# 附录 华东师范大学软件工程学院实验报告

|  |  |  |
| --- | --- | --- |
| **实验课程**：计算机网络 | **年级**：2019 | **实验成绩**： |
| **实验名称**：TCP | **姓名**：邓文倩 |  |
| **实验编号**：lab6 | **学号**：10195101525 | **实验日期**：2020/12/24 |
| **指导教师**：章玥 | **组号**： | **实验时间**：2020/12/24 |

**一、实验目的**

1.熟悉使用wireshark抓取TCP数据包；

2.分析抓取到的数据包，掌握TCP数据包的结构；

3.掌握TCP数据包各字段的含义；

4.掌握TCP连接建立和释放的步骤；

5.掌握TCP数据传输过程。

**二、实验内容与实验步骤**

1. 抓包

1.以http://old.ecnu.edu.cn/site/xiaoli/2016.jpg为例，使用wget确认URL有效，或者使用你感兴趣的URL；

2.启动Wireshark，在菜单栏的捕获->选项中进行设置，选择已连接的以太网，设置捕获过滤器为tcp and host xx.xx.xx，xx.xx.xx是要从中获取内容的服务器名称，如上述例子中的old.ecnu.edu.cn；

3.捕获开始后，重复第一步，重新发送请求；

4.命令完成后，停止捕获。

（二）实验结果分析

**选择一个TCP帧，观察其协议层：**

1.根据你的理解，绘制TCP报文段的结构图（包括头部各字段的位置及大小）。

**在你捕获到的结果中，找到设置了SYN标志的TCP段及其后的数据包，完成以下问题：**

1.绘制三次握手的时序图，直到并包括建立连接后计算机发送的第一个数据包（HTTP GET请求），包括

每个数据段的序列号和Ack标号；

本地计算机发送或接收每个数据段的时间（以毫秒为单位）；

本地计算机从发送SYN段到接收到SYN-ACK段的往返时间；

2.SYN数据包上携带哪些TCP 选项？

3.传输完成后，TCP连接会以四次挥手或一端发送RST数据包的方式断开，同1一样，绘制TCP连接释放的时序图（从发出第一个FIN或RST到连接断开为止）。

**在“统计”菜单下，选择“ IO图表”，以查看数据包速率。**

* 调整过滤器为“tcp.srcport==80”仅查看下载数据包，重新绘图；
* 调整过滤器为“tcp.dstport==80”仅查看上传数据包，重新绘图；

**通过你对数据传输的理解，回答以下问题：**

1.实验中下载的大概速率为多少？（以packets/s和bits/s为单位）

2.下载内容（即TCP有效负载）占下载率的百分比是多少？

3.实验中上传的大概速率为多少？（以packets/s和bits/s为单位）

4.如果最近从服务器收到的TCP数据段的序列号是X，那么下一个发送TCP报文中的Ack号是多少？

（三）问题与思考

**在完成本实验后继续探索TCP协议：**

* 探索TCP的拥塞控制和经典AIMD策略。
* 更深入地探索TCP的可靠性机制。捕获包括段丢失的TCP连接，查看什么触发重新传输以及何时触发，另外查看往返时间估算工具。
* 查看包括SACK在内的选项的使用以了解详细信息。
* TCP是Web的基础传输层。可以通过设置并发连接来查看浏览器如何使用TCP。

