**中国地质大学课程设计报告**



|  |  |  |
| --- | --- | --- |
| 课程名称 | 计算机网络课程设计B | |
| 姓 名 | 单凯 | C:\Users\单凯\Desktop\stuPic.jpg |
| 学 院 | 计算机学院 |
| 班 级 | 191134班 |
| 学 号 | 20131004087 |
| 联系电话 | 18186482897 |
| 邮 箱 | 1431173103@qq.com |
| 指导老师 | 陈喆 |
| 报告时间 | 2016年06月15日 |

**目录**

[**一、** **wireshark实验** 1](#_Toc455653871)

[**1.** **TCP协议分析** 1](#_Toc455653872)

[**实验一.** **TCP三次握手wireshark分析** 1](#_Toc455653873)

[**实验二.** **连接释放分析** 2](#_Toc455653874)

[**实验三.** **捕获一个从你的电脑到远程服务器的 TCP 数据** 4](#_Toc455653875)

[**(1)** **操作步骤** 4](#_Toc455653876)

[**(2)** **问题回答** 5](#_Toc455653877)

[**实验四.** **TCP congestion control in action** 6](#_Toc455653878)

[**2.** **ICMP协议分析** 6](#_Toc455653879)

[**实验一.** **ICMP与ping** 7](#_Toc455653880)

[**（1）实验步骤：** 7](#_Toc455653881)

[**（2）回答问题：** 7](#_Toc455653882)

[**实验二：ICMP与traceroute** 8](#_Toc455653883)

[**（1）实验步骤：** 8](#_Toc455653884)

[**（2）回答问题：** 8](#_Toc455653885)

[**3.** **HTTP协议分析** 10](#_Toc455653886)

[**实验一：基本的 HTTP GET/Response 操作** 10](#_Toc455653887)

[**（1）实验步骤：** 10](#_Toc455653888)

[**（2）回答问题：** 10](#_Toc455653889)

[**实验二：HTTP GET/Response 有条件的相互作用** 13](#_Toc455653890)

[**（1）实验步骤：** 13](#_Toc455653891)

[**（2）回答问题：** 14](#_Toc455653892)

[**实验三：得到较长的文档** 15](#_Toc455653893)

[**（1）实验步骤：** 16](#_Toc455653894)

[**（2）回答问题：** 16](#_Toc455653895)

[**实验四：带有内嵌对象的HTML文档** 17](#_Toc455653896)

[**（1）实验步骤：** 17](#_Toc455653897)

[**（2）回答问题：** 17](#_Toc455653898)

[**实验五：HTTP认证** 18](#_Toc455653899)

[**（1）实验步骤：** 18](#_Toc455653900)

[**（2）回答问题：** 18](#_Toc455653901)

[**二、** **Java编程** 19](#_Toc455653902)

[**1.** **MD5加密** 20](#_Toc455653903)

[**(1)** **运行效果** 20](#_Toc455653904)

[**(2)** **流程框图** 20](#_Toc455653905)

[**2.** **SHA1加密** 20](#_Toc455653906)

[**(1)** **运行结果** 21](#_Toc455653907)

[**(2)** **流程框图** 21](#_Toc455653908)

[**(3)** **关键代码** 21](#_Toc455653909)

[**3.** **Des加密** 23](#_Toc455653910)

[**(1)** **运行效果** 23](#_Toc455653911)

[**(2)** **流程框图** 24](#_Toc455653912)

[**加密** 24](#_Toc455653913)

[**解密** 24](#_Toc455653914)

[**(3)** **关键代码** 24](#_Toc455653915)

[**4.** **Tcp的套接字编程** 24](#_Toc455653916)

[**(1)** **Java中流的相关知识：** 24](#_Toc455653917)

[**(2)** **Tcp通信的示意图** 26](#_Toc455653918)

[**(3)** **通过控制台输入进行通信** 26](#_Toc455653919)

[**(4)** **通过tcp传输对象** 26](#_Toc455653920)

[**(5)** **带界面的tcp通信** 27](#_Toc455653921)

[**(6)** **多（客户端）对一（服务器端）的Tcp通信** 27](#_Toc455653922)

[**(7)** **Tcp传输文件** 28](#_Toc455653923)

[**5.** **下载文件** 29](#_Toc455653924)

[**(1)** **下载图片** 29](#_Toc455653925)

[**6.** **Udp套接字编程** 29](#_Toc455653926)

[**(1)** **基本的Udp套接字编程** 29](#_Toc455653927)

[ **基本原理** 29](#_Toc455653928)

[ **运行效果** 30](#_Toc455653929)

[**(2)** **广播** 30](#_Toc455653930)

[ **原理** 30](#_Toc455653931)

[**(3)** **密钥协商** 30](#_Toc455653932)

[ **原理** 30](#_Toc455653933)

[ **运行效果** 31](#_Toc455653934)

[**(4)** **传输对象** 31](#_Toc455653935)

[ **总体思路** 31](#_Toc455653936)

[ **关键代码：** 31](#_Toc455653937)

[**(5)** **传输文件** 31](#_Toc455653938)

[ **总体思路** 31](#_Toc455653939)

[ **流程框图** 31](#_Toc455653940)

[ 关键代码 32](#_Toc455653941)

[**7.** **仿QQ** 32](#_Toc455653942)

[**1)** **仿QQ的登录界面** 32](#_Toc455653943)

[ **意图** 32](#_Toc455653944)

[ **运行效果** 32](#_Toc455653945)

[ **连接mysql数据库** 33](#_Toc455653946)

[ **验证码** 35](#_Toc455653947)

[**(1)** **主要思路** 35](#_Toc455653948)

[**(2)** **流程框图和关键代码** 35](#_Toc455653949)

[**2)** **仿QQ的好友列表以及好友聊天功能（应老师要求）** 35](#_Toc455653950)

[ **运行效果** 35](#_Toc455653951)

[ **JavaGui编程的相关知识** 36](#_Toc455653952)

[ **一般的Java的Gui编程的流程** 36](#_Toc455653953)

[ **Swing基本框架** 37](#_Toc455653954)

[ **卡片布局** 37](#_Toc455653955)

[ **好友列表界面的布局构思** 37](#_Toc455653956)

[ **群聊的服务器端的设计** 38](#_Toc455653957)

[ **好友信息从哪里来？** 38](#_Toc455653958)

[**8.** **前向搜索算法** 38](#_Toc455653959)

[**(1)** **运行效果** 39](#_Toc455653960)

[**(2)** **流程框图** 39](#_Toc455653961)

[**(3)** **关键代码** 39](#_Toc455653962)

[**9.** **玩一下tomcat** 41](#_Toc455653963)

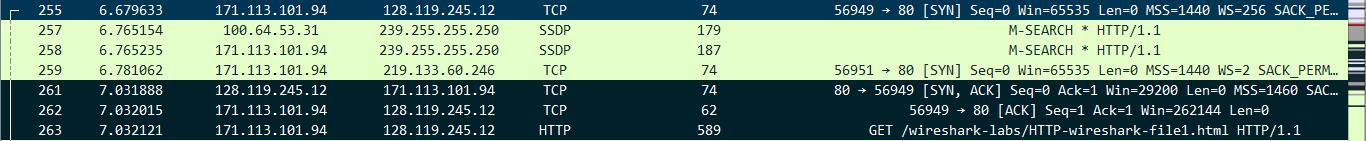
[**三、** **个人感悟** 42](#_Toc455653964)

[**四、** **参考材料** 43](#_Toc455653965)

**wireshark实验**

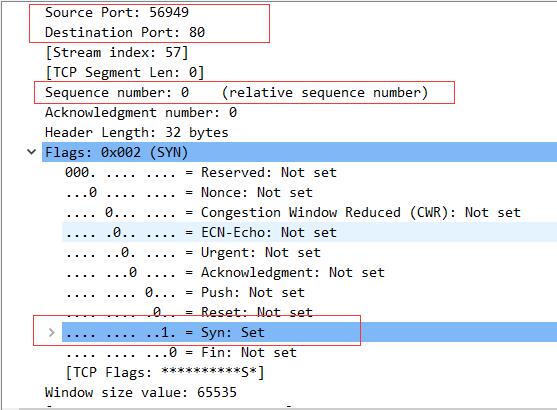
1. **TCP协议分析**
2. **TCP三次握手wireshark分析**

打开wireshark，选择本地连接接口并开始抓包。打开浏览器，自动打开浏览器首页，页面加载完毕，停止抓包。观察封包列表，可以得到三次握手的数据：

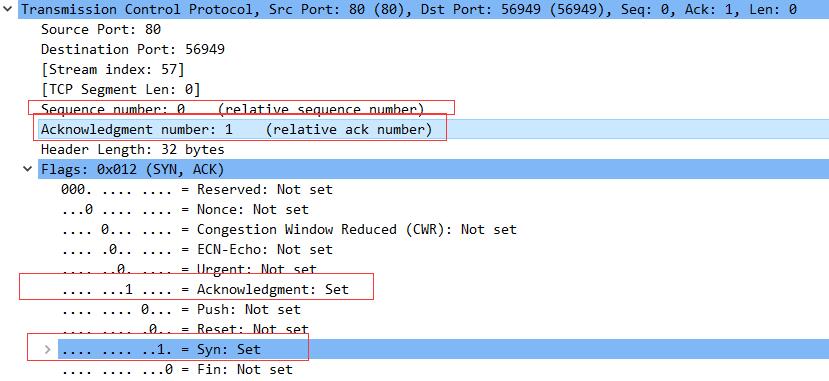


由此我们可以验证HTTP的确是通过TCP建立连接的。

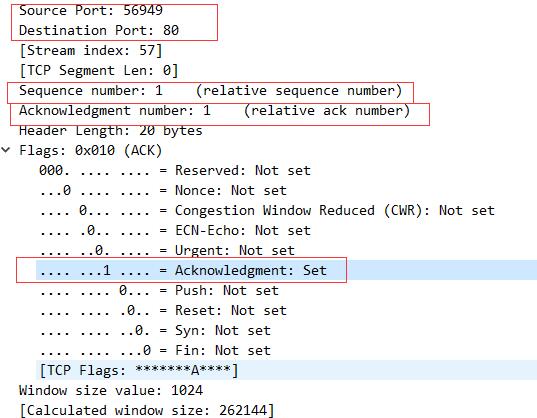
第一次握手数据包，可以看到客户端发送一个TCP，标志位为SYN，序列号为0， 代表客户端请求建立连接。如下图：



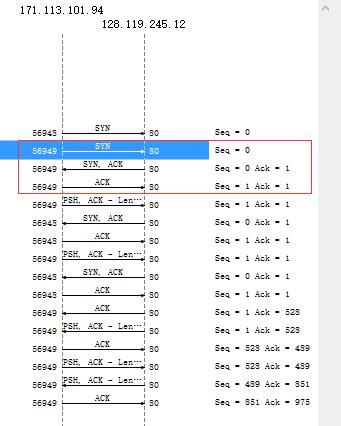
第二次握手数据包，可以看到服务器发回确认包, 标志位为 SYN,ACK. 将确认序号ACK设置为1.如下图：



第三次握手数据包，可以看到客户端再次发送确认包(ACK) ，标志位为ACK，将sequence number+1.如下图：



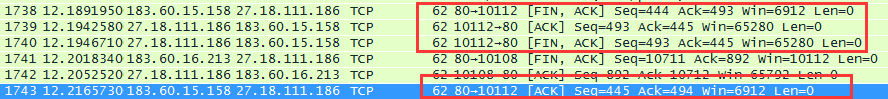
使用wireshark的工具flow graph生成TCP连接的图像，此次三次握手正如下图红框中所示。



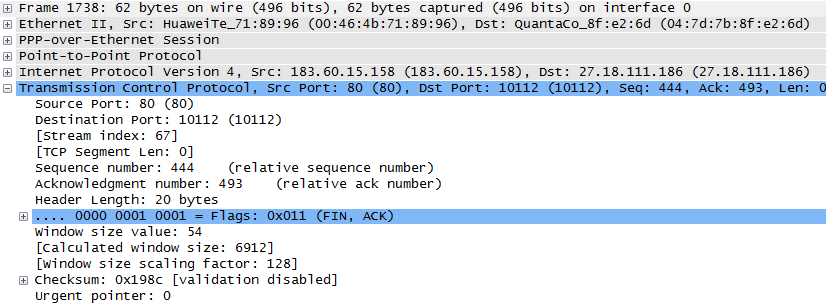
经过上述三次握手过程，即建立了HTTP连接。

1. **连接释放分析**

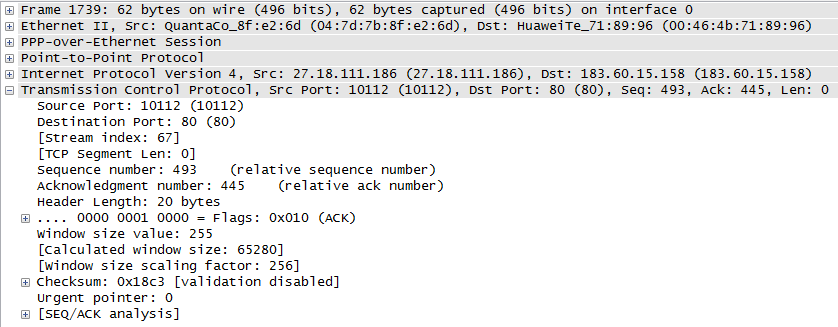
打开wireshark，选择一个接口并开始抓包，打开浏览器浏览网页再关闭，停止抓包。查看所抓取的封包，得到连接释放的封包数据如下：



