哈尔滨工业大学

<<计算机网络>>

实验报告

(2017 年度春季学期)

姓名:	张茗帅
学号:	**1140310606**
学院:	计算机科学与技术学院
教师:	聂兰顺

实验一 HTTP 代理服务器的设计与实现

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。

实验环境：
（1）硬件设备：我的笔记本电脑
（2）开发环境：Windows 10, MyEclipse
（3）开发语言：Java

二、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。

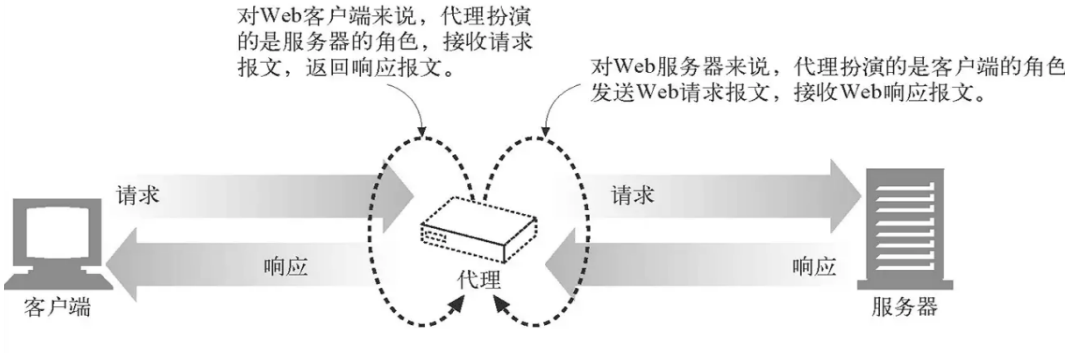
(2) 设计并实现一个支持 Cache 功能的 HTTP 代理服务器。要求能缓存原服务器响应的对象，并能够通过修改请求报文（添加 if-modified-since 头行），向原服务器确认缓存对象是否是最新版本。

