实验五 TCP

张舒恒 PB19030888

实验目的

1. 了解 TCP 的 SEQ 和 ACK 序列号和确认号在 TCP 协议中的作用
2. 了解 TCP 建立连接三次握手的过程
3. 了解 TCP 的拥塞控制算法
4. 进行 TCP 连接性能的计算
5. 加强对 TCP 报文段结构的了解

实验环境

pc一台，win10操作系统，wireshark工具，浏览器

实验步骤

1.启动 Web 浏览器。 在 http://gaia.cs.umass.edu/wireshark-labs/alice.txt 查看Alice in Wonderland 的 ASCII 档案文件， 将此文件存储在本地。

2.打开 http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.

Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

If you have followed the instructions for the TCP Wireshark Lab, you have *already* downloaded an ASCII copy of Alice and Wonderland from http://gaia.cs.umass.edu/wireshark-labs/alice.txt and you also *already* have the Wireshark packet sniffer running and capturing packets on your computer.

Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your computer.

选择文件 未选择任何文件

Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to send a copy of alice.txt over an HTTP connection (using TCP) to the web server at gaia.cs.umass.edu. After clicking on the button, wait until a short message is displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin analyzing the TCP transfer of alice.txt from your computer to gaia.cs.umass.edu!!

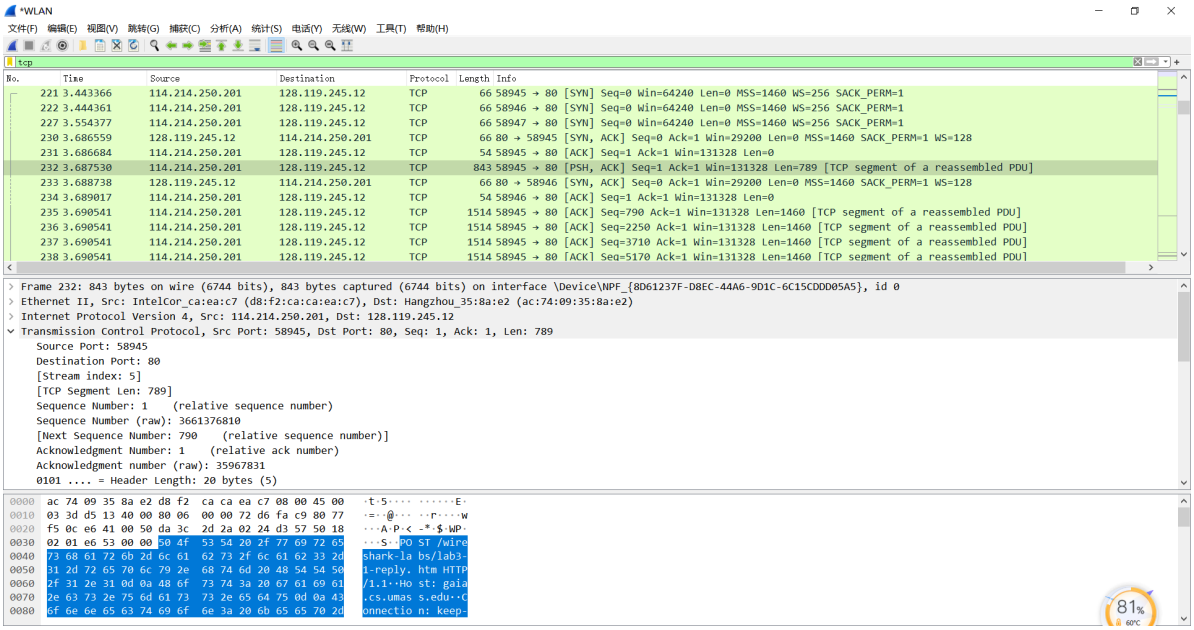
Upload alice.txt file

3.使用此表单中的"选择文件"按钮在计算机上输入包含 Alice in Wonderland的文件名(完整路径名)

4.启动 Wireshark 并开始数据包捕获

5.返回浏览器，按"Upload alice.txt file"按钮将文件上传到 gaia.cs.umass.edu服务器。文件上传后，浏览器窗口中会显示一条简短的祝贺消息。

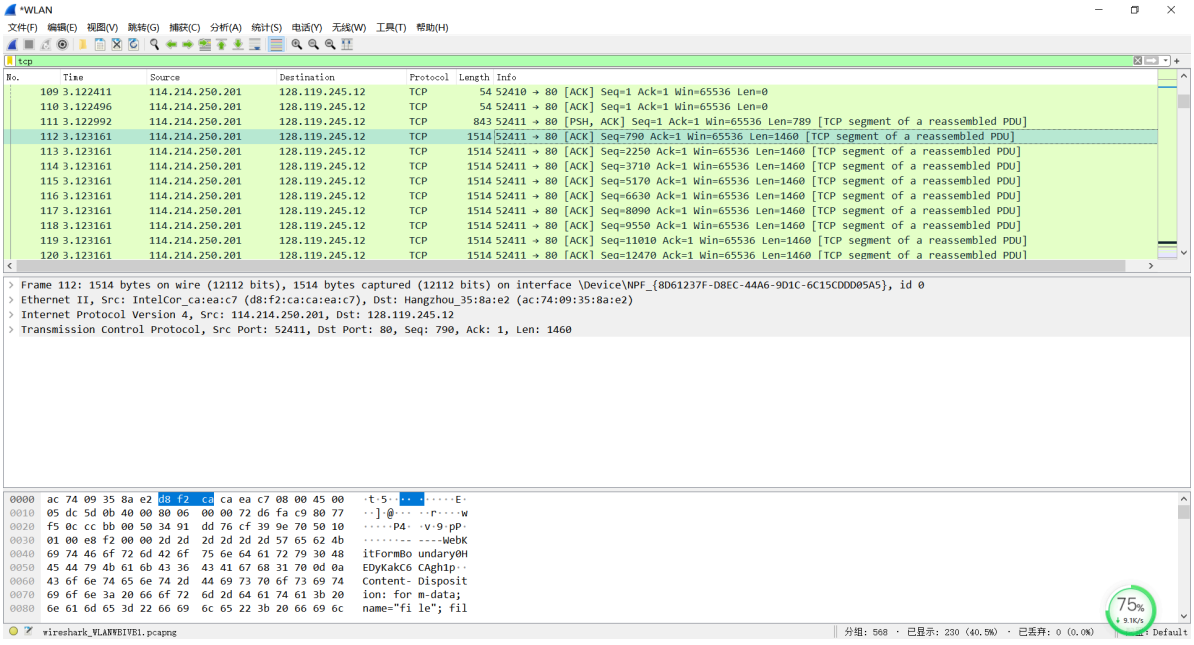
6.停止 Wireshark 数据包捕获，截图并分析



问题回答

1.将文件传输到 gaia.cs.umass.edu 的客户端计算机（源）使用的 IP 地址和 TCP 端口号是什么?

作者 IP：192.168.1.102，TCP 发送端口 1161(下图为作者抓包结果)



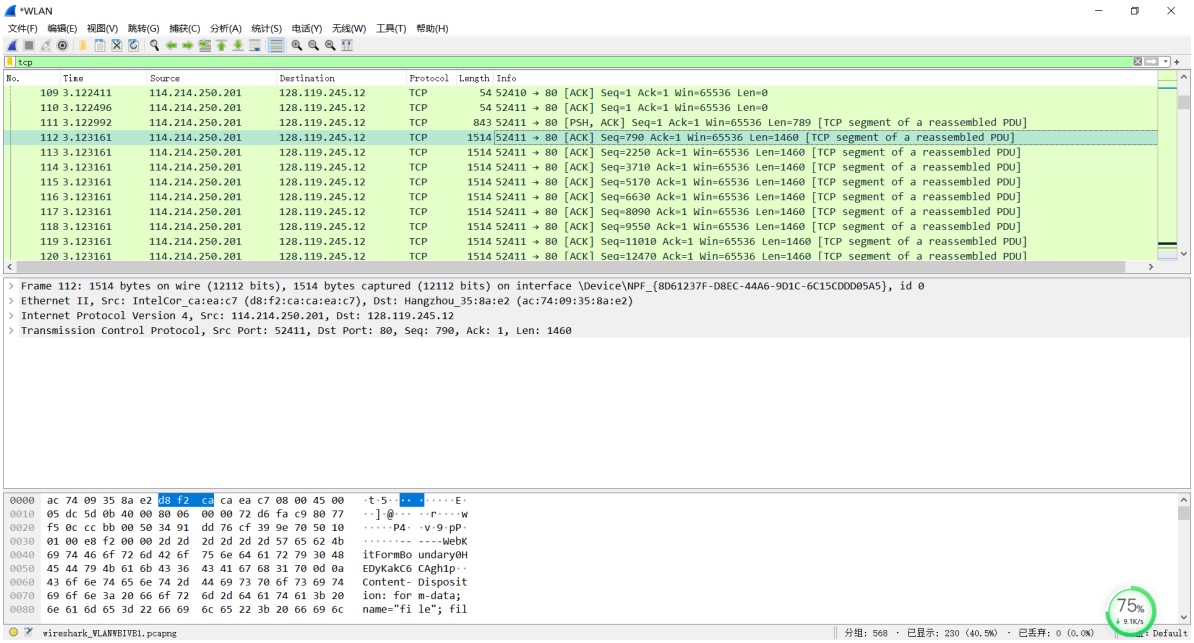
2.gaia.cs.umass.edu 的 IP 地址是什么？在哪个端口号上发送和接收此连接的 TCP 区段?

