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	张儒	院系	软件工程		
班级	2137101	学号	2021112678		
任课教师	李全龙	指导教师	李全龙		
实验地点	格物 207	实验时间	10.16 18:30		
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					

哈尔滨工业大学计算学部
FACULTY OF COMPUTING, HIT

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。

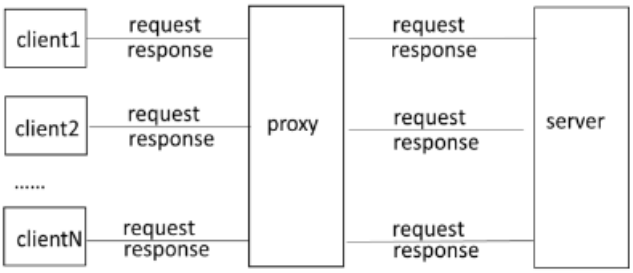
实验内容：

设计并实现一个HTTP代理服务器，要求指定端口接受来自客户的 HTTP 请求并访问原服务器，并接收 HTTP 服务器的响应报文，之后将响应报文转发给对应的客户端。并且代理服务器支持Cache 功能，能缓存原服务器响应的对象，并能够通过修改请求报文，向原服务器确认缓存对象是否是最新版本。并且支持网站过滤、用户过滤、网站引导功能