**三、实验环境**

1.Wireshark

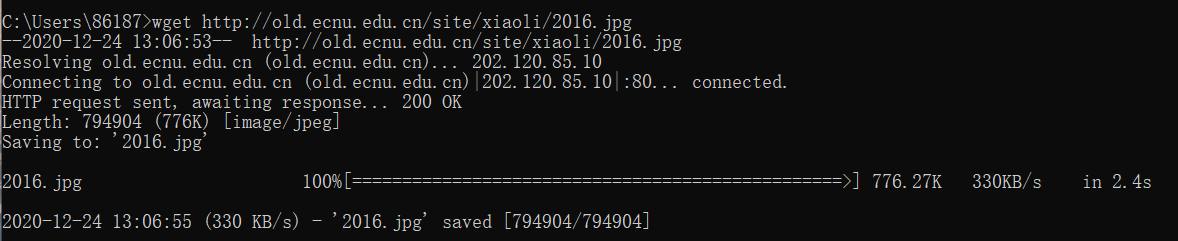
2.wget

3.Windows系统

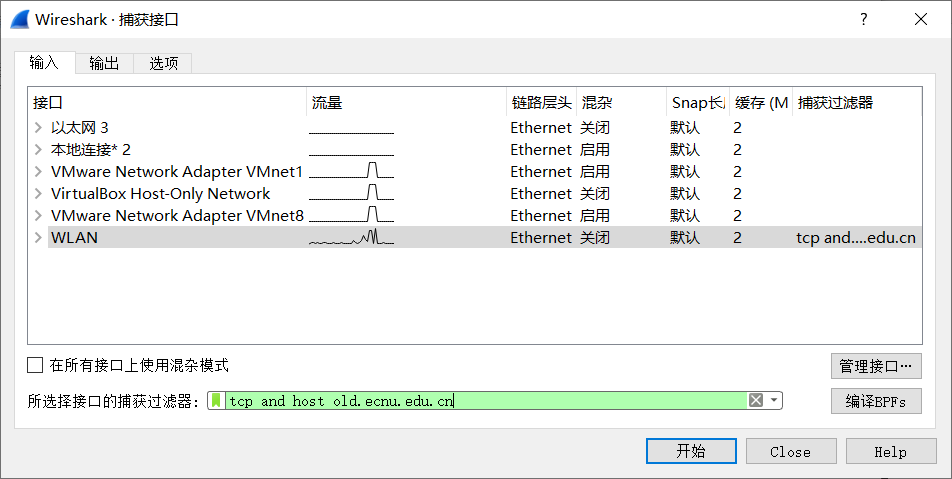
**四、实验过程与分析**

（一）抓包

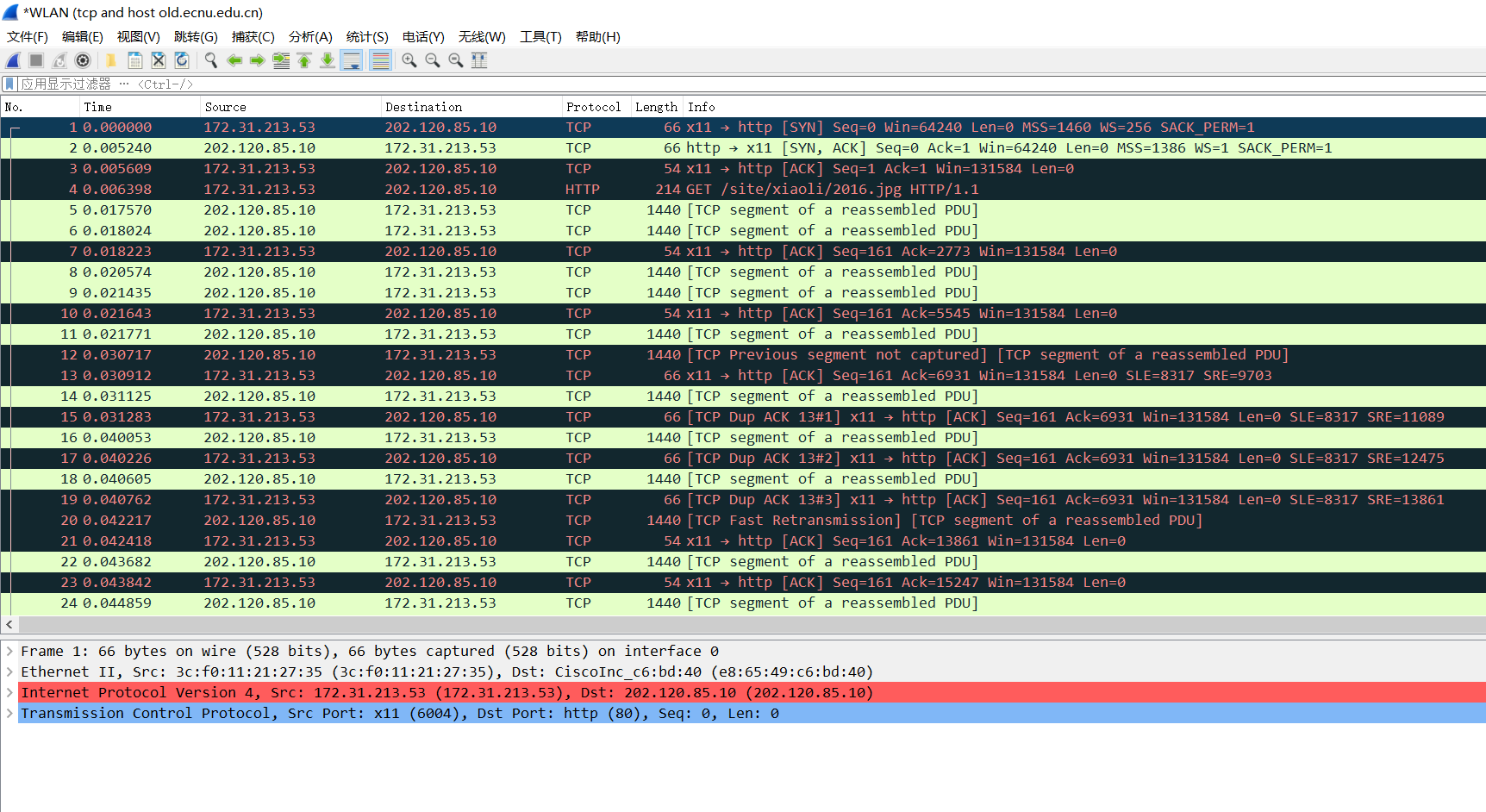