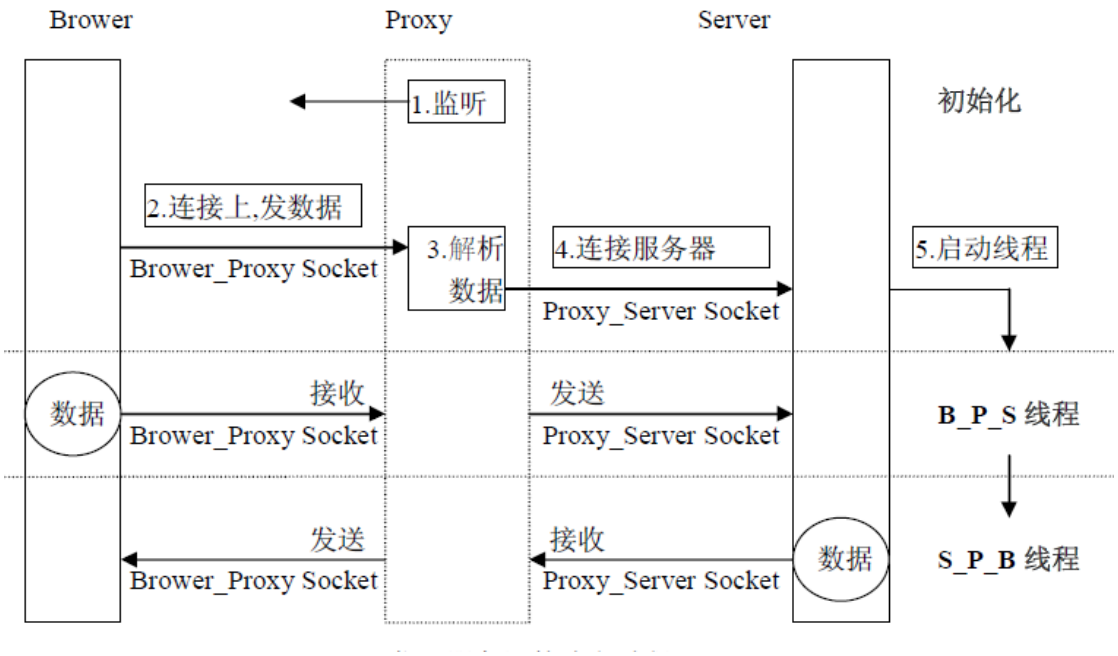
(3) 扩展 HTTP 代理服务器，支持如下功能：

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（钓鱼网站）

三、实验过程及结果

1 实验原理图：





代理服务器的工作原理是：

当客户在浏览器中设置好代理服务器后，你使用浏览器访问所有 WWW 站点的请求都不会直接发给目的主机，而是先发给代理服务器，代理服务器接受了客户的请求以后，由代理服务器向目的主机发出请求，并接受目的主机的数据，存于代理服务器的硬盘中作为缓存，然后再由代理服务器将客户要求的数据发给客户。即再进行报文解析后代理服务器在本地查找缓存，如果有缓存并且没有过期的情况下，直接使用缓存的内容，这样能真正的减少服务器端的负荷，提高客户端的访问速度。

下面我们来详细说明其工作过程：

1. 首先，代理服务器创建HTTP代理服务的TCP主套接字，通过该主套接字监听等待客户端的连接请求初始化结构体套接字。

2. 当客户端连接之后，读取客户端的 HTTP 请求报文，通过请求行中的 URL，解析客户期望访问的原服务器 IP 地址。然后根据这个 IP 地址在代理服务器在本地查找缓存，如果缓存未过期则，直接使用缓存的内容，组成报文回发给客户端，否则进行下一步。

3. 创建访问原（目标）服务器的TCP套接字，将HTTP请求报文转发给目标服务器，

4. 接收目标服务器的响应报文，当收到响应报文之后，将响应报文转发给客户端，同时将该响应报文存储在代理服务器的缓存中。

5. 最后关闭套接字，等待下一次连接。

2 Socket 编程的客户端和服务器端主要步骤

服务器端：

1. 使用 socket 函数创建一个 socket 描述符，来唯一标识一个 socket。
2. 使用 bind 函数绑定 IP 地址，端口信息等。
3. 使用 listen 函数进行监听创建的 socket。
4. 使用 accept 函数接收请求，此时 socket 连接也就建立好了。
5. 使用 read(),write()等函数调用网络 I/O 进行读写操作，来实现网络中不同进程