实验过程：

一、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。



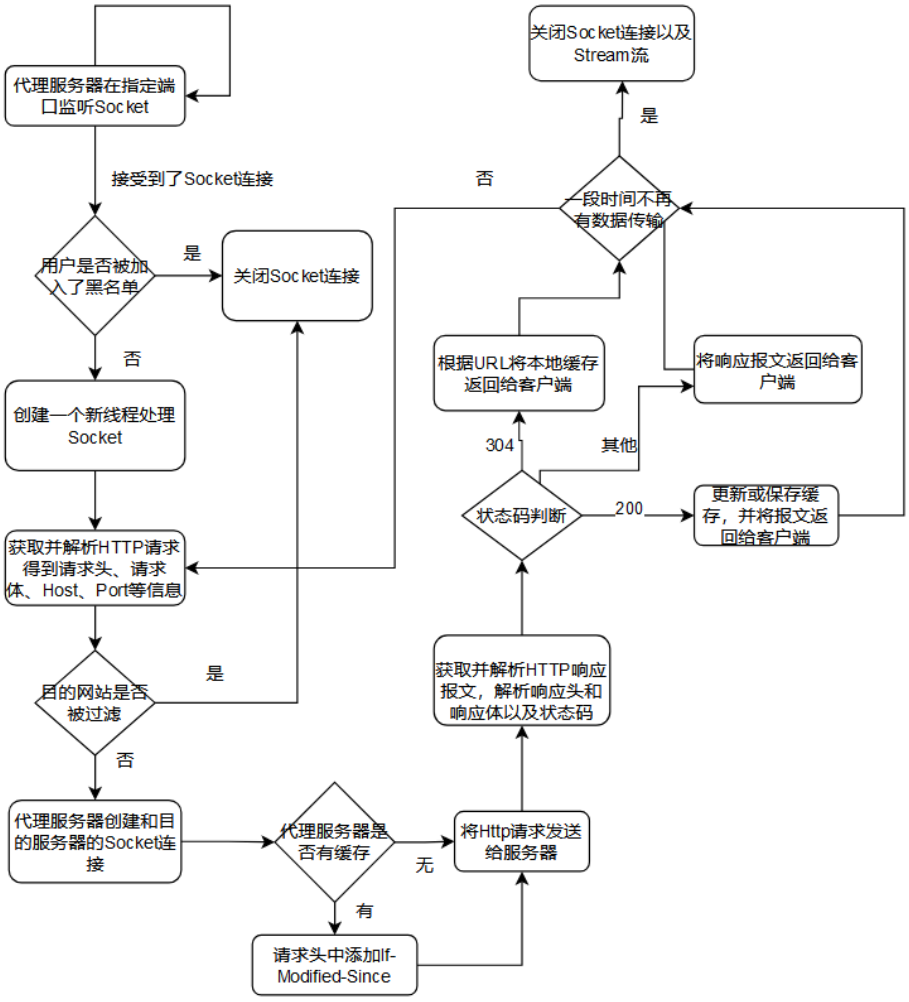
使用代理的B/S

代理服务器在指定端口（例如 8080）监听浏览器的访问请求（需要在客户端浏览器进行相应的设置），接收到浏览器对远程网站的浏览请求时，代理服务器开始在代理服务器的缓存中检索 URL 对应的对象（网页、图像等对象），找到对象文件后，提取该对象文件的最新被修改时间；代理服务器程序在客户的请求报文首部插入<If-Modified-Since: 对象文件的最新被修改时间>，并向原 Web 服务器转发修改后的请求报文。如果代理服务器没有该对象的缓存，则会直接向原服务器转发请求报文，并将原服务器返回的响应直接转发给客户端，同时将对象缓存到代理服务器中。代理服务器程序会根据缓存的时间、大小和提取记录等对缓存进行清理。

二、基本步骤

1. 首先将浏览器的代理服务器设置为127.0.0.1:8080，从而将浏览器的所有请求发送给自己的代理服务器。
2. 首先创建一个ServerSocket，调用其accpet()函数之后，会进入阻塞状态，负责在8080端口进行监听，监听是否有客户端发出的Socket。如果有Socket连接建立成功，先检查用户是否被加入黑名单，如果没有就会创建一个线程去处理Socket。同时原来的线程继续监听。
3. 收到客户端发送的HTTP请求之后，对请求报文进行分析，提取出来目标主机IP和端口号以及URL，以及头部信息。
   - 首先获取Client的InputStream，然后接受从浏览器发送的HTTP数据报，将接收到的数据报首先输入到一个ByteOutPutStream中去，便于操作。
   - 之后通过BufferedReader进行读取数据包的内容，使用正则表达式匹配头部的相应字段，解析出目标服务器的Host以及Port以及URL等头部信息。
4. 检查目的主机IP和端口号是否在黑名单（网站过滤），以及是不是需要进行网站引导的IP（网站引导/钓鱼），然后进行禁止访问、引导操作。

5. 代理服务器根据目的主机IP和端口号创建ProxySocket，先检查本地有没有缓存，如果有在请求头中添加If-Modified-Since字段，并发送给服务器。
   - 根据目的服务器的Host以及Port创建ProxySocket，然后根据解析出来的URL来判断代理服务器中有没有缓存（缓存文件按照URL命名）
   - 如果有缓存，则在请求头添加If-Modified-Since字段，其中时间采用请求头所规定的标准时间。如果没缓存就不做修改。之后将请求头和请求体发送给服务器。
6. 代理服务器接受服务器返回的响应报文，如果状态码为200，说明缓存过期了或者没有缓存。那么就以URL为文件名，将响应报文写入文件。如果状态码为304，说明缓存没有过期，那么就根据文件名查找相应的缓存，然后返回给客户端。
   - 创建一个新的类去处理代理服务器和服务器之间的通。使用ProxySocket的Inputstream来接受HTTP响应报文
   - 解析响应报文，解析出状态码、请求头、请求体等信息
   - 状态码为200，说明缓存过期或者没有缓存。那么就以URL为文件名，将响应报文写入文件。状态码为304，说明缓存没有过期，那么就根据文件名URL查找相应的缓存，然后返回给客户端。其余状态码则将响应报文直接返回给客户端
7. 通信结束后，关闭Socket以及各种Stream流，避免资源浪费
8. 流程图：



实验结果：

1. 首先代理服务器在8080端口就进行监听

```
"C:\Program Files\Java\jdk1.8.0_333\bin\java.exe" ...
创建serverSocket成功!在ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]监听
```