打开命令行wget：



设置Wireshark：

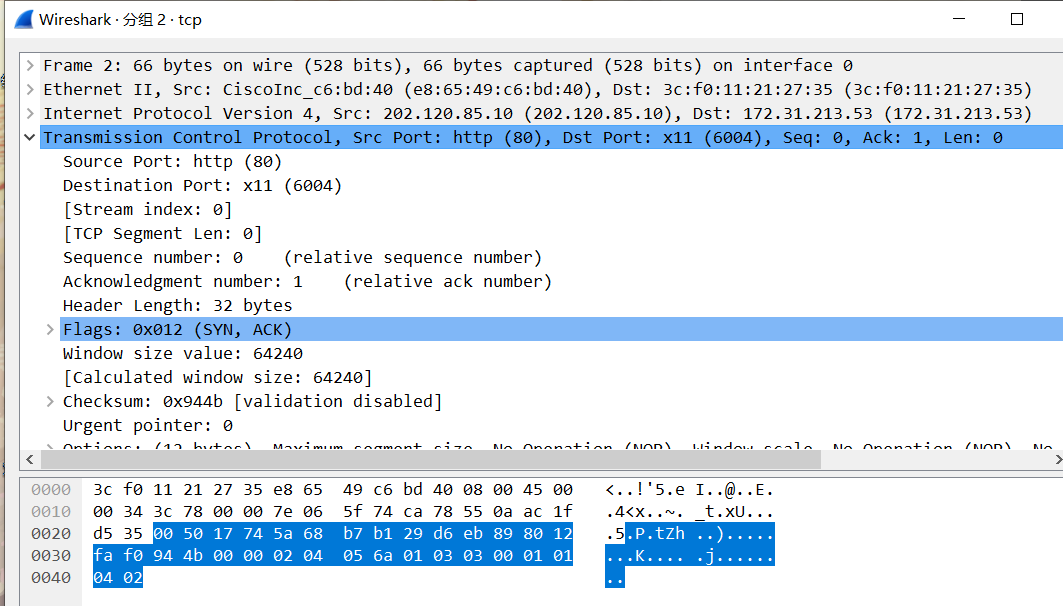


获取抓包结果：

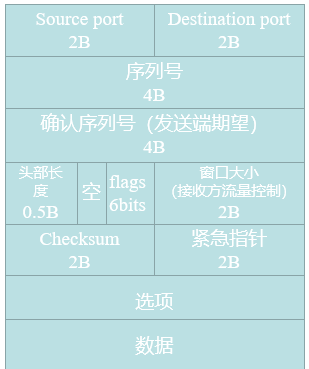


1. 实验结果分析

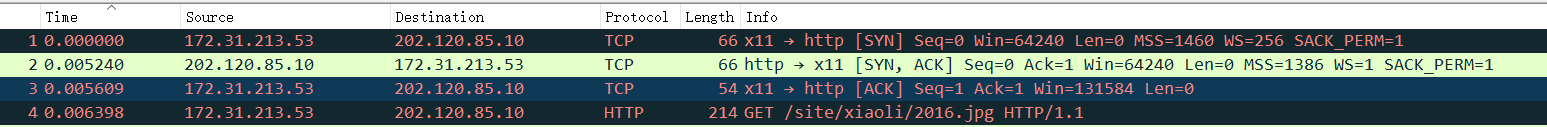