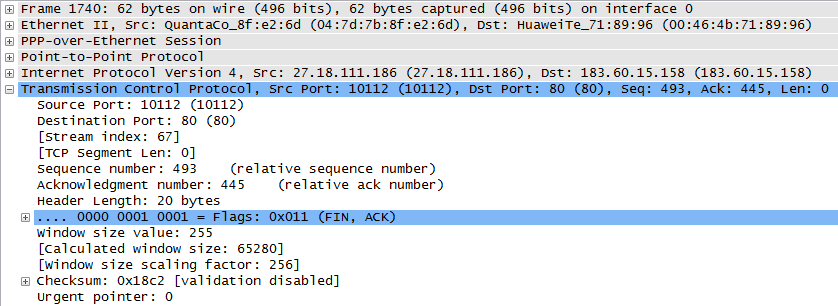
上图验证了TCP四次挥手原理，一一分析这四个TCP封包：

第一次挥手：客户机发送一个Fin=1,Seq=444,ACK=493,请求关闭客户机到服务器的数据传送：

第二次挥手：服务器收到客户端的Fin,它发回一个ACK,确认序号为收到的序号加1，即ACK=444+1=445，SEQ=493：



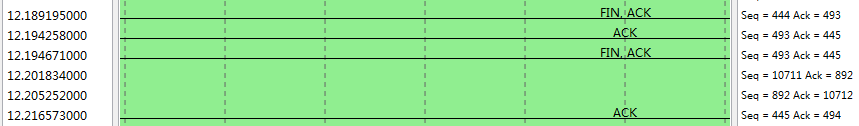
第三次“挥手”：客户机收到服务器发送的一个Fin=1,用来关闭服务器到客户机的数据传送，Seq=493,ACK=445：



第四次“挥手”：客户机收到服务器所发的Fin后，发送一个ACK给Server,确认序号为收到序号+1，ACK=493+1=494,服务器进入关闭状态，完成四次挥手。



使用wireshark的工具flow graph生成TCP连接的图像，此次三次握手正如下图所示（由于箭头太长。无法显示全部，故截取片段展示），由FIN,ACK和左边的时间和右边的SEQ,ACK可以看出此图正是上例的流量图。



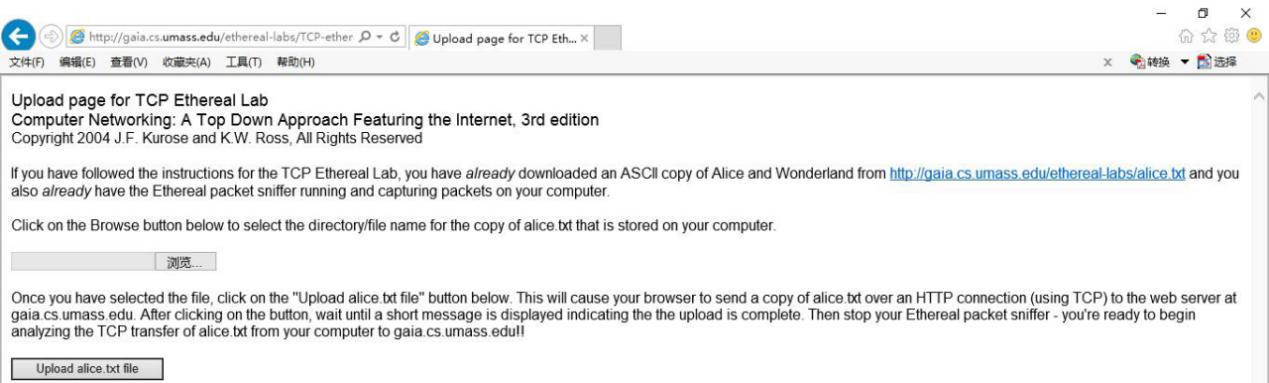
经过上述连接释放过程，即断开了HTTP连接。

1. **捕获一个从你的电脑到远程服务器的 TCP 数据**
2. **操作步骤**

1． 打开你的浏览器。输入：http://gaia.cs.umass.edu/ethereal-labs/alice.txt你将会看到 Alice in Wonderland 的文本版，然后请你暂时另存到你的电脑中；

2． 接下来输入：http://gaia.cs.umass.edu/ethereal-labs/TCP-ethereal-file1.html

3． 你将会看到一个像这样的屏幕：

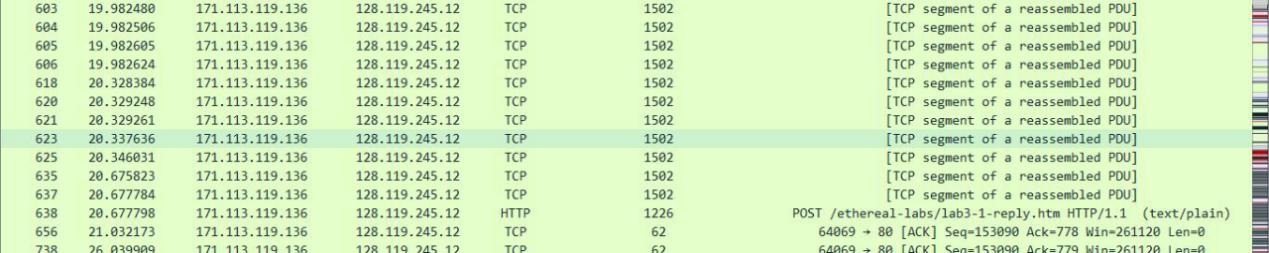


4． 用“浏览…”按钮选取你刚才保存的文件，请先不要按“Upload alice.txt file”按钮；

5． 现在打开 Ethereal，开始包的捕获；

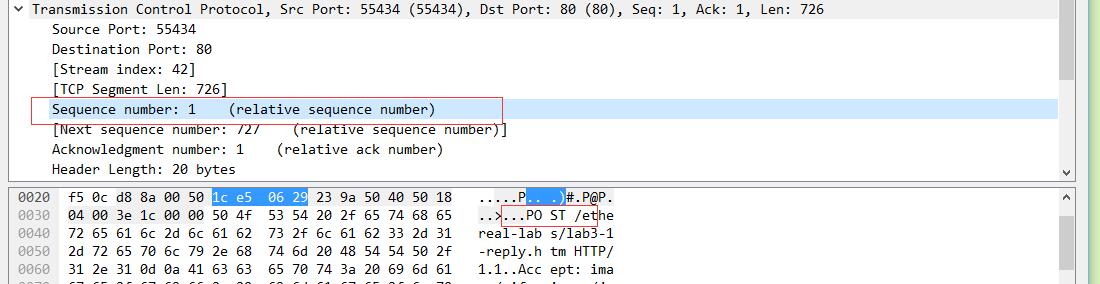
6． 再回到你的浏览器，按下“Upload alice.txt file”按钮向 gaia.cs.umass.edu 服务器来上载文件（一旦这个文件被上载，一个短小的祝词将会显示在你的浏览器窗口）；

7． 停止 Ethereal 的包捕获，你的 Ethereal 窗口应该和下面所示的窗口相类似：



1. **问题回答**

**Q5包含 HTTP POST 命令的 TCP 片段的序列号是多少？记录这个是为了找到 POST命令，你需要研究在 Ethereal 窗口底部的包的内容，寻找一个含有“POST”和它的数据字段的片段。**



序列号是1.

**Q6考虑在 TCP 连接中含有 HTTP POST 并把它作为第一个片段的 TCP 片段。在 TCP连接（包括含有 HTTP POST 的片段）中最先的六个片段的序列号是多少？每一个片段是什么时候发送的？每一个片段接收到ACK是什么时候？请给出每一个TCP片段发送和确认被收到时的间隔，即六个片段中的每一个 RTT 值是多少？当接收到每一个 ACK时的EstimatedRTT值是多少？假设对于第一个片段来说，EstimatedRTT值和标准的RTT值相同。Ethereal 有一个不错的功能，允许你为每一个 TCP 片段发送进行 RTT 划分。在捕获包的列表中选择一个 TCP 片段，它是从客户端发送到 gaia.cs.umass.edu 服务器的。填写下表并回答以下问题：**

前6个报文段为No.11，12，14，15，17，18

序列号为：1，619，2027，3435，4843，6251.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Send time | ACK Received Time | RTT seconds |
| Segment 1 | 2.513203 | 2.799560 | 0.286357 |
| Segment 2 | 2.513393 | 2.803157 | 0.289764 |
| Segment 3 | 2.799582 | 2.803157 | 0.003575 |
| Segment 4 | 2.799592 | 3.088572 | 0.28898 |
| Segment 5 | 2.803168 | 3.088579 | 0.285411 |
| Segment 6 | 2.803176 | 3.088589 | 0.285413 |

EstimatedRTT=0.875\* EstimatedRTT+0.125\*SampleRTT

接受到报文段1之后的EstimatedRTT为：

EstimatedRTT=RTT for segment 1=0.286157 s

接受到报文段2之后的EstimatedRTT为：

EstimatedRTT=0.875\*0.286357+0.125\*289764=0.286783 s

接受到报文段3之后的EstimatedRTT为：

EstimatedRTT=0.875\*0.286783+0.125\*0.003575=0.251382 s

接受到报文段4之后的EstimatedRTT为：

EstimatedRTT=0.875\*0.251382+0.125\*0.28898=0.256082 s

接受到报文段5之后的EstimatedRTT为：

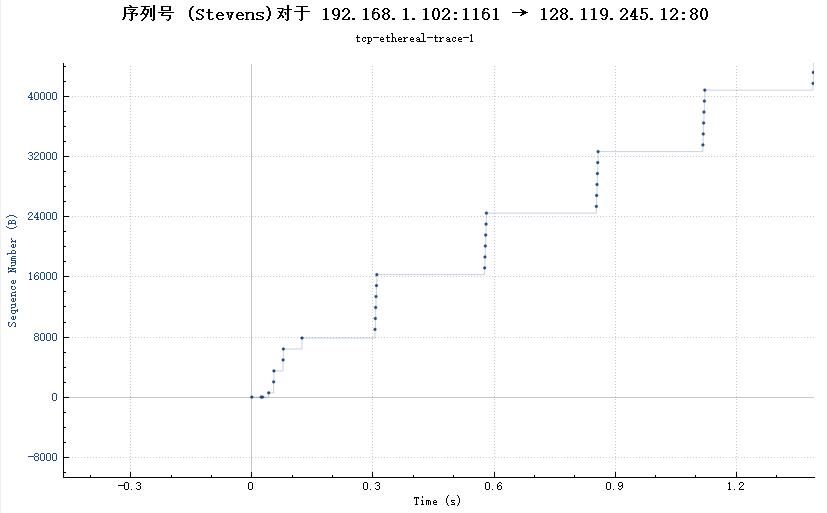
EstimatedRTT=0.875\*0.256082 +0.125\*0.285411= 0.259748 s

接受到报文段6之后的EstimatedRTT为：

EstimatedRTT=0.875\*0.259748+0.125\*0.285413= 0.262956 s

1. **TCP congestion control in action**

**Q13.使用时间序列-图（史蒂文斯）绘图工具来查看从客户端被发送到gaia.cs.umass.edu服务器段的序号与时间的关系曲线图。你能找出TCP的slowstart阶段的开始和结束，并在拥塞避免接管？**

A13.

慢启动阶段即从HTTP POST 报文段发出时开始，但是无法判断什么时候慢启动结束，拥塞避免阶段开始。慢启动阶段和拥塞避免阶段的鉴定取决于发送方拥塞窗口的大小。拥塞窗口的大小并不能从时间—序号图（time-sequence-graph）直接获得。然而在一个发送方中未被确认的数据量（即in flight 数据量）不会超过CongWin（拥塞窗口）和RcvWindow（接收窗口）中的最小值，即LastByteSend-LastByteAcked<=min{CongWin,RcvWindow}。同时，在第9题中看到，接收方通告给发送方的窗口大小并没有遏制发送速率。因此，未被确认的数据量（即in flight 数据量），是由拥塞窗口决定的，所以通过发出而未被确认的数据量（即in flight 数据量），我们可以估计拥塞窗口大小的下界。

**Q14讨论你看到的数据和教材上讲的TCP的理想化行为的区别。**

这个图形类似于阶梯状，跟教材上还是有很大区别的。

1. **ICMP协议分析**
2. **ICMP与ping**

**（1）实验步骤：**

①让我们进入 Windows 命令行提示符；

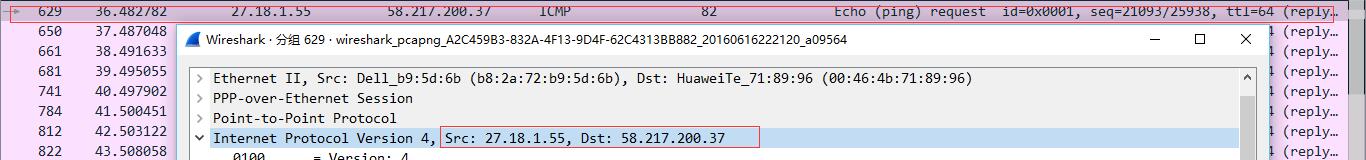
②开启 Wireshark Network Analyzer，开始捕捉数据包；

③在 MS-DOS 命令行下输入：ping –n 10 www.baidu.com

④当 Ping 程序终止后，停止用 Wireshark 捕获包。

**（2）回答问题：**

**Q1.你主机的 IP 地址是什么？远程主机的 IP 地址是什么？**

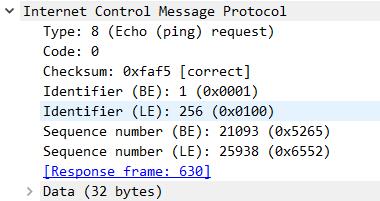


本机IP是27.18.1.55；远程主机的IP地址是58.217.200.37.