2. 当我们访问一个网页（发送HTTP请求）时，代理服务器会收到请求并缓存转发，页面也可以正常加载


```
"C:\Program Files\Java\jdk1.8.0_333\bin\java.exe" ...
创建serverSocket成功!在ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]监听
检测到HTTP请求: today.hit.edu.cn
保存缓存:httptodayhiteducnarticle20231015107933
检测到HTTP请求: today.hit.edu.cn
保存缓存:httptodayhiteducnsitestoday1prod1dpweb1hiteducnfilescsscssFmqpmwjc6HKFrE7ST9dU3ITMGgMwVKqi9LH0xtFOFccss
connect: Address is invalid on local machine, or port is not valid on remote machine
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTPS请求: safebrowsing.googleapis.com
保存缓存:httptodayhiteducnsitestodaythemeshittodaylogopng
保存缓存:httptodayhiteducnsitestodayfilesinlineimageslogomobilepng
保存缓存:httptodayhiteducnsitestodaythemeshittodaylogomobilepng
保存缓存:httptodayhiteducnsitestoday1prod1dpweb1hiteducnfilesstylesmediumpublicimages201805wechatpngitokAdJT00
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
检测到HTTP请求: today.hit.edu.cn
```

```
httptodayhiteducnboostreplacecallbackccmscorestatisticsbuilder3AbuildCountargs5B05Dnodeargs5B15D107933tokenZOXlibJpkX9
httptodayhiteducnboostreplacecallbackDrupal5CCore5CRender5CElement5CStatusMessages3A3ArenderMessagesargs5B05Dtoken
httptodayhiteducnboostreplacecallbackhitmisclazybuilder3AbuildAdvertiseShowargs5B05D2args5B15D5args5B25D5args5B35D100
httptodayhiteducnboostreplacecallbackhitmisclazybuilder3AbuildDefaultHeadertokenhRBI6B8mBplSkQq7PVQo5M5h23k7BgvZH9J8
httptodayhiteducnboostreplacecallbackhitmisclazybuilder3AbuildErrorCommentargs5B05D107933tokencOfpmohBPQsK7qPKz52CN
httptodayhiteducnboostreplacecallbackhitmisclazybuilder3AbuildSpaceWebrecommendargs5B05D5args5B15D30args5B25D1args5
httptodayhiteducnboostreplacecallbackhitmisclinkbuilder3Abuildargs5B05Dnodeargs5B15D107933args5B25DcollecttokenPRGAn7I
httptodayhiteducnboostreplacecallbackhitmisclinkbuilder3Abuildargs5B05Dnodeargs5B15D107933args5B25DrecommendtokenWF
httptodayhiteducncoremodulesstatisticsstatisticsphp
httptodayhiteducncorethemesclassyimagesiconsxofficespreadsheetpng
httptodayhiteducnsitestoday1prod1dpweb1hiteducnfilescssscssFmqpmwjc6HKFrE7ST9dU3ITMGgMwVKqi9LH0xtFOFccss
httptodayhiteducnsitestoday1prod1dpweb1hiteducnfilesimages20190410zqywjpg
httptodayhiteducnsitestoday1prod1dpweb1hiteducnfilesstylesmediumpublicimages201805wechatpngitokAdJT00
httptodayhiteducnsitestodayfilesinlineimageslogomobilepng
httptodayhiteducnsitestodaythemeshittodaylogomobilepng
httptodayhiteducnsitestodaythemeshittodaylogopng
httptodayhiteducnthemescustomhitfrontimagesheaderbgpng
httptodayhiteducnthemescustomhitfrontimageslinebgpng
httptodayhiteducnthemescustomhitfrontimagespluspng
httptodayhiteducnthemescustomhitfrontjspluginsslickajaxloadergif
```