1.绘制TCP报文结构：打开一个TCP包。

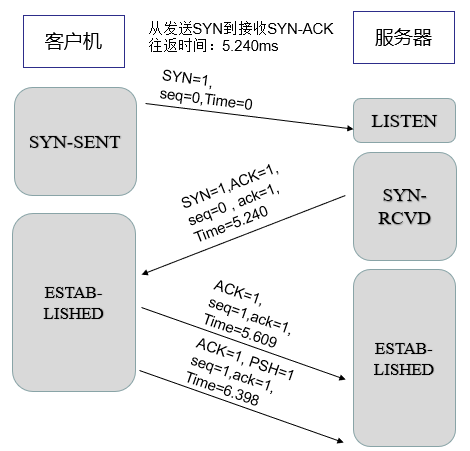


按照包中提供的信息绘制结构图：



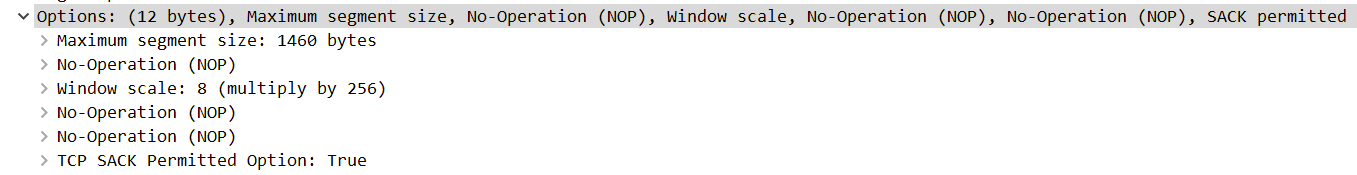
2.找到SYN标志的TCP并绘图：





3. SYN数据包携带的TCP选项：

