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	田一间		院系	计算机学院	
班级	1636101		学号	1160300617	
任课教师	李全龙		指导教师	李全龙	
实验地点	格物 213		实验时间	2018 年 10 月 27 日	
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：

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） 初始化Server

使用java的ServerSocket，在相应端口初始化，并调用accpet()函数等待客户端发请求。

关键代码：

```
public HttpProxy() throws IOException {
    serverSocket = new ServerSocket(port);

while (true) {
    Socket socket = null;
    try {
        socket = serverSocket.accept();
```

（2） 多线程建立，处理客户端请求

使用java.util.concurrent 包提供的线程池，根据CPU数量创建固定工作线程数目的线程池，服务器接收到客户端请求时，将子线程加入线程池去执行。

关键代码：

```
//根据CPU数目创建线程池
executorService = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors() * POOL_SIZE);

executorService.execute(new ClientHandle(socket, count++));
```

（3） 用户过滤功能

通过socket.getInetAddress()判断请求的IP地址是否在禁止的IP地址列表中，若在，返回403 Forbidden，并返回相应html数据，结束该进程。

代码：

```
// 进行用户过滤
private boolean userFilter() throws IOException {

    if (ForbiddenUsers.contains(socket.getInetAddress().toString())) {
        System.err.println("Current user is forbidden to access!" + socket.getInetAddress());
        sos.write("HTTP/1.1 403 Forbidden\r\n\r\n".getBytes());
        String HtmlUserForbidden = "<!DOCTYPE html><html><head><meta charset=\"utf-8\"><title>403 F
                + "<body><h1 align=\"center\">403 Forbidden</h1><p align=\"center\">Current user is

        sos.write(HtmlUserForbidden.getBytes());
        sos.flush();
        return true;
    }
    return false;
}
```

（4） 解析端口号和主机

若用户过滤未成功，则通过客户端请求的HTTP报文的请求行解析出主机名称以及访问服务器的端口号。

代码：

```
// 从请求行中解析主机和端口号
// 解析主机和端口号
private void parseHostAndPort(String requestLine) {
    // e.g. GET http://www.baidu.com/ HTTP/1.1
    String[] tokens = requestLine.split(" ");
    type = tokens[0];
    host = tokens[1];
    int index = host.indexOf("//");
    if (index != -1) {
        host = host.substring(index + 2);
    }
    index = host.indexOf("/");
    if (index != -1) {
        host = host.substring(0, index);
    }
    index = host.indexOf(":");
    if (index != -1) {
        port = Integer.parseInt(host.substring(index + 1));
        host = host.substring(0, index);
    }
```

注：在这里写的太复杂了，后来了解到使用 java.net.url 类可直接进行信息获取

（5） 屏蔽网站、钓鱼等功能

通过解析出的主机名称判断是否进行屏蔽或者钓鱼。若被屏蔽，则写回403 Forbidden等消息。若被钓鱼，则构建发向钓鱼网站的请求报文，在发送数据时使用。

代码：

```
// 进行网站过滤，过滤成功返回true
// 屏蔽某些网站
private boolean netFilter() throws IOException {
    if (ForbiddenHosts.contains(host)) {
        System.err.println("This site is forbidden to visit: " + host);
        sos.write("HTTP/1.1 403 Forbidden\r\n\r\n".getBytes());
        String HtmlNetForbidden = "<!DOCTYPE html><html><head><meta charset=\"
                + "<body><h1 align=\"center\">403 Forbidden</h1><p align=\"cen
        sos.write(HtmlNetForbidden.getBytes());
        return true;
    }
    return false;
}
```

```
// 将要访问的网页引导向另一个网页，成功引导返回true
// 钓鱼
private void Fishing() throws IOException {
    if (!FishingHosts.contains(host)) {
        fishSuccess = false;            // 钓鱼失败，置全局变量为false
        return;
    }
    fishSuccess = true;     //钓鱼成功，置全局变量为true
    System.err.println("Fishing success! From " + host + " to http://nstool.netease.com/");
    host = "nstool.netease.com";        //修改host
    StringBuffer sb = new StringBuffer();   //构建钓鱼使用的报文
    String URL = "http://nstool.netease.com/";
    sb.append("GET " + URL + " HTTP/1.1\r\n");
    sb.append("Accept: */*\r\n");
    sb.append("Accept-Encoding: gzip, deflate, br\r\n");
    sb.append("Accept-Language: zh-Hans-CN, zh-Hans; q=0.5\r\n");
    sb.append("Cache-Control: max-age=0\r\n");
    sb.append("Connection: Keep-Alive\r\n");
    sb.append("Host: nstool.netease.com\r\n");
    sb.append("Upgrade-Insecure-Requests: 1\r\n");
    sb.append(
            "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
    fishHeader = sb.toString();
    return;
}
```