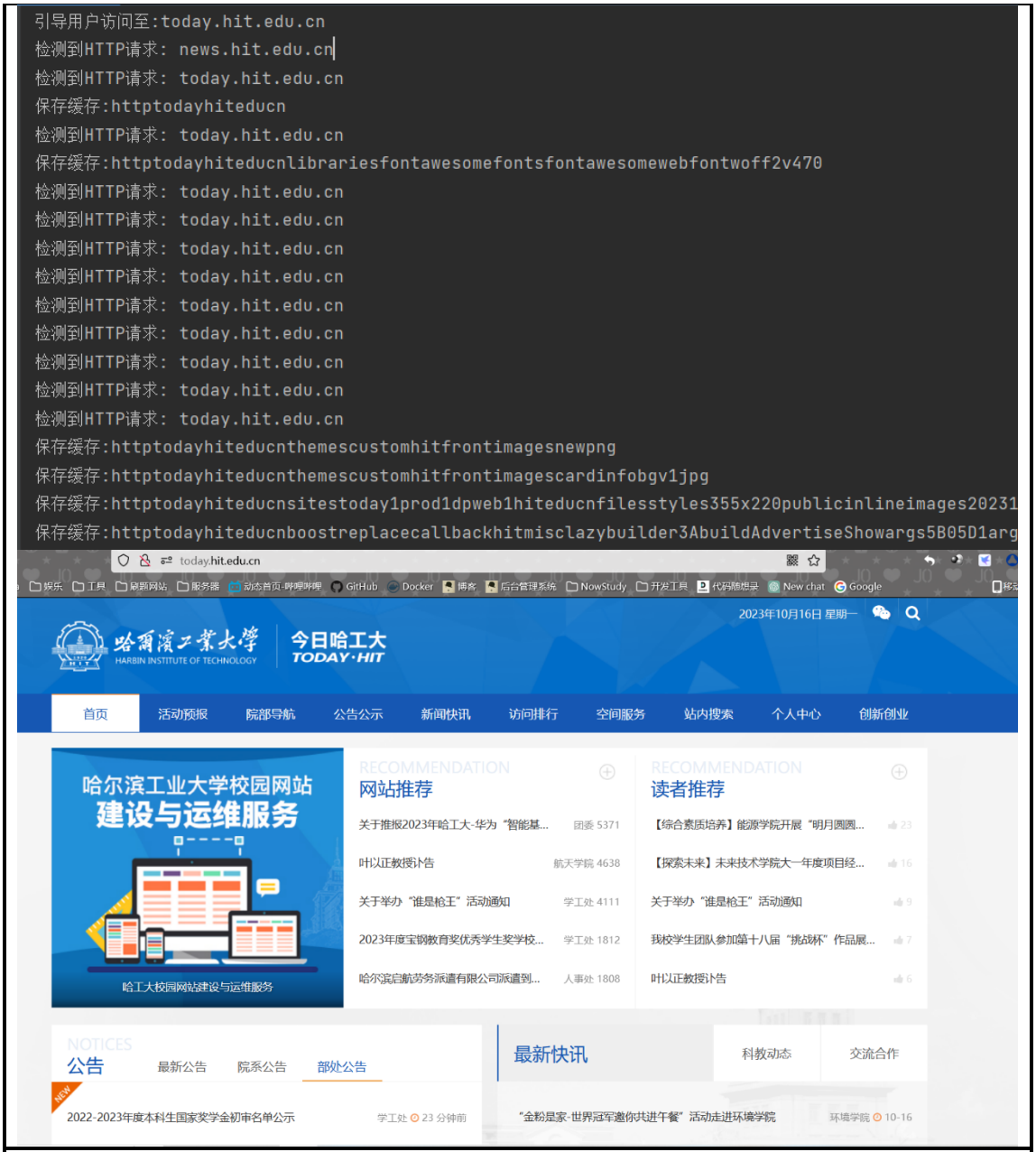
3．如果用户访问了被过滤的网站，代理服务器不允许其访问

```
创建serverSocket成功!在ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]监听
防火墙不允许你访问当前主机!
检测到HTTP请求: today.hit.edu.cn
防火墙不允许你访问当前主机!
检测到HTTP请求: today.hit.edu.cn
防火墙不允许你访问当前主机!
检测到HTTP请求: today.hit.edu.cn
防火墙不允许你访问当前主机!
检测到HTTP请求: today.hit.edu.cn
防火墙不允许你访问当前主机!
检测到HTTP请求: today.hit.edu.cn
防火墙不允许你访问当前主机!
检测到HTTP请求: today.hit.edu.cn
```

4．如果用户（客户端）被禁止访问，被加入了黑名单，那么不允许访问代理服务器

```
创建serverSocket成功!在ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8080]监听
当前用户禁止访问!
```

5．网站引导：当用户访问某个特定网站时，引导至另一个网站



问题讨论：

1.在实现HTTP代理服务器基本功能时,socket接受请求时会一直阻塞。
读取socket的inputStream时,如果读取完毕了，但是不会停止，而是继续等待信息的到来，会阻塞。
查阅资料,发现Socket的inputStream的数据传输与读取文件的不同,因为读取文件时,当读取到文件的结尾时,inputStream会结束,但是Socket的inputSream只有在连接结束时才会结束,所以要想结束读取,我设置了Socket的outTime,当超时时会抛出SocketTimeoutException异常,通过捕捉这个异常就可以退出inputStream的读取。

