****

计算机网络课程

实验报告

学 号 09019106

姓 名 牟倪

东南大学计算机科学与工程学院

二0 21 年 12 月

**目 录**

[一、实验一：应用层实验 2](#_Toc90715015)

[1.1 Wireshark抓包软件初探 2](#_Toc90715016)

[1.1.1 使用显示过滤器 3](#_Toc90715017)

[1.1.2 使用捕获过滤器 3](#_Toc90715018)

[1.2 基于Wireshark抓包结果的HTTP协议分析 4](#_Toc90715019)

[1.3基于Wireshark抓包结果的DNS协议分析 5](#_Toc90715020)

[1.4测试curl命令（选做） 7](#_Toc90715021)

[1.5基于telnet的get命令测试（选做） 8](#_Toc90715022)

[1.6基于telnet的SMTP服务测试（选做） 9](#_Toc90715023)

[1.7 测试tracert命令并解析其过程 11](#_Toc90715024)

[1.7.1 tracert 的执行过程与原理 11](#_Toc90715025)

[1.7.2 实验过程与结果 11](#_Toc90715026)

[1.8 使用nslookup查询域名信息 12](#_Toc90715027)

[1.8.1 直接查询 12](#_Toc90715028)

[1.8.2 查询其它记录 12](#_Toc90715029)

[1.9 心得体会 13](#_Toc90715030)

[二、实验二：TCP协议分析 14](#_Toc90715031)

[2.1建立TCP连接：三次握手 14](#_Toc90715032)

[2.1.1 寻找三次握手数据包 15](#_Toc90715033)

[2.1.2 分析三次握手数据包 16](#_Toc90715034)

[2.2 结束TCP连接：四次挥手 17](#_Toc90715035)

[2.2.1 寻找四次挥手数据包 18](#_Toc90715036)

[2.2.2 分析四次挥手数据包 20](#_Toc90715037)

[2.3 TCP报文格式分析 22](#_Toc90715038)

[2.4 重置TCP连接：RST数据包 23](#_Toc90715039)

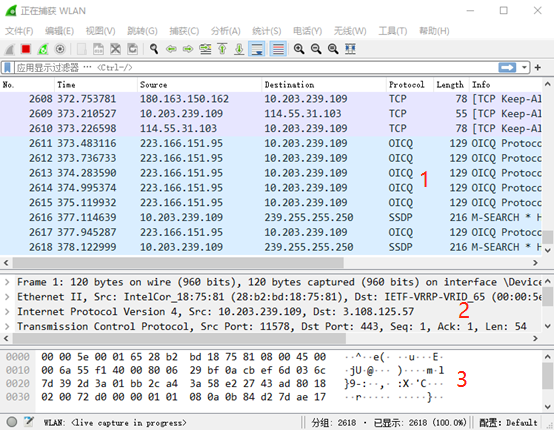
[2.5 心得体会 23](#_Toc90715040)

# 一、实验一：应用层实验

1.1 Wireshark抓包软件初探

Wireshark是目前全球使用最广泛的开源网络数据包分析工具。通过Wireshark抓包，我们能够分析各层网络协议的性能、掌握通信主体的运行情况，确认带宽分配和时延大小、查看应用的快慢并改进优化，识别网络中存在的攻击或恶意行为、解决网络异常和故障。

Wireshark界面如图所示：



1号窗口展示的是wireshark捕获到的所有数据包的列表。注意最后一列Info列是wireshark组织的说明列并不一定是该数据包中的原始内容。

2号窗口是1号窗口中选定的数据包的分协议层展示。底色为红色的是因为wireshark开启校验和验证而该层协议校验和又不正确所致。

3号窗口是1号窗口中选定的数据包的源数据，其中左侧是十六进制表示右侧是ASCII码表示。另外在2号窗口中选中某层或某字段，3号窗口对应位置也会被高亮。

1.1.1 使用显示过滤器

在页面上端的“应用显示过滤器…”框内输入过滤语句，即可使用显示过滤器，Wireshark只会显示符合过滤条件的包。

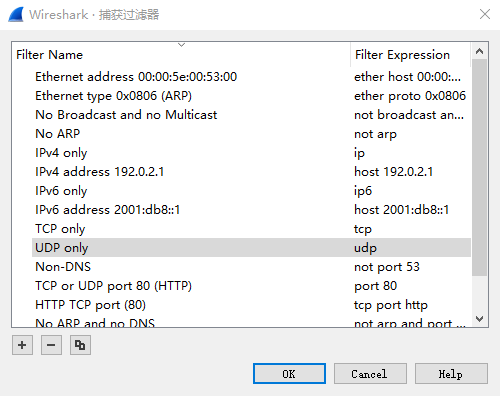
以下为示例语句：

|  |  |
| --- | --- |
| 语句 | 含义 |
| tcp.port == 80 || udp.port == 80 | 80端口的数据包 |
| !udp | 不显示UDP协议数据包 |
| tcp.len == 0 | 长度=0的TCP数据包 |
| frame.time > 2021-12-17 | 帧到达时间晚于2021-12-17 |