接收 IP：128.119.245.12，TCP 接受端口 80

3.客户端计算机（源）将文件传输到 gaia.cs.umass.edu 所使用的 IP 地址和 TCP 端口号是多少?

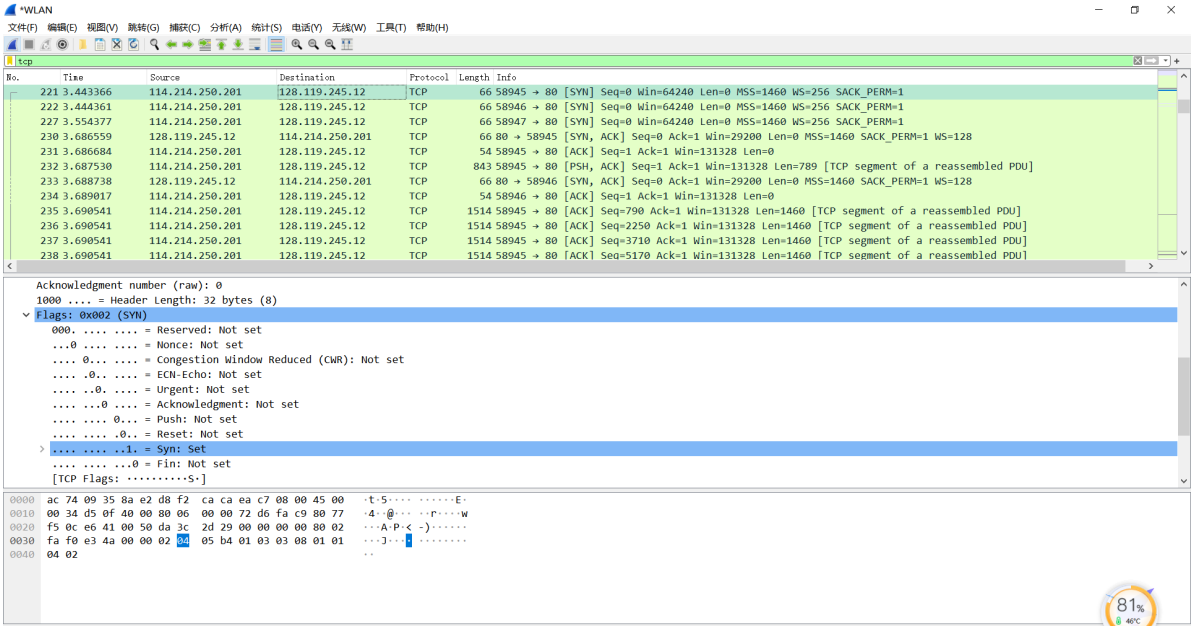
我的 IP：114.214.250.201，TCP 发送端口 52411

接收 IP：128.119.245.12，TCP 接受端口 80(下图为我的抓包结果)



4.

SEQ(序列号)=0，客户端应该发送 SYN 请求请求建立连接，这里找到发送的第一个请求并且发现客户端将 SYN 标志置为 1 用来请求建立连接。

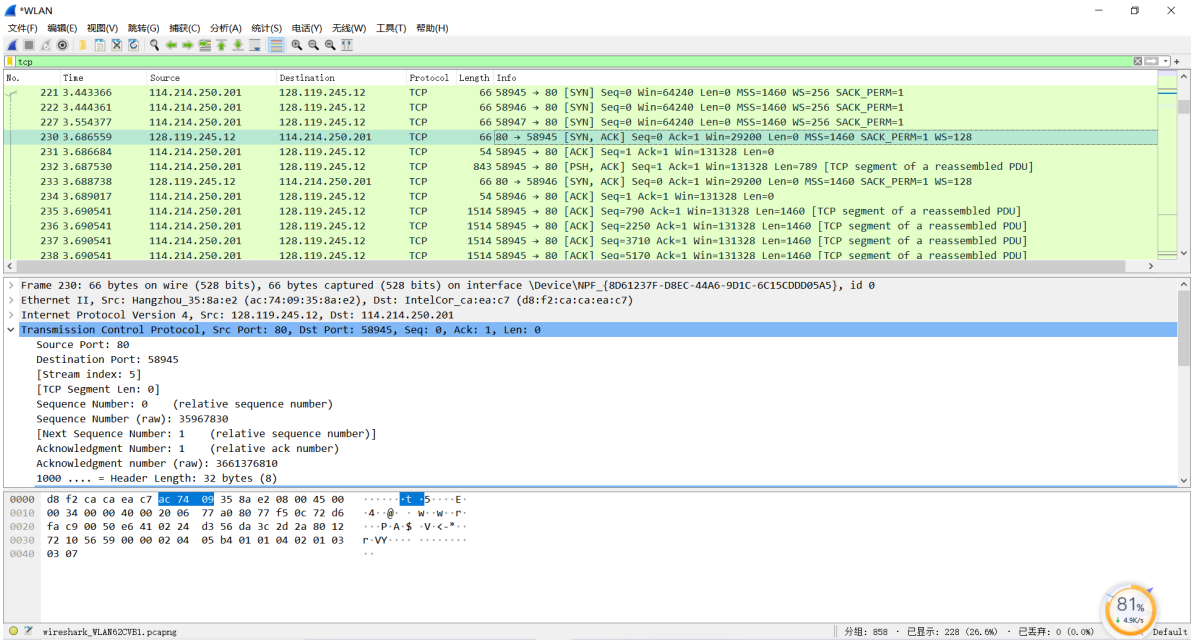


5.

seq = server_isn = 0 ;

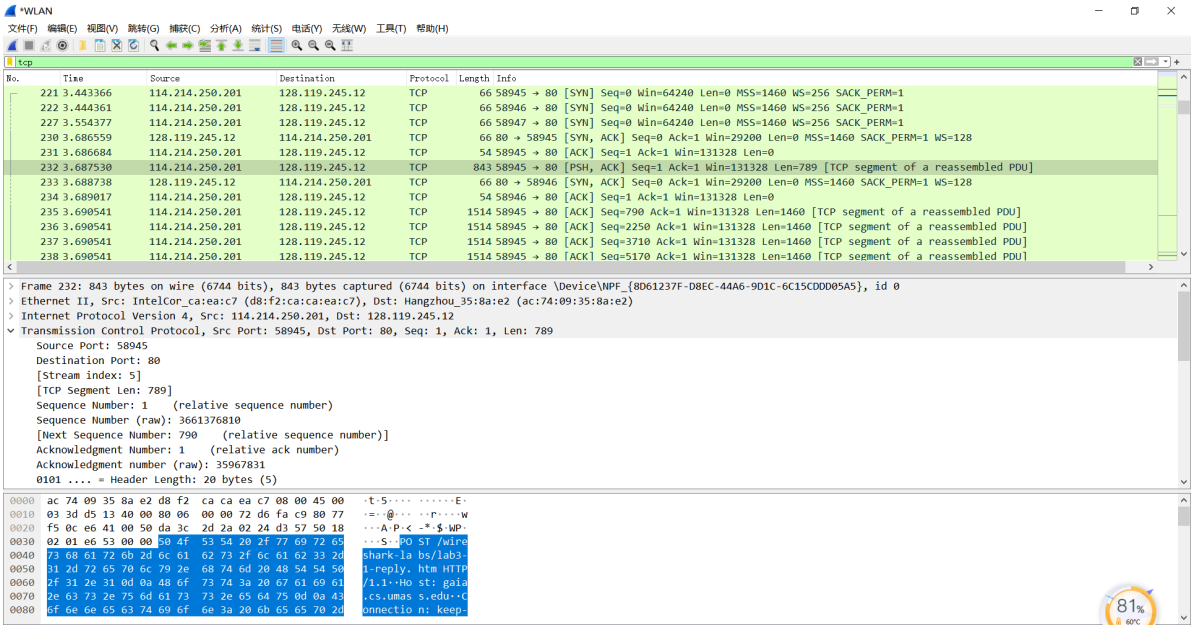
ack = client_isn + 1 =1;

标志状态位：SYN=1 且 ACK=1，说明这是一个SYNACK



6.