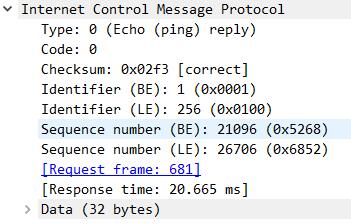
**Q2.为什么一个 ICMP 包没有源和目标端口号？**

A2. 因为ICMP是网络层协议，因此没有源和目的端口号。

**Q3. 检查由你主机发送的一个 Ping 请求包。 ICMP type 和 code 是多少？这个 ICMP 包包含了哪些其他的字段？ Checksum、 sequence number 和 identifier 字段的值各为多少？**

****

**Q4.检查相应的 Ping 回应包， ICMP type 和 code 是多少？这个 ICMP 包包含了哪些其他的字段？ Checksum、 sequence number 和 identifier 字段的值各为多少？**

****

**实验二：ICMP与traceroute**

**（1）实验步骤：**

①进入 Windows 命令提示符；

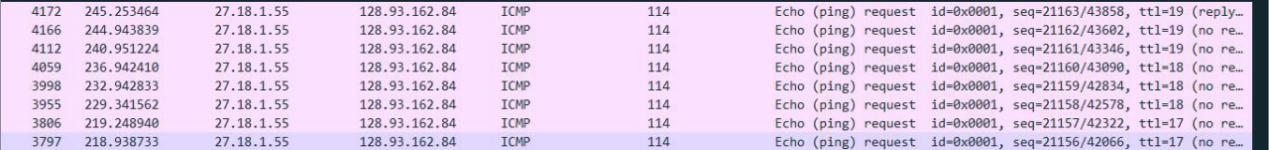
②开启 Wireshark Network Analyzer，开始捕捉数据包；

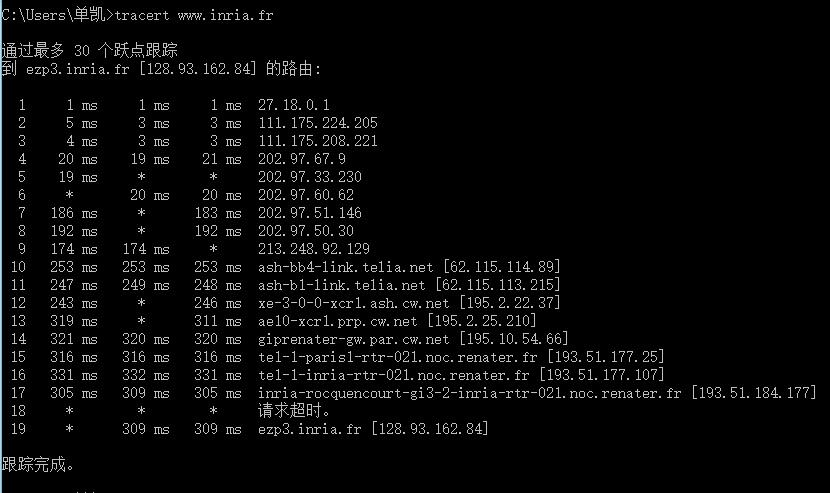
③在 MS-DOS 命令行下输入：tracert www.inria.fr

④当 Traceroute 程序终止后，停止用 Wireshark 捕获包。

**（2）回答问题：**

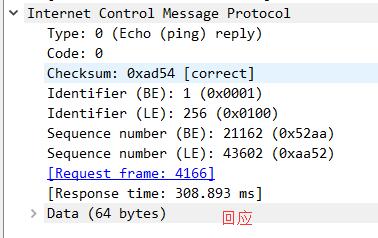
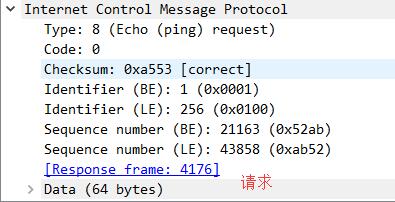
**Q5.你主机的 IP 地址是什么？远程主机的 IP 地址是什么？**





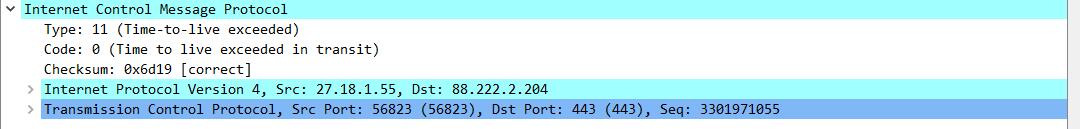
A5. 我主机的IP地址是27.18.1.55，远程主机IP是128.93.162.84。

**Q6.检查 ICMP echo 包，它和前半部分实验中的 ICMP ping 查问包有否不同？如有，具体在何处？**



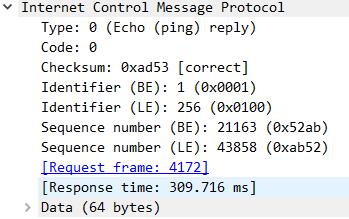
|  |  |  |
| --- | --- | --- |
| 不同点 | 请求包 | 回应包 |
| Type | 8 | 0 |

**Q7.检查 ICMP error 包，它的类型是什么，code 是什么。它和 echo 包相比有更多的字段，这些字段中包含了什么？**



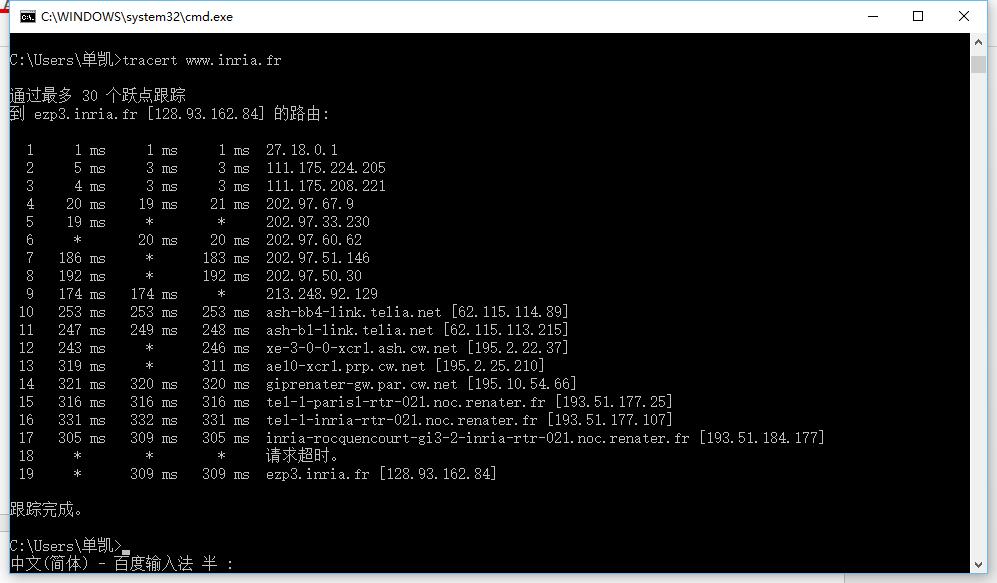
A7. 类型是11，code是0，与echo包相比多了Internet protocol version字段中的若干字段。

**Q8.检查源端收到的最后三个 ICMP 包。它们和 ICMP error 包有何不同？为什么不同？**



A8. 没有下属的Internet protocol version字段和Internet control message protocol字段，多了64字节的data字段。因为功能不同。

**Q9.在 tracert 测算中是否有延迟明显高于其它的链路？基于路由器的名字，你能否猜测一下这两条链路末端路由器的所在位置？**



1. **HTTP协议分析**

**实验一：基本的 HTTP GET/Response 操作**

**（1）实验步骤：**

①打开浏览器；

②开启 Wireshark Network Analyzer，但不开始数据包捕获。在 Wireshark 主窗口顶部的 Filter中输入“ http”，因此只有捕获的 HTTP 消息稍后会显示在数据包列表窗口中；

③等待大约一分钟，然后开始 Wireshark 数据报捕获；

④在浏览器中输入如下地址：http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html浏览器将会显示一个很简单的且只有一行的 HTML 文件；

⑤停止 Wireshark 的数据报捕获。

得到如下数据报：

第127和第140条数据报正是我们要获取的。

**（2）回答问题：**

**Q1.你的浏览器运行的 HTTP 是 1.0 版本还是 1.1 版本？服务器运行 HTTP 是哪个版本？**

C:\Users\单凯\Desktop\1.jpg

A1.我的浏览器运行的是1.1版本，服务器运行的也是1.1版本。

**Q2.你的浏览器能接受服务器的哪些语言？**

C:\Users\单凯\Desktop\1.jpg

A2.我的浏览器能接受zh-cn和zh等语言。

**Q3.你的电脑的 IP 地址是多少？ gaia.cs.umass.edu服务器的 IP 地址是多少？**

C:\Users\单凯\Desktop\1.jpg

A3.我的IP地址是192.168.1.162，服务器的IP地址是128.119.245.12

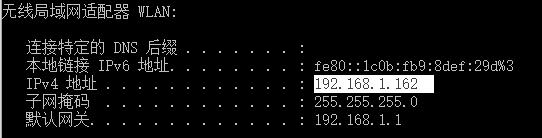


图 1 ipconfig查看自己的IP

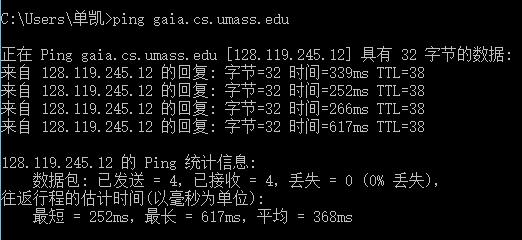
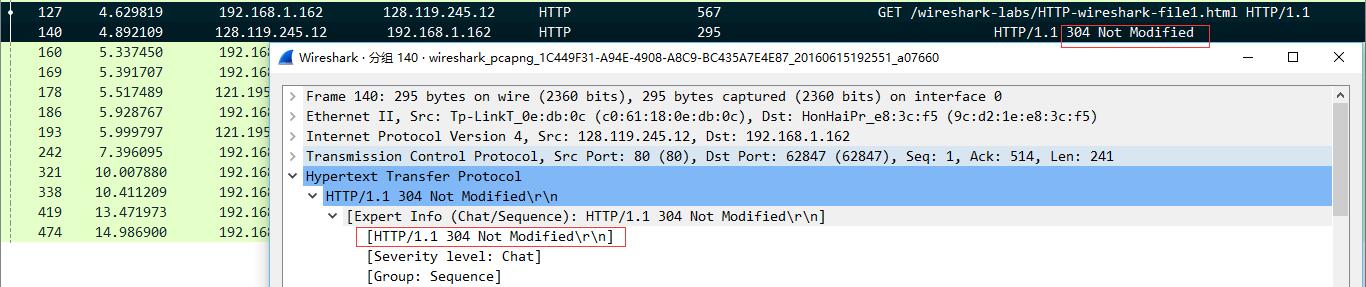


图 2 ping命令得到目的主机地址

由wireshark捕获的包的地址和dos命令下的一致。  
**Q4.从服务器返回到你的浏览器的状态码是什么？**



A4.状态码是304.



图 3 304状态码的含义

**Q5.什么时候在服务器端得到最后修正的 HTML 文件？**

C:\Users\单凯\Desktop\2.jpg

A5. 2016.6.15 11:25:52

**Q6.多少字节的内容已经返回到你的浏览器？**

如果是首次请求应该是如下的情况



返回到浏览器的内容长度是200字节。

但是，我的不是首次请求，服务器通知浏览器，从缓存中去取，所以并没有如上图所示的两个首部字段。

**Q7.除以上已回答过的字段外，头部还有哪些字段？在数据包内容窗口中检查原始数据，是否有未在数据包列表中显示的头部？**

A7.

1. **HTTP请求报文中缺少：**
2. **Accept-Charset**: ISO-8859-1 [**接受字符编码 iso-8859-1**]
3. **Referer**: <http://www.sohu.com/index.jsp> [**告诉服务器，我来自哪里,该消息头，常用于防止盗链**]
4. Cookie [**cookie??是否使用cookie？？**]

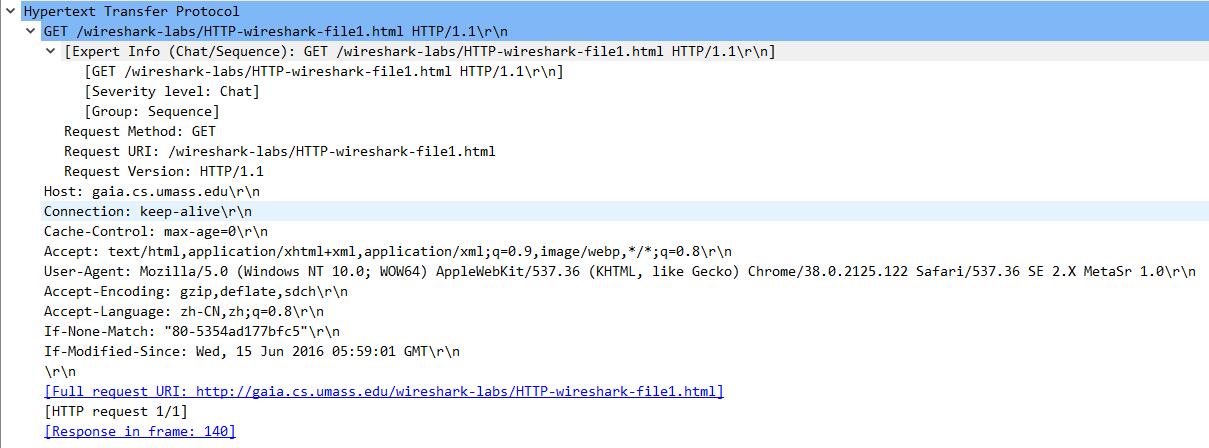


图 4 http请求报文

1. **http响应报文中缺少：**

* **Location: http://www.baidu.org/index.jsp 【让浏览器重新定位到url】**
  + **Content-Encoding**: gzip 【告诉浏览器我使用 gzip】
  + **Content-Length**: 80 【告诉浏览器会送的数据大小80节】
  + **Content-Language**: zh-cn 【支持中文】
  + **Content-Type**: text/html; charset=GB2312 [内容格式text/html; 编码gab2312]
  + **Last-Modified**: Tue, 11 Jul 2000 18:23:51 GMT 【告诉浏览器，该资源上次更新时间】
  + **Refresh**: 1;url=http://www.baidu.com 【过多久去，刷新到 http://www.baidu.com】
  + **Content-Disposition**: attachment; filename=aaa.zip 【告诉浏览器，有文件下载】
  + **Transfer-Encoding**: chunked [传输的编码]
  + **Set-Cookie**:SS=Q0=5Lb\_nQ; path=/search[后面详讲]
  + **Expires**: -1[告诉浏览器如何缓存页面IE]
  + **Cache-Control**: no-cache [告诉浏览器如何缓存页面火狐]
  + **Pragma**: no-cache [告诉浏览器如何缓存页面]