1.1.2 使用捕获过滤器

捕获过滤器表达式作用在wireshark开始捕获数据包之前，只捕获符合条件的数据包，不记录不符合条件的数据包。

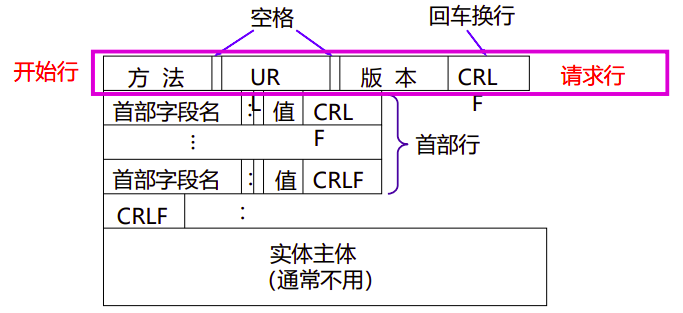
点开菜单栏的“捕获”，选择“捕获过滤器”，即可使用捕获过滤器。界面如下所示：



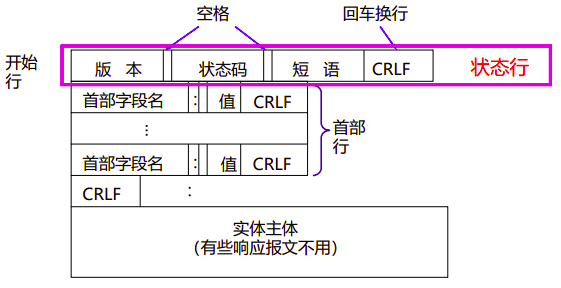
1.2 基于Wireshark抓包结果的HTTP协议分析

HTTP报文格式如图所示：

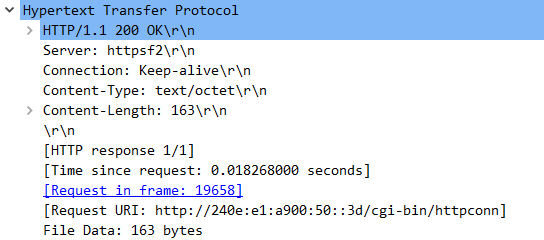
（请求报文）



（响应报文）



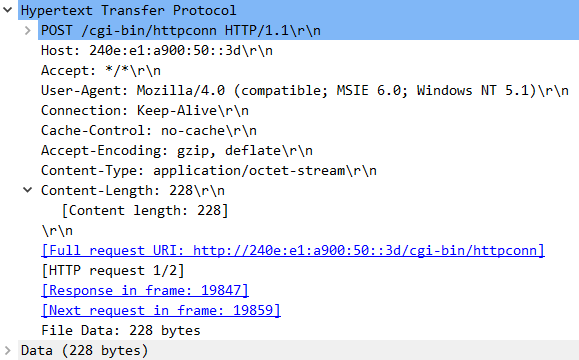
如图所示，这是一个捕获的HTTP数据包。



第一行为开始行，在响应报文中也被叫做状态行，由版本+状态码+短语组成。在这个包里，HTTP版本为1.1，状态码为200，短语为ok。

接下来为HTTP首部行。可以看出server信息，connection为keep-alive，content-type内容类型和相应的content-length等。

我们再捕获一个HTTP包：

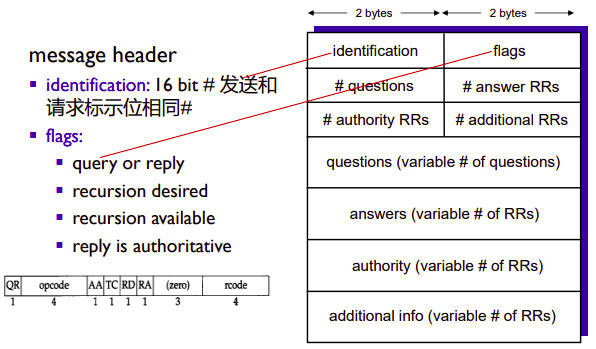


如图所示，这个包的首部有更多信息，证明HTTP首部是可变长的。

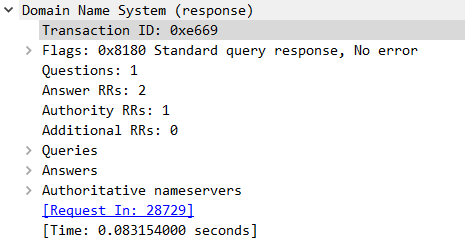
第一行为开始行，在请求报文中也被叫做请求行，由方法+请求资源的url+版本组成。在这个包里，方法为POST，url为/cgi-bin/htpconn，HTTP版本为1.1。

1.3基于Wireshark抓包结果的DNS协议分析

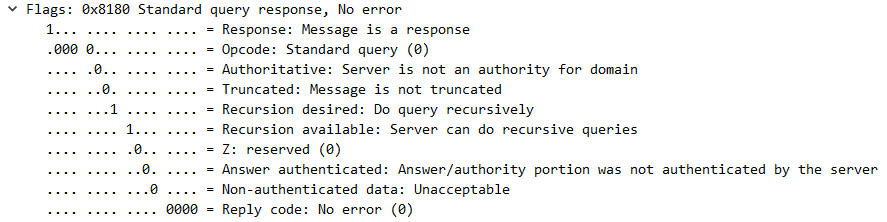
DNS报文格式如图所示：



如图所示，这是一个捕获的DNS数据包。

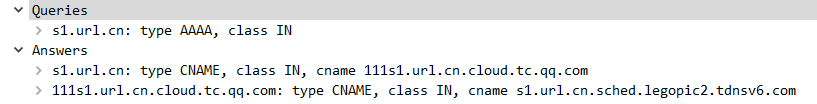


点开Flags，如图所示：



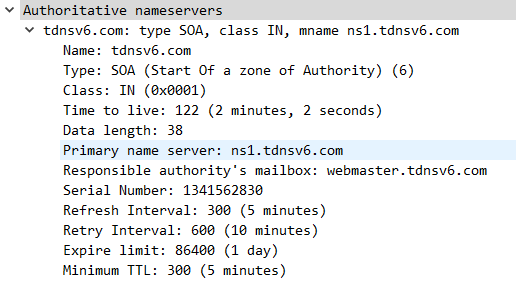
可以看出，该报文的性质为response，查询类型为standard query，AA位=0表示server不是authoritative server。

点开Queries和Answers，如图所示：



代表了查询的内容与相应的结果。

点开Authoritative nameservers，如图所示：

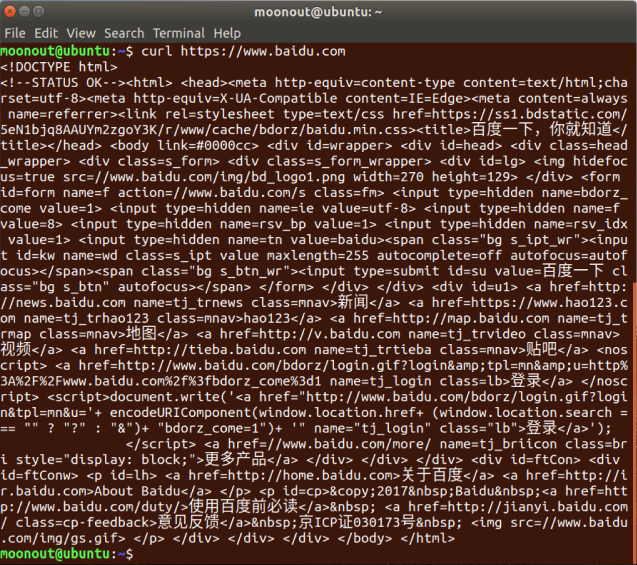


这应该是关于的Authoritative servers的信息。可以看出Authoritative servers的域名、TTL（本域名→IP地址映射信息保留时长）等。