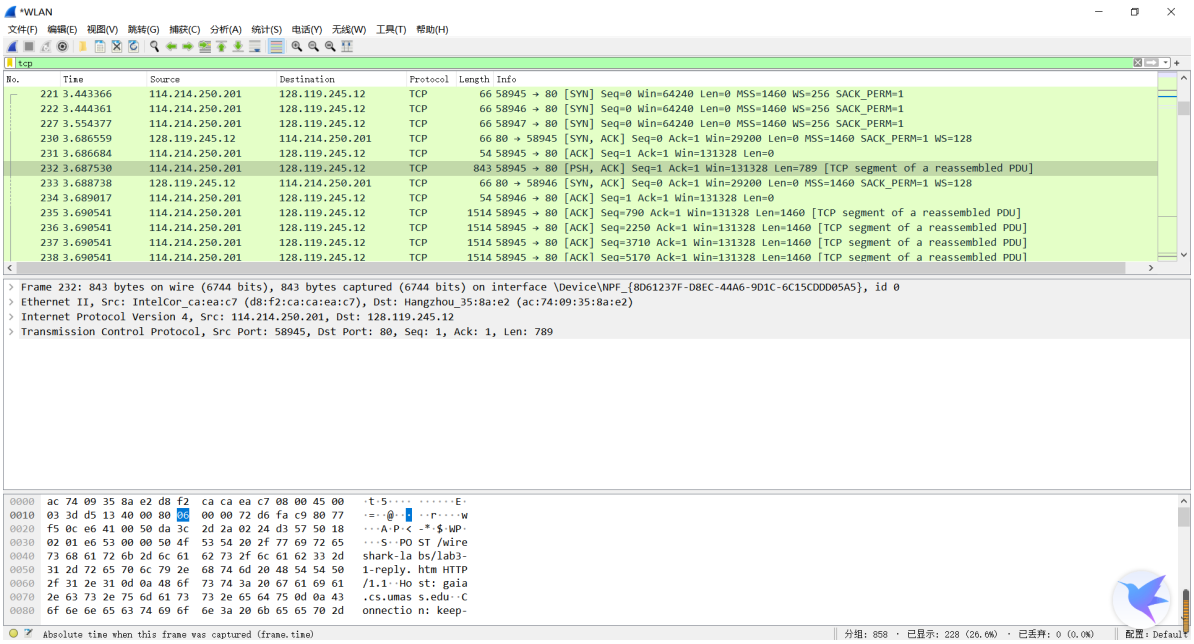
包含HTTP POST命令的TCP报文段位置紧随三次握手之后，seq=1

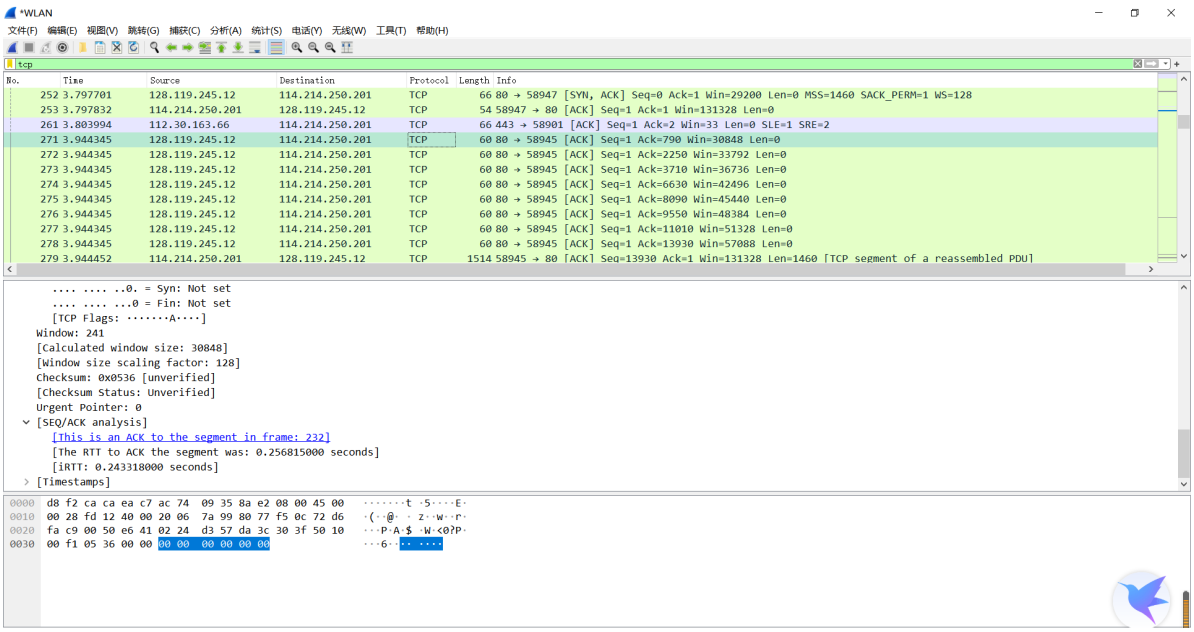


7.

NO.1 HTTP-POST 连接的 TCP 区段

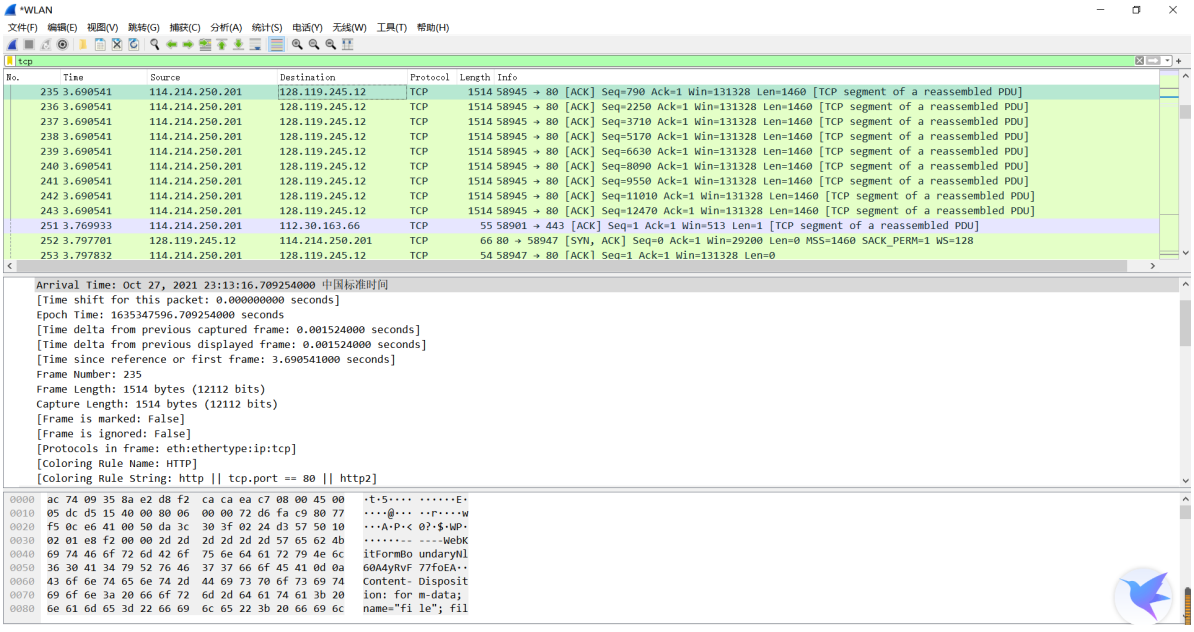
序列号：1，发送时间：3.687530s，接受时间：3.944345s，RTT：0.256815s