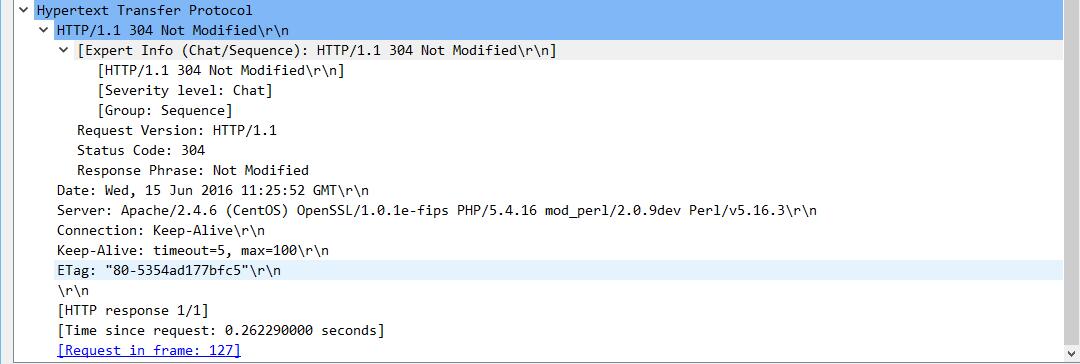
****

图 5 http响应报文

**实验二：HTTP GET/Response 有条件的相互作用**

**（1）实验步骤：**

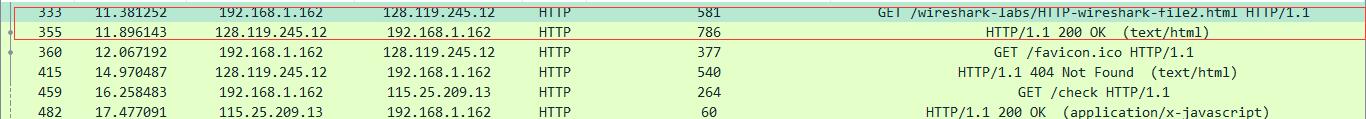
在开始前先确信你的浏览器缓存是空的，对于 IE 浏览器选择工具-Internet 选项-删除文件，钩选“删除全部文件”从你的浏览器中移除缓存的文件，然后点击确定按钮。现在按以下步骤做：  
①打开 Wireshark Network Analyzer，开始数据包捕获。  
②在你的浏览器中输入如下的 URL 地址：<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html>

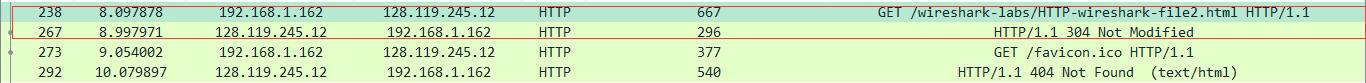
浏览器将会显示一个很简单的 5 行的 HTML 文件；

  
③快速地在你的浏览器中再次输入相同的 URL（或者点一下刷新）；

④停止 Wireshark 数据包捕获，在在 Wireshark 主窗口顶部的 Filter 中输入“ http”，因此只有捕获的 HTTP 消息稍后将会显示在数据包列表窗口中。

得到如下数据报：





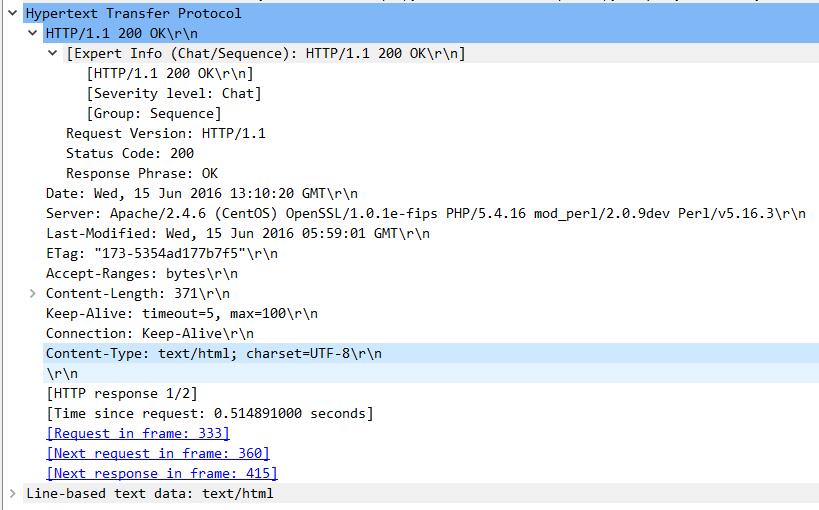
红框中的数据正是我们要获取的。

**（2）回答问题：**

Q8.从你的浏览器到服务器得到的请求中检查第一个 HTTP GET 的内容。在 HTTP GET  
中你有没有看到一行“ IF-MODIFIED-SINCE” ?为什么？



A8.没有这行，因为这是浏览器首次获取该页面。  
**Q9.检查服务器回应内容，服务器明确地返回了文件的内容吗？你怎样断定？**



A9.明确返回了。从line-based text data中的数据可以看出。  
**Q10.现在从你的浏览器到服务器得到的请求中检查第二个 HTTP GET 的内容。 在 HTTP GET 中你有没有看到一行“ IF-MODIFIED-SINCE” ?如果有，在“ IF-MODIFIED-SINCE”头部有什么信息？**



If-Modified-Since是标准的HTTP请求头标签，在发送HTTP请求时，把浏览器端缓存页面的最后修改时间一起发到服务器去，服务器会把这个时间与服务器上实际文件的最后修改时间进行比较。

如果时间一致，那么返回HTTP状态码304（不返回文件内容），客户端接到之后，就直接把本地缓存文件显示到浏览器中。

如果时间不一致，就返回HTTP状态码200和新的文件内容，客户端接到之后，会丢弃旧文件，把新文件缓存起来，并显示到浏览器中。



图 6 第一次的服务器响应报文

**Q11.第二次 HTTP 返回的状态码是多少？从服务器返回的响应第二个 HTTP GET 的短语是什么？服务器明确地返回了文件的内容吗？为什么？**

A11.状态码是304，短语是Not Modified，并没有明确返回文件内容，因为该网页内容的最后修改时间和缓存中的最后修改时间一致。所以浏览器可以直接从缓存中读取，服务器不再发送，以便节省带宽。

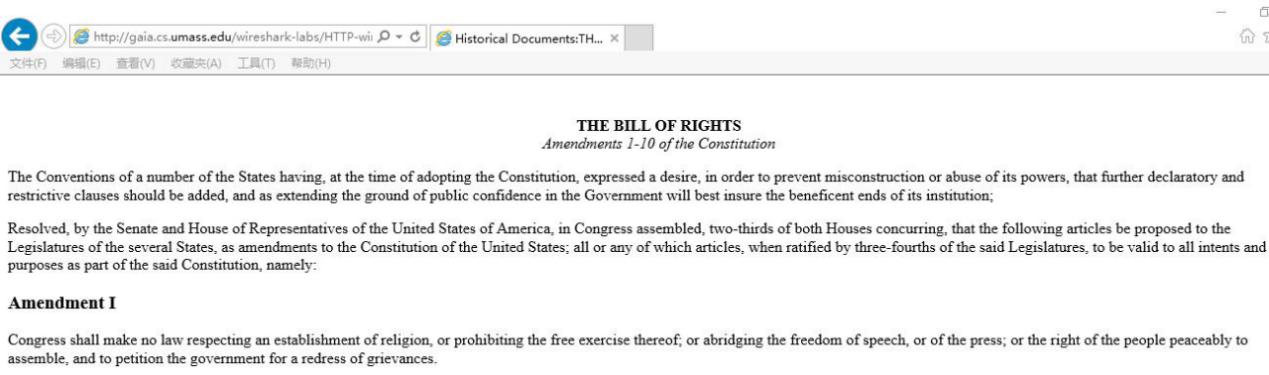


**实验三：得到较长的文档**

**（1）实验步骤：**

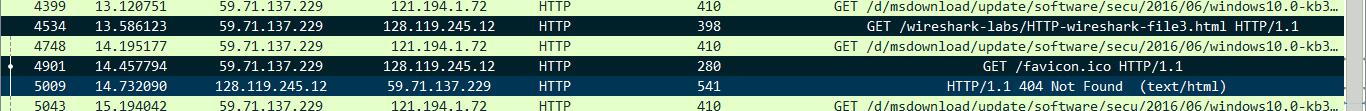
在前面实验得到的文档是简单和短小的 HTML 文件， 让我们来看看下载一个长的 HTML文件时会出现什么。按以下要求做：  
①打开你的浏览器，确信你的浏览器缓存是空的，就如上面讨论的；  
②打开 Wireshark Network Analyzer，开始数据包捕获；  
③在你的浏览器中输入如下的 URL 地址：  
<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html>

你的浏览器将会显示相当长的美国权利法案；

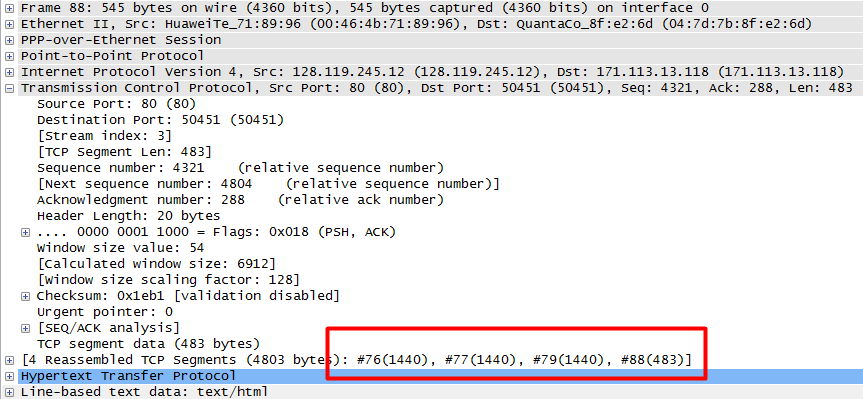
  
④停止 Wireshark 数据包捕获，在 Filter 栏输入“ http”，因此只有捕获的 HTTP 消息稍后将会显示在数据包列表窗口中。

**（2）回答问题：**

**Q12.你的浏览器发送了多少个HTTP GET请求消息？**



A12.2个。  
**Q13．传输这一个HTTP响应需要多少个TCP数据段？**



A13.4个TCP数据段。  
Q14.响应HTTP GET请求的相关的状态码和短语是什么？



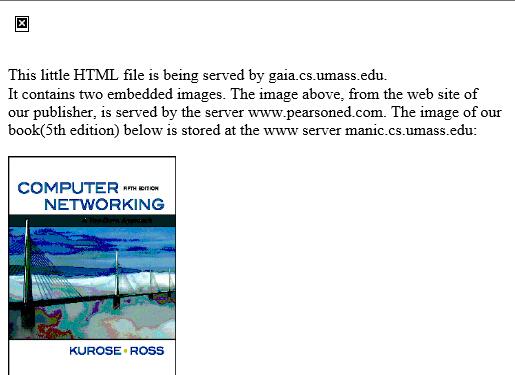
A14.200 OK.  
Q15．在TCP“Continuation”附加关联的传输数据中有没有HTTP状态码和短语？

A15.有。

**实验四：带有内嵌对象的HTML文档**

**（1）实验步骤：**

①打开浏览器，确信你的浏览器缓存是空的，就如上面讨论的；  
②打开 Wireshark Network Analyzer，开始数据包捕获；  
③在你的浏览器中输入如下的 URL 地址：<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html>



④停止 Wireshark 数据包捕获，在 Filter 窗口中输入“ http”，因此只有捕获的 HTTP 消息稍后将会显示在数据包列表窗口中。

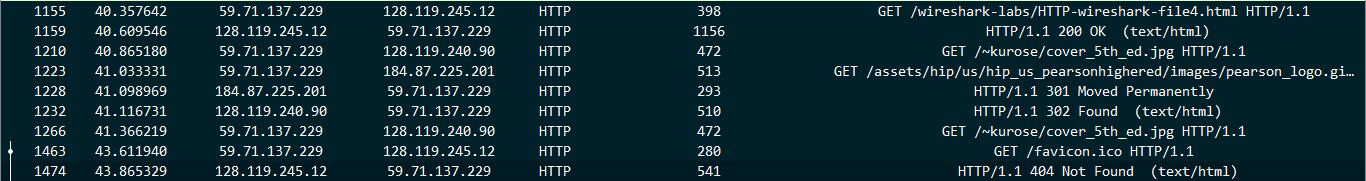


图 7 请求带图片的文件

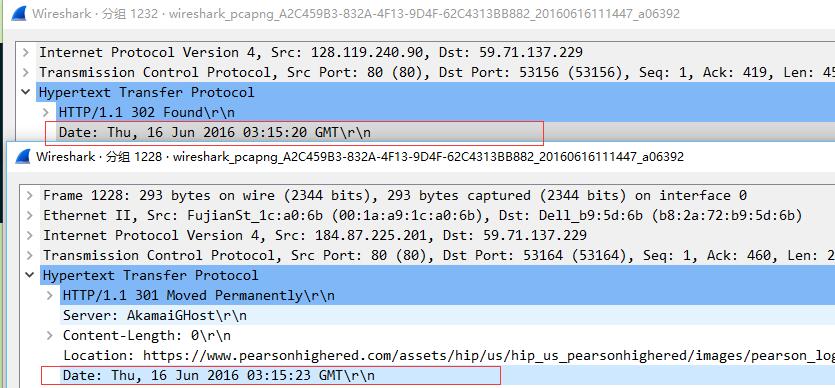
**（2）回答问题：**

**Q16. 你的浏览器发送了多少个HTTP GET请求消息？这些GET请求发送到哪些IP地址？**

A16.