1.4测试curl命令（选做）

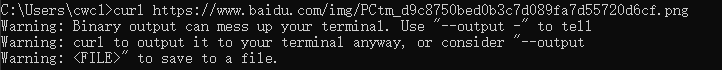
curl（Command-line Uniform Resource Locator，命令行统一资源定位器）是使用命令行访问网页URL的工具 。Linux系统自带curl工具。

打开linux虚拟机，打开命令行，输入curl https://www.baidu.com，执行结果如下图所示：



curl 命令直接得到了网页的源代码，没有HTTP报文头部等信息。

用curl访问png图片，如图所示：

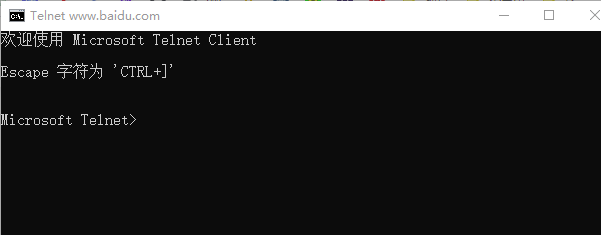


curl提示我，直接输出二进制文件，可能满屏都是乱码，可以用<FILE>来存储文件。

1.5基于telnet的get命令测试（选做）

Telnet协议是TCP/IP协议家族中的一员，是Internet远程登陆服务的标准协议和主要方式。在电脑上用telnet 连接到服务器，在telnet程序中输入命令，这些命令就会在服务器上运行，就像直接在服务器的控制台上输入一样。

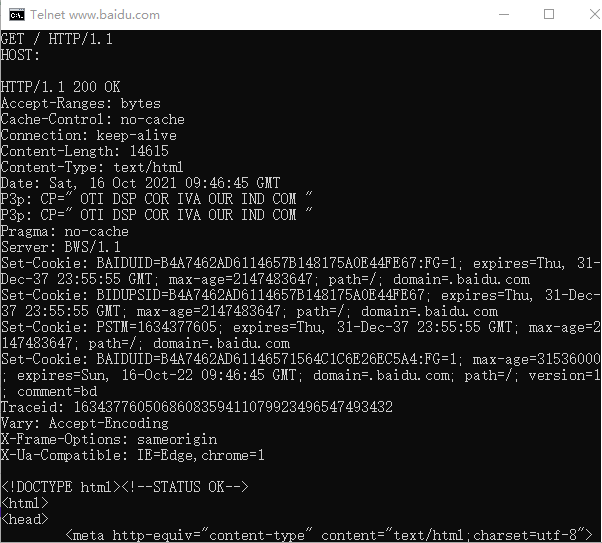
打开cmd，输入telnet www.baidu.com 80。回车后，屏幕为全黑。此时我们用快捷键Ctrl+]打开本地回显功能，这样就可以看见我们所键入的内容了。



单击回车，进行编辑状态。输入GET / HTTP/1.1后回车，接着输入HOST:然后按回车，一个简单的HTTP请求就完成了；只要再次按下回车，便可以向服务器递交这个请求了。

GET表示请求方式，/表示请求的根目录下的文件，HTTP/1.1表示HTTP协议版本，HOST:是一个消息头。GET HTTP必须大写，否则请求无法发送。

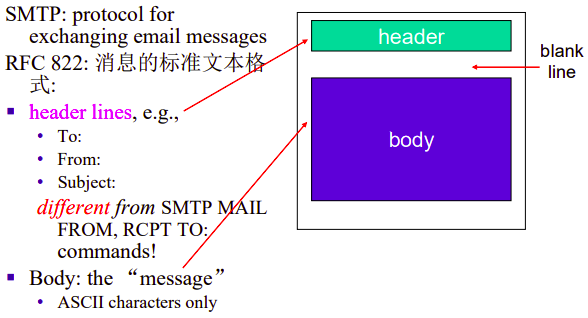
服务器的返回内容HTTP响应报文，如图所示：



虽然telnet很强大，但是换一个视角，一旦入侵者与远程主机建立了 Telnet 连接，入侵者便可以使用目标主机上的软、硬件资源，而入侵者的本地机相当于一个只有键盘和显示器的终端。因此，telnet也可以作为远程入侵的工具。

1.6基于telnet的SMTP服务测试（选做）

SMTP（simple mail transfer protocol），简单邮件传输协议，定义了将邮件在 mail server 之间推送的服务。SMTP报文格式如图所示：