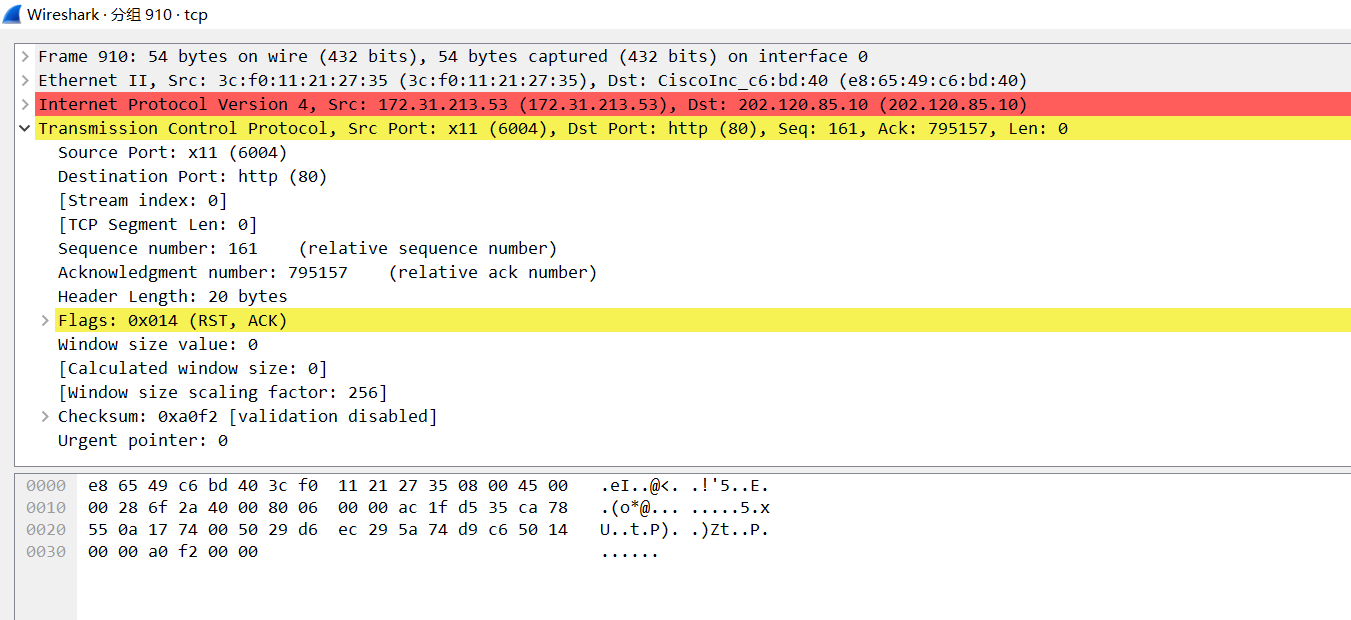
查找options字段：



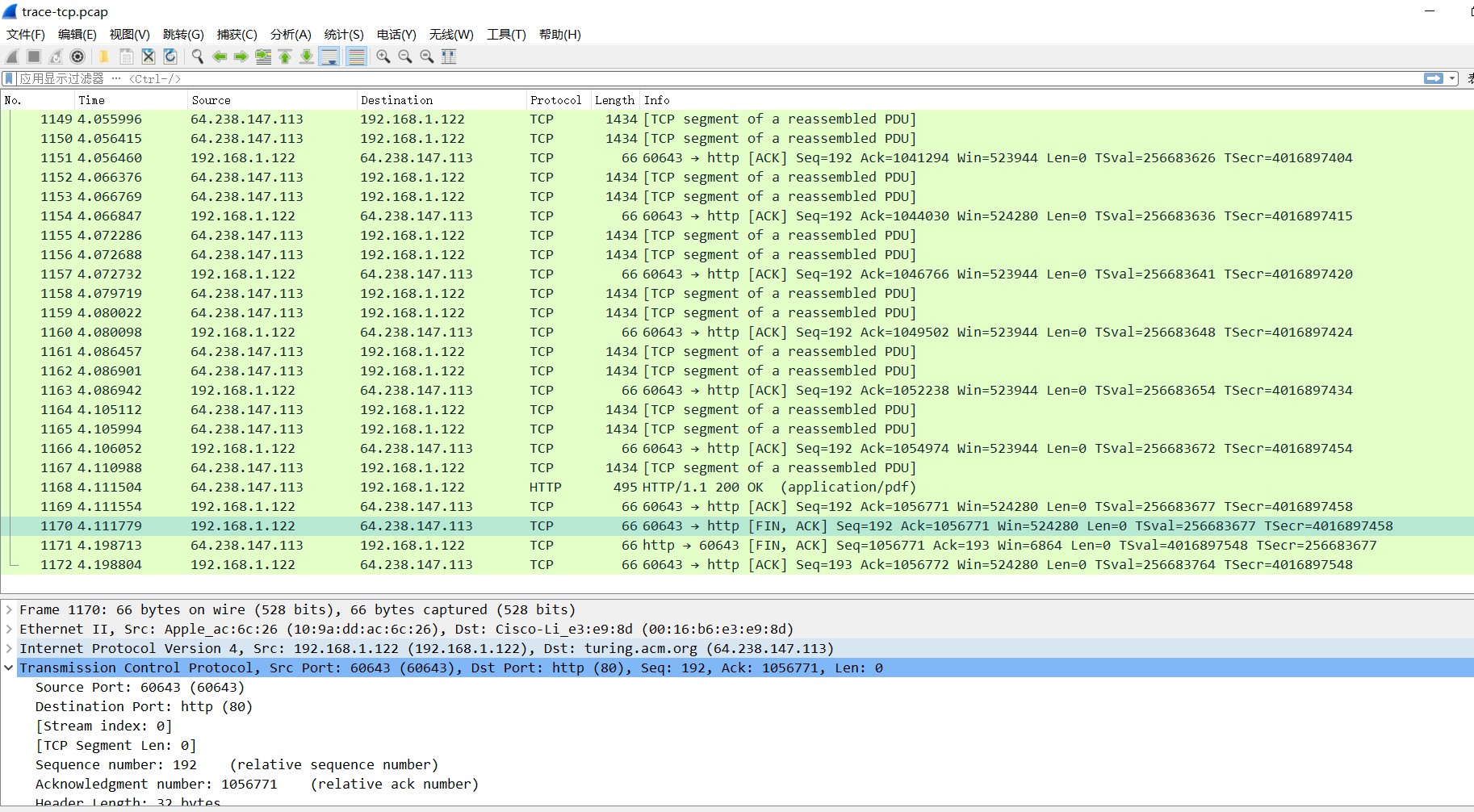
含有最大段长、NOP、窗口范围、SACK等。

4.连接释放的时序图：

找了一下自己的材料，没有四次握手的。都是本地向服务器发送RST包然直接断开的。