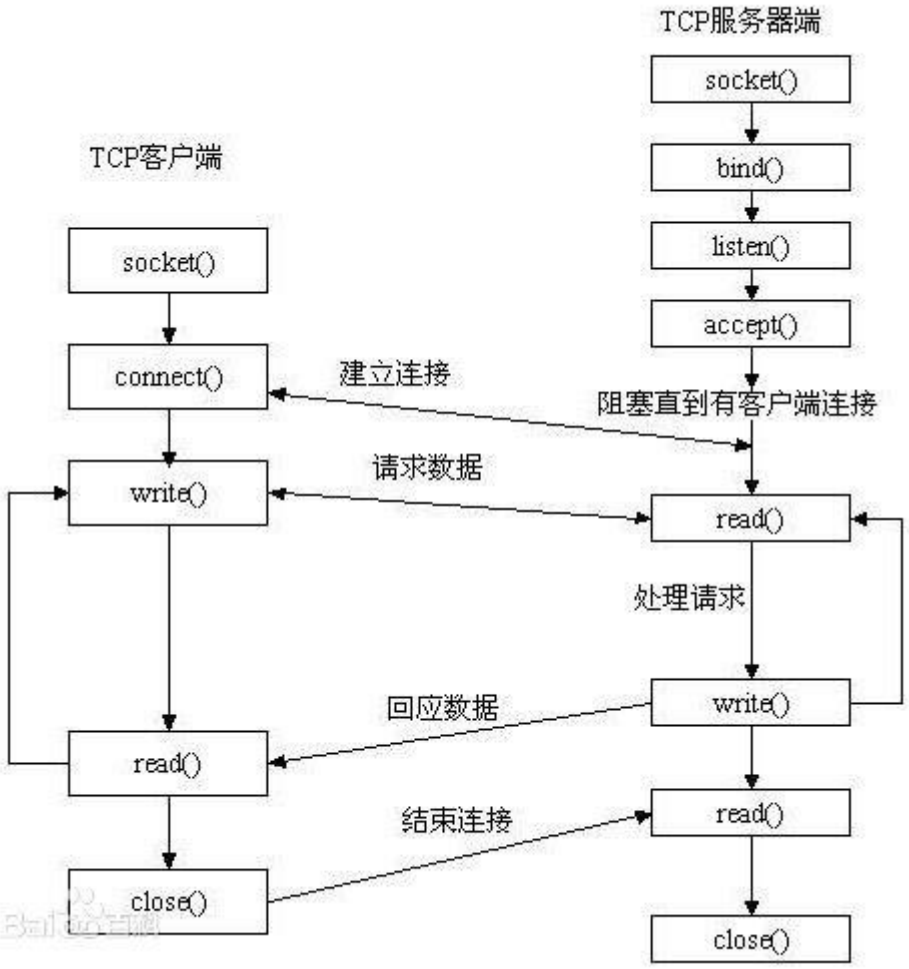
之间的通信

6. 使用 close 函数关闭网络连接，即关系相应的 socket 描述字。

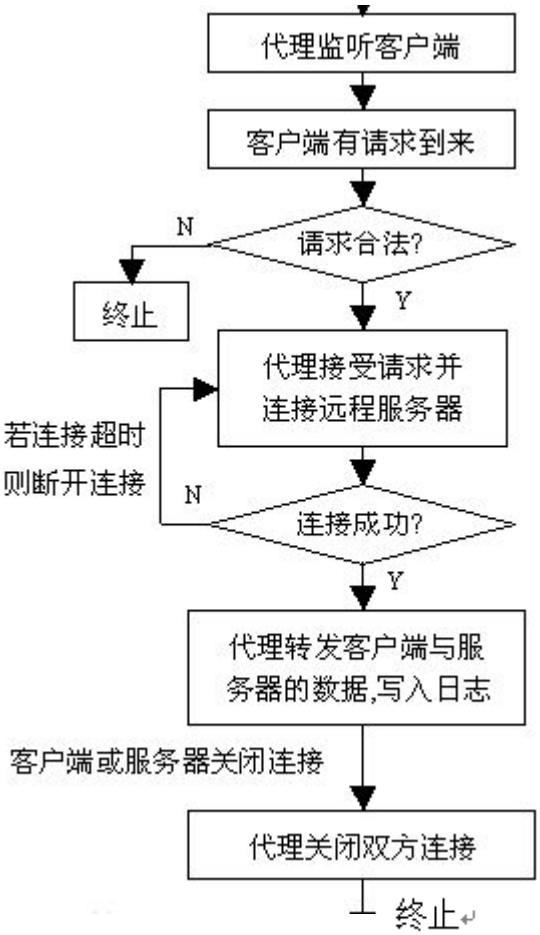
客户端：

1. 创建 socket
2. 绑定 IP 地址，端口信息
3. 设置要连接对方的 IP 地址和端口属性
4. 使用 connect 函数连接服务器。
5. 使用 read(),write()等函数进行网络 I/O 的读写
6. 关闭网络连接

3 服务器端和客户端 Socket 编程的流程图如下：



4 HTTP 代理服务器的程序流程图



5 实现 HTTP 代理服务器的关键技术及解决方案

1. Socket 编程

   HTTP 的请求都是遵循相应的标准格式，因此我们针对标准的格式要进行正确的处理，来得到正确结果。

2. 代理流程的正确组织

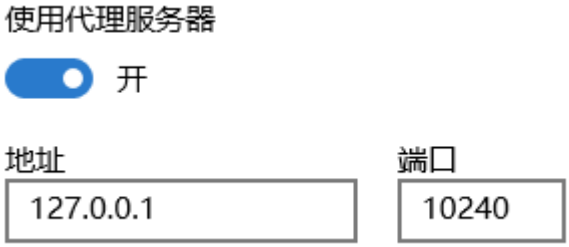
   整个代理服务器中，相关的流程不能出现差错。即必须遵循这样的一个流程：先建立相关的 socket 连接，然后真正的客户端首先去请求代理服务器，代理服务器去解析请求的内容，然后由他像真正的服务器发出 connect 的 socket 连接，真正的服务器去响应代理服务器的这个请求，然后做出响应发送给代理服务器，然后代理服务器再发送给真正的服务器，这样就完成了一次 HTTP 的请求与响应。