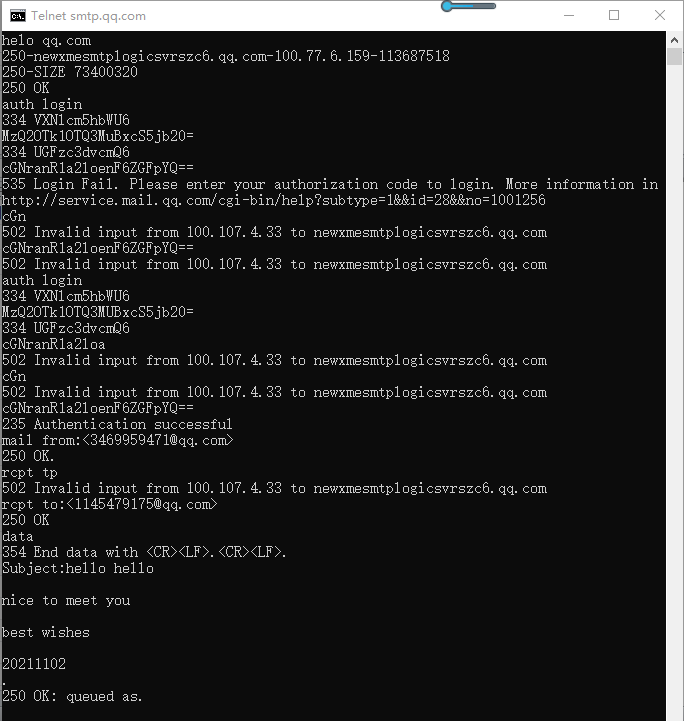


我们利用telnet，远程登陆QQ邮箱的SMTP服务器，测试SMTP服务。

操作步骤：

* 打开cmd，输入telnet smtp.qq.com 587；
* 输入helo qq.com，向服务器表明身份；
* 输入auth login；
* 输入邮箱的 base64 编码（[3469959471@qq.com](mailto:3469959471@qq.com)的base64编码）；
* 输入开启 IMAP/SMPT 时授权码的base64 编码（无空格）；
* 输入发送人邮箱mail from:<xxxxxxxxxx@qq.com>；
* 输入收信人邮箱rcpt to:<xxxxxxxxxx@qq.com>；
* 输入 data+回车，成功后就可以进行邮件正文编辑；
* 邮件输入格式：
  + Subject: 主题文本
  + （空一行）
  + 邮件内容…
  + . （句点单独一行，表示邮件结束）
* 输入quit退出。

实验结果：



对方收到的邮件：



1.7 测试tracert命令并解析其过程

Tracert 命令（trace router的缩写）是路由跟踪实用程序，用于确定 IP 数据包访问目标所采取的路径。命令用 IP 生存时间 (TTL) 字段和 ICMP 错误消息来确定从一个主机到网络上其他主机的路由。

1.7.1 tracert 的执行过程与原理

tracert 利用 IP 生存时间字段（TTL）来确定数据包的路径：

* 首先，tracert 向目的地送出一个 TTL 为1的数据包。当路径上的第一个路由节点收到这个数据包时，它将TTL减1，此时 TTL 变成0，所以该路由器会将此数据包丢掉，并送回一个“ICMP time exceeded”消息（包括发 IP 包的源地址、IP 包的所有内容、路由器的 IP 地址）。tracert 收到这个消息后，便得到了路由节点的地址。
* 接着，tracert 再送出另一个 TTL 是2的数据包，发现第2个路由器，再送出TTL为3的数据包……
* tracert 每次将TTL加1来发现下一个路由器，直到数据包到达目的地为止。由于tracert通过UDP数据包向不常见端口（30000以上）发送数据包，因此数据包到达目的地后，会收到“ICMP port unreachable”消息，可以根据这个消息判断是否到达目的地。

1.7.2 实验过程与结果

打开 cmd，输入tracert www.baidu.com，执行结果如下图所示：



* 最左侧一列：数据包经过路由节点的标号；
* 中间三列：连接到每个路由节点的时间、返回时间、多次连接反馈的平均值。时间越小，连接速度越快。
* 最后一列：路由节点的 IP 地址。

第7个路由节点的结果是“\*”，表示ICMP包返回时间超时，无法联通。可能的原因有很多，最有可能是“该路由节点某些端口禁止ping”。

1.8 使用nslookup查询域名信息

nslookup可以用来查询DNS记录，查看域名解析是否正常，在网络故障的时候用来诊断网络问题。

1.8.1 直接查询

使用nslookup domain[dns-server]，可以直接查询域名对应的IP地址。

打开cmd，输入nslookup baidu.com。成功查询到了IP地址。

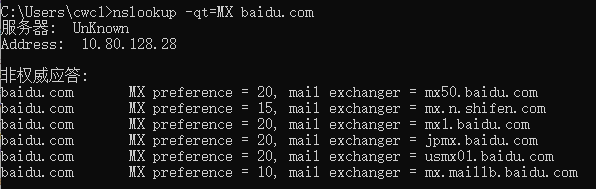


发现两次应答中，IP地址顺序不同，或许与负载均衡算法中“将一串IP地址以循环队列的顺序发送”有关。

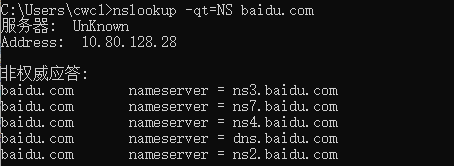
1.8.2 查询其它记录

命令格式：nslookup -qt=type domain[dns-server]。其中，type为查询类型，如A和AAAA都表示地址记录，CNAME表示别名记录，MX表示邮件服务器记录，NS表示名字服务器记录，PTR表示反向记录，RP表示负责人记录，SRV表示TCP服务器信息记录。

打开cmd，输入nslookup -qt=MX baidu.com，即可得知baidu.com的邮件服务器记录记录。如图所示：



输入nslookup -qt= NS baidu.com，即可得知baidu.com的名字服务器记录。如图所示：



1.9 心得体会

经过本次实验，我熟悉了Wireshark的界面和筛选器操作，但对筛选器代码的编写还不是很熟练。我了解了curl、telnet、nslookup等命令的基本使用，并通过抓包/telnet连接对HTTP、SMTP、DNS协议建立了更加直观的概念。

看着Wireshark抓到满屏的数据包，我深深地为计算机网络体系结构的庞大、宏伟、完备、无处不在而感到震撼感动，似乎也理解了计算机网络研究/应用的迷人之处。信息流一刻不停地刷新，我的电脑每时每刻都在接收海量的数据包，想到这里就不仅感慨计算机网络技术对传统生活方式的颠覆，如果我出生在30年前，想必完全料不到如今计算机网络的发展。

遇到问题：nslookup查询不到域名，更改WLAN连接的属性，将DNS服务器从手动设置改为自动设置即可。

# 二、实验二：TCP协议分析

TCP（Transmission Control Protocol，传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议。

实验内容：

* 观察 TCP 三次握手与四次挥手报文。注意报文收发过程中，双方TCP状态的变化。
* 以捕获的报文为依据，分别画出本次TCP连接三次握手与四次挥手的时序图，结合 TCP 状态机，在双方各阶段标出对应的TCP状态。
* 选择其中一个 TCP 报文，配合Wireshark截图，分析该报文TCP首部各字段的定义、值及其含义。

预期目标：

* 理解 TCP 的工作过程，掌握 TCP 工作原理与实现；
* 学会运用 Wireshark 分析 TCP 连接管理、流量控制和拥塞控制的过程，发现 TCP 的性能问题。

