型变量挨个读取。

另外还有一个问题就是在将byte进行本地缓存的时候出现的问题，这将会在cache环节进行介绍。

2. 添加cache功能

2.1 暂存网页信息

添加cache功能，必须要做的就是将之前从服务器传过来的网页文件进行保存，这个过程可以有两种方法实现。一种是直接将信息保存在本地文件中，这样做的好处是可以顺便记录访问时间的时间戳，获取比较方便，操作简单易行，但是也有不好的一点，那就是数据格式的问题，不能直接将数据按照string 类型或者byte类型存储，需要将不同的字符进行分割，因为都是255之内的整数，如果直接存储就会造成数据之间之间直接相连，不能做到单个数据之内同步。另一种方法是，使用本地缓存存储，也就是开一个ArrayList将访问文件作为string 存储，在使用的时候再调用g etByte函数进行转换。这个做法有所欠妥，每次程序关闭都会将获取到的信息恢复为空，并且对内存有着比较大的要求。但是有点也很明显，操作起来十分简单，使用一个HashMap就可以实现cache的功能。核心代码展示如下：

```
if (matcher.f ind()) {
    String    host = matcher.g roup().replace("Host: ", "");
```

```
                if (haveThisF ile(host) == f alse) {
                    Socket serverSocket = new Socket(host, 80);
                    serverOutputStream = serverSocket.g etOutputStream();
                    serverOutputStream.write(bytes);
                    (new thread2(inputStream,serverOutputStream)).start();
                    serverInputStream = serverSocket.g etInputStream();
                        //开一个新的list存储网页文件信息
                    ArrayList<Integ er> tmp = new ArrayList<Integ er>();
                    while(true) {
                        int next = serverInputStream.read();
                        if (next == -1) break;
                        outputStream.write(next);
                        tmp.add(new Integ er(next));
                    }
                    proxy.cache.put(host, tmp);//将网页信息与网站名对应，
存储映射表中
                        //创建文件，将信息存储在本地文件中
                    F ile f ile = new F ile(host + ".txt");
                    f ile.createNewF ile();
                    PrintStream printor = new PrintStream(f ile);
                    printor.write(tmp.toString ().g etBytes());
                    printor.close();
                }
```

cache信息存储

2.2 判断更新

使用cache最大的意义就是将之前访问过的文件直接传给客户端，而不是重新发送请求，但是在使用的时候，也应该保证内容的实时性，所以应该判断一下是否增加了新的内容。按照报告要求的，要在请求报文中添加if -modif ied-since头行，并在响应报文中获取信息判断是否更新。但是实际上这是一种

浪费资源的方法，因为发给服务器请求报文之后，服务器会处理所有的请求，而不仅仅是是否更新这一条。所以，个人认为应该构建一个http请求报文，向服务器请求上一次更新时间，并和本地文件的last-modif ied时间进行一个对比。如果在上次请求之后更新了文件，那就重新发送请求，按照之前的过程再走一遍，如果没有更新，就将本地文件发送给客户即可。这个过程的实现只需要一个简单的逻辑判断。关于时间戳的获取，请求报文的建立以及相应核心代码的展示如下：

```
/**
 * check the last update time of    the web-site
 * @param host
 * @return
 * @throws IOException
 */
private long    g etUpdateTime(String    host) throws IOException {
    URL u = new URL("http://" + host);
     HttpURLConnection http = (HttpURLConnection) u.openConnection();
     http.setReq uestMethod("HEAD");
     Date lastModif y =new Date(http.g etLastModif ied());
     return lastModif y.g etTime();
}
```

这个函数用于新建一个http请求报文，并从网站获取上次更新时间对应的时间戳作为函数返回值。

至于这个部分的核心逻辑就是一个判断，非常简单，就不再进行展示了。

3. 扩展功能

此次实验针对一下三个扩展功能进行了简单的实现。由于整个过程就是简单的判断，所以就不再进行代码展示。

3.1 网站过滤

过滤特定的网站，在提取了主机名之后，判断一下是不是要过滤的目标

网站，如果是就转发请求，否则转发。

3.2 用户过滤

与网站过滤类似，从请求报文中提取用户名，并进行相应的操作。

3.3 网站引导

当检测到牟勇访问特定的源网站之后，向相应的目标网站发送请求，并将其相应转发给客户即可。

实验结果：

采用演示截图、文字说明等方式，给出本次实验的实验结果。

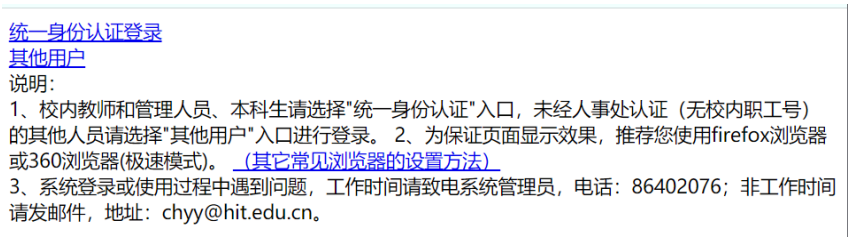
1. 基本代理服务器的实现

通过查找使用http协议的网站，发现有如下网站满足条件，并将其定为此次实验的目标网站，www.4399.com, today.hit.edu.cn, acm.hit.edu.cn, jwts.hit.edu.cn,等。