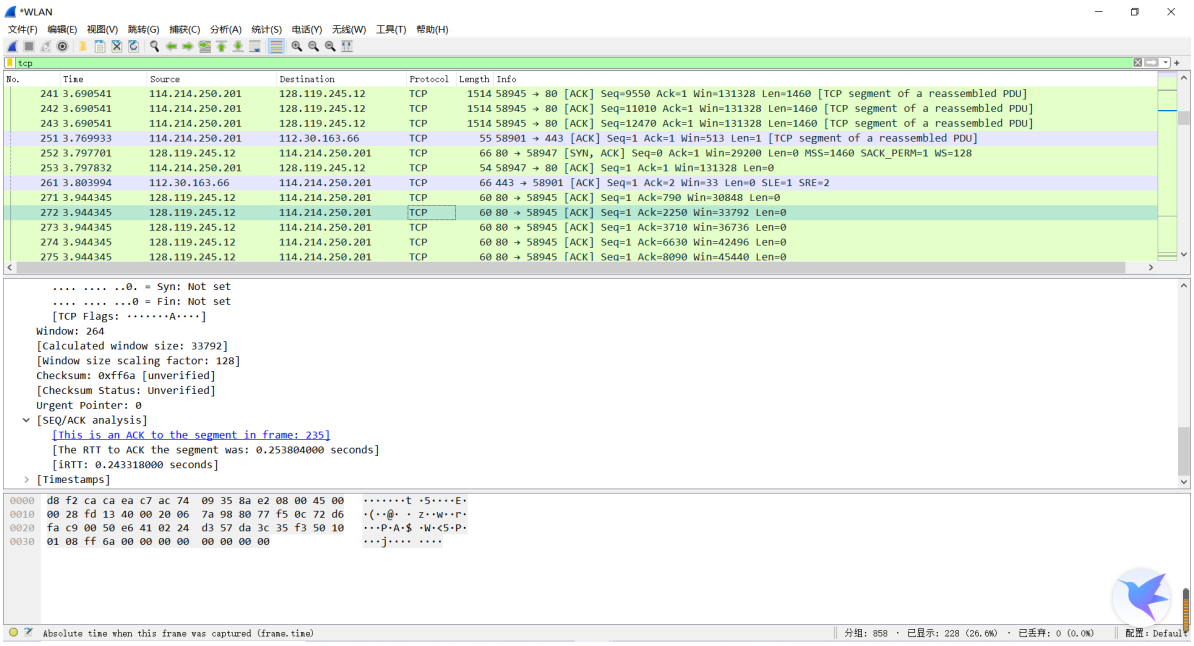




NO.2 第二个 TCP 区段

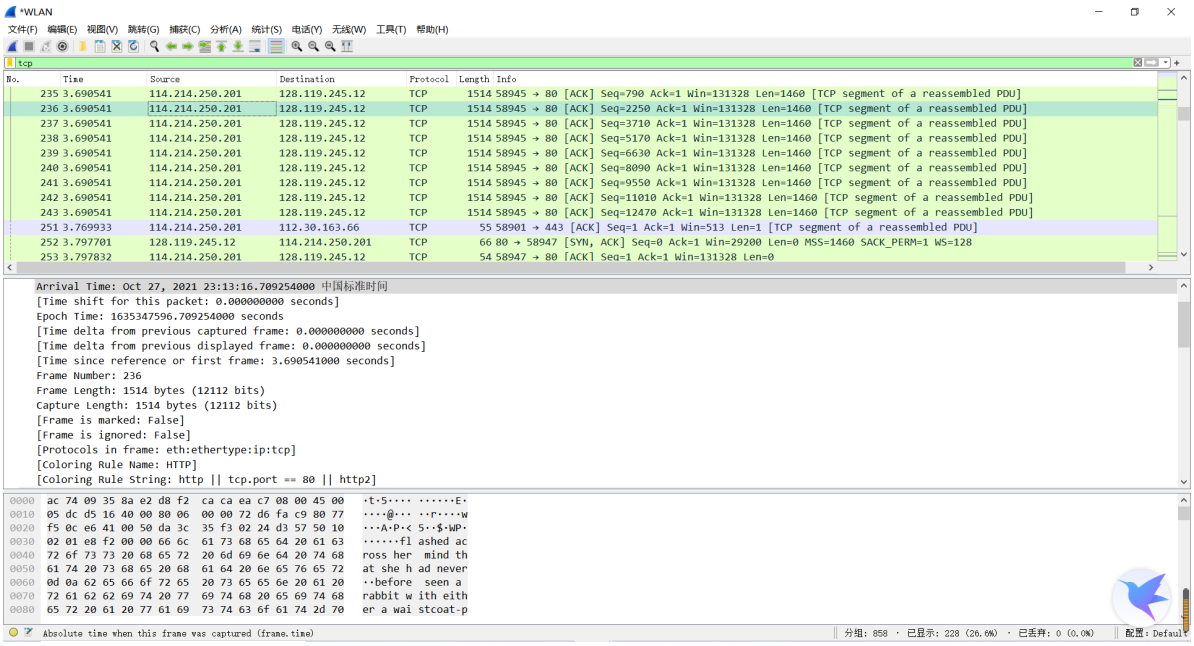
序列号：790，发送时间：3.690541s，接受时间：3.944345s，RTT：0.253804s

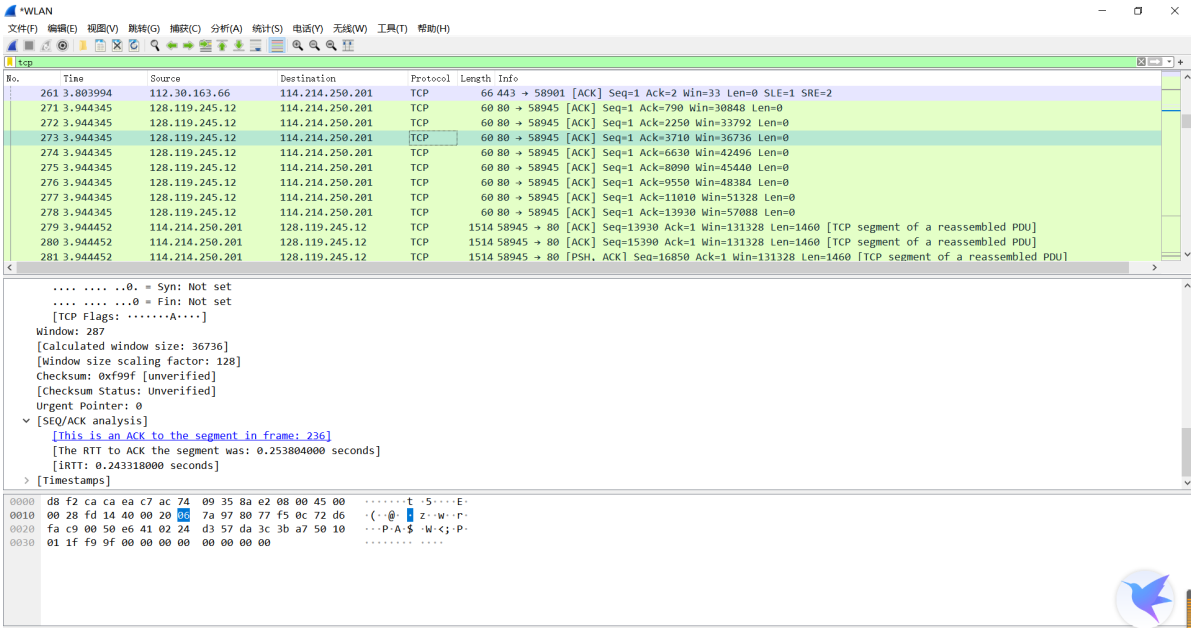




NO.3 第三个 TCP 区段

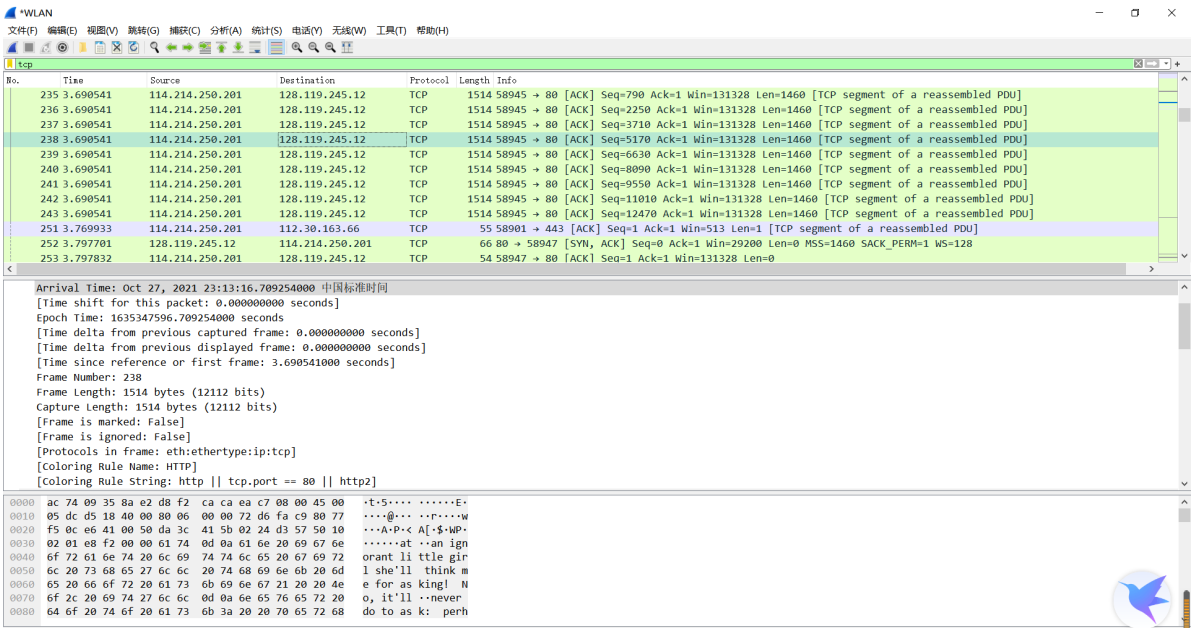
序列号：2250，发送时间：3.690541s，接受时间：3.944345s，RTT：0.253804s