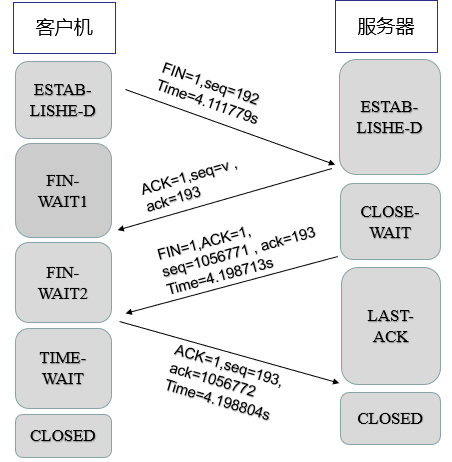


因为RST画出来效果不好，所以用了lab4stu的材料：



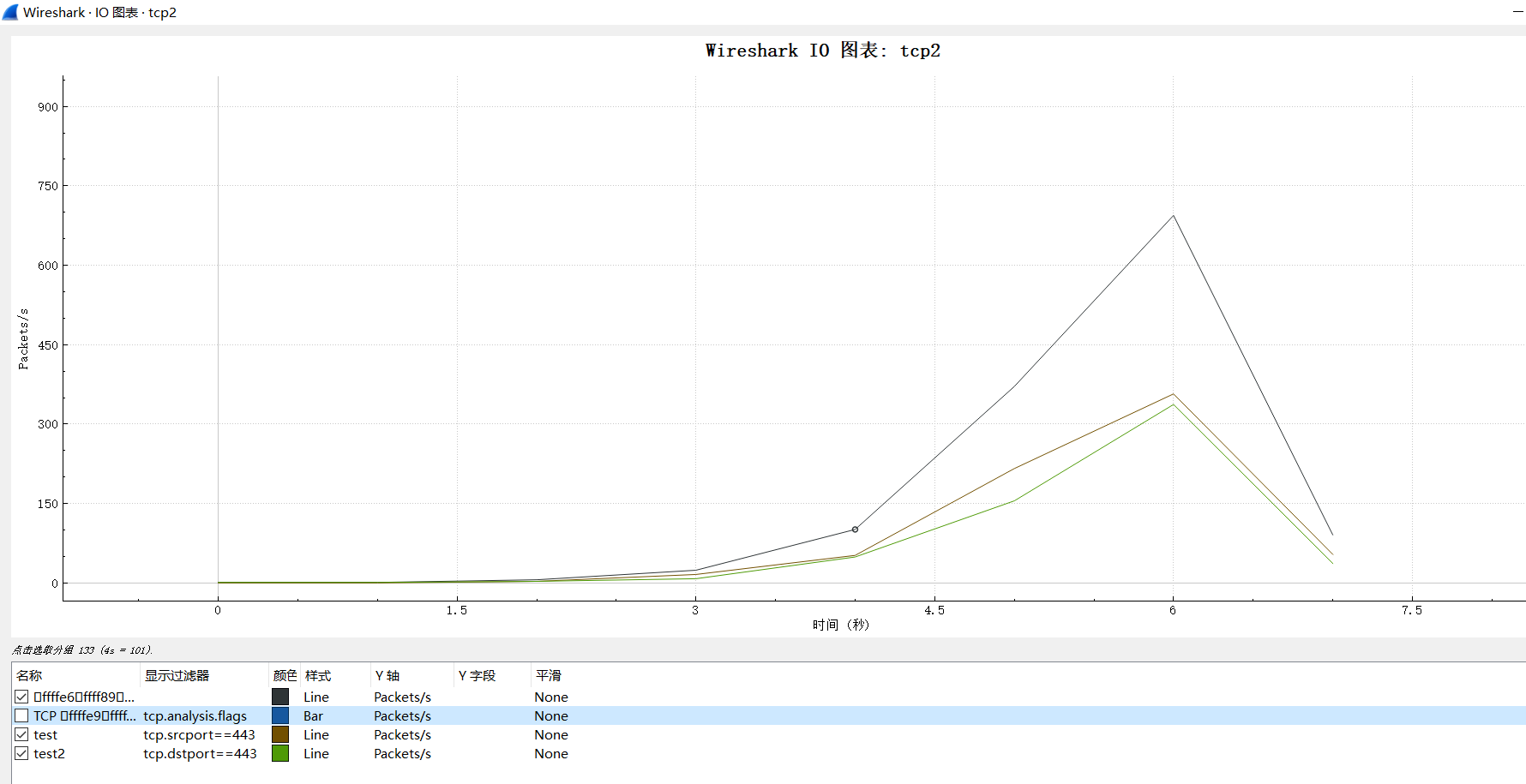
但是发现这个材料也不是四次挥手，实际上是三次。探究原因应该是服务器将FIN和ACK放在同一次里发送了。