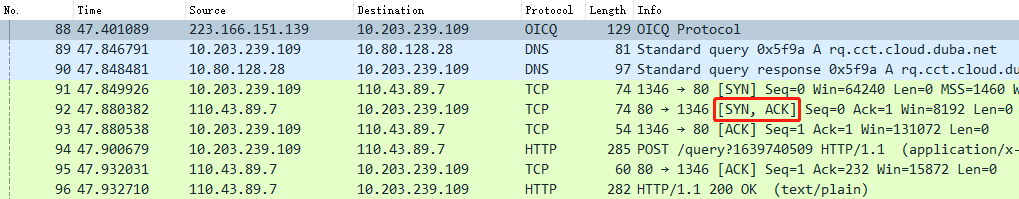
2.1建立TCP连接：三次握手

TCP连接的三次握手过程大致如下：

* 第一次握手：请求建立连接。
  + 客户端发送连接请求报文段，将SYN置1，Sequence Number置为x；
  + 然后，客户端进入SYN\_SEND状态，等待服务器的确认；
* 第二次握手：服务器收到SYN报文段。
  + 服务器收到客户端的SYN报文段，需要对这个SYN报文段进行确认，设置Acknowledgment Number为x+1（即Sequence Number+1，表示希望接收到下一个包）；
  + 同时，服务器自己还要发送SYN请求信息，将SYN置1，Sequence Number置为y；
  + 服务器端将上述所有信息放到一个报文段（即SYN+ACK报文段）中，一并发送给客户端，此时服务器进入SYN\_RECV状态；
* 第三次握手：客户端收到服务器的SYN+ACK报文段。
  + 设置Acknowledgment Number为y+1，向服务器发送ACK报文段。
  + 客户端发送这个报文段后，进入ESTABLISHED状态；服务器收到ACK报文段，也进入ESTABLISHED状态，完成TCP三次握手。
* FLAGS置位：SYN → SYN+ACK → ACK。

2.1.1 寻找三次握手数据包

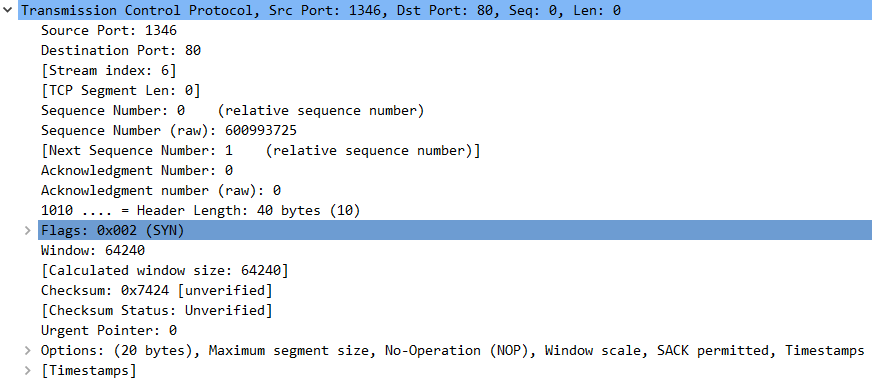
因为三次握手的第二个数据包非常特殊，SYN、ACK同时置位，所以寻找三次握手的数据包，可以使用TCP报文的FLAGS标志位进行检索。检索语句为tcp.flags.syn == 1 && tcp.flags.ack == 1。



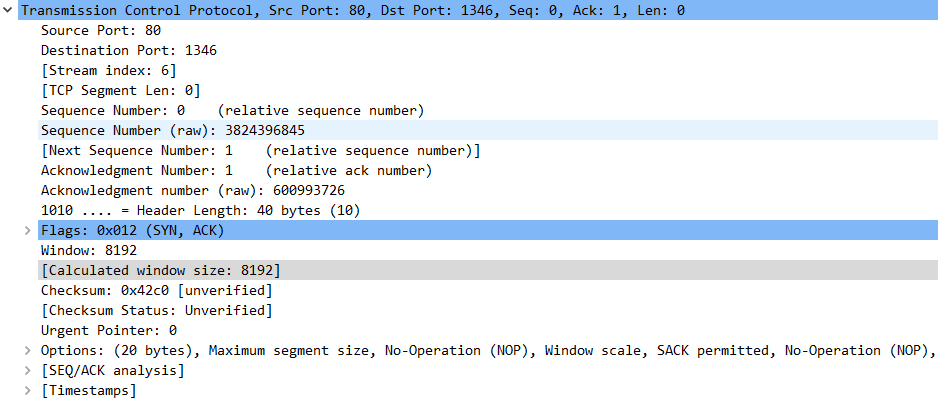
如图上的红框所示，92号包同时置位了SYN和ACK，因此91 92 93号包是一个TCP连接的三次握手。

91 92 93号包的详细信息如下：

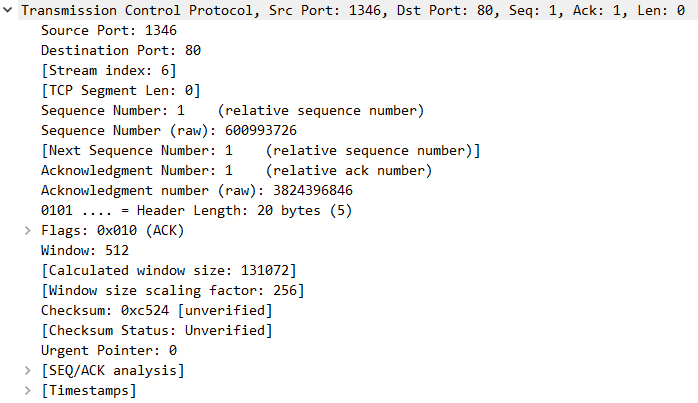
91：



92：



93：



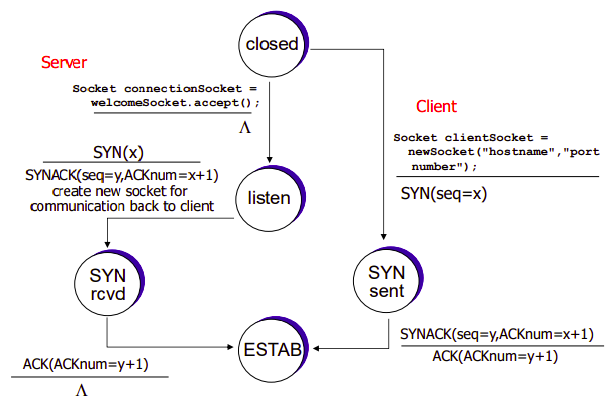
2.1.2 分析三次握手数据包

观察91 92 93号包的具体信息，可以发现：

* 91：Sequence Number=600993725，Acknowledgement Number=0；
* 92：Sequence Number=3824396845，Acknowledgement Number=600993726；
* 93：Sequence Number=600993726，Acknowledgement Number=3824396846；