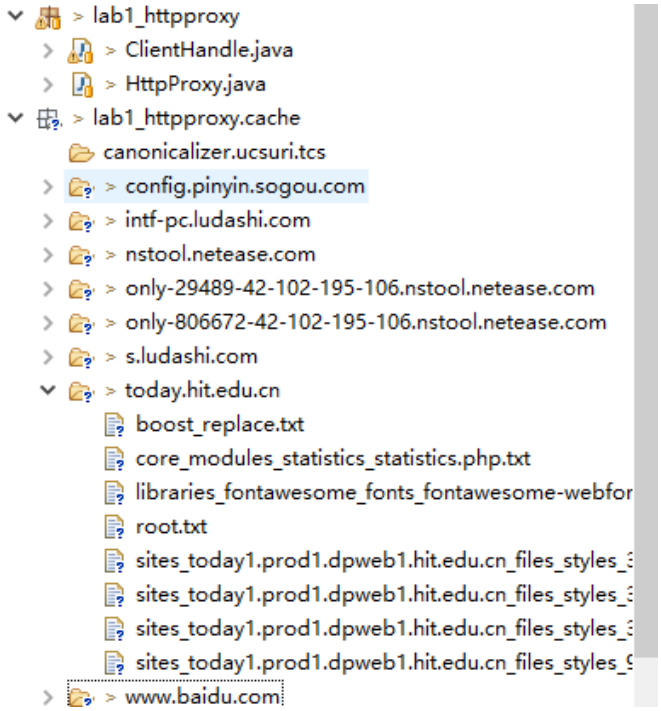
（6） 寻找本地缓存是否存在

通过URL的解析，将指定请求的响应报文存储在指定文件夹的文件中，通过判断该文件是否存在即可判断是否有缓存。有缓存的情况下，从中取出响应报文的最后更新时间。

代码：

```
// 寻找本地是否存在缓存，通过相应文件是否存在来判断
private void FindCache() throws IOException {
    foldName = "src/lab1_httpproxy/cache/" + url.getHost() + "/";
    fileName = url.getPath().replace("/", "_").substring(1) + ".txt";
    if (fileName.equals(".txt")) {
        fileName = "root.txt";
    }
    File fold = new File(foldName);
    //如果文件夹不存在则创建
    if (!fold.exists()) {
        fold.mkdir();
        System.err.println("Cache does not exist.");
        return;
    } else {
        File file = new File(foldName + fileName);
        if (!file.exists()) {
            file.createNewFile();    //如果文件不存在，创建
            System.err.println("Cache does not exist!");
            return;
        }
        foundCache = true;       //该文件存在，代表有缓存
        System.err.println("Find Cache");
        // 从缓存中获得最新时间
        BufferedReader reader = new BufferedReader(new FileReader(file));
        String line = null;
        while((line = reader.readLine())!=null) {
            if (line.contains("Date")) {
                ifModifiedSince = line.trim().substring(line.indexOf(" ")+1);
                  System.err.println("-----------------" + ifModifiedSince);
            }
        }
        reader.close();
        return;
    }
}
```

缓存截图：



（7） 向目标服务器发送请求

代理服务器构建发向目标服务器的socket，在发送数据时，判断是否钓鱼与有缓存。若钓鱼成功，则发送钓鱼函数构建的相应请求报文。若存在缓存，则修改客户端的请求报文，添加If-Modified-Since字段。