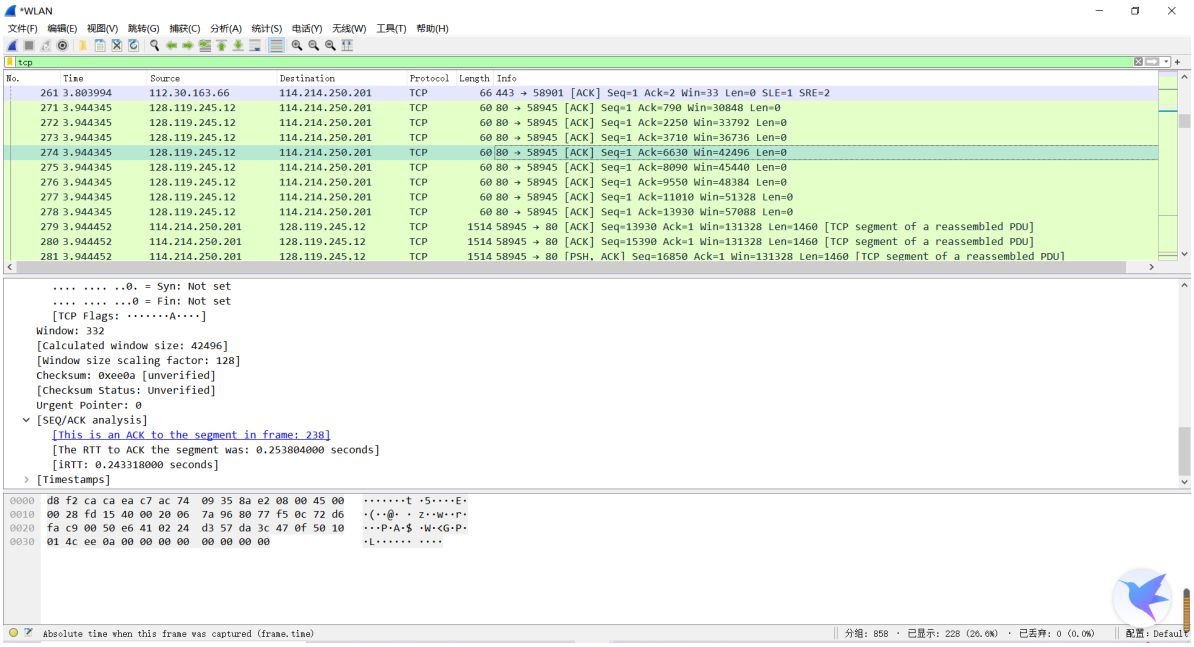




NO.4 第四个 TCP 区段

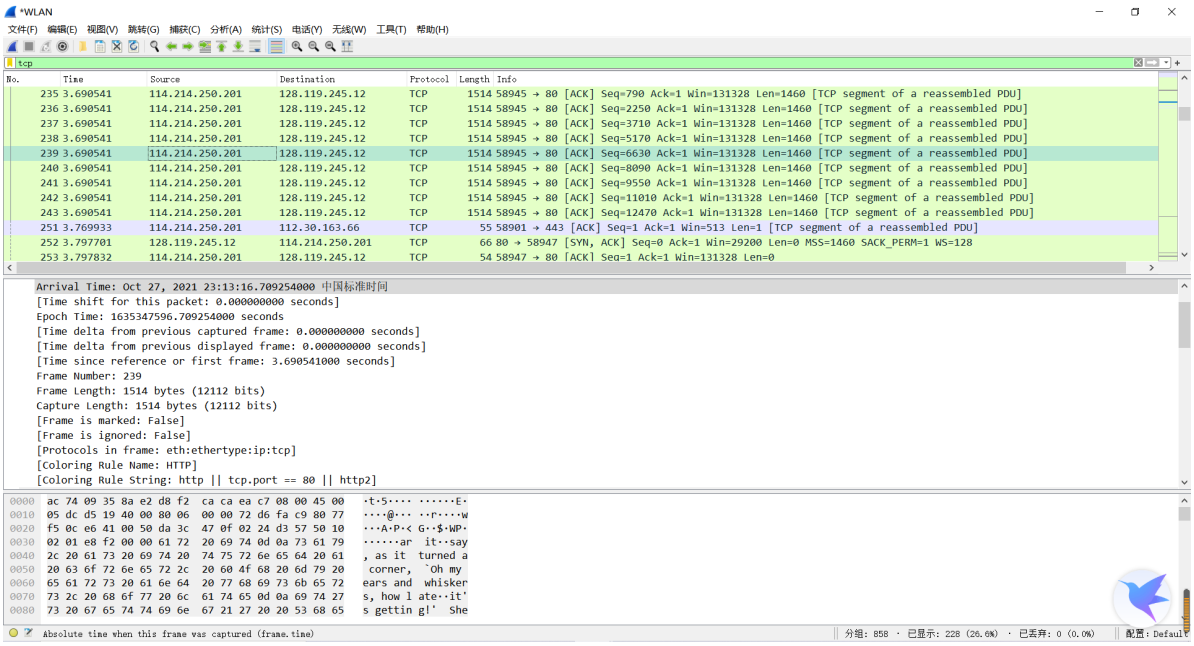
序列号：5170，发送时间：3.690541s，接受时间：3.944345s，RTT：0.253804s

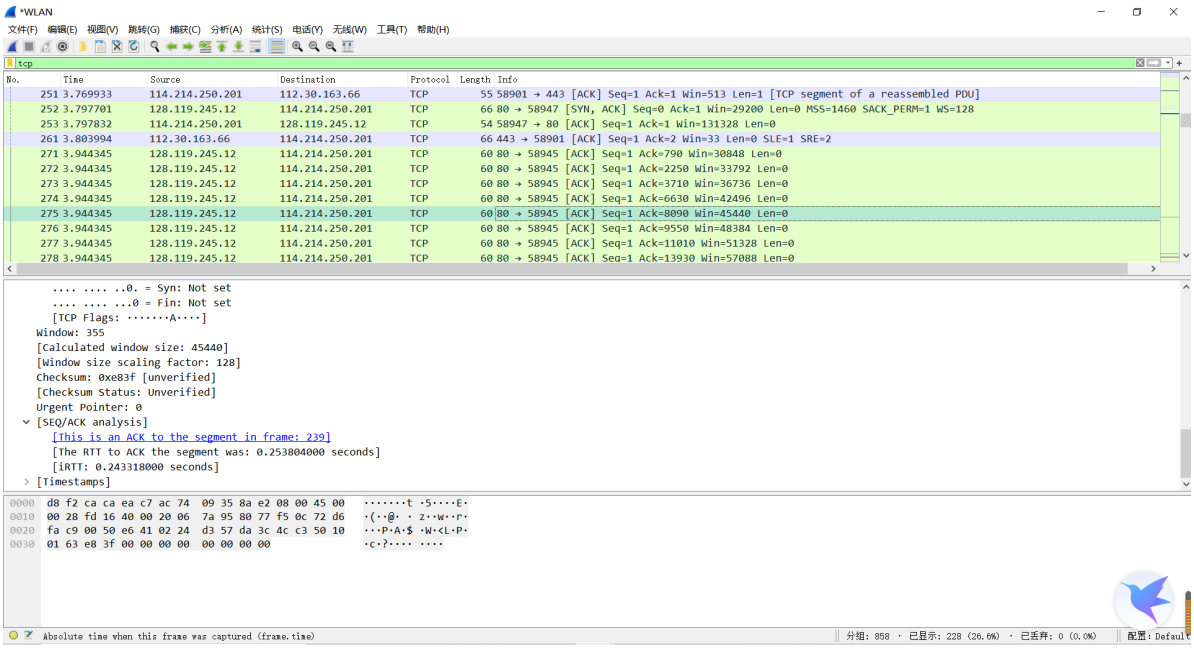




NO.5 第五个 TCP 区段

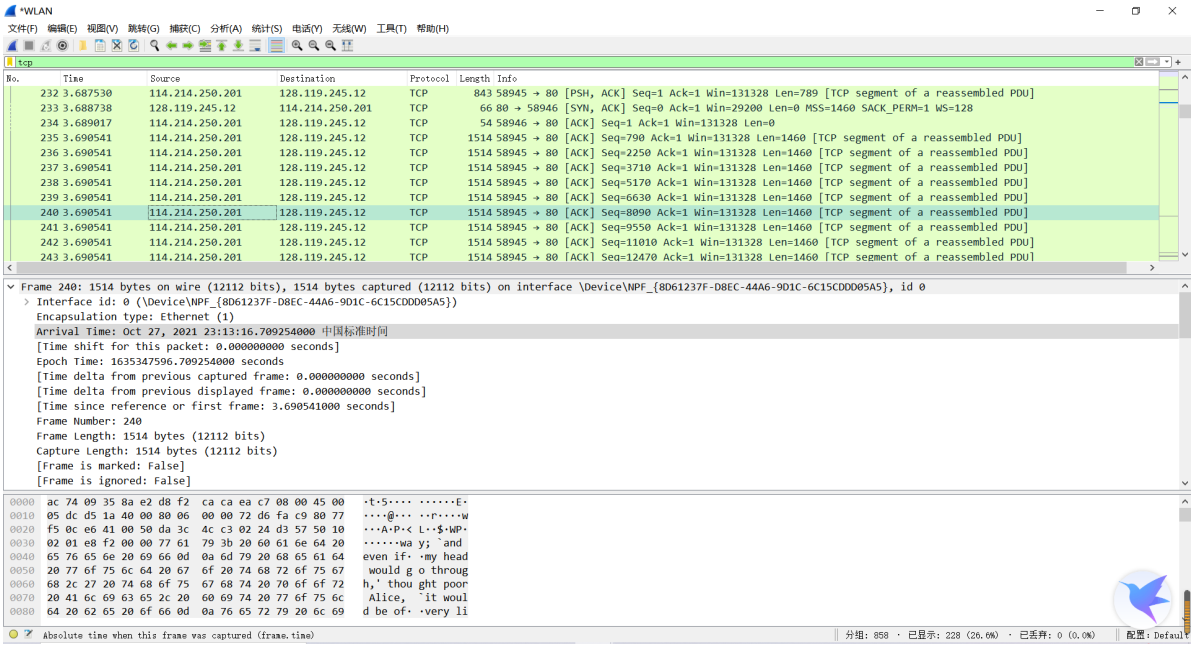
序列号：6630，发送时间：3.690541s，接受时间：3.944345s，RTT：0.253804s