访问4399：

4399访问界面

访问哈工大教务系统：



jwts访问界面

在这里仅给出两个例子，上面的代码，是开启cache功能的首次运行结果，也相当于是基本的代理服务器功能，但是从上面的展示结果可以看出来增加了cache功能之后会出现一定的不稳定性。

2. 添加cache功能

关于网站的访问结果就不再进行展示了，在程序运行时输出一些提示进行展示，每次输出1代表接受了一次请求，在接受请求之后会输出目标网站的网站名，有时会抛出一些异常，但是不影响访问，在进行了一次访问之后，第二次访问时，就输出了像在表格最后的信息，已经有这个文件了，就直接将这个文件传输为客户就行了，并且从时间上判断，网页还没有更新，所以这一轮的决策是从本地上传：

```
1
- - - - - - - - - - - - - - - - - - www.4399.com
1
- - - - - - - - - - - - - - - - - - www.4399.com
```

```
1
- - - - - - - - - - - - - - - - -  www.4399.com
Maybe something  wrong  happened!
- - - - - - - - - - - - - - - - -  www.4399.com
1
- - - - - - - - - - - - - - - - -  beacons2.g vt2.com
1
- - - - - - - - - - - - - - - - -  g .live.com
1
- - - - - - - - - - - - - - - - -  beacons3.g vt2.com
1
- - - - - - - - - - - - - - - - -  log in.live.com
1
- - - - - - - - - - - - - - - - -  log in.live.com
1
- - - - - - - - - - - - - - - - -  log in.live.com
1
- - - - - - - - - - - - - - - - -  log in.live.com
1
- - - - - - - - - - - - - - - - -  www.baidu.com
1
- - - - - - - - - - - - - - - - -
nav.smartscreen.microsof t.com
1
- - - - - - - - - - - - - - - - -  log in.live.com
Maybe something  wrong  happened!
1
1
- - - - - - - - - - - - - - - - -  www.4399.com
- - - - - - - - - - - - - - - - -  img a2.5054399.com
1
- - - - - - - - - - - - - - - - -  g .live.com
1
- - - - - - - - - - - - - - - - -  www.4399.com
have this f ile!!!!!!!!!!!
www.4399.com.txt
lastModif ied time
1572517624008
upload f rom localhost
```

cache使用展示

3. 扩展

3.1 网站过滤

在此次实验中过滤两个网站：

```
/**
 * def ine some websites that couldn't be visited!
 */
    private String  wall1 = "acm.hit.edu.cn";
    private String  wall2 = "www.4399.com";
```

屏蔽网站

访问结果如下，一直等待响应：

屏蔽效果

3.2 用户过滤

实验时把自己墙了就OK了。

屏蔽用户

3.3 网站引导

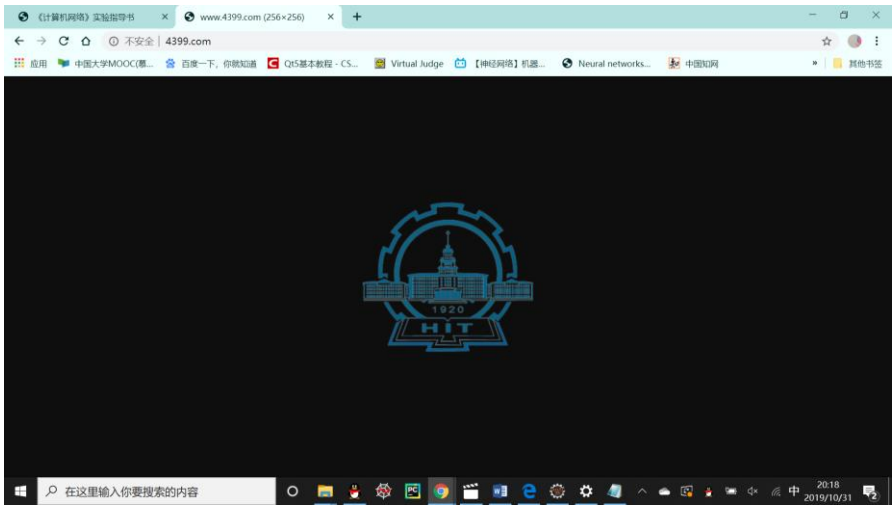
将源和目的分别设置为如下，并在用户访问4399小游戏网站时，将访问引导至哈工大oj的一个图片上去：

```
/**
```

```
* f ishing  source web- site and targ et web- site
*/
    private String  source = "www.4399.com";
    private String  targ et = "acm.hit.edu.cn";
```

钓鱼网站转换

访问结果如下：



钓鱼结果展示

问题讨论：

在实验中还是遇到了不少的问题的，现在总结为以下几点：

1. 实现方法

在实现方法上，有一些可以自己解决的问题，但是并不一定能解决地很好，在博客里面看到一些，有所借鉴，也很有收获，当然这不是抄袭。

2. 实现细节

在实现细节上，有一些很小的错误，比如不能直接将数据存储到.txt文件中，这样会导致数据出现紊乱等问题。

3. 实现逻辑

此次实验的控制逻辑还是很明了的，想清楚以后只要按部就班的使用高级语言实现出来就可以了。

心得体会：

结合实验过程和结果给出实验的体会和收获。

总而言之，此次实验过程简单，实验目标清晰，但是也复习运用了课上学习的内容，作为简单的练习很有意义。