3. HTTP 请求头的正确解析

   HTTP 的请求头的格式是一定的，我们要针对 HTTP 的请求头去正确的分析出服务器的 host，端口号，cookies 等其他信息。

6 实验测试

首先打开代理服务器



然后启动程序，输入 http://www.qq.com/

程序中输出如下

请求信息

```
a client connect
HeaderContent : GET http://www.qq.com/ HTTP/1.1
begin:

GET http://www.qq.com/ HTTP/1.1
Host: www.qq.com
Proxy-Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
Cookie: eas_sid=I1N4J980n8I872S693P6e9s6M4; RK=qIUyQ676em; pgv_pvi=1623507968; tvfe_boss_uuid=70582adc1f9
```

返回信息

HTTP/1.1 200 OK成功~~~200 OK

HTTP/1.1 200 OK成功~~~200 OK

HTTP/1.1 200 OK成功~~~200 OK

HTTP/1.1 200 OK成功~~~200 OK

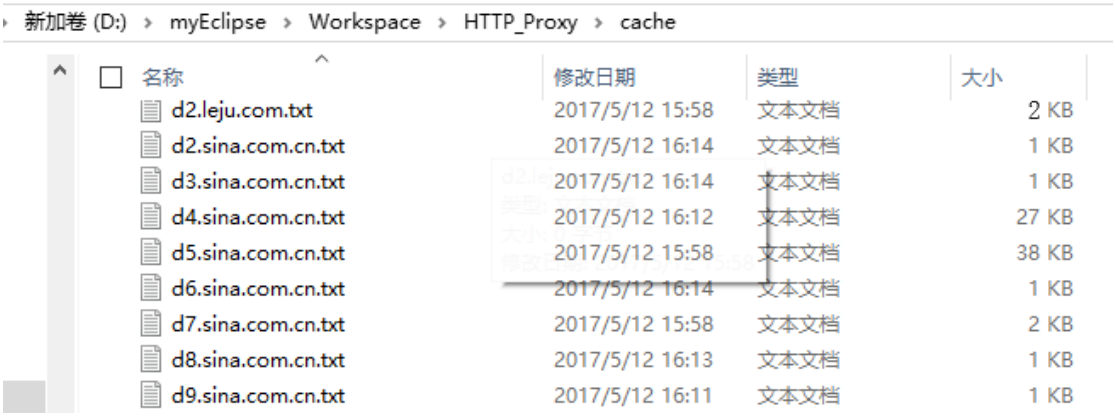
输入http://www.baidu.com/

程序中显示如下：

```
a client connect
HeaderContent : GET http://www.baidu.com/ HTTP/1.1
begin:

GET http://www.baidu.com/ HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: zh-Hans-CN,zh-Hans;q=0.8,en-US;q=0.5,en;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
DNT: 1
Host: www.baidu.com
Proxy-Connection: Keep-Alive
Cookie: BAIDUID=A358791D231AF0AAB2B62E3B27896DCA:FG=1; BIDUPSID=1769577569D0C07E8D8C3C9F2E7D2F91; PSTM=14
```

而打开我的缓存文件夹，发现访问过的网站信息成功存入了对应的本地缓存文件中



设置网站禁止名单，设为 baidu.com

```
14 public class Control {
15     final static String[] gfw_url_list = {
16         "baidu.com"
17     };
```

这个时候发现网页无法正确显示了（之前还能够正确显示百度的）

设置客户 IP 禁止名单，设为本机地址

```
18    final static String[] gfw_ip_list = {
19            "127.0.0.1"
20    };
```

再次访问 www.baidu.com

设置重定向名名单，将 hit.edu.cn 重定向至http://www.sm.cn/

```
21    final static String[] gfw_redirect_list = {
22            "hit.edu.cn"
23    };
```