与课堂上的内容一样，Seq#与ACK#存在对应关系。

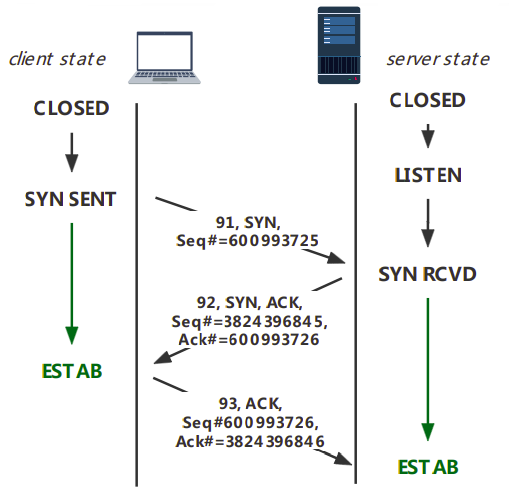
TCP三次握手的FSM（有限状态机）如图所示：



可以看出，TCP三次握手的过程大致如下：

* server首先启动并进入listen状态，准备随时接收client的连接请求。
* client发送SYN报文，进入SYN sent状态。
* server收到client发送的SYN报文，进入SYN received状态，并发送SYNACK报文。
* SYN sent状态的client收到server发送的SYNACK报文，发送ACK报文，进入Established状态，client建立连接。
* SYN received状态的server收到client发送的ACK报文，进入Established状态，server建立连接。

根据FSM的状态转换，TCP三次握手的时序图与状态如图所示：



2.2 结束TCP连接：四次挥手

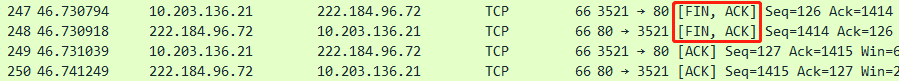
TCP通过client和server的4次交互来关闭连接，被称为四次挥手。大致过程如下：

* 第一次挥手：主机1（可以是客户端，也可以是服务器端）设置Seq#和Ack #，向主机2发送一个FIN报文段，然后主机1进入FIN\_WAIT\_1状态；这表示主机1没有数据要发送给主机2了，但仍然可以接收数据；
* 第二次挥手：主机2收到了 FIN 报文段，向主机1回一个 ACK 报文段，Ack#为Seq#加1（表示：我回复的是你上一句话）；主机1进入FIN\_WAIT\_2状态；
* 第三次挥手：主机2向主机1发送FIN报文段，请求关闭连接，同时主机2进入CLOSE\_WAIT状态；主机2告诉主机1，我也没有数据要发送了，可以进行关闭连接了；
* 第四次挥手：主机1收到主机2发送的FIN报文段，向主机2发送ACK报文段，然后主机1进入TIME\_WAIT状态；主机2收到主机1的 ACK 报文段以后，就关闭连接；TIME\_WAIT时，主机1等待 2个MSL（Maximum Segment Lifetime）后依然没有收到回复，则证明主机2已正常关闭，那好，主机1也可以关闭连接了。
* 总结：
  + 因为TCP连接是全双工的，所以每个方向都要单独关闭。
  + 收到一个FIN包（FINbit=1），意味着这一方向没有数据流动。也就是说，你给我发FIN，代表你不跟我说话了，但我还可以跟你说话。
  + 因此，两个方向都要经历“发FIN包→对方应答表示收到FIN包”的过程，加起来是四次挥手。
  + FLAGS：FIN→ACK；FIN→ACK。

2.2.1 寻找四次挥手数据包

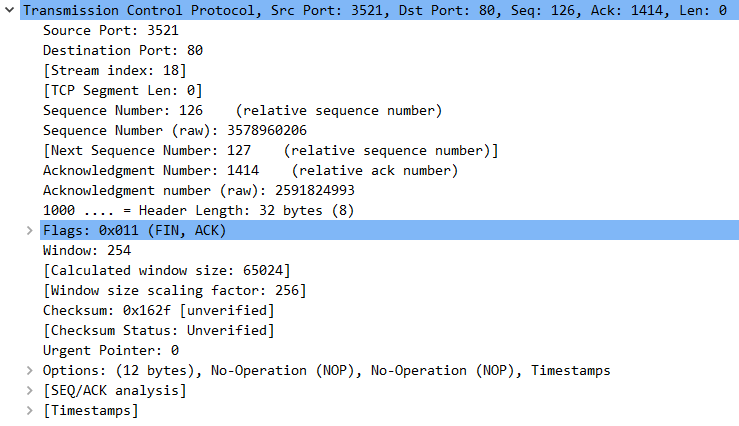
FLAGS里的FIN置位代表四次挥手数据包，因此我们使用tcp.flags.fin == 1语句来检索。