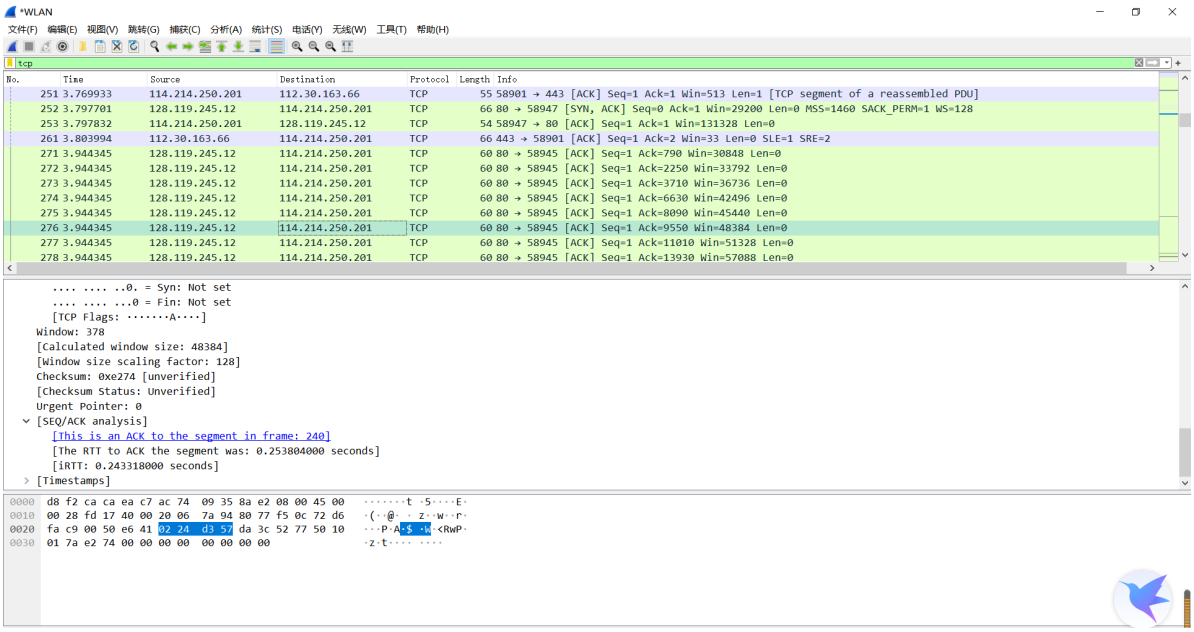




NO.6 第六个 TCP 区段

序列号：8090，发送时间：3.690541s，接受时间：3.944345s，RTT：0.253804s





计算EstimatedRTT，这里使用Excel

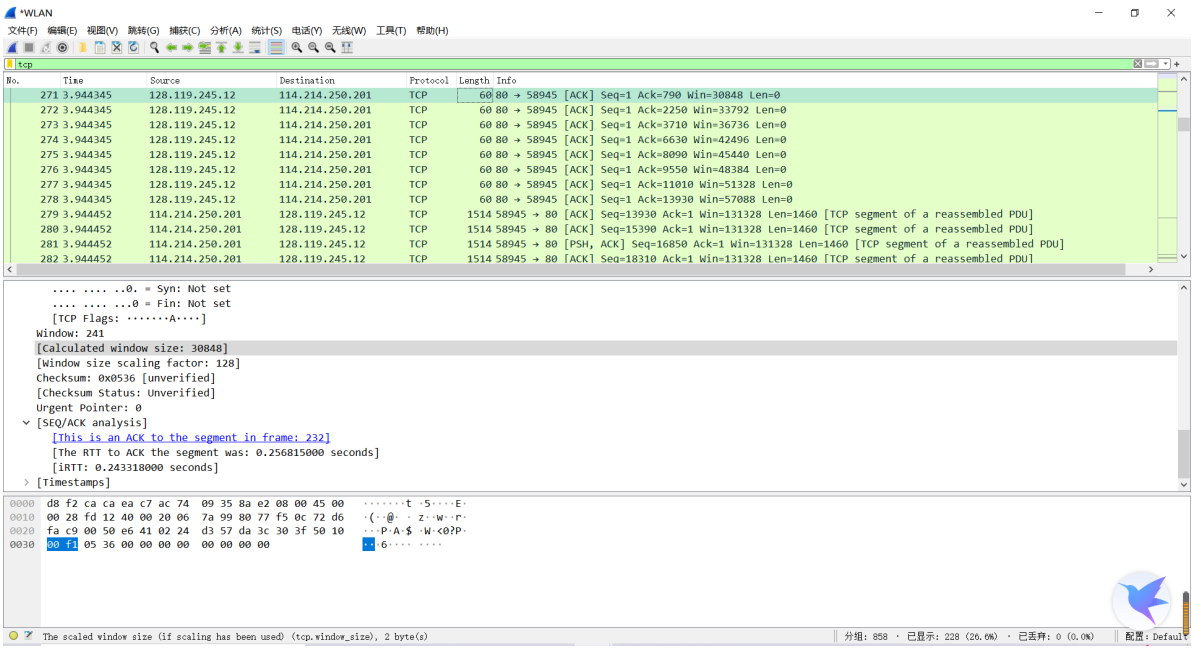
	A	B	C	D
1	Seq	RTT(s)	EstimatedRTT(s)	
2	1	0.256815	0.256815000	
3	790	0.253804	0.256438625	
4	2250	0.253804	0.253804000	
5	5170	0.253804	0.253804000	
6	6630	0.253804	0.253804000	
7	8090	0.253804	0.253804000	
8				

8.

由第7题截图，前6个区段length：789,1460,1460,1460,1460,1460

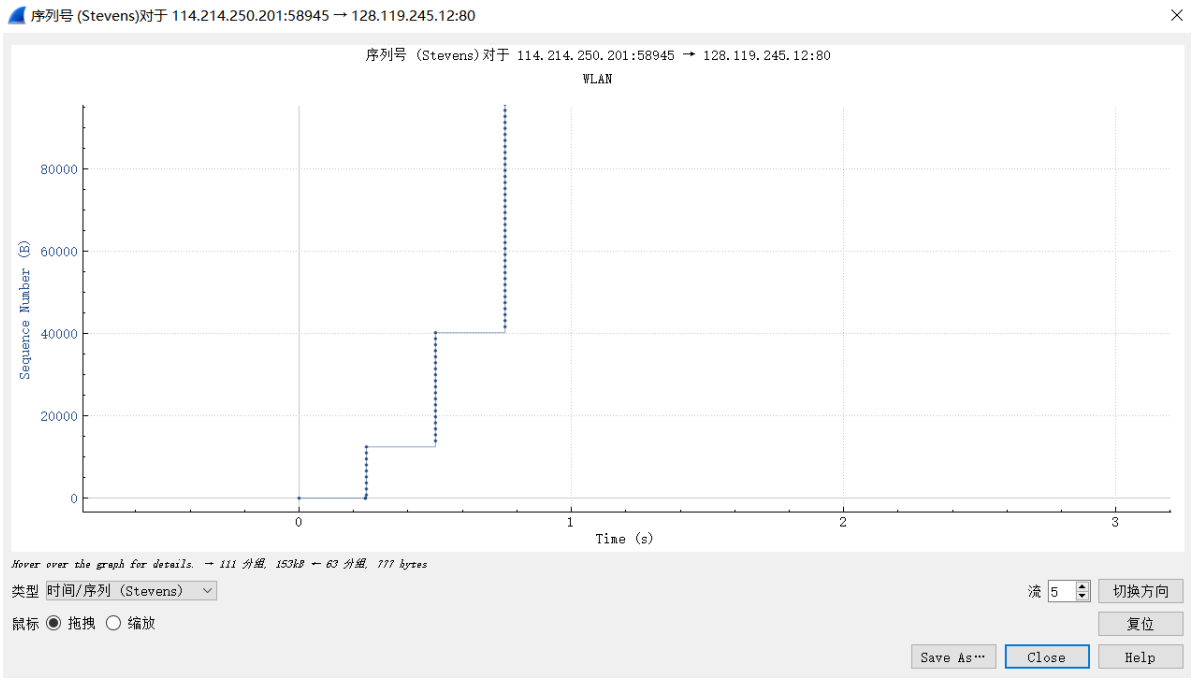
9.