代码：

```
// 向真正的服务器发送请求
private void sendToRealServer() throws IOException {
    cSocket = new Socket(host, port); // 代理服务器作为客户端，向真正的服务器发送请求
    cSocket.setSoTimeout(TIMEOUT); // 设置超时
    cos = cSocket.getOutputStream();
    cis = cSocket.getInputStream();
    System.out.println("Request Information:  "); // 正常访问
    if (fishSuccess) {       //钓鱼成功，则发送指定的HTTP 请求报文
        System.out.println(fishHeader);
        cos.write(fishHeader.getBytes());
        cos.flush();
        return;
    }
    StringBuffer sb = new StringBuffer(); // 从客户端接收报文发给真服务器
    String line = requestLine;
    while (!line.isEmpty()) {
        sb.append(line + "\r\n");
        line = sbr.readLine();
    }
    if (foundCache) {    //如果cache存在，修改请求报文，添加If-Modified-Since字段
        sb.append("If-Modified-Since: "+ ifModifiedSince +"\r\n");
    }
    sb.append("\r\n");
    System.out.println(sb.toString());
    cos.write(sb.toString().getBytes());
    cos.flush();
}
```

（8） 获得目标服务器的数据发送给用户

代理服务器将数据发给客户端，其中需要判断本地缓存是否最新的，若是，服务器会返回报文 http 304 Not Modified，此时需要将本地的缓存返回给客户端。若不是最新的，则将服务器的返回信息给客户端。

```
// 获得目标服务器的数据返回给用户
private void returnDataToClient() throws IOException {
    File file = new File(foldName + fileName);
    FileOutputStream fpos = new FileOutputStream(file);
    boolean cacheIsNew = false;
    byte[] buff = new byte[1024];
    int len = -1;
    while ((len = cis.read(buff)) != -1) {  // 从真服务器获取数据发给客户端
        if (new String(buff).contains("304 Not Modified")) {     //如果服务端返回304，表示本地 cache 是新的
            cacheIsNew = true;
            break;
        }
        sos.write(buff, 0, len);
        fpos.write(buff);
    }
    sos.flush();
    fpos.close();
    if (cacheIsNew) {    //缓存是最新的，直接将本地数据返回给客户端
        InputStream fis = new FileInputStream(file);
        System.err.println("The cache is up to data!Return local data!");
        while ((len = fis.read(buff)) != -1) {
            sos.write(buff, 0, len);
        }
        sos.flush();
        fis.close();
    }
}
```

（9） 线程执行的总过程

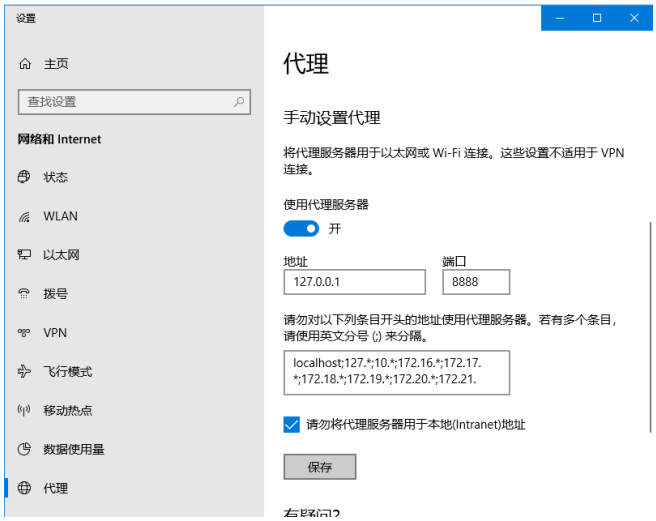
其顺序即按照上文的说明过程进行，当成功把信息交给客户端时，该线程结束。

```
public void run() {
    try {
        System.err.println("Socket " + order + " accepted: " + socket.getInetAddress()

        if (userFilter()) {
            return; // 进行用户过滤
        }

        requestLine = sbr.readLine();
        parseHostAndPort(requestLine); // 获取端口号和主机

        if (netFilter()) {
            return; // 屏蔽某些网站
        }

        Fishing();      //钓鱼
        FindCache();    //寻找缓存
        sendToRealServer(); // 向真正的服务器发出请求
        returnDataToClient(); // 获得目标服务器的数据返回给用户

    } catch (IOException e) {
        System.out.println(e.getMessage());
    } finally {
        try {
```