找到了一个四次挥手过程，如图所示：

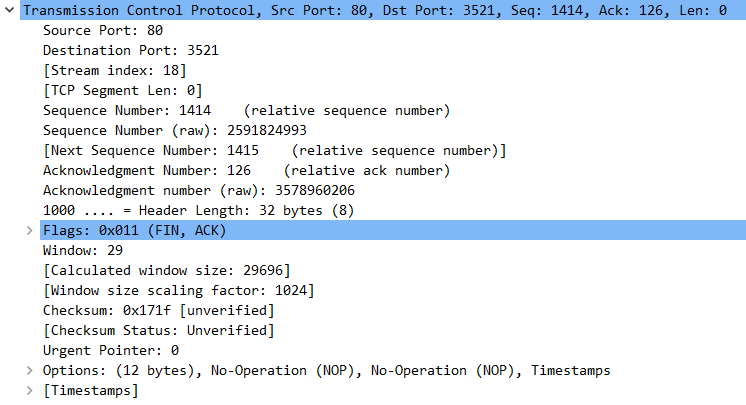


247 248 249 250号包为一个四次挥手过程。数据包的详细信息如图所示：

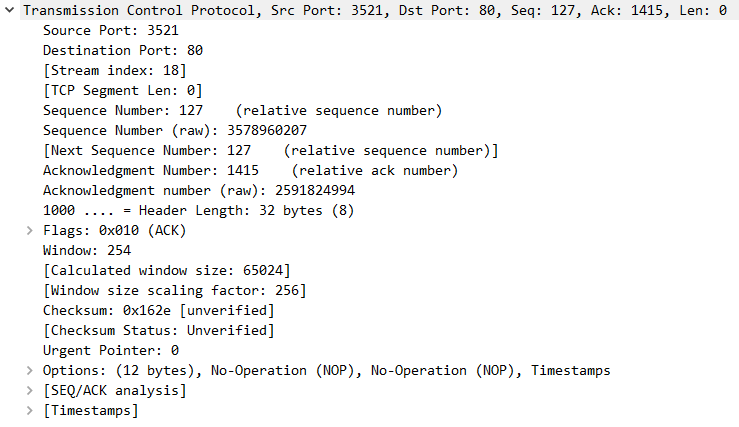
247：



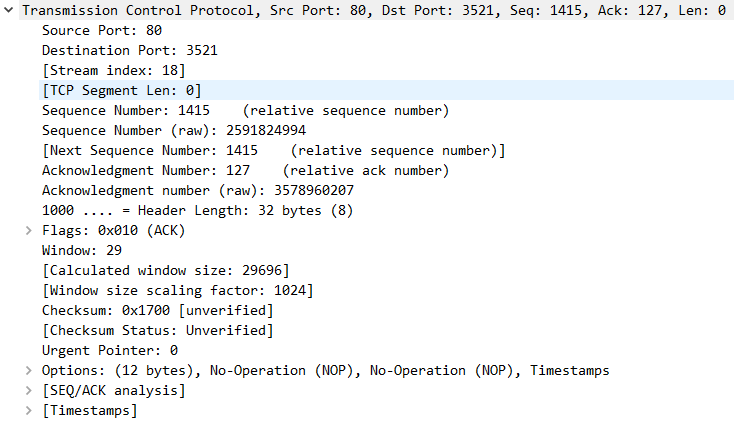
248：



249：



250：



2.2.2 分析四次挥手数据包

观察247 248 249 250号包的具体信息，可以发现：

* 247：Seq#=3578960206，Ack#=2591824993；
* 248：Seq#=2591824993，Ack#=3578960206；
* 249：Seq#=3578960207，Ack#=2591824994；
* 250：Seq#=2591824994，Ack#=3578960207；

发现了与课堂讲述不符的情况：248号包的Ack# == 247号包的Seq#，而非248号包的Ack# == 247号包的Seq#+1，249 250号包也是如此。为了验证这种现象是否普遍存在，我们再抓几组TCP四次挥手数据包。

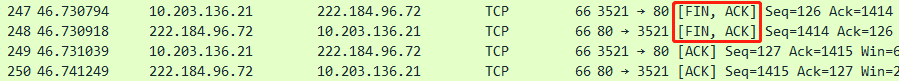
E:\temp\1639788736(1).png

这是另一组四次挥手数据包。为什么看起来只有3个包呢？四次挥手的4代表两个FIN→ACK过程，而48号包不仅应答了47号包的FIN（即发送ACK响应），也发送了自己的FIN，把两个操作合在一起了，因此变成了3个包。

这一组包的Seq#和Ack#很正常。我们再抓一组：

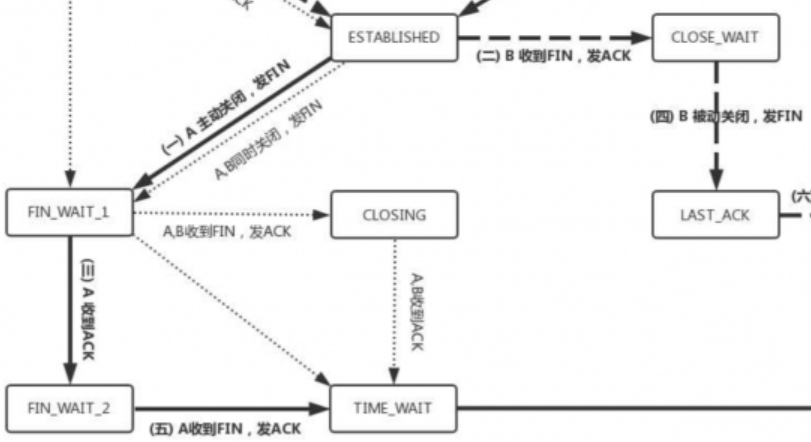
E:\temp\1639788936(1).png

这是另一组四次挥手数据包，也是看起来只有3个包。Seq#和Ack#也很正常。再回顾一下最开始的反常四次挥手包：



据此，我们合理推测，该四次挥手过程如下： client和server同时想结束通信，因此发出了247 248号包。250号包响应的是247号包的FIN，249号包回应的是248号包的FIN。