2. 在确定缓存的时间的时候遇到了一些问题
一开始我是采用文件的修改日期作为If-Modified-Since字段的日期，但是发现这样是不规范的，其实服务器发回来的响应报文里有个Modified-Since字段，里面就会有缓存的最新修改日期，应该以此为准。

心得体会：

通过这次试验，我深刻领悟了 HTTP 协议的原理和结构，同时成功地运用 Socket 编程，为今后的学习和工作中处理计算机网络应用场景打下了坚实的基础。这对我来说至关重要。在试验中，我还实现了 HTTPS 协议的转发，从而对 HTTPS 协议也有了初步了解。此外，通过实现网站屏蔽和用户访问代理服务器的控制功能，我对防火墙的原理有了一定的认识。通过自己动手创建一个钓鱼网站，我学会了如何保护网络免受钓鱼攻击，以及如何确保通信的安全性。在实际生产环境中，使用更安全的 HTTPS 协议是防范网络钓鱼网站出现的有效方式。

代码：

Server.java

```
public class Server {

    private static final int SERVER_PORT = 8080;
    //用户过滤
    private static final HashSet<String> CLIENT_FIREWALL = new HashSet<>();

    static{
        //添加不允许访问的用户
        //CLIENT_FIREWALL.add("127.0.0.1");
    }

    public static void main(String[] args) throws IOException {
        //创建一个 ServerSocket，并在 8080 端口进行监听
        ServerSocket serverSocket = new ServerSocket(SERVER_PORT);
        System.out.println("创建 serverSocket 成功!在" + serverSocket + "监听");
        //一直让其在 8080 端口监听
        while (true) {
            Socket connect = serverSocket.accept();
            String hostName = connect.getInetAddress().getHostName();
            //用户过滤操作
            if(CLIENT_FIREWALL.contains(hostName)){
                System.out.println("当前用户禁止访问!");
            }else {
                HttpProxy httpProxy = new HttpProxy(connect);
                //开启一个新线程来处理 Http 请求
                new Thread(httpProxy).start();

            }
        }
    }
}
```