7 核心模块程序代码（含详细注释）

访问限制控制模块

```
package com.zhangmingshuai.server;

/**
 * CreateDate：2017-5-4 time：02:58:42
 * Location:HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */
import com.zhangmingshuai.bean.HeaderContent;
import java.io.IOException;
import java.io.OutputStream;

public class Control {
    final static String[] gfw_url_list = { //设置网站禁止访问名单
        "baidu.com"
    };
    final static String[] gfw_ip_list = {  //设置禁止的客户端的 IP 地址
           "127.0.0.1"
    };
    final static String[] gfw_redirect_list = {//设置重定向的网站源地址
            "hit.edu.cn"
    };
    final static String redirect = "HTTP/1.1 200 OK\n" +
                                   "Server: nginx\n" +
                                   "Content-Type: text/html\n\n" +
                                   "www.sm.cn";//设置重定向目的地址
    public static boolean a(HeaderContent HeaderContent){  //依次判断
        for(String url:gfw_url_list){
            if(HeaderContent.url.contains(url)){   //是否在禁止名单内
                return false;
            }
        }
        return true;
    }

    public static boolean b(HeaderContent HeaderContent){
        for(String ip:gfw_ip_list){        //是否在禁止名单内
            if(HeaderContent.ip.contains(ip)){
                return false;
            }
        }
        return true;
    }

    public static boolean c(HeaderContent HeaderContent,OutputStream osIn)
throws IOException {
        for(String url:gfw_redirect_list){ //是否在重定向名单内
```

```
                if(HeaderContent.url.contains(url)){
                    byte[] bytes = redirect.getBytes();
                    osIn.write(bytes,0,bytes.length);
                    osIn.flush();
                    return false;
                }
            }
            return true;
        }
    }
```

报文头解析模块

```
package com.zhangmingshuai.server;

/**
 * CreateDate: 2017-5-4 time: 02:31:17
 * Location:HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */
import com.zhangmingshuai.bean.HeaderContent;

public class HeaderParse {
    public static HeaderContent parse(byte[] buffer, int len) {
        String str = new String(buffer, 0, len);
        HeaderContent HC = new HeaderContent();
        //System.out.println("&& Now Line="+str);
//      if (str.charAt(0) == 'C')
//      {
////          return null; // call connect
////          System.out.println("#%$%$&%&^%"+str.substring(0,7));
////          HC.port = 80;
////          str = "GET" + str.substring(7);
//      }
        String[] source = str.split("\\r\\n");

        //HeaderContent HC = new HeaderContent();

        String[] firstLine = source[0].split(" "); //解析获得HTTP请
                  //求头的协议版本号，url，方法，主机，等等全部有用的信息
        HC.method = firstLine[0];
        HC.protocol = firstLine[2];
        HC.url = firstLine[1];
        for (String line : source) {
            //System.out.println("$$$line = "+line);
```

```
            String[] map = line.split(":");
            if (map[0].equals("Host")) {   //按照字符串匹配得到相关信息
                int i = 0;
                if (map[i + 1].startsWith("http"))
                    i++;
                HC.host = map[i + 1].substring(1).replace("/", "");
                if (map.length > i + 2){
                    HC.port = Integer.parseInt(map[i + 2]);
//                  System.out.println("HC.PORT="+HC.port);
                }
            } else if (map[0].equals("Cookie")) {
                HC.cookie = map[1].substring(1);
            }
        }
        HC.bytes = str.getBytes();
        HC.len = HC.bytes.length;
        return HC;
    }
    }
```