最小可用缓冲区空间量是30848，缺少接收器缓冲区空间不会限制发送方传送 TCP 区段。



10.

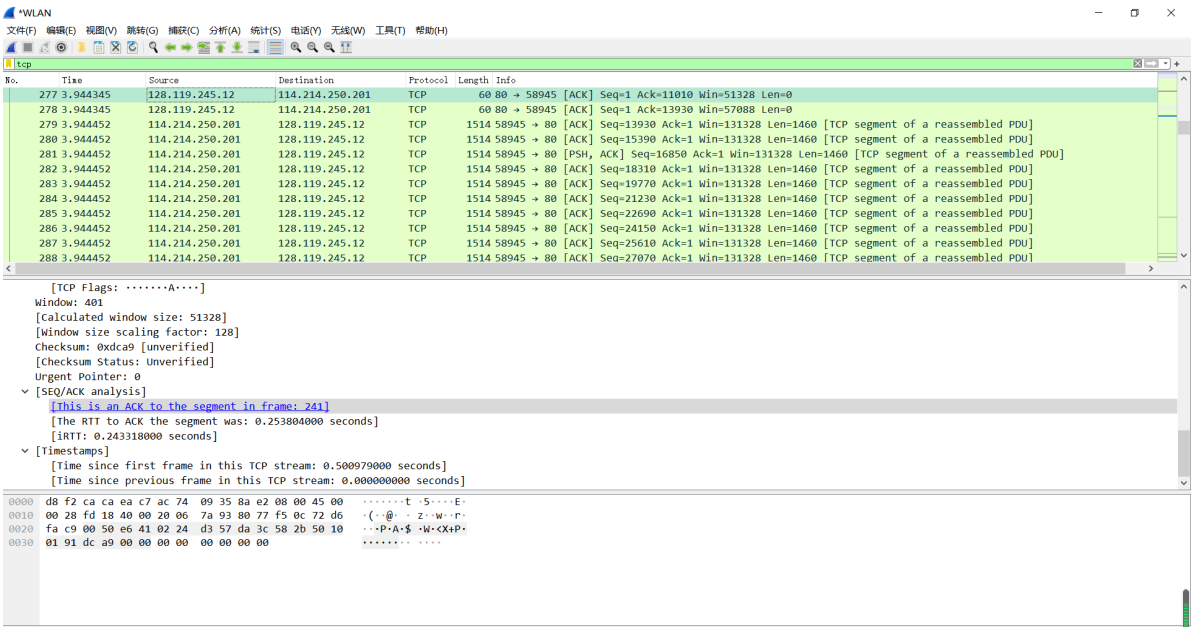
没有发现重传的情况，因为序列号一直在增大，且并未在抓包结果中发现有重传的相关数据包提示。

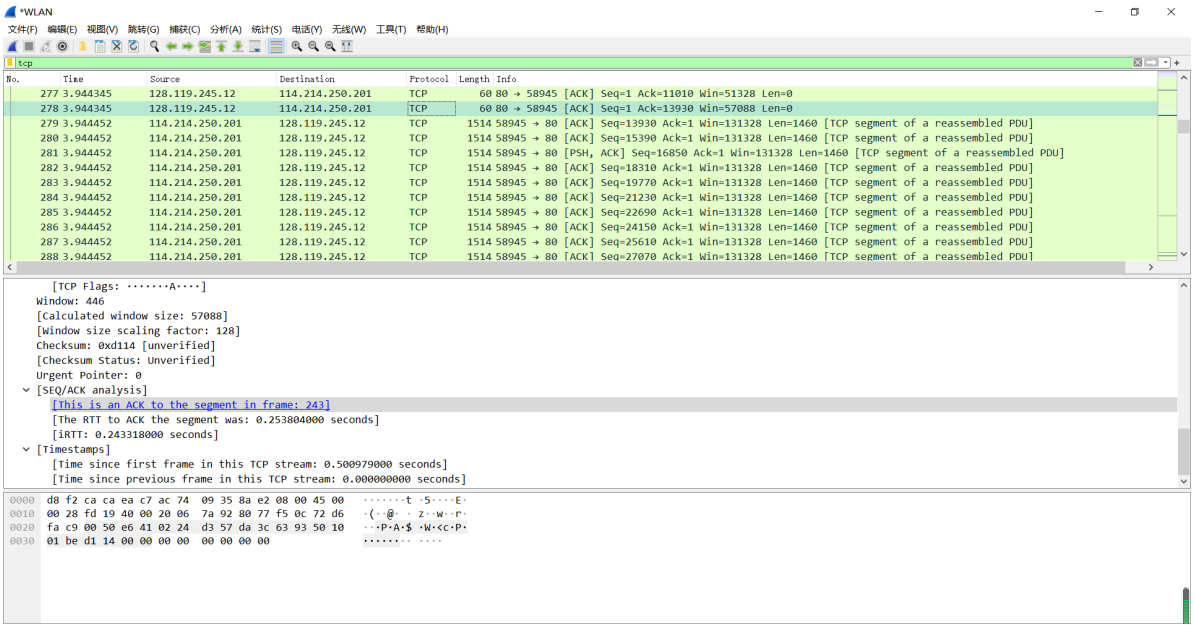


11.

基本上一个ACK接收1460B，因为链路层协议都是1500B的MTU（Maximum Transmission Unit），所以减去TCP/IP首部的40B，还剩1460B的MSS（Maximum Segment Size）可以用来传输真正的data。

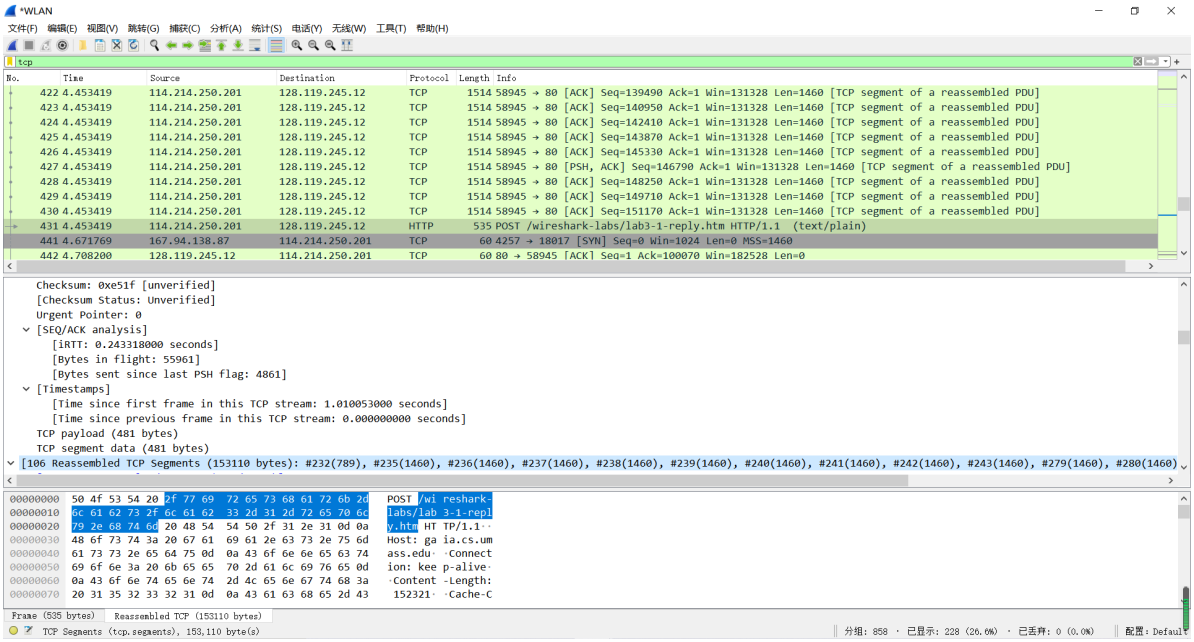
找一个累计确认的ACK的时候，发现下图frame标号跳过了其中一个（即242），所以客户收到的ACK在确认了frame241之后，直接通过确认frame243的ACK累计确认了242和243两个frame。还发现通过观察后面Info里面相邻[ACK]中ack的差值是不是大于1460也可以找到用于累计确认的ACK，这里相邻[ACK]的ack差值就是13930-11010=2920。显然，一个segment是装不下这么多字节的数据的，所以这两个ACK确认的一定不是相邻的两个发送过来的segment，也就是说这两个ACK确认了多个segment，即累计确认。