共发送了5个get请求，目的IP地址分别是：128.119.245.12、128.119.240.90、184.87.225.201。

**Q17．你的浏览器是否同时下载这两张图片（并发的从两个网站上下载图片）？你是如何判断的？**



A17.不是并发下载的，是先后请求先后到达的。

如图7 所示：浏览器在得到html页面之后，连续发出两个请求，请求两张图片，然后服务器端连续发回两个响应。

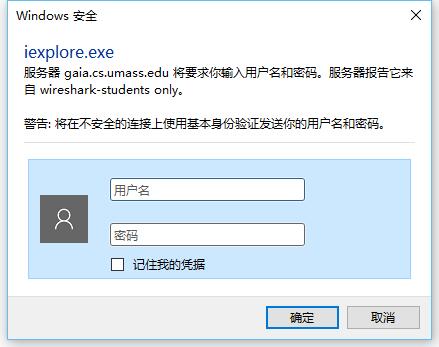
**实验五：HTTP认证**

**（1）实验步骤：**

① 确信你的浏览器缓存是空的，就如上面讨论的。关闭你的浏览器，然后再打开；  
② 打开 Wireshark Network Analyzer，开始数据包捕获；  
③ 在你的浏览器中输入如下的 URL 地址：

<http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html>

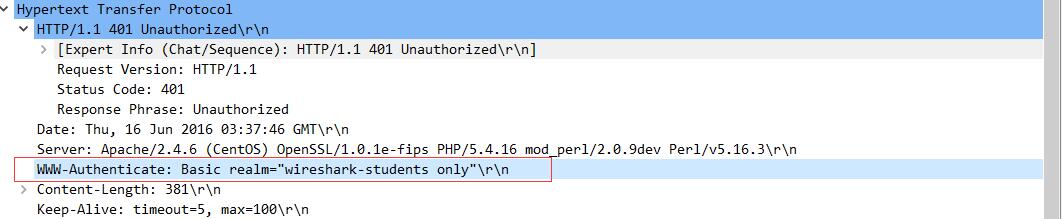
在出现的对话框中输入用户名和密码，用户名是“wireshark-students”，密码是“network”；

  
④停止Wireshark 数据包捕获，在Filter窗口中输入“ http”，因此只有捕获的HTTP消息稍后将会显示在数据包列表窗口中。

**（2）回答问题：**

**Q18．服务器对起初的HTTP GET消息的响应（状态码和短语）是什么？与前一部分实验相比，在这个响应消息中出现了什么新的字段？**





A18. 401 Unauthorized，出现了www-Authenticate字段。  
**Q19．当你的浏览器第二次发送HTTP GET消息时，有什么新的字段被包含在HTTP GET消息中？对应的服务器响应有没有新的字段出现？**

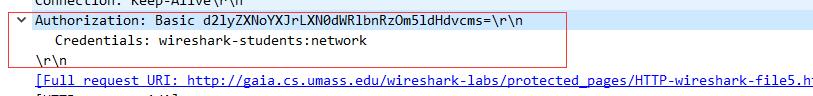


图 8 get消息

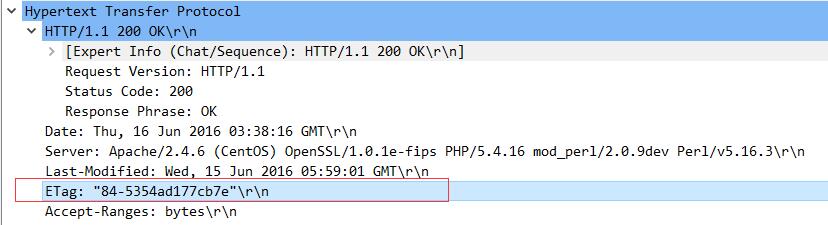
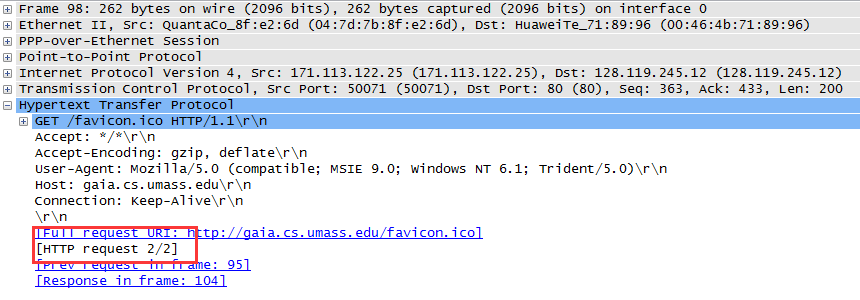
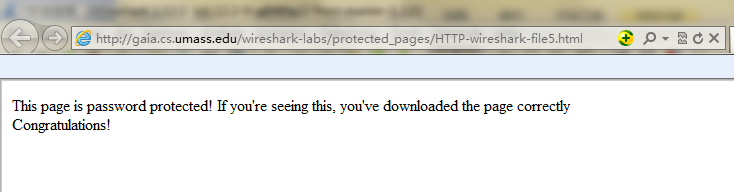


图 9 响应消息

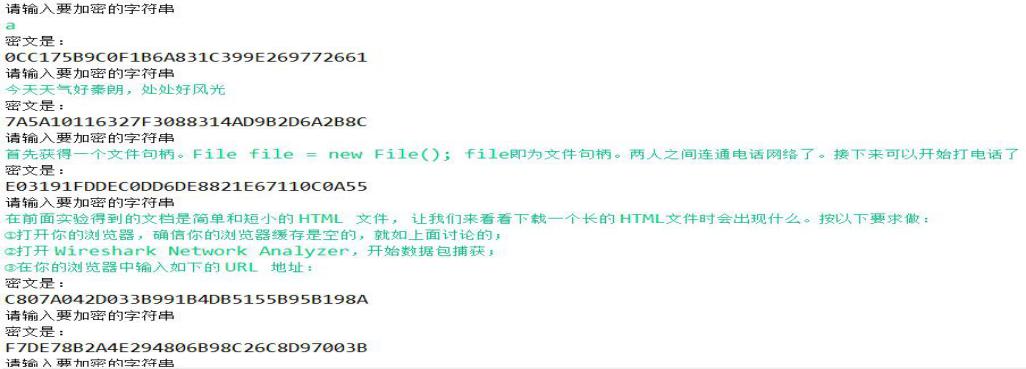
A19. GET消息中出现了新的Authorization字段。服务器响应出现了ETag字段。  
**Q20．当你的浏览器第三次发送HTTP GET消息时， 有什么新的字段被包含在HTTP GET消息中？当你输入正确的用户名和密码后，服务器给出的响应是什么？**

A20. HTTP request字段。给出如下页面响应。

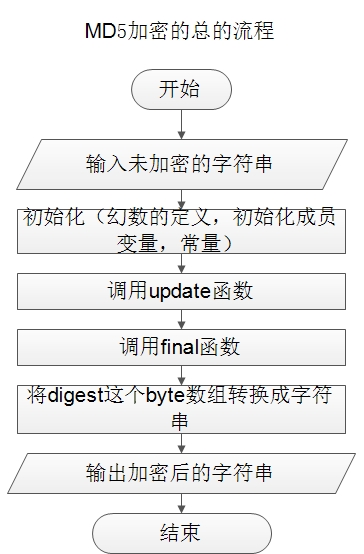
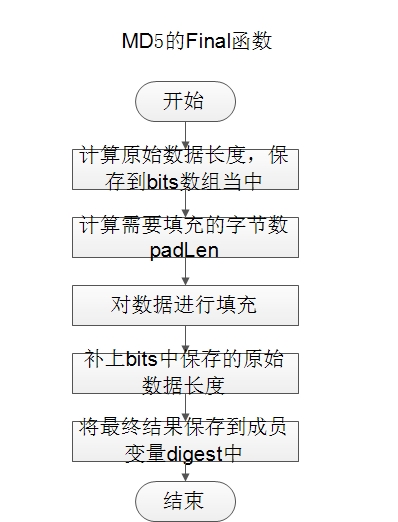




1. **Java编程**
2. **MD5加密**
3. **运行效果**

****

1. **流程框图**

****

1. **SHA1加密**
2. **运行结果**

****

1. **流程框图**

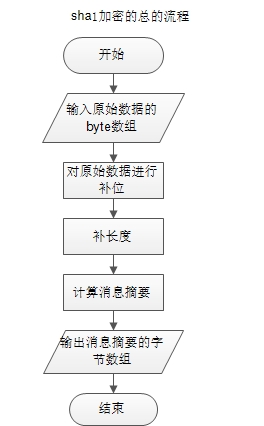
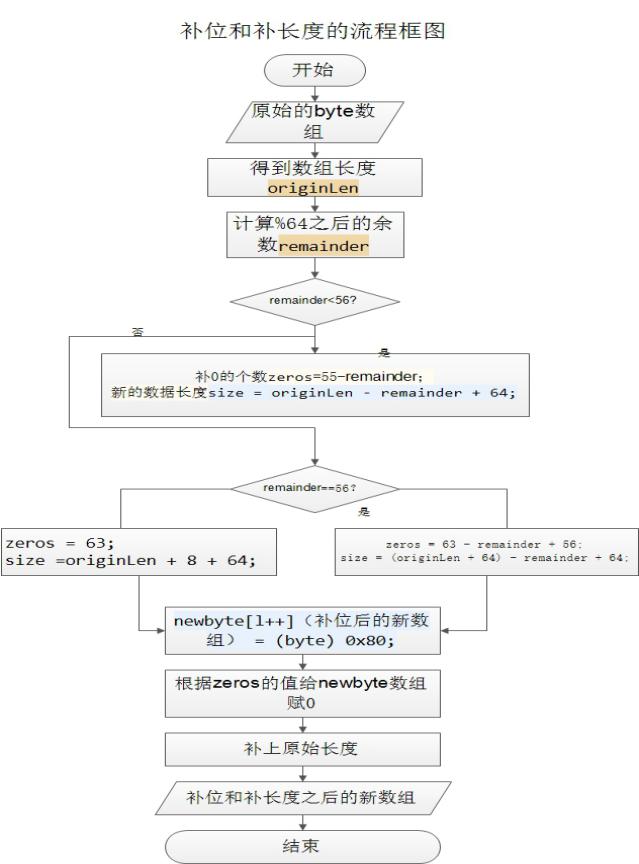
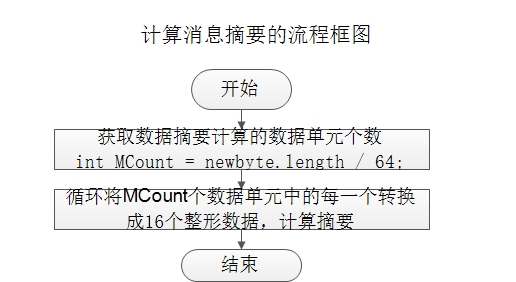
****

图 10 sha1总的流程



1. **关键代码**

计算摘要的算法是固定的，首先定义四个需要用到的函数

**private** **int** f1(**int** x, **int** y, **int** z){**return** (x & y) | (~x & z);}

**private** **int** f2(**int** x, **int** y, **int** z){**return** x ^ y ^ z;}

**private** **int** f3(**int** x, **int** y, **int** z){**return** (x & y) | (x & z) | (y & z);}

**private** **int** f4(**int** x, **int** y){**return** (x << y) | x >>> (32 - y);}

和两个需要用到的缓冲区：

//五个字的缓冲区

**private** **final** **int**[] abcde =

{ 0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476, 0xc3d2e1f0 };

// 摘要数据存储数组

**private** **int**[] digestInt = **new** **int**[5];

// 计算过程中的临时数据存储数组，80个字的缓冲区

**private** **int**[] tmpData = **new** **int**[80];

接下来就开始计算摘要

**for** (**int** i = 16; i <= 79; i++)

{

tmpData[i] = f4(tmpData[i - 3] ^ tmpData[i - 8] ^ tmpData[i - 14]

^ tmpData[i - 16], 1);

}

**int**[] tmpabcde = **new** **int**[5];

**for** (**int** i1 = 0; i1 < tmpabcde.length; i1++)

{

tmpabcde[i1] = digestInt[i1];

}

**for** (**int** j = 0; j <= 19; j++)

{

**int** tmp = f4(tmpabcde[0], 5)

+ f1(tmpabcde[1], tmpabcde[2], tmpabcde[3]) + tmpabcde[4]

+ tmpData[j] + 0x5a827999;

tmpabcde[4] = tmpabcde[3];

tmpabcde[3] = tmpabcde[2];

tmpabcde[2] = f4(tmpabcde[1], 30);

tmpabcde[1] = tmpabcde[0];

tmpabcde[0] = tmp;

}

**for** (**int** k = 20; k <= 39; k++)

{

**int** tmp = f4(tmpabcde[0], 5)

+ f2(tmpabcde[1], tmpabcde[2], tmpabcde[3]) + tmpabcde[4]

+ tmpData[k] + 0x6ed9eba1;

tmpabcde[4] = tmpabcde[3];

tmpabcde[3] = tmpabcde[2];

tmpabcde[2] = f4(tmpabcde[1], 30);

tmpabcde[1] = tmpabcde[0];

tmpabcde[0] = tmp;

}

**for** (**int** l = 40; l <= 59; l++)

{

**int** tmp = f4(tmpabcde[0], 5)

+ f3(tmpabcde[1], tmpabcde[2], tmpabcde[3]) + tmpabcde[4]

+ tmpData[l] + 0x8f1bbcdc;

tmpabcde[4] = tmpabcde[3];

tmpabcde[3] = tmpabcde[2];

tmpabcde[2] = f4(tmpabcde[1], 30);

tmpabcde[1] = tmpabcde[0];

tmpabcde[0] = tmp;

}

**for** (**int** m = 60; m <= 79; m++)

{

**int** tmp = f4(tmpabcde[0], 5)

+ f2(tmpabcde[1], tmpabcde[2], tmpabcde[3]) + tmpabcde[4]

+ tmpData[m] + 0xca62c1d6;

tmpabcde[4] = tmpabcde[3];

tmpabcde[3] = tmpabcde[2];

tmpabcde[2] = f4(tmpabcde[1], 30);

tmpabcde[1] = tmpabcde[0];

tmpabcde[0] = tmp;

}

**for** (**int** i2 = 0; i2 < tmpabcde.length; i2++)

{

digestInt[i2] = digestInt[i2] + tmpabcde[i2];

}

**for** (**int** n = 0; n < tmpData.length; n++)

{

tmpData[n] = 0;