对 HTTPS 隧道模式的 CONNECT 特殊处理模块

```
package com.zhangmingshuai.laoxiao;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;

public class ProxyServer { //对 CONNECT 模式进行特殊处理

    public static void  syncStreams(InputStream in,OutputStream out)
    {
        int data;
        int written=0;
        try {
            while( (data=in.read()) != -1)
            {
                if(written==1024)   //输出数据
                {
                    out.flush();
```

```
                    written=0;
                }
//              System.out.print((char)data);
//              System.out.println("sending:"+(char)data);
                out.write(data);
                written++;
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
//          e.printStackTrace();
        }finally{
            try {
                out.flush();
            } catch (Exception e) {
                // TODO Auto-generated catch block
//              e.printStackTrace();
            }
        }
    }

    public static void process(final Socket client,final Socket server)
    {
        Thread t1=new Thread(new Runnable() {   //设置多线程

            @Override
            public void run() {
                // TODO Auto-generated method stub
                try {   //进行同步

ProxyServer.syncStreams(client.getInputStream(),server.getOutputStream());
                } catch (IOException e) {
                    // TODO Auto-generated catch block
//                  e.printStackTrace();
                }
            }
        });
        Thread t2=new Thread(new Runnable() {
```

```
                @Override
                public void run() { //写线程的 run 方法，用于多线程 start 调用
                    // TODO Auto-generated method stub
                    try {

    ProxyServer.syncStreams(server.getInputStream(),client.getOutputStream());
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            });

            t1.start();
            t2.start();
            try {
                t1.join();          //通过 join 对多线程的时序进行控制
                t2.join();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            System.out.println("t1,t2 ended.");
        }
        public static void build2(Socket client,String host,int port)
        {
            System.out.println("host is "+host+" ,port is "+port);
            try {
                Socket server=new Socket(host,port);
                client.getOutputStream().write("HTTP/1.1     200     Connection
Established\r\n\r\n".getBytes());       //往客户端回写 HTTP 相应头
                client.getOutputStream().flush();
                System.out.println("sent 200");
                ProxyServer.process(client, server);
                server.close();
            } catch (Exception e) {
                // TODO Auto-generated catch block
```

```
    //          e.printStackTrace();
                try {
                    System.out.println("sending 500");
                    client.getOutputStream().write("HTTP/1.1    500    Internal
Error\r\n\r\n".getBytes());     //往客户端回写 HTTP 相应头
                    client.getOutputStream().flush();
                } catch (IOException e1) {
                    // TODO Auto-generated catch block
    //              e1.printStackTrace();
                }
            }

            try {
                client.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
    //          e.printStackTrace();
            }
        }
        public static void buildConnection(Socket client,String data)
        {                   //建立连接
            System.out.println("request data is :"+data);
            String host=null;
            int port=443;
            int i="CONNECT".length();
            int ihostend=data.indexOf(":");
            if(ihostend<0)
            {
                host=data.substring(i+1,data.indexOf(" ",i+1));
            }else{              //获取端口和主机号
                host=data.substring(i+1,ihostend);
                port = Integer.valueOf(data.substring(ihostend+1,data.indexOf("
",ihostend+1)));
            }
            ProxyServer.build2(client, host, port);

        }
        public static void main22(String[]args)
```

```
        {
            try {
                ServerSocket socket=new ServerSocket(8080);
                byte[] data=new byte[1500];
                while(true)
                {
                    final Socket client=socket.accept();
                    int len=client.getInputStream().read(data);
                    System.out.println("len is "+len);
                    if(len==-1)continue;
                    final String strdata=new String(data,0,len);
                    if(strdata.startsWith("CONNECT"))
                    {               //建立多线程
                        Thread t1=new Thread(new Runnable() {

                            @Override
                            public void run() {
                                // TODO Auto-generated method stub
                                ProxyServer.buildConnection(client, strdata);
                            }
                        });
                        t1.start();
                    }else{
                        client.getOutputStream().write("HTTP/1.1    404    Not
Found\r\n\r\n".getBytes());     //往客户端回写 HTTP 相应头
                        client.close();
                    }
                }
            } catch (UnknownHostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }
```

```
    }
```