按照理想情况绘图如下（四次挥手）：



5. 查看I/O图表。由于之前下载的文件较小，效果不好，所以重新更换了一个资源地址。

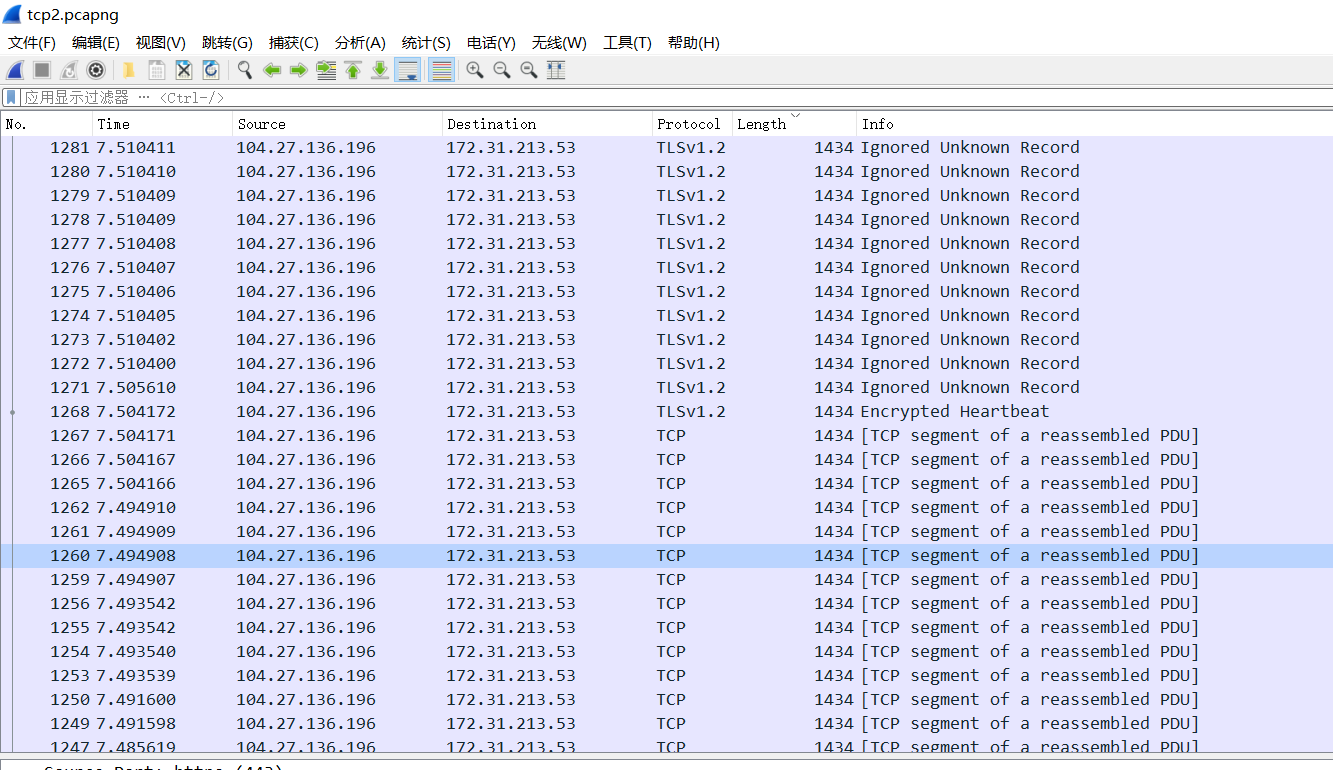
得到的新的图表如下：

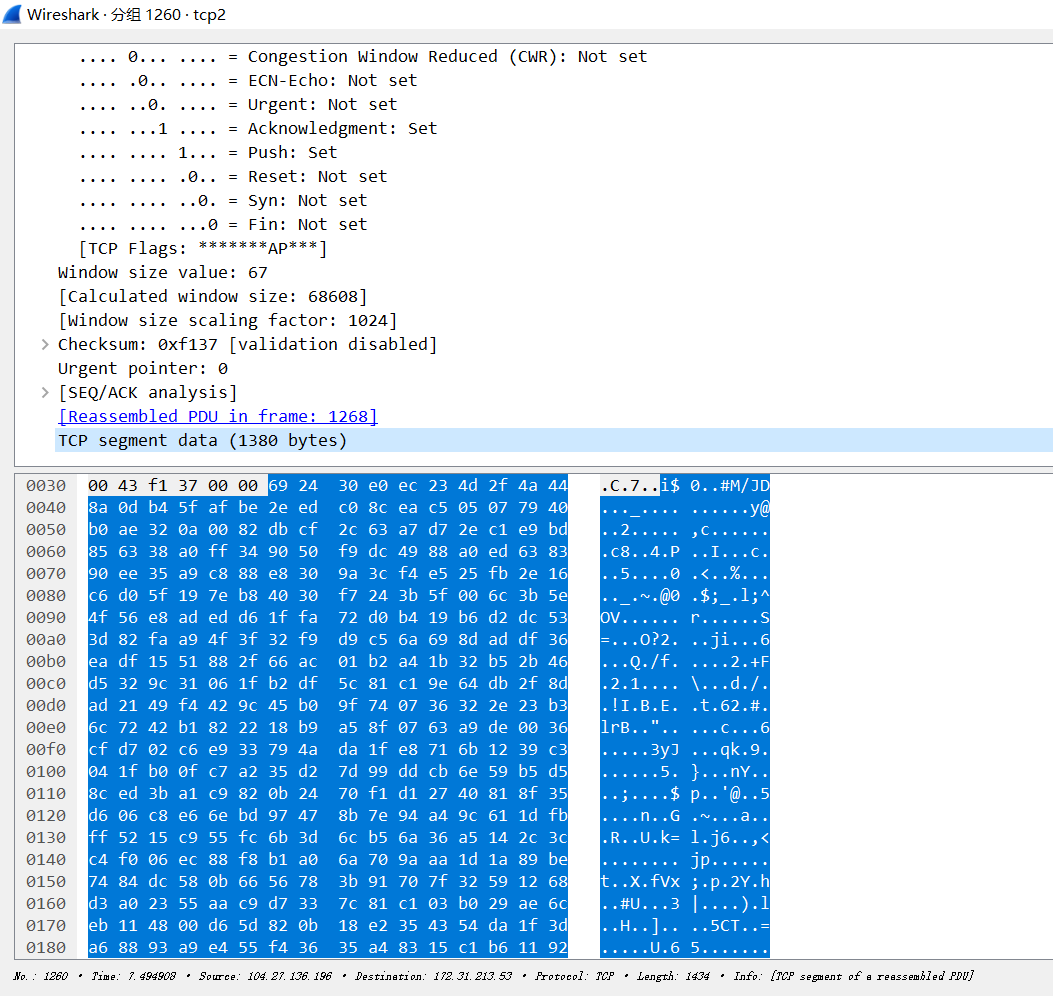


设置相应的端口过滤器，查看图表。由于数据大小等种种限制，图表形状可能与预期的不完全相同。

可以看到，下载的峰值速率为350packets/s，上传的峰值速率为340 packets/s。

下载内容占下载率的百分比通过有效负载占整个数据包的百分比进行计算：





打开最长的数据包（1434B）其中数据占1380B，那么可以得到百分比为96.3%。

由前面内容可知，如果最近从服务器收到的TCP数据段的序列号是X，那么下一个发送TCP报文中的Ack号是X+1。

1. 问题与思考

1.TCP拥塞控制：

TCP的四种拥塞控制算法：

1.慢开始

2.拥塞避免

3.快重传

4.快恢复

慢开始和拥塞避免：

发送方维持一个叫做拥塞窗口cwnd，根据网络来进行动态的调整大小，网络拥塞的时候，路由器会丢弃报文，当发送方没有按时收到确认报文，那么就知道网络发生了拥堵。

慢开始的“慢”指的是，初始cwnd=1。

拥塞避免算法的思路是让拥塞窗口cwnd缓慢增大，即每经过一个往返时间RTT就把发送方的拥塞窗口cwnd+1，而不是加倍（也就是收到两个，四个确认，仍然+1），这样cwnd就按线性增大。

快重传和快恢复：

快重传：

当接收方收到了一个失序的报文，马上报告给发送方，假如M2收到了，M3没有收到，之后的M4,M5,M6又发送了，此时接收方一共连续给发送方反馈了4个M2确认报文。那么快重传规定，发送方只要连续收到3个重复确认，立即重传对方发来的M3