}

1. **Des加密**
2. **运行效果**

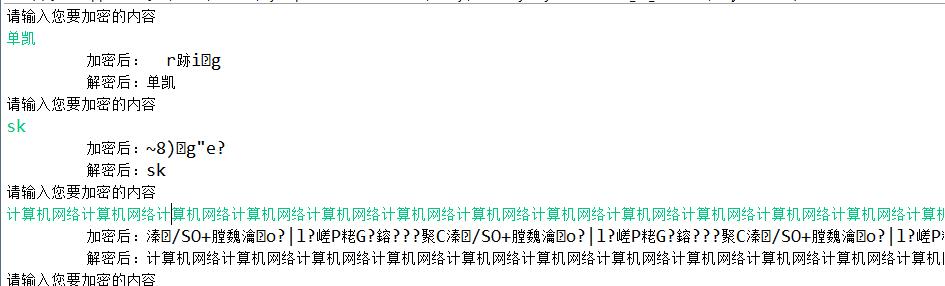
****

图 11 des加密的效果

1. **流程框图**

|  |  |
| --- | --- |
| **加密** | **解密** |
|  |  |

1. **关键代码**

**// DES算法要求有一个可信任的随机数源**

**SecureRandom random = new SecureRandom();**

**// 创建一个DESKeySpec对象**

**DESKeySpec desKey = new DESKeySpec(password.getBytes());**

**// 创建一个密匙工厂**

**SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");**

**// 将DESKeySpec对象转换成SecretKey对象**

**SecretKey securekey = keyFactory.generateSecret(desKey);**

**// Cipher对象实际完成解密操作**

**Cipher cipher = Cipher.getInstance("DES");**

**// 用密匙初始化Cipher对象**

**cipher.init(Cipher.DECRYPT\_MODE, securekey, random);**

**// 真正开始解密操作**

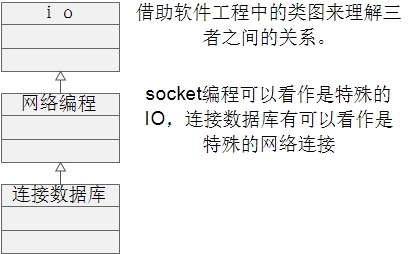
**return cipher.doFinal(src);**

1. **Tcp的套接字编程**

这部分包括Tcp简单通信，一对多聊天，传送文件等，结合后面的“仿ＱＱ好友列表部分”的多对多聊天基本上全面覆盖了ｔｃｐ编程的基础知识。

1. **Java中流的相关知识：**

个人认为io和网络编程和连接数据库之间的关系如下

****

从这部分开始开始接触socket编程，所以和有必要了解一下java中的IO。

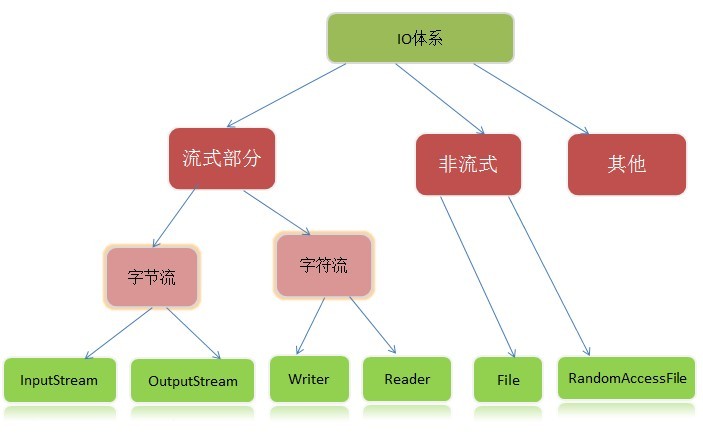


图 12 java中IO体系结构



图 13 java流类框架

了解了Java中的流类之间的关系才能更好的理解流类对象的构造函数的使用。

下面介绍一下流的另一种分类。

* 节点流：可以从或向一个特定的地方（节点）读写数据。如FileReader.
* 处理流：是对一个已存在的流的连接和封装，通过所封装的流的功能调用实现数据读写。如BufferedReader.处理流的构造方法总是要带一个其他的流对象做参数。一个流对象经过其他流的多次包装，称为流的链接。

然后说一下流的关闭规则

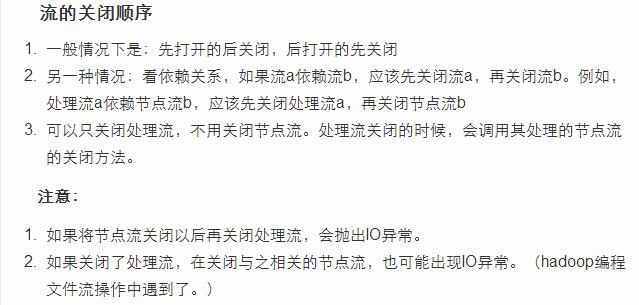
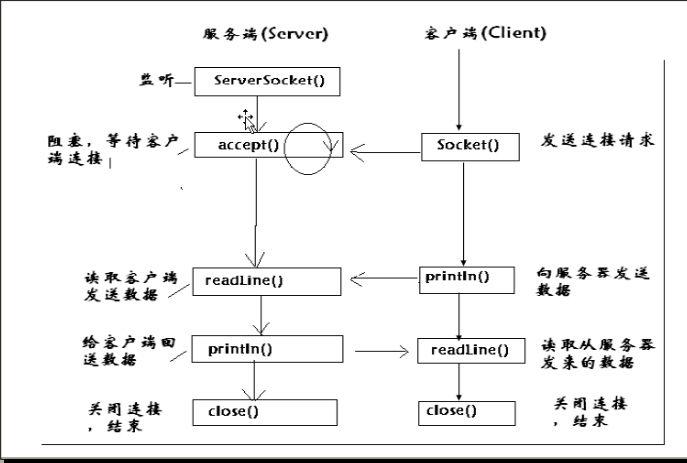


图 14 java中流的关闭顺序

1. **Tcp通信的示意图**



1. **通过控制台输入进行通信**

关键代码如下：

|  |  |
| --- | --- |
| 服务器端： | 客户端： |
| // 在9999端口监听  ServerSocket ss=**new** ServerSocket(9999);  //等待某个客户端来连接，该函数会返回一个socket链接  Socket s=ss.accept();  BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));  PrintWriter pw=new PrintWriter(s.getOutputStream(),true);  String infoFromClient=br.readLine();  Pw.println(); | //链接某个服务器， ip 端口号  Socket s=new Socket("127.0.0.1", 9999);  //给服务器发送数据  //通过pw向s写数据，true表示即时刷新  PrintWriter pw=new PrintWriter(s.getOutputStream(),true);  pw.println(brConsole.readLine()); |

1. **通过tcp传输对象**

|  |  |
| --- | --- |
| 服务器端 | 客户端 |
| 双方公用的类：  **public** **class** User **implements** Serializable  { //用户名和密码 **private** String name; **private** String pass;} | |
| ObjectInputStream ois=**new** ObjectInputStream(s.getInputStream());  User u=(User)ois.readObject(); | //s是socket对象，u是User对象  ObjectOutputStream oos=new ObjectOutputStream(s.getOutputStream());  oos.writeObject(u); |

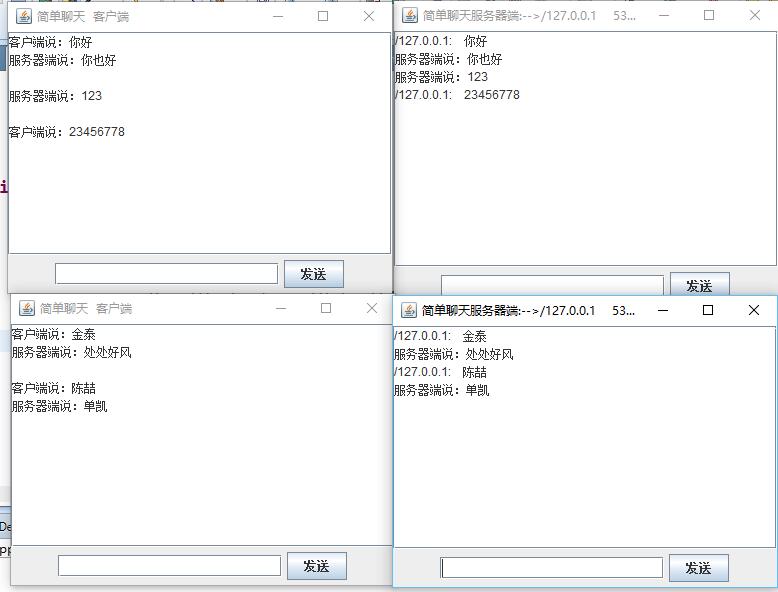
1. **带界面的tcp通信**



图 15 带界面的tcp通信的效果

1. **多（客户端）对一（服务器端）的Tcp通信**

* 效果图



* **原理简介**

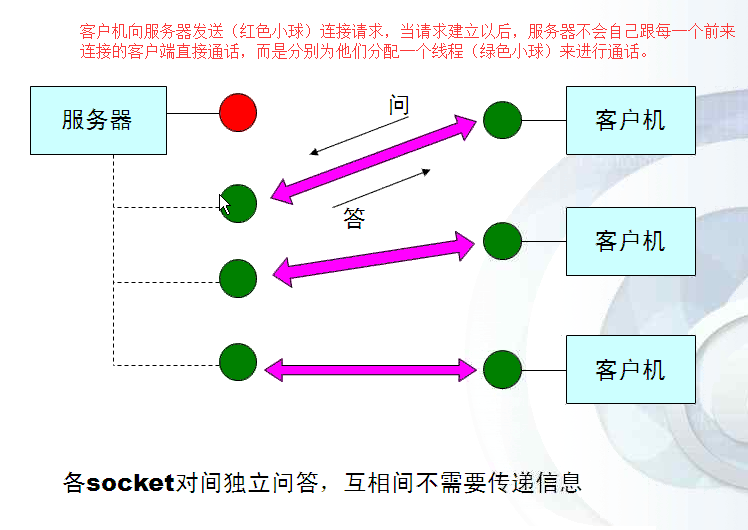
****

图 16 一对多聊天原理图

* **实现**

服务器端要实现线程，每当客户端连过来的时候，服务器端就启动一个线程，来负责与客户端进行通信，关键是将套接字（Socket对象）传递给线程

|  |  |
| --- | --- |
| 服务器端 | 线程 |
| ss =newServerSocket(myListeningPort);  System.out.println("我在9999端口监听");  while (true)  {  Socket clientSocket=ss.accept();  new CreateServerThread(clientSocket);  } | **public** CreateServerThread(Socket s) **throws** IOException  {  client = s;  setInteface();  thread=**new** Thread(**this**);  thread.start();  bufferedReader = **new** BufferedReader(**new** InputStreamReader( client.getInputStream()));  printWriter = **new** PrintWriter(client.getOutputStream(), **true**); |

1. **Tcp传输文件**

* 效果



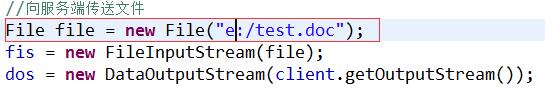


图 17 源文件所在地址

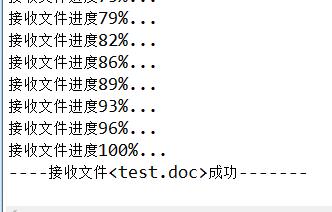
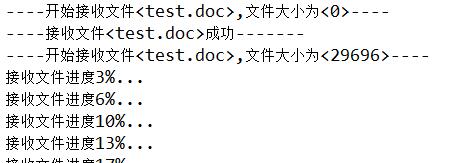


图 18 传输过程

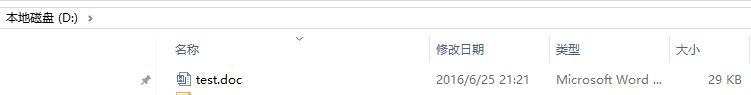


图 19 目的文件

* 关键代码

|  |  |  |  |
| --- | --- | --- | --- |
| 客户端 | | 服务器端 | |
| 先  传输文件名和长度 | //向服务端传送文件  File file = new File("e:/test.doc");  fis = new FileInputStream(file);  dos=newDataOutputStream(client.getOutputStream());  //文件名和长度  dos.writeUTF(file.getName());  dos.flush();  dos.writeLong(file.length());  dos.flush(); | 接收文件名和长度 | dis = new DataInputStream(client.getInputStream());  //文件名和长度  String fileName = dis.readUTF();  long fileLength = dis.readLong();  fos = new FileOutputStream(new File("d:/" + fileName)); |
| 再传文件内容 | //传输文件  byte[] sendBytes = new byte[1024];  int length = 0;  while((length = fis.read(sendBytes, 0, sendBytes.length)) > 0){  dos.write(sendBytes, 0, length);  dos.flush();  } | 接受内容显示进度 | byte[] sendBytes = new byte[1024];  int transLen = 0;  System.out.println("----开始接收文件<" + fileName + ">,文件大小为<" + fileLength + ">----");  while(true){  int read = 0;  read = dis.read(sendBytes);  if(read == -1) break;  transLen += read;  System.out.println("接收文件进度" + 100 \* transLen/fileLength + "%...");  fos.write(sendBytes, 0, read);  fos.flush();} |

1. **下载文件**

|  |  |  |
| --- | --- | --- |
|  | 1. **下载图片** | 1. **下载html页面** |
| **流程图** |  | **这个跟读取html文件差不多** |
| **运**  **行效果** |  |  |

1. **Udp套接字编程**
2. **基本的Udp套接字编程**

* **基本原理**

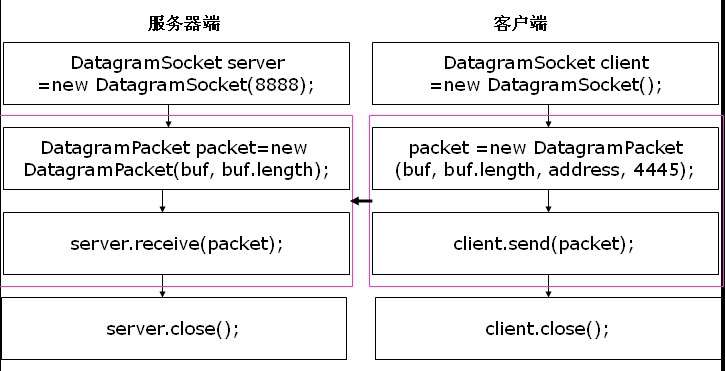
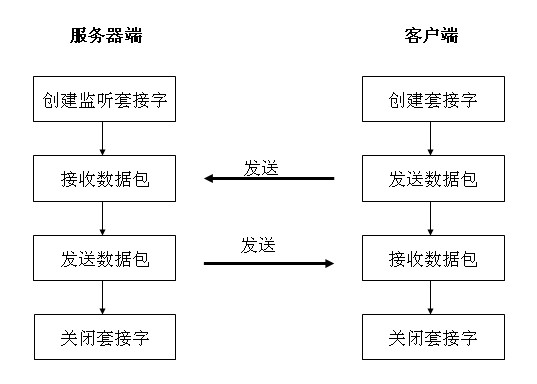
****