多线程 SOCKET 主模块

```
package com.zhangmingshuai.Socket;

/**
 * CreateDate: 2017-5-4 time: 06:39:44
 * Location:HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */
import com.sun.org.apache.bcel.internal.generic.InstructionConstants.Clinit;
import com.zhangmingshuai.bean.HeaderContent;
import com.zhangmingshuai.laoxiao.ProxyServer;
import com.zhangmingshuai.server.Control;
import com.zhangmingshuai.server.HeaderParse;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;

public class SocketThread extends Thread {
    private Socket socketIn,socketOut;   //代理服务器分别与客户端和服务器端相连
    private InputStream isIn,isOut;      //的Socket
    private OutputStream osIn,osOut;
    private byte[] bytes = new byte[2048];

    public SocketThread(Socket socket) {
        this.socketIn = socket;
    }
    public static void syncStream(InputStream in,OutputStream out)
    {
        int data;

        try{
            while((data=in.read())!=-1)
            {
                out.write(data);
```

```
                System.out.println("sending--->"+(char)data);
                out.flush();
            }
            out.flush();
        }catch(Exception e){
            try {
                out.flush();
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    }

    public static void processConnect(Socket cclient,Socket sserver)
    {
        final Socket client=cclient;
        final Socket server=sserver;
        Thread t1=new Thread(new Runnable() {
            @Override
            public void run() {
                // TODO Auto-generated method stub
                try {
                    SocketThread.syncStream(client.getInputStream(),
server.getOutputStream());
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
        Thread t2=new Thread(new Runnable() {
            @Override
            public void run() {
//                TODO Auto-generated method stub
                try {
                    SocketThread.syncStream(server.getInputStream(),
client.getOutputStream());
                } catch (IOException e) {
//                    TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });
```

```
        t1.start();
        t2.start();

        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


    }

    public void run() {
        try {
            System.out.println("\n\na client connect ");
            isIn = socketIn.getInputStream();
            osIn = socketIn.getOutputStream();

            int len;
            HeaderContent hc = null;
            if ((len = isIn.read(bytes)) != -1&&len > 0){
                hc = HeaderParse.parse(bytes,len);
                hc.ip = socketIn.getLocalAddress().getHostAddress();
              //System.out.println("%%%ip="+HeaderContent.ip);
            }
            System.out.println("HeaderContent : " + hc.toString()); //HTTP头部
            if(hc.method.equals("CONNECT"))
            {
                ProxyServer.build2(socketIn, hc.host,hc.port);
                return;
            }
//

            if(Control.c(hc,osIn)){
                //System.out.println("!!!sockeout_HOST="+hc.host);
                //System.out.println("!!!sockeout_PORT="+hc.port);
                socketOut = new Socket(hc.host,hc.port);
                isOut = socketOut.getInputStream();
                osOut = socketOut.getOutputStream();
                System.out.println("begin:\n");
                System.out.println(new String(hc.bytes, 0, hc.len));
                System.out.println("end");
```

```
//                System.out.println("***END OF THE HEADERCONTENT***");
                osOut.write(hc.bytes, 0, hc.len);
               osOut.flush();

                if(Control.a(hc)&&Control.b(hc)){   //此部分对Cache进行处理
                    SocketThreadOutput out = new SocketThreadOutput(isIn, osOut);
                    out.start();
                    File my = new File("cache/"+hc.host+".txt");
                    if(my.exists()){
                        System.out.println("Cache Entered");
                        FileInputStream fs = new FileInputStream(my);
                        BufferedReader br= BufferedReader(new
InputStreamReader(fs));
                        String line = null;
                        osIn.write("HTTP/1.0 200 OK\r\n\r\n".getBytes());
                        while((line=br.readLine())!=null){
                            osIn.write(line.getBytes());//从Cache中读取缓存内容
                        }
                        fs.close();

                    }else{
                    SocketThreadInput in = new SocketThreadInput(isOut,
osIn,hc.host);
                    in.start();

//                    out.join();
                    in.join();
                    }
                }
            }
        } catch (Exception e) {
            System.out.println("a client leave?"+e.getMessage());
        } finally {
            try {
                if (socketIn != null) {
                    socketIn.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    /**
     * CreateDate: 2017-5-6上午12:56:42
```

```
     * Location: HIT
     * Author: Zhang Mingshuai
     * TODO
     * return
     */
    private BufferedReader BufferedReader(InputStreamReader inputStreamReader) {
        // TODO Auto-generated method stub
        return null;
    }
    }
```

回发给 client 报文模块

```
    package com.zhangmingshuai.Socket;

    /**
     * CreateDate: 2017-5-4 time: 05:48:01
     * Location:HIT
     * Author: Zhang Mingshuai
     * TODO
     * return
     */
    import java.io.FileOutputStream;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.io.File;

    public class SocketThreadInput extends Thread {
        private InputStream isOut;
        private OutputStream osIn;
        private String url;
        public SocketThreadInput(InputStream isOut, OutputStream osIn, String url)
{
            this.isOut = isOut;
            this.osIn = osIn;
            this.url = url;

        }

        private byte[] buffer = new byte[1024];

        public void run() {
            try {
                int len;
                int flag = 0;
                File my = new File("cache/"+url+".txt");
                if(!my.exists()){       //写入 Cache 缓存
                    System.out.println("Error   happening   when   open   'cache'
file!");
                    my.createNewFile();
```

```
                }
                FileOutputStream fs = new FileOutputStream(my);
                StringBuffer temp = new StringBuffer();
                while ((len = isOut.read(buffer)) != -1) {
                    if (len > 0) {
                       String sb = new String(buffer,0,len);
                       if(-1!=sb.indexOf("\r\n\r\n")&&flag ==0){
                           sb = sb.substring(sb.indexOf("\r\n\r\n")+4);
                           flag = 1;
                       }
                     //  System.out.println("Send to client:"+sb);
                        osIn.write(buffer, 0, len);
                        osIn.flush();
                        if(flag ==1 ){
                           temp.append(sb);
                           temp.append("\r\n");
                        }

                    }
                }
              //  System.out.println("^^SocketInput_TEMP:"+temp+"^^^^^");
                fs.write(temp.toString().getBytes("utf-8"));
                fs.close();
            } catch (Exception e) {
                System.out.println("SocketThreadInput leave");
            }
        }
    }
```

注：下面的代码模块只包含在上面没有给出的核心代码（因为不是核心模块）

(注：此模块在实际代码中并未使用，因为这部分代码与 SocketThread.java 中的一部分代码意义上重复了，写的时候没注意...)