HttpProxy.java

```
/**
 * 执行连接请求的类
 */
public class HttpProxy implements Runnable {
    //服务端监听到的客户端的请求 Socket
    private Socket clientSocket;
    //request 的信息
    private String strHeader;
    private HashMap<String,String> headers;
    private String body;
    private String method;
    private HostAndPortAndURL hostAndPort;
    //代理 Socket 类
    private Socket proxyClient;
```

```
    //网站过滤防火墙
    private static final HashSet<String> SERVER_FIREWALL = new HashSet<>();

    public HttpProxy(Socket clientSocket) {
        this.clientSocket = clientSocket;
        this.proxyClient  = null;
        headers = new HashMap<>();
        hostAndPort = new HostAndPortAndURL();
        method = null;
        body = null;
        strHeader = null;
        //添加要过滤的网站
        //todo 通过外部文件配置形式
        //SERVER_FIREWALL.add("today.hit.edu.cn");
    }


    //收到连接请求之后，开启线程，执行 run 方法
    @Override
    public void run() {
        //在这个流中，存储从客户端接收到的 HTTP 请求或其他数据。
        ByteArrayOutputStream clientSocketInfo = new ByteArrayOutputStream();
        try {
            //设置超时时间
            clientSocket.setSoTimeout(400);
            //将 socket 中的数据写入到 clientSocketInfo
            storageRequestInfo(clientSocketInfo,clientSocket.getInputStream());
        } catch (SocketTimeoutException e){

        } catch (IOException e) {
            throw new RuntimeException("储存请求信息失败!");
        }

        try {
            //获取 header 以及 body 的信息
            getHeaderAndBodyInfo(clientSocketInfo);
            String targetHost = hostAndPort.getHost();
            int targetPort = hostAndPort.getPort();

            //检查防火墙
            if(SERVER_FIREWALL.contains(targetHost)){
                System.out.println("防火墙不允许你访问当前主机!");
                CloseAllConnect();
            }

            //如果是某一个网站，就钓鱼
            if( hostAndPort.getHost()!=null &&
hostAndPort.getHost().equals("news.hit.edu.cn")){
                System.out.println("引导用户访问至:today.hit.edu.cn");
                String response = "HTTP/1.1 302 Found\r\nLocation: http://today.hit.edu.cn";
                clientSocket.getOutputStream().write(response.getBytes());
                clientSocket.close();
            }

            // 创建代理连接到目标服务器
            proxyClient = new Socket(targetHost, targetPort);

            if (targetPort == 80) {
                System.out.println("检测到 HTTP 请求: " + targetHost);
```

```
                //得到缓存文件的最新修改时间
                String url = hostAndPort.getUrl();
                if(url == null){

proxyClient.getOutputStream().write((strHeader+"\r\n\r\n"+body).getBytes());
                    proxyClient.getOutputStream().flush();
                }
                else {
                    String cacheName = url.replaceAll("[^A-Za-z0-9]", "");
                    File cache = new File("cache/" + cacheName);
                    //如果有缓存，那么就查询是不是最新的
                    if (cache.exists()) {
                        String createCacheTime =
TransferDataWithCache.getCreateCacheTime(cache);
                        // Day, DD Month YYYY HH:MM:SS GMT
                        // 添加 If-Modified-Since 请求头的新请求
                        //If-Modified-Since: Thu, 31 Aug 2023 02:44:38 GMT
                        byte[] newRequest = (strHeader + "If-Modified-Since: " +
createCacheTime + "\r\n\r\n" + body).getBytes();
                        ByteArrayOutputStream a = new ByteArrayOutputStream();
                        a.write(newRequest);
                        proxyClient.getOutputStream().write(a.toByteArray());
                        proxyClient.getOutputStream().flush();
                    } else {
                        proxyClient.getOutputStream().write((strHeader + "\r\n\r\n" +
body).getBytes());
                        proxyClient.getOutputStream().flush();
                    }
                }
            } else if (targetPort == 443) {
                //System.out.println("检测到HTTPS请求，暂不处理....");
                System.out.println("检测到HTTPS请求: " + targetHost);
                //发送特殊响应，告诉客户端，与代理服务器的连接已经成功建立，客户端可以开始进
行TLS/SSL握手。
                //客户端接收到这个响应后，会发送TLS/SSL握手请求，建立一个安全通道，然后通过
该通道进行加密通信。
                //这里是clientSocket!!!
                clientSocket.getOutputStream().write("HTTP/1.1 200 Connection
established\r\n\r\n".getBytes());
                proxyClient.getOutputStream().flush();
            }

            //设置超时时间，避免阻塞
            if (proxyClient != null)
                proxyClient.setSoTimeout(200);

            //数据传输
            TransferData transferData;
            if (targetPort == 80) {
                //缓存版本
                TransferDataWithCache transferDataWithCache = new TransferDataWithCache();
                transferDataWithCache.transfer(clientSocket,proxyClient,hostAndPort);


            } else if (targetPort == 443) {
                transferData = new TransferData();
                transferData.transfer(clientSocket, proxyClient);
            }
```

```
            //关闭连接
            CloseAllConnect();

        } catch (ConnectException e) {
            System.err.println(e.getMessage());
            CloseAllConnect();
        } catch (UnknownHostException e) {
            System.err.println("未知的主机名 " + e.getMessage());
            CloseAllConnect();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }



    /**
     * 将 request 的信息先存储起来，便于我们进行操作
     * @param clientSocketInfo
     * @param inputStream
     * @throws IOException
     */
    private void storageRequestInfo(ByteArrayOutputStream clientSocketInfo, InputStream
inputStream) throws IOException {
        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputStream.read(buffer)) != -1) {
            clientSocketInfo.write(buffer, 0, length);
        }
    }

    /**
     * 拿到请求头的信息
     * @param clientSocketInfo
     * @throws IOException
     */
    private void getHeaderAndBodyInfo(ByteArrayOutputStream clientSocketInfo) throws
IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(new
ByteArrayInputStream(clientSocketInfo.toByteArray())));
        String line;
        boolean isFirst = true;
        StringBuilder header = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            //如果读到了一行空的，就说明 header 读完了
            if (line.isEmpty()) {
                break;
            }
            header.append(line).append("\r\n");
            //处理第一行  GET today.hit.edu.cn:80  HTTP/1.1
            if (isFirst) {
                Pattern urlPattern = Pattern.compile(" (.*?) ");
                Matcher urlMather = urlPattern.matcher(line);
                Pattern methodPattern = Pattern.compile("(CONNECT|GET|PUT|DELETE|POST)");
                Matcher methodMather = methodPattern.matcher(line);
                //获取请求方式
                if (methodMather.find()) {
                    this.method = methodMather.group();
                }
```

```
                if(urlMather.find()){
                    String group = urlMather.group(1);
                    //if(group.length() <200) {
                        this.hostAndPort.setUrl(group);
                    //}
                }
                isFirst = false;
            } else {
                String[] keyAndValue = line.split(": ");
                if (keyAndValue.length == 2) {
                    this.headers.put(keyAndValue[0], keyAndValue[1]);
                }
            }
        }
        strHeader = header.toString();
        //获取 host 和 port
        if(headers.containsKey("Host")){
            String host = headers.get("Host");
            String[] split = host.split(":");
            //HTTPS 请求格式:  www.baidu.com:443
            if(split.length == 2) {
                this.hostAndPort.setHost(split[0]);
                this.hostAndPort.setPort(Integer.parseInt(split[1]));
            }
            //HTTP 请求格式:   Host: hit.edu.cn
            else if (split.length == 1){
                this.hostAndPort.setHost(split[0]);
                this.hostAndPort.setPort(80);
            }
        }
        //获取 body 信息
        StringBuilder body = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            body.append(line);
        }
        this.body = body.toString();
    }


    /**
     * 关闭所有的连接
     */
    private void CloseAllConnect() {
        try {
            if (clientSocket != null && !clientSocket.isClosed())
                clientSocket.close();
            if (proxyClient != null && !proxyClient.isClosed())
                proxyClient.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TransferDataWithCache.java