图 20　基本Udp编程思路与代码的对应关系

* **运行效果**

****

1. **广播**

* **原理**

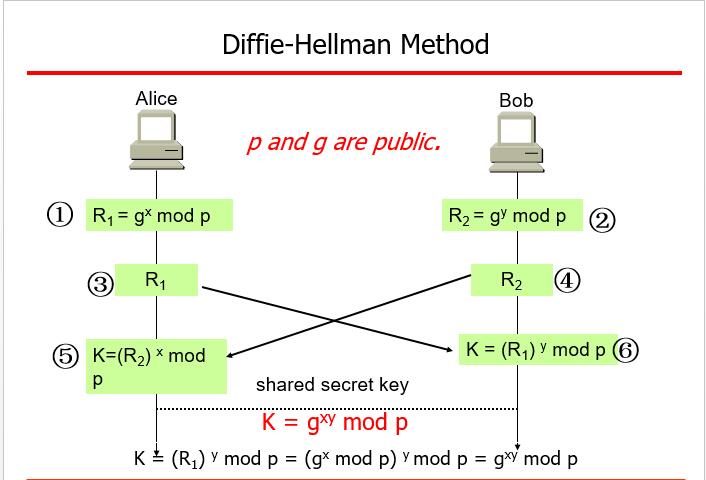
广播UDP与单播UDP的区别就是IP地址不同，广播使用广播地址255.255.255.255，将消息发送到在同一广播网络上的每个主机。值得强调的是：**本地广播信息是不会被路由器转发**。当然这是十分容易理解的，因为如果路由器转发了广播信息，那么势必会引起网络瘫痪。这也是为什么IP协议的设计者故意没有定义互联网范围的广播机制。

广播地址通常用于在网络游戏中处于同一本地网络的玩家之间交流状态信息等。其实广播顾名思义，就是想局域网内所有的人说话，**但是广播还是要指明接收者的端口号的**，因为不可能接受者的所有端口都来收听广播。

代码在“我的Udp套接字编程\src\com\udp\broadcast”和“我的Udp套接字编程\src\com\udp\broadcast2”。

1. **密钥协商**

* **原理**

****

* **运行效果**

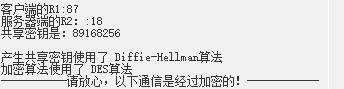
****

图 21 Client端 图 22 客户端

1. **传输对象**

* **总体思路**

跟tcp传输对象基本一样，发送的时候，先创建ObjectOutputStream对象，然后调用ObjectOutputStream.writeObject(u);将对象发送出去。

接受的时候，同样，先拿到ObjectInputStream对象，然后调用readObject方法。

* **关键代码：**

|  |  |
| --- | --- |
| 发送方 | 接收方 |
| DatagramSocket da = new DatagramSocket();  Bytearray b = new Bytearray();  b.setMsg("你好啊！");  byte[] by = new byte[1024 \* 1024];  ByteArrayOutputStream bs = new ByteArrayOutputStream();  ObjectOutputStream bo = new ObjectOutputStream(bs);  bo.writeObject(b); | DatagramSocket da = new DatagramSocket(9099);  byte[] by = new byte[1024 \* 1024];  DatagramPacket data = new DatagramPacket(by, by.length);  da.receive(data);  ByteArrayInputStream bs = new ByteArrayInputStream(data.getData());  ObjectInputStream os = new ObjectInputStream(bs);  Bytearray m = (Bytearray) os.readObject(); |

1. **传输文件**

* **总体思路**

每次传送一个byte数组，直到整个文件传送完毕。但是因为Udp是不可靠的传输，所以增加确认和重传的控制，这个控制非常简单，每收到一个Udp数据报就回复一个确认，发送方在收到确认之后才继续发送下一个数据报，类似于Arq协议。

* **流程框图**

|  |  |
| --- | --- |
| 发送方 | 接收方 |
|  |  |

* 关键代码

|  |  |
| --- | --- |
| 发送方 | 接收方 |
| while ((len = bis.read(buf)) != -1)  {pkg = new DatagramPacket(buf, len, new InetSocketAddress(ip, serverPort));  // 设置确认信息接收时间，3秒后未收到对方确认信息，则重新发送一次  send.setSoTimeout(3000);  send.send(pkg);  send.receive(messagepkg);  System.out.println(new String(messagepkg.getData()));  } | while (true)  {receiveSocket.receive(pkg);  if (new String(pkg.getData(), 0, pkg.getLength()).equals("end"))  {System.out.println("文件接收完毕");  bos.close();  receiveSocket.close();  break;}  receiveSocket.send(ackPkt);  System.out.println(new String(ackPkt.getData()));  bos.write(pkg.getData(), 0, pkg.getLength());  bos.flush();} |

1. **仿QQ**
2. **仿QQ的登录界面**

* **意图**

用一下Md5加密，验证码和套接字编程中的传输对象

* **运行效果**

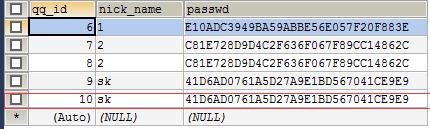
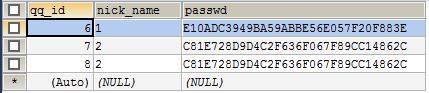


图 23 注册前后的表



图 24 登录界面

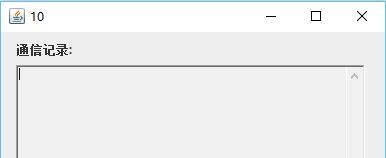


图 25 注册界面和注册后的聊天界面（标题为用户id）

* **连接mysql数据库**

用户表中存放用户昵称，qq号，密码（Md5加密）

将连接数据库的操作专门写成一个类。

|  |  |
| --- | --- |
| 成员变量 | public static final String url = "jdbc:mysql://127.0.0.1/myqq?useUnicode=true&characterEncoding=utf-8&useSSL=false";  public static final String name = "com.mysql.jdbc.Driver";  public static final String user = "root";  public static final String password = "955219";  public Connection conn = null;  public PreparedStatement pst = null; |
| 成员函数 | public DBHelper(String sql)  {  Class.forName(name);// 指定连接类型  conn = DriverManager.getConnection(url, user, password);// 获取连接  pst = conn.prepareStatement(sql);// 准备执行语句  } |
| public void close()  {  this.conn.close();  this.pst.close();  } |

类的应用的一个例子

/\*\*

\* 根据QQ号检查用户的密码

\* @param qqId

\* @return

\*/

public static boolean checkUser(String qqId,String passWd)

{

boolean isOK=false;

String sqlString=MessageFormat.format("select passwd from user where qq\_id={0}", new Integer[]{new Integer(qqId)});

db1=new DBHelper(sqlString);

String tempPwdString="";

try

{

ret=db1.pst.executeQuery();

while(ret.next())

{

tempPwdString=ret.getString(1);

}

if(passWd.equals(tempPwdString))

{

return true;

}

}

catch (SQLException e)

{

e.printStackTrace();

}

return false;

}

* **验证码**

1. **主要思路**

利用随机函数和java的绘图操作

1. **流程框图和关键代码**

|  |  |
| --- | --- |
| **流程框图** | **关键代码** |
|  | **private void DrawVerificationCode()**  **{ Random r = new Random();**  **string str = null;**  **for (int i = 0; i < 5; i++)**  **{ int rNum = r.Next(0, 10); str += rNum; }**  **//创建GDI对象**  **Bitmap bmp = new Bitmap(120, 30);**  **Graphics g = Graphics.FromImage(bmp);**  **for (int i = 0; i < 5; i++)**  **{**  **Point p = new Point(i \* 20, 0);**  **string[] fonts = { "微软雅黑", "宋体", "仿宋", "华文行楷", "隶书" };**  **Color[] colors = { Color.Blue, Color.Brown, Color.Chocolate, Color.DarkOrange, Color.Red };**  **g.DrawString(str[i].ToString()， new Font(fonts[r.Next(0, 5)], 20, FontStyle.Bold), new SolidBrush(colors[r.Next(0, 5)]), p); }**  **//画一些杂乱的线条**  **for (int i = 0; i < 15; i++)**  **{Point p1 = new Point(r.Next(0, bmp.Width), r.Next(0, bmp.Height));**  **Point p2 = new Point(r.Next(0, bmp.Width), r.Next(0, bmp.Height));**  **g.DrawLine(new Pen(Brushes.Green), p1, p2); }**  **//画一些杂乱的点**  **for (int i = 0; i < 500; i++)**  **{ Point p = new Point(r.Next(0, bmp.Width), r.Next(0, bmp.Height));**  **bmp.SetPixel(p.X, p.Y, Color.Black); }**  **//将图片镶嵌到picturebox种**  **pbVerificationCode.Image = bmp; }** |

1. **仿QQ的好友列表以及好友聊天功能（应老师要求）**

**首先声明一下，这一部分和登陆界面是两个不同的程序，没有综合在一起。并且这一部分是我和訚实松同学两个人共同完成的。**

* **运行效果**

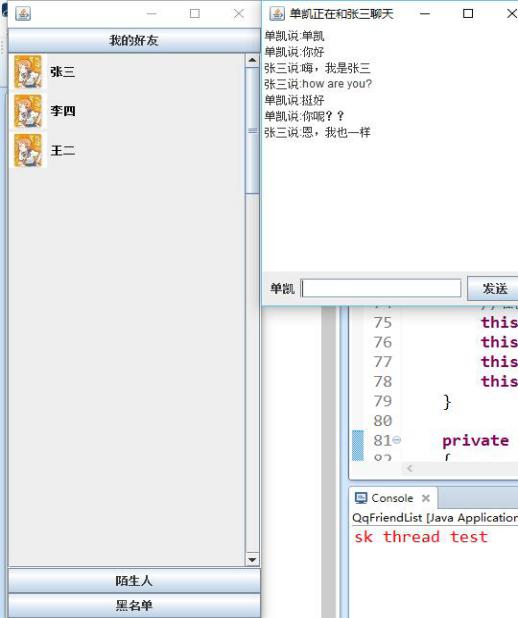
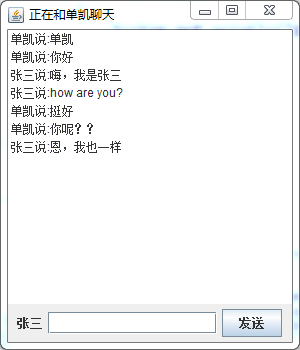
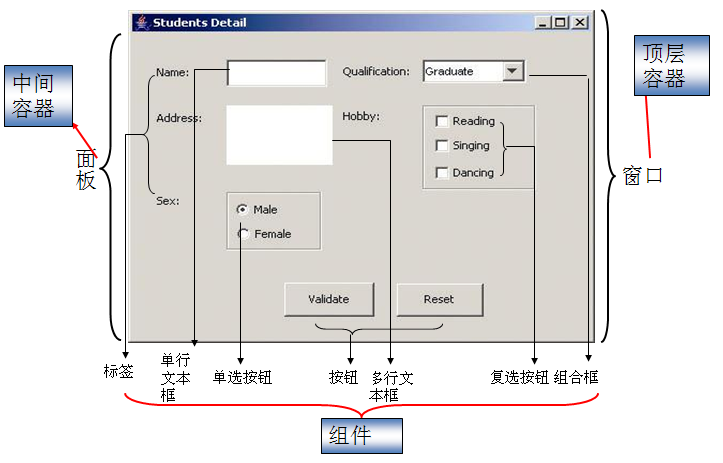
**** ****

图 26 我处情形和partner处的情形

* **JavaGui编程的相关知识**
* **一般的Java的Gui编程的流程**

****

**1、 创建顶层容器**

对应于程序的初始显现窗口，窗口中放入其它菜单、工具栏、文本框、按钮等组件。

顶层容器图形化界面显示的基础，其它所有的组件（控件）都是直接或间接显示在顶层容器中的。在java中顶层容器有三种，分别是JFrame（框架窗口，即通常的窗口）、JDialog（对话框）、JApplet（用于设计嵌入在网页中的java小程序）。

**2、 创建中间容器、组件**

对应于程序中出现的菜单、工具栏（中间容器）、文本框、按钮、单选框、复选框等控件。

有很多Swing组件可以使用，见前面的Swing UI组件表。

**3、 将组件加入容器**

在java中创建组件后，还需要将组件放入相应的容器，才能在顶层容器，如窗口中显示出组件。

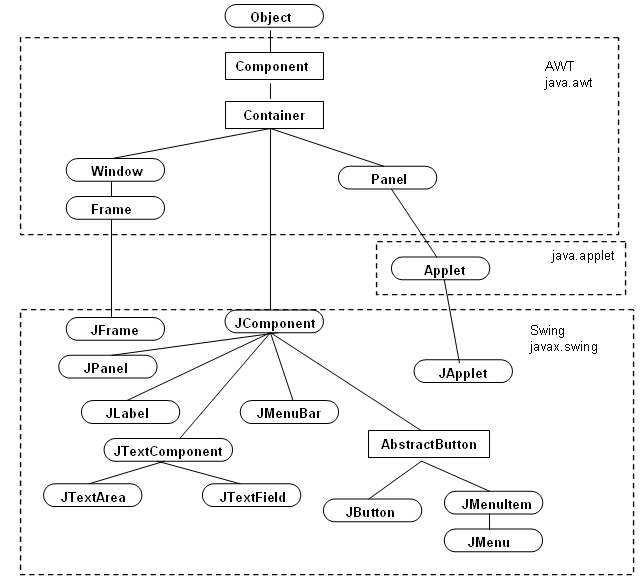
**4、 设置容器内组件的位置**

组件添加到容器中，还必须设置好组件的显示位置，一般有两种方法来设置组建的显示位置，一是按照与容器的相对距离（以像素为单位），精确固定控件的位置；二是用布局管理器来管理组件在容器内的位置。

**5、 处理组件所产生的事件**

即用户执行选择菜单、单击按钮等操作时，就要执行相应的命令，进行相关的程序处理，这就需要设置组件的事件。

* **Swing基本框架**



Java中组件容器包含顶层容器和中间容器。

在java中顶层容器有三种，分别是JFrame（框架窗口，即通常的窗口）、JDialog（对话框）、JApplet（用于设计嵌入在网页中的java小程序），顶层容器是容纳其它组件的基础，即设计图形化程序必须要有顶层容器。