代理服务器发给 Server 相应的 HTTP 报文

```
package com.zhangmingshuai.Socket;

/**
 * CreateDate: 2017-5-5 下午 05:48:18
 * Location: HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */
import java.io.InputStream;
import java.io.OutputStream;
```

```
public class SocketThreadOutput extends Thread {
    private InputStream isIn;
    private OutputStream osOut;

    public SocketThreadOutput(InputStream isIn, OutputStream osOut) {
        this.isIn = isIn;
        this.osOut = osOut;
    }

    private byte[] buffer = new byte[1024];

    public void run() {
        try {
            int len;
            while ((len = isIn.read(buffer)) != -1) {
                if (len > 0) {
                  //  System.out.println(new String(buffer, 0, len));
                    osOut.write(buffer, 0, len);
                    osOut.flush();
                }
            }
            byte[]     bytes     =     new     String("if-modified-since:
lalala").getBytes();
            osOut.write(bytes, 0, bytes.length);
            osOut.flush();
            //System.out.println(new String(bytes, 0, len));
        } catch (Exception e) {
            System.out.println("SocketThreadOutput leave");
        }
    }
}
```

创建服务器端 Socket 套接字，调用其他子模块

```
package com.zhangmingshuai.server;

/**
 * CreateDate：2017-5-4 time：03:39:35
 * Location:HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */

import com.zhangmingshuai.Socket.SocketThread;
```

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;



public class Server {
    private ServerSocket server=null;

    public Server(int port) throws IOException {
        server=new ServerSocket(port);
        while(true){
            Socket socket=null;
            try{
                socket=server.accept();
                new SocketThread(socket).start();
            }catch(Exception e) {
                System.out.println("Error."+e);
            }
        }
    }

    public void close() throws IOException {
        if(server!=null) server.close();
    }

}
```

用于保存HTTP头部的重要信息

```
package com.zhangmingshuai.bean;

/**
 * CreateDate: 2017-5-4 time: 01:22:59
 * Location: HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */
public class HeaderContent {
    public String method;
    public String host;
    public int port = 80;
```

```
    public String protocol;
    public String cookie;
    public String url;
    public byte[] bytes;
    public int len;
    public String ip;

    public String toString()
    {
        return method + " " +  url + " " + protocol;

    }
}
```

主函数，调用 Server.java，并配置端口

```
package com.zhangmingshuai;

/**
 * CreateDate：2017-5-5 time：07:54:50
 * Location:HIT
 * Author: Zhang Mingshuai
 * TODO
 * return
 */

import com.zhangmingshuai.server.Server;

import java.io.IOException;

public class Main {

    public static void main(String[] args) {
        @SuppressWarnings("unused")
        Server server = null;
        try {
            server = new Server(10240);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

四、实验心得

通过本次实验，我在实践过程中对 HTTP 的请求报文格式和响应报文格式记忆深刻，弥补了课堂上短暂学习过后的忘记效应，同时学到了 TCP 协议在传输数据的流程和方式，熟悉并且掌握了 Java 中对 Socket 网络编程的过程与技术，也再一次提升了自己的代码能力，同时也更清晰地掌握了 HTTP 代理服务器的基本工作原理，掌握了 HTTP 代理服务器设计与编程实现的基本技能。并且也在这基础上了解了浏览器在进行搜索网页过程中，网络所起的作用以及具体的工作原理，数据的传输方式等等，收获颇丰。

与此同时，我在设计网站过滤、网站引导和用户过滤以及 Cache 的过程中，也了解到了很多 Java 语言 scoket 编程中函数的一些用法和技巧，对并且第一次接触 Java 中多线程的设计方法，对多线程的知识有了较为深入的理解和认识。最后本次实验使我对计算机网络产生了巨大的兴趣，渴望在以后的学习和实验中获得更多的知识来不断丰富自己，成为一个对国家和社会有用的人。