```
/**
 * 带有缓存的数据传输
 */
public class TransferDataWithCache {
```

```
    private String status;
    private HashMap<String, String> headers;
    private String strHeader;
    private String body;
    private long MaxWaitTime = 8000;

    public TransferDataWithCache() {
        status = null;
        strHeader = null;
        headers = new HashMap<>();
        body = null;
    }

    public void transfer(Socket client, Socket proxyClient, HostAndPortAndURL
hostAndPortAndURL) {
        long LatestDataTransportTime = System.currentTimeMillis();
        try {
            while (proxyClient != null && !(client.isClosed() || proxyClient.isClosed())) {
                //1.proxyClient 接受服务器的返回信息
                ByteArrayOutputStream serverInfo = new ByteArrayOutputStream();
                try {
                    proxyClient.setSoTimeout(200);
                    transferData(serverInfo, proxyClient.getInputStream());
                } catch (SocketTimeoutException e) {

                }
                handleResponse(serverInfo);
                //如果 body 不为空
                if (body != null && !body.isEmpty()) {
                    //2.如果是 HTTP 状态码 200
                    if (status.equals("200")) {
                        String cacheName = hostAndPortAndURL.getUrl().replaceAll("[^A-Za-z0-
9]", "");
                        File cache = new File("cache/" + cacheName);
                        //2.1 缓存过期了，body 里面有资源的最新版本，更新缓存
                        //2.2 更新资源更新时间
                        if (hostAndPortAndURL.getUrl() != null) {
                            //todo 异步处理
                            cache.createNewFile();
                            System.out.println("保存缓存:" + cacheName);
                            setCacheToFile(cache, serverInfo, body);
                        }
                        //2.3 将资源返回给客户端
                        client.getOutputStream().write(serverInfo.toByteArray());
                        client.getOutputStream().flush();
                    }
                    //2.2.如果是 HTTP 状态码 304 Not Modified,那么就返回缓存给客户端即可
                    else if (status.equals("304")) {
                        String cacheName =hostAndPortAndURL.getUrl().replaceAll("[^A-Za-z0-
9]", "");
                        File cacheFile = new File("cache/" + cacheName);
                        FileInputStream fileInputStream = new FileInputStream(cacheFile);
                        ByteArrayOutputStream cacheFromFile =
getCacheFromFile(fileInputStream);
                        //发送给客户端
                        client.getOutputStream().write(cacheFromFile.toByteArray());
                        client.getOutputStream().flush();
                    } else {
                        System.out.println("状态码" + status);
```