Java中间容器是可以包含其它相应组件的容器，但是中间容器和组件一样，不能单独存在，必须依附于顶层容器。

常见的中间容器有：

• JPanel：最灵活、最常用的中间容器。

• JScrollPane：与 JPanel 类似，但还可在大的组件或可扩展组件周围提供滚动条。

• JTabbedPane：包含多个组件，但一次只显示一个组件。用户可在组件之间方便地切换。

• JToolBar：按行或列排列一组组件（通常是按钮）。

* **卡片布局**

卡片布局能够让多个组件共享同一个显示空间，共享空间的组件之间的关系就像一叠牌，组件叠在一起，初始时显示该空间中第一个添加的组件，通过CardLayout类提供的方法可以切换该空间中显示的组件。

* **好友列表界面的布局构思**

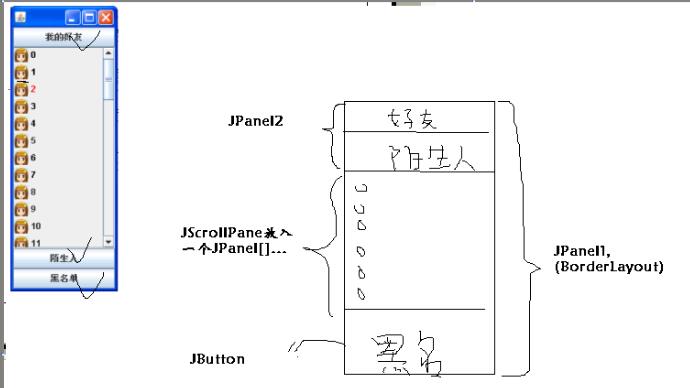
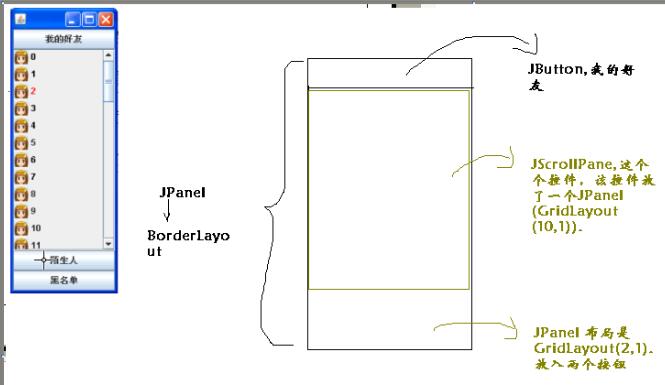


图 27 好友列表界面的构思

* **群聊的服务器端的设计**

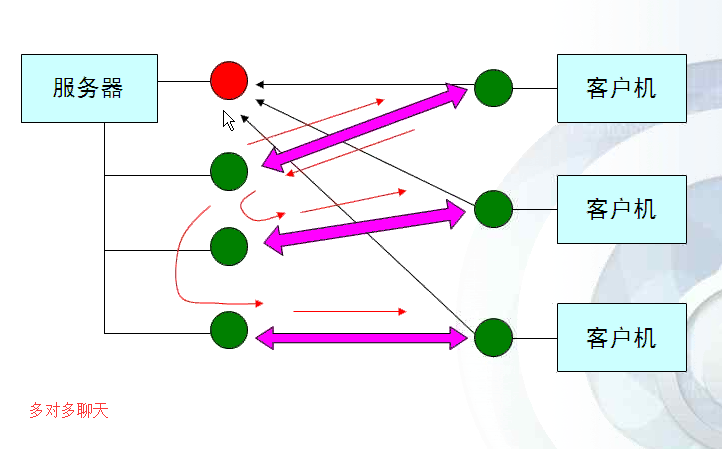
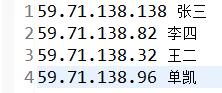
服务器端有多个socket，各socket对间**不是独立的**，从一个socket接受的信息，需要转发到其他的socket上去。将构建的所有的socket放到一个数组中，当需要将一个客户机发送的消息显示给其它的客户机时就遍历该数组，创建输出流，将消息发送出去。 原理图如下：  


图 28 多对多聊天的原理图

**至此，结合“4.Tcp套接字编程”中的相关内容，个人觉得Tcp编程中的基本的常用知识我还是学习掌握全了。**

* **好友信息从哪里来？**

为了简单起见，凸出网络编程和多线程编程的重点，所以我们是事先将打算聊天的几个人的ＩＰ地址和昵称以文本文件的形式保存起来，通过读取文本文件的方式显示到列表上面去。

1. **前向搜索算法**
2. **运行效果**

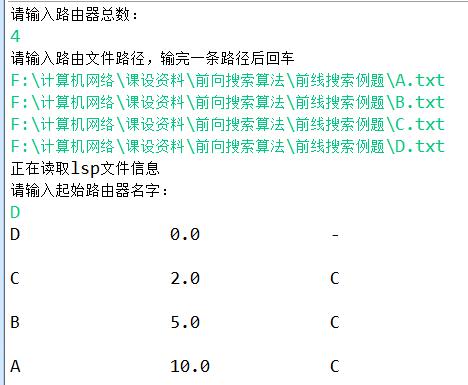
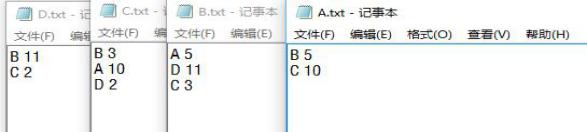
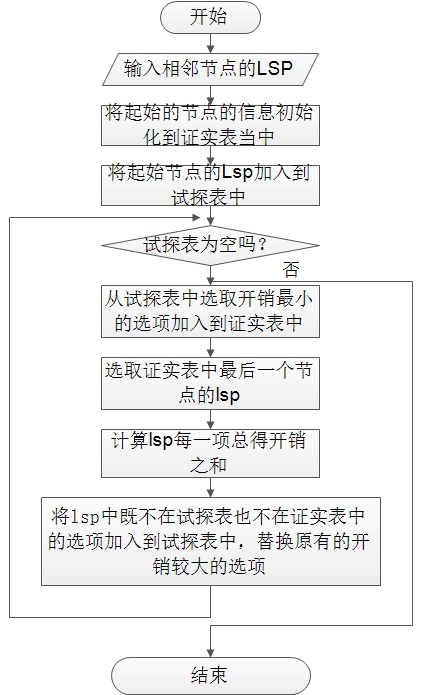
****

图 29 系统方法书中例题的各路由器的lsp信息 图 30 result of forward search

如上图所示结果与《计算机网络系统方法》书中的例题的结果一模一样。

1. **流程框图**

****

1. **关键代码**

**/\*\***

**\* 前向搜索算法**

**\* @param startName 起始的路由器的名字**

**\*/**

**private static void forwardSearch(String startName)**

**{**

**ArrayList<ConfirmAndTentativeNode> confirmed=new ArrayList<ConfirmAndTentativeNode>();//证实表**

**ArrayList<ConfirmAndTentativeNode> tentative=new ArrayList<ConfirmAndTentativeNode>();//试探表**

**//得到起始路由器**

**Router routerStart=GetRouterFromName(startName);**

**//构造比较器,升序**

**Comparator<LSPNode> comparator1=new Comparator<LSPNode>()**

**{**

**public int compare(LSPNode o1, LSPNode o2)**

**{**

**if(o1.cost>=o2.cost)**

**{**

**return 1;**

**}**

**return -1;**

**}**

**};**

**Collections.sort(routerStart.getLsp(),comparator1);**

**//得到nexthop**

**String strNextHop=routerStart.getLsp().get(0).getNeibor();**

**confirmed.add(new ConfirmAndTentativeNode(startName,0,"-"));**

**routerStart.setConfirmed(confirmed);**

**routerStart.setTentative(tentative);**

**for (int i = 0; i < routerStart.getLsp().size(); i++)**

**{**

**tentative.add(new ConfirmAndTentativeNode(routerStart.getLsp().get(i).getNeibor(),**

**routerStart.getLsp().get(i).getCost(),**

**routerStart.getLsp().get(i).getNeibor()));**

**}**

**while(!tentative.isEmpty())**

**{**

**//取得证实表的最后一个节点**

**ConfirmAndTentativeNode tempNode=routerStart.confirmed.get(routerStart.confirmed.size()-1);**

**//获得了他的名字**

**String nextName=tempNode.getDestintion();**

**Router routerTemp=GetRouterFromName(nextName);**

**for(int i=0;i<routerTemp.lsp.size();i++)**

**{**

**if(!routerStart.getLsp().contains(new LSPNode(tempNode.destintion, tempNode.cost)))**

**{**

**nextName=strNextHop;**

**}**

**ConfirmAndTentativeNode node=new ConfirmAndTentativeNode(routerTemp.lsp.get(i).neibor,**

**tempNode.cost+routerTemp.lsp.get(i).cost,**

**nextName);**

**int tempIndex,tempIndex1;**

**//如果节点在试探表中 且 开销《试探表中的节点就替换**

**if((tempIndex=isInTentative(node,tentative))>=0 &&**

**node.getCost()<tentative.get(tempIndex).getCost())**

**{**

**tentative.remove(tempIndex);**

**tentative.add(node);**

**}**

**else if ((tempIndex = isInTentative(node, tentative)) < 0**

**&& (tempIndex1 = isInTentative(node, confirmed)) < 0)**

**{**

**tentative.add(node);**

**}**

**}**

**//构造比较器**

**Comparator<ConfirmAndTentativeNode> comparator=new Comparator<ConfirmAndTentativeNode>()**

**{**

**public int compare(ConfirmAndTentativeNode o1, ConfirmAndTentativeNode o2)**

**{**

**if(o1.cost>=o2.cost)**

**{**

**return 1;**

**}**

**return -1;**

**}**

**};**

**//对试探表按开销进行排序**

**Collections.sort(tentative,comparator);**

**//将试探表中的最小的那个节点加入到证实表中**

**confirmed.add(tentative.get(0));**

**tentative.remove(0);**

**}**

**routerStart.setConfirmed(confirmed);**

**routerStart.setTentative(tentative);**

**}**

1. **玩一下tomcat**

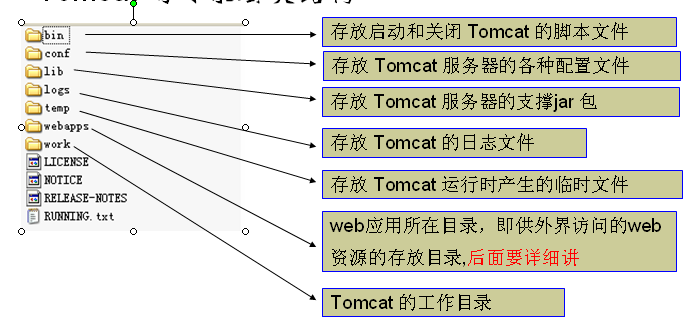


图 31 tomcat的目录结构

一个helloworld程序（配置默认主页面）。

①在web文件夹下配置WEB-INF文件夹

②在 web.xml 文件中添加配置的代码:

<welcome-file-list>

<welcome-file>hello1.html</welcome-file>

</welcome-file-list>

③通过<http://localhost:8080/web1>来访问hello1.html

web-inf目录下的 classes目录将来是存放 class文件

lib 目录将来时存放 jar文件

web.xml 配置当前这个web应用的信息.

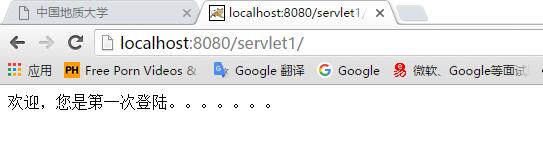


图 32 运行效果

1. **个人感悟**

首先，老师您说过要给我“优”的。

还有，五个自问自答的问题需要纸质档和电子档结合着看，因为写的时间不

一样，纸的写的早一些，并且写字太累，有些地方表述的不是太好。另外，五个问题是单独的一个文件“5个问题.doc”.

本次课程设计主要包括了wireshark，md5等几种加密算法，tcp和udp套接字编程等，通过wireshark实验，我对wireshark软件有了一定的了解，基本能达到运用的要求，用wireshark软件我们做了三次握手，拥塞，和http协议等，这三个实验让我们对所学的知识有了更深一层的理解.在我们的编程实验中，它不但提高了我的编程能力，还使我对tcp的三次握手，md5加密以及javad的ＩＯ编程有了更深层次的理解，另外有老师建议，我还做了一个ＱＱ的好友列表下选择好友进行通信的实验，通过这个实验我更加深入的了解了多线程编程，此次课程设计提高了我的实际动手能力了。

在本次课程设计中，有过失败也有过成功，有过喜悦也有过悲伤，我曾遇到过很多困难，如tomcat的安装配置等，但是在老师和同学们的帮助下我最终克服了这些困难，最终成功的做出了实验，此次课程设计在老师和同学们的共同努力下取得了圆满的成功，使我对网络有更深一步的了解，达到了此次课程设计的要求。对老师和同学表示由衷的感谢。

1. **参考材料**
2. <https://www.ibm.com/developerworks/cn/java/javamd5/> （Md5）
3. RFC1312 （Md5）
4. <http://blog.163.com/zhaowenchao_vip/blog/static/1715151442011752234456/> (sha1)
5. http://www.cnblogs.com/hellojava/archive/2013/01/25/2861127.html（tcp套接字）
6. <http://blog.csdn.net/todd911/article/details/9378319>
7. <http://www.cnblogs.com/zh2000g/archive/2010/03/22/1692002.html>
8. <http://blog.csdn.net/hguisu/article/details/7418161>
9. <http://www.tuicool.com/articles/NFBvye>
10. <http://blog.csdn.net/wintys/article/details/3525643/>
11. <http://158067568.iteye.com/blog/901052>
12. http://blog.csdn.net/reille/article/details/7106195
13. http://blog.csdn.net/liujun13579/article/details/7773945
14. 《计算机网络系统方法 第四版 中文版》
15. 课程PPT
16. 19212-01-17-何小平.pdf 和 计算机网络课程设计B(2016).doc（老师给的模板）等