快恢复：

1.当发送方连续收到三个重复确认，执行乘法减小，ssthresh减半

2.由于发送方可能认为网络现在没有拥塞，因此与慢开始不同，把cwnd值设置为ssthresh减半之后的值，然后执行拥塞避免算法，线性增大cwnd。

2.经典AIMD：

乘法减小（Multiplicative Decrease）：不论在慢开始阶段或拥塞避免阶段，只要出现超时，就把慢开始门限值减半（当前拥塞窗口的一半）

加法增大（Additive Increase）：执行拥塞避免算法后，使拥塞窗口缓慢增大，以防止网络过早出现拥塞

这两种算法结合起来被称为AIMD算法。

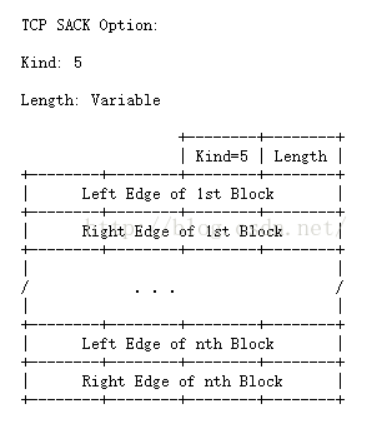
3.SACK：

标准的TCP确认机制中，如果发送方发送了0-1000序号之间的数据，接收方收到了0-100、300-1000，那么接收方只能向发送方确认101，这时发送方会重传所有101-1000之间的数据，实际上这是不必要的，因为有可能仅仅是丢了一小段而已，但是在标准的TCP确认机制中，发送方无法感知这一事情，只能重传从101开始的所有数据。

为了优化这种情况，必须让发送方知道更多的接收信息，所以发展出了SACK选项。SACK特性是TCP的一个可选特性，是否启用需要收发双发进行协商，通信双发在SYN段或SYN+ACK段中添加SACK允许选项通知对端本端是否支持SACK，如果双发都支持，那么后续连接态通信过程中就可以使用SACK选项了。所以SACK允许选项只能出现在SYN段中。

SACK是一个TCP的选项，来允许TCP单独确认非连续的片段，用于告知真正丢失的包，只重传丢失的片段。





**五、实验结果总结**

在老师和助教的讲解下，我熟悉了TCP数据包的结构，掌握了TCP各字段的含义和TCP的传输过程。

**六、附录**