```
                        client.getOutputStream().write(serverInfo.toByteArray());
                        client.getOutputStream().flush();
                    }
                    LatestDataTransportTime = System.currentTimeMillis();
                }
                //如果 body 为空的话，直接返回
                else {
                    //如果超时了
                    if (System.currentTimeMillis() - LatestDataTransportTime > MaxWaitTime)
{
                        break;
                    }
                    client.getOutputStream().write(serverInfo.toByteArray());
                    client.getOutputStream().flush();
                }
            }

            client.close();
            proxyClient.close();

        } catch (IOException e) {
            try {
                if (!client.isClosed()) {
                    client.close();
                }
                if (!proxyClient.isClosed()) {
                    proxyClient.close();
                }
            } catch (IOException ignored) {
            }
        }
    }

    /**
     * 获得文件创建的请求头日期
     *
     * @param cache
     * @return
     * @throws IOException
     */
    public static String getCreateCacheTime(File cache) throws IOException {
        Path path = Paths.get(cache.getAbsolutePath());
        BasicFileAttributes attrs = Files.readAttributes(path, BasicFileAttributes.class);
        FileTime fileTime = attrs.lastModifiedTime();
        long millis = fileTime.toMillis();
        SimpleDateFormat dateFormat = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss
'GMT'", Locale.US);
        Date date = new Date();
        date.setTime(millis);
        return dateFormat.format(date);
    }

    /**
     * 将缓存写入到文件中去
     *
     * @param cache
     * @param body
     * @throws IOException
     */
```

```
    private void setCacheToFile(File cache, ByteArrayOutputStream body, String bodyInfo)
throws IOException {
        FileOutputStream fileOutputStream = new FileOutputStream(cache);
        fileOutputStream.write(body.toByteArray());
        fileOutputStream.flush();
        fileOutputStream.close();
    }


    /**
     * 从文件中读取缓存的信息
     *
     * @param fileInputStream
     * @return
     * @throws IOException
     */
    private ByteArrayOutputStream getCacheFromFile(FileInputStream fileInputStream) throws
IOException {
        ByteArrayOutputStream cache = new ByteArrayOutputStream();
        cache.write((strHeader + "\r\n\r\n").getBytes());
        byte[] temp = new byte[1024];
        int length;
        while ((length = fileInputStream.read(temp)) != -1) {
            // bytesRead 包含了实际读取的字节数
            // 将读取的数据转换为字符串并打印
            cache.write(temp, 0, length);
        }
        fileInputStream.close();
        return cache;
    }

    /**
     * 处理 Http 响应
     *
     * @param serverInfo
     * @throws IOException
     */
    private void handleResponse(ByteArrayOutputStream serverInfo) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(new
ByteArrayInputStream(serverInfo.toByteArray())));

        String line;

        boolean isFirst = true;
        StringBuilder header = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            //如果读到了一行空的，就说明 header 读完了
            if (line.isEmpty()) {
                break;
            }
            header.append(line).append("\r\n");
            //处理第一行  HTTP/1.1 状态码  abcd
            if (isFirst) {
                Pattern methodPattern = Pattern.compile("\\d\\d\\d");
                Matcher methodMather = methodPattern.matcher(line);
                //获取返回的状态码
                if (methodMather.find()) {
                    this.status = methodMather.group();
                }
```

```
                isFirst = false;
            } else {
                String[] keyAndValue = line.split(": ");
                if (keyAndValue.length == 2) {
                    this.headers.put(keyAndValue[0], keyAndValue[1]);
                }
            }
        }
        strHeader = header.toString();
        //获取 body 信息
        StringBuilder body = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            body.append(line);
        }
        this.body = body.toString();
    }

    /**
     * 从 socket 中读取数据到 data 输出流
     *
     * @param data
     * @throws IOException
     */
    private void transferData(ByteArrayOutputStream data, InputStream inputStream) throws
IOException {
        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputStream.read(buffer)) != -1) {
            data.write(buffer, 0, length);
        }
    }
}
```