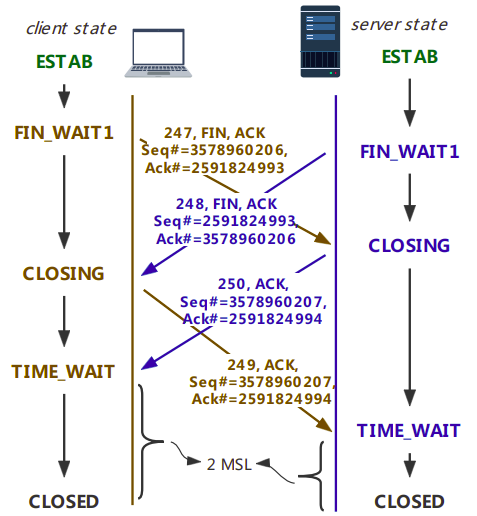
TCP四次挥手的FSM（有限状态机）如图所示（图源网络）：



可以看出，

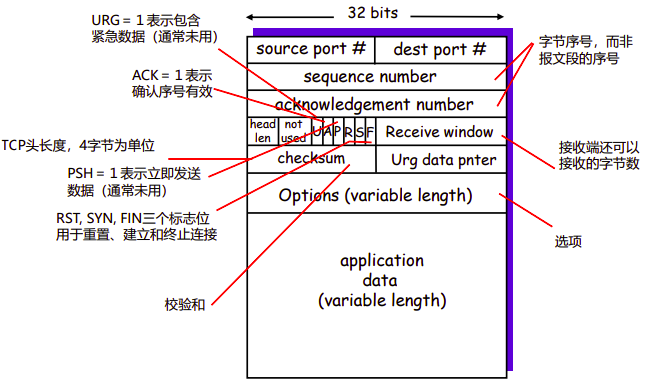
* 经典情形：
  + 主动发FIN的一方进入FIN\_WAIT\_1状态，被动收到FIN的一方进入CLOSE\_WAIT状态，并发送ACK。
  + FIN\_WAIT\_1状态的主动方收到被动方发来的ACK，进入FIN\_WAIT\_2状态。
  + CLOSE\_WAIT状态的被动方发送FIN，进入LAST\_ACK状态；FIN\_WAIT\_2状态的主动方收到被动方发来的FIN，进入TIME\_WAIT状态，并发送ACK。
  + LAST\_ACK状态的被动方收到主动方发来的ACK，进入CLOSED状态；TIME\_WAIT状态的主动方等待 2个MSL（Maximum Segment Lifetime），也进入CLOSED状态。
* 我们遇到的情形（设通信双方为AB）：
  + AB同时希望关闭连接，发送FIN包，都进入FIN\_WAIT1状态。
  + AB收到了对方发来的FIN包，都进入CLOSING状态，并发送响应FIN包的ACK。
  + AB收到了对方发来的ACK，进入TIME\_WAIT状态。
  + AB等待 2个MSL，进入CLOSED状态。

根据FSM的状态转换，TCP四次握手的时序图与状态如图所示：

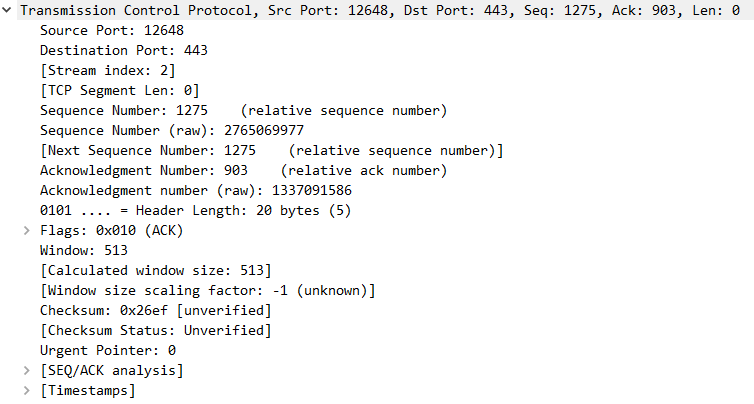


2.3 TCP报文格式分析

TCP报文格式如下：



我们选取一个TCP报文，分析它的格式：



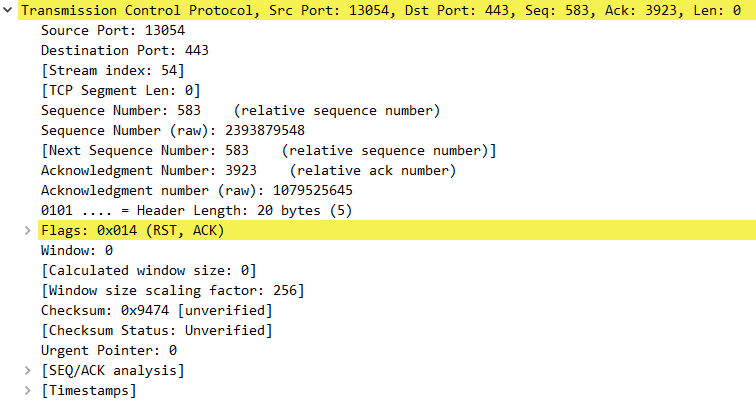
该TCP报文：

* source port=12648，dest port=443；
* seq#=2765069977，ack#=1337891586；
* FLAGS仅置位了ACK；
* 和流量控制有关的window=513；
* checksum=0x26ef；
* urgent data pointer指示接收方是否应该尽快将该报文段交给应用层，这里为0，表示不紧急。

2.4 重置TCP连接：RST数据包

在理想情况下，TCP连接都是正常关闭的。但在现实中，TCP连接会突然断掉，比如说网络的瞬时拥塞、潜在的攻击者等等。在这些情况下，就可以使用 RST标志置位的数据包，指出连接被异常终止或拒绝连接请求。

一个RST包除了RST、ACK标志置位以外，没有其他任何信息。我们可以通过tcp.flags.reset == 1语句来筛选出RST包：



如图所示，这是一个RST包的详细信息。确实，除了必要的src port、dest port、seq#、ack#和自动生成的checksum，报文头部几乎没有其他有效信息。

2.5 心得体会

经过本次实验，我对TCP三次握手和四次挥手的过程建立了直观的概念，初步熟悉了Wireshark过滤器语句的书写规范，温习了TCP报文的格式，浅显了解了Reset包的特点和应用场景。

值得一提的是，我在抓四次挥手包的时候，非常巧的抓到了“通信双方同时希望停止连接”的包，而课堂上讲的模型总是有一个主动方主动停止连接、然后被动方也跟着停止连接。这两个情形的FSM、时序图不一样，Seq# Ack#对应关系也不尽相同。经过查阅资料，我找到了包含“通信双方同时停止连接”情形的TCP FSM（即完整的FSM），大致了解了“通信双方同时停止连接”的四次挥手过程。

遇到问题：选做实验看起来非常复杂，考试周deadline太多、时间不充裕，虽然很想尝试，但也有心无力。