计算机网络 TCP 实验报告

姓名：陈鹤影　　学号：PB21061287　　日期：2022.10.23

一、　实验目的：

1. 熟悉并了解 TCP 报文的具体格式，掌握 Seq、ACK、WIN 等。
2. 掌握 TCP 的建立过程——三次握手。
3. 通过对 TCP 流的分析，掌握 TCP 的拥塞控制机制。
4. 学习利用 Wireshark 绘制 TCP 流图形。
5. 掌握 TCP 连接性能的相关计算。

二、　实验流程及问题回答：

Part 1：Capturing a bulk TCP transfer from your computer to a remote server

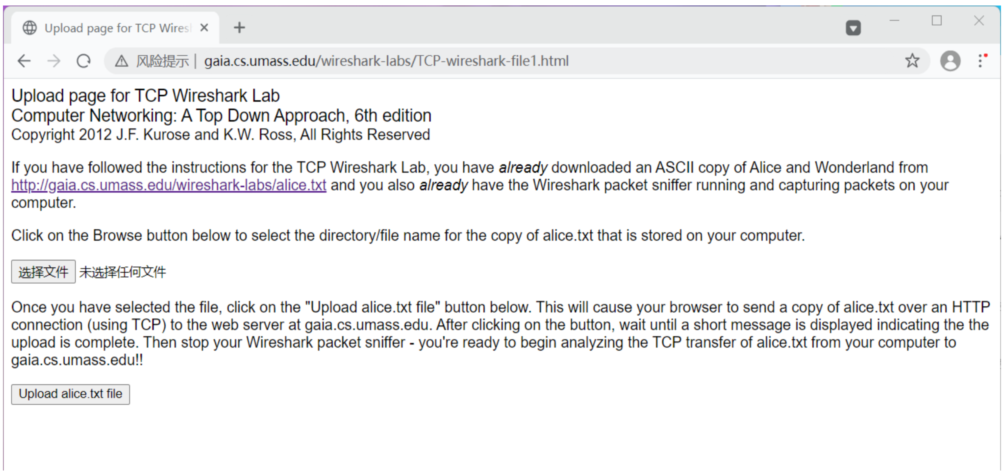
1. **实验内容：**

　　使用 Wireshark 跟踪文件从主机到远程服务器的 TCP 传输，利用 HTTP POST 方法将文件发送到 Web 服务器（传输较大的数据量）。

2. **实验流程：**

　1）获取需要传输的文件（获取文件储存在 alice.txt 中）

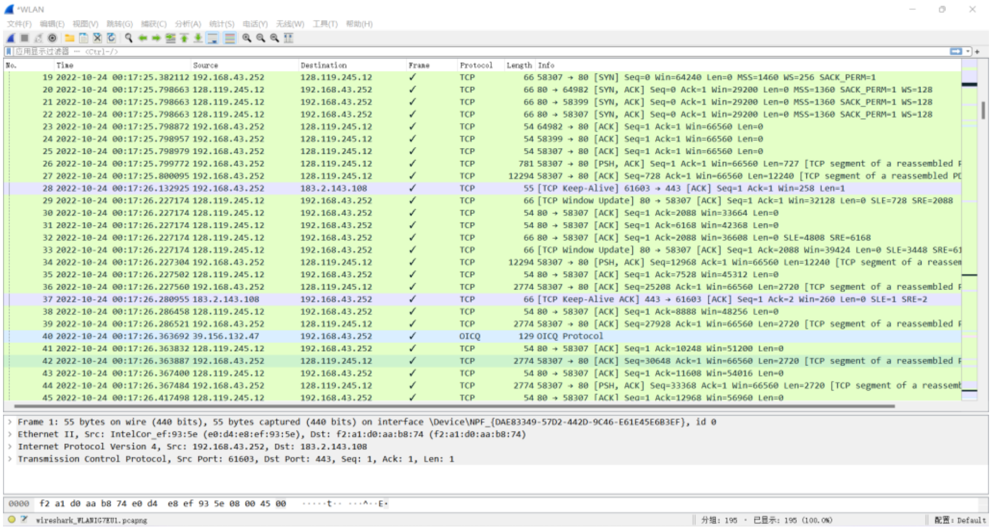
　2）登入 http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html 网页。界面如图所示：



（图 1-1 文件上传登入界面）

　3）浏览并选择对应的文件（alice.txt），启动 Wireshark 抓包，上传文件。

　4）上传成功，结束 Wireshark 抓包。抓包结果如图所示：



（图 1-2 Wireshark 对大文件上传过程的抓包）

Part 2：A first look at the captured trace

1. **实验内容：**

   初步分析获得的 trace。

2. **实验流程：**

   1） 过滤出“tcp”数据包

   2） 分析数据可以观察到 TCP 的三次握手过程：

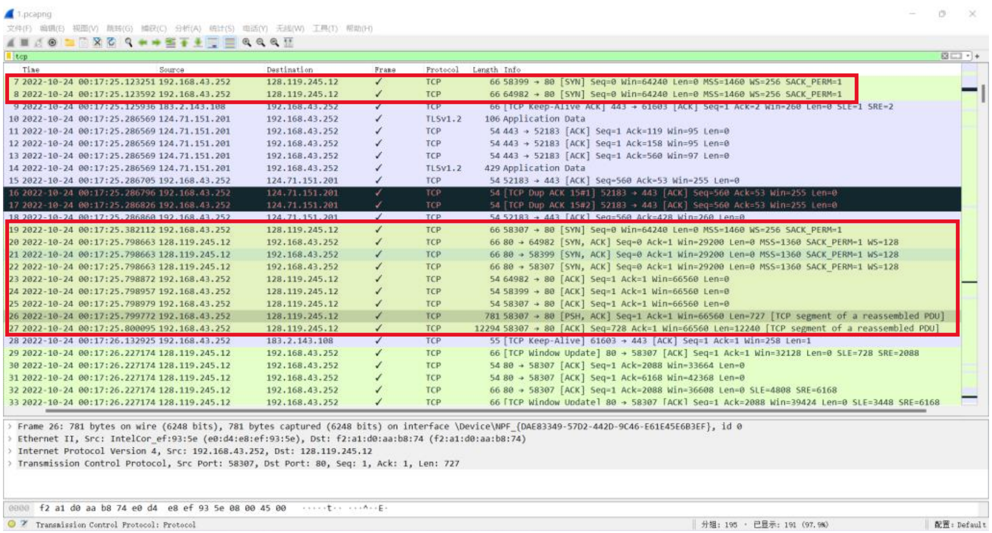
   a. 客户端 TCP 首先向服务器端 TCP 发送 SYN 报文段（SYN = 1）。客户随机选择初始序列号（Seq = 0）。

   b