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）设置代理服务器



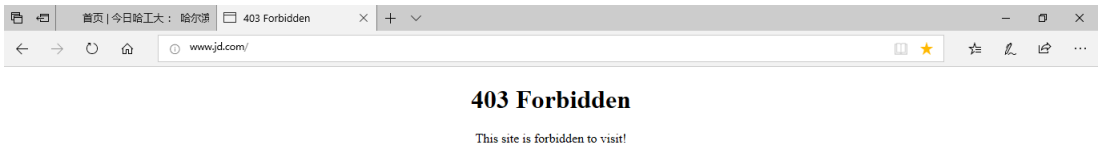
（2）访问 http:today.hit.edu.cn

可以看到浏览器成功访问 http 网站，同时在控制台也做了信息打印，注意第一次访问，打印出了缓存不存在，且请求报文中没有 If-Modified-Since 字段：

```
Socket 10 accepted: /127.0.0.1:60217
Socket 9 accepted: /127.0.0.1:60216
Socket 8 accepted: /127.0.0.1:60215
Cache does not exist!
Cache does not exist!
Cache does not exist!
Cache does not exist!
Request Information:
GET http://today.hit.edu.cn/sites/today1.prod1.dpweb1.hit.edu.cn/files/styles/355x220/pub
Referer: http://today.hit.edu.cn/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec
Accept: image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Language: zh-Hans-CN,zh-Hans;q=0.5
Accept-Encoding: gzip, deflate
If-Modified-Since: Fri, 02 Nov 2018 13:30:52 GMT
If-None-Match: "1b84b-579ae8b590530"
Host: today.hit.edu.cn
Proxy-Connection: Keep-Alive
Cookie: _ga=GA1.3.213772372.1540610207
```

（3）网站屏蔽：

对京东、360 等官网进行了屏蔽：



控制台打印：

```
Socket 36 accepted: /127.0.0.1:62562
This site is forbidden to visit: www.jd.com
```

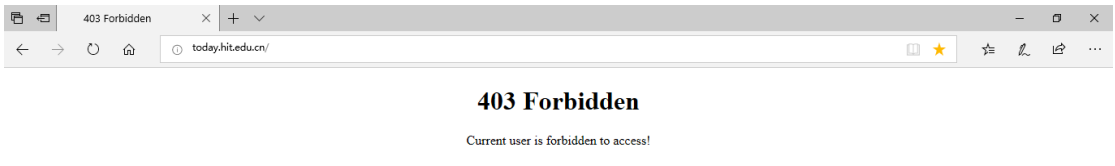
（4）钓鱼

将淘宝、百度等官网引导向网易DNS检测工具

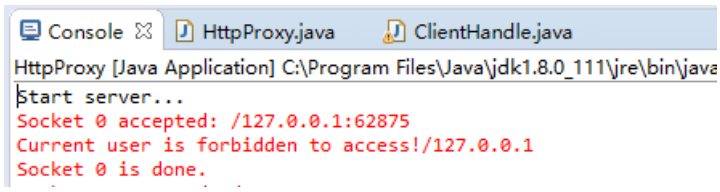
控制台打印：

```
Socket 2 accepted: /127.0.0.1:62849
Fishing success! From www.taobao.com to http://nstool.netease.com/
Request Information:
GET http://nstool.netease.com/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-Hans-CN, zh-Hans; q=0.5
Cache-Control: max-age=0
Connection: Keep-Alive
Host: nstool.netease.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.
```