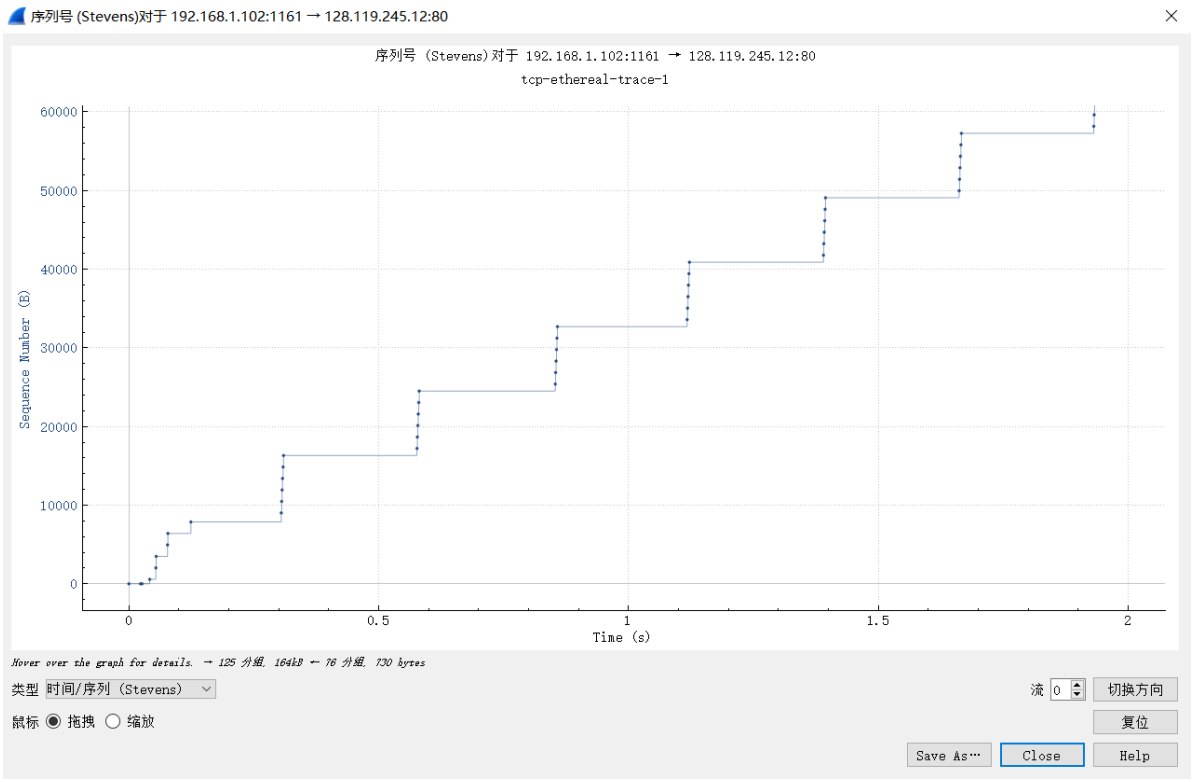
12.

153110Bytes/1.010053s=151586.10Byte/s=148.03Kb/s



13.

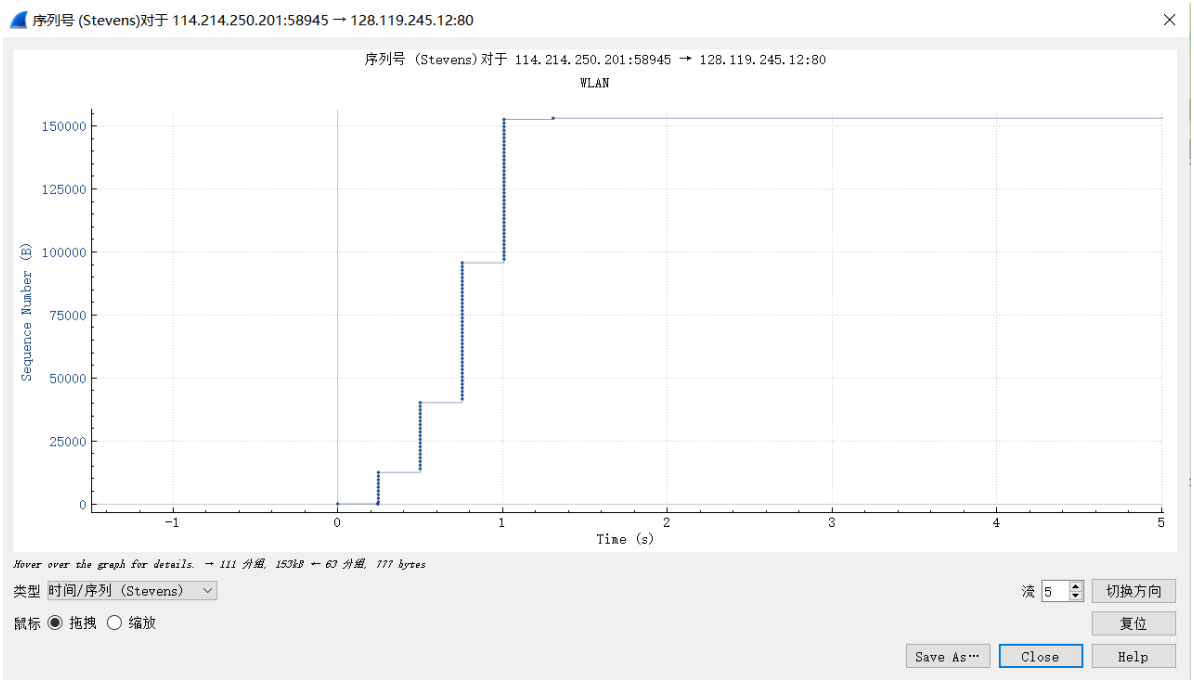
0-0.3s为慢启动阶段，然后进入拥塞避免阶段。教材中的理论是“在慢启动状态，cwnd的值以1个MSS开始并且每当传输的segment首次被确认时就增加一个MSS”。观察抓到的包，在三次握手后，客户相继发送了frame4 和frame5，frame 5是图中横坐标接近0.05s的那个点，此时是相继发送，说明cwnd为一个包大小左右。接着，frame 6即是对frame 4的ACK，所以在接收到frame 6之后，cwnd增加到了2个MSS，同时发送了frame 7 和frame 8，即图中横坐标刚过0.05s的那两个stack在一列上的点，这个cwnd持续了一段时间。frame14-17连续ACK了4次，所以cwnd的值从2加到了6，所以客户直接同时发送了frame18-23这六个包，即图中横坐标为0.3s的那6个叠在一起的点。后面都是发6个收6个，说明cwnd遇到了ssthresh，进入拥塞避免阶段，cwnd不再增加（但是教材上说会线性增加直到遇到超时，所以这里的机制与教材上的机制有区别）



1	0.000000	192.168.1.102	128.119.245.12	TCP	62	1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
2	0.023172	128.119.245.12	192.168.1.102	TCP	62	80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
3	0.023265	192.168.1.102	128.119.245.12	TCP	54	1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
4	0.026477	192.168.1.102	128.119.245.12	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565 [TCP segment of a reassembled PDU]
5	0.041737	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
6	0.053937	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	0.054026	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
8	0.054690	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
9	0.077294	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	0.077405	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
11	0.078157	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
12	0.124085	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	0.124185	192.168.1.102	128.119.245.12	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147 [TCP segment of a reassembled PDU]
14	0.169118	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	0.217299	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	0.267802	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0
17	0.304807	128.119.245.12	192.168.1.102	TCP	60	80 → 1161 [ACK] Seq=1 Ack=9013 Win=23360 Len=0
18	0.305040	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=9013 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
19	0.305813	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=10473 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
20	0.306692	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=11933 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
21	0.307571	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=13393 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
22	0.308699	192.168.1.102	128.119.245.12	TCP	1514	1161 → 80 [ACK] Seq=14853 Ack=1 Win=17520 Len=1460 [TCP segment of a reassembled PDU]
23	0.309553	192.168.1.102	128.119.245.12	TCP	946	1161 → 80 [PSH, ACK] Seq=16313 Ack=1 Win=17520 Len=892 [TCP segment of a reassembled PDU]

14.

0s发送1个frame，0.245s发送了9个frame，0.501s发送了18个frame，0.757s发送了36个frame，1.01s发送了37个frame。0~0.757s为慢启动阶段，然后进入拥塞避免阶段。这里先是在0s发送1个frame，随后在0.245s发送窗口就增长到9，与教材上所说的“在慢启动状态，cwnd的值以1个MSS开始并且每当传输的segment首次被确认时就增加一个MSS”的机制有所不同。


序列号 (Stevens)对于 114.214.250.201:58945 → 128.119.245.12:80
序列号（Stevens）对于 114.214.250.201:58945 → 128.119.245.12:80
WLAN
Sequence Number (B)
150000
125000
100000
75000
50000
25000
0
-1
0
1
2
3
4
5
Time (s)
Hover over the graph for details. → 111 分组, 153kB ← 63 分组, 777 bytes
类型 时间/序列（Stevens）
流 5
切换方向
鼠标 拖拽 缩放
复位
Save As…
Close
Help