（5）用户过滤

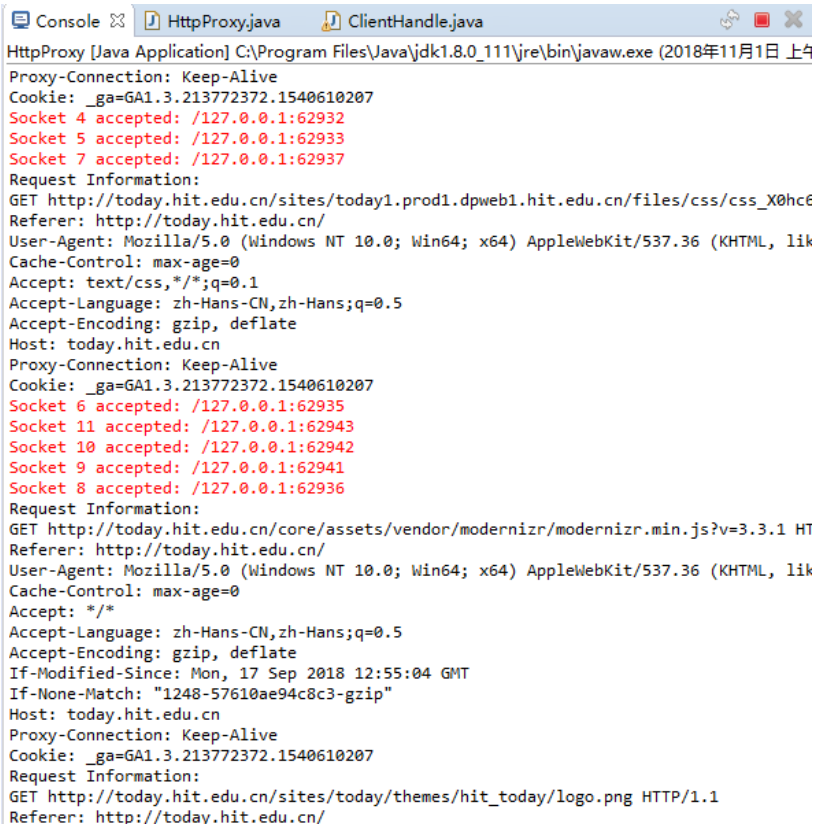


控制台打印：



（6）多线程

控制台打印每个线程的访问信息，红字打印socket 后的序号即为线程序号:



（7）本地缓存

再次访问今日哈工大 http:today.hit.edu.cn，控制台打印的信息是本地缓存是最新的，从本地返回数据给客户端。且可以看到，请求报文的最后都有 If-Modified-Since 字段。

```
The cache is up to data!Return local data!
Socket 7 is done.
The cache is up to data!Return local data!
Socket 6 is done.
Socket 8 accepted: /127.0.0.1:59832
Find Cache
Socket 9 accepted: /127.0.0.1:59834
Find Cache
Socket 10 accepted: /127.0.0.1:59835
Find Cache
Socket 11 accepted: /127.0.0.1:59838
Find Cache
Request Information:
POST http://today.hit.edu.cn/boost/replace?callback=hit_misc.lazy_builder%3AbuildDefaultHeade
Origin: http://today.hit.edu.cn
Referer: http://today.hit.edu.cn/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
Accept-Language: zh-Hans-CN,zh-Hans;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 946
Host: today.hit.edu.cn
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: _ga=GA1.3.213772372.1540610207
If-Modified-Since: Fri, 02 Nov 2018 14:31:07 GMT
```

问题讨论：

对实验过程中的思考问题进行讨论或回答。

（1） Socket编程的客户端和服务器端主要步骤

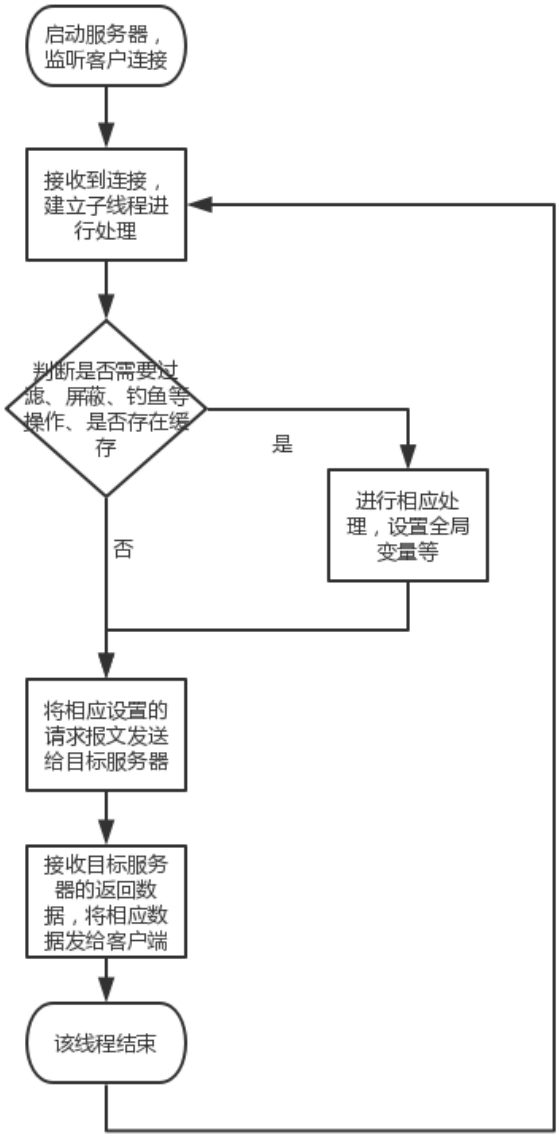
客户端：构建Socket并连接:new Socket(host,port) -> 发送数据报：socket.getOutPutStream.write() -> 接收返回数据: socket.getInputStream() -> 关闭: close()

服务器端： 构建ServerSocket(), 绑定端口: new ServerSocket(port) -> 监听socket请求：socket.accept() -> 建立线程进行处理 -> 关闭 close().

（2） HTTP代理服务器的基本原理

代理服务器允许一个网络终端（一般为客户端）通过这个服务与另一个网络终端（一般为服务器）进行非直接的连接。其在指定端口监听浏览器的访问请求，在自己的缓存中检索URL对应的对象，判断是否为最新缓存。若是最新，则将缓存数据发给客户端。若无缓存，则向原服务器转发请求报文，并将原服务器返回的响应转发给客户端。

（3） HTTP代理服务器的程序流程图



（4） 实现HTTP代理服务器的关键技术及解决方案

1） 多线程建立：

代理服务器应具备多用户同时访问，客户端可同时访问多目标服务器的功能，因此具备多线程处理能力十分重要。本实验采用线程池管理技术，根据CPU处理能力设定线程池最大数量，有效解决该问题。

2） 报文信息的获取和处理

HTTP报文具备固定的格式，在相关信息处理时要特别重视，比如每行的结尾为'\r\n'，请求头和数据之间有空行等，都需要注意。

而且目标服务器端、服务器端、客户端之间传递数据时，用字符串处理很容易发生错误，后来改用了流之间的操作。

3） 缓存技术

将目标服务器的响应报文按照其URL的相关路径建立对应文件夹进行存储。接受到客户请求时，先查询文件是否存在，存在则有缓存，从中提取出If-Modified-Since信息，修改客户端请求报文，以判断是否为最新对象。若是，则直接使用本地缓存返回给客户端，若不是，则将服务器的返回消息发给客户端。

心得体会：

结合实验过程和结果给出实验的体会和收获。

通过本次试验，成功实现了一个简易的HTTP代理服务器，还具备钓鱼、过滤、屏蔽、缓存等功能，这在实验前是想象不到的，想不到代理服务器是这么实现的，也想不到自己也能做一个代理服务器。

计算机网络学了很多网络方面的理论知识，但是一直没有实际化，这次的实际编程让自己受益匪浅，也感受到了网络的美妙。希望自己在以后的实验中也能有所收获。

遗憾的是，实验验收前的那周任务繁忙，没来得及实现缓存功能，现在成功完成，不知道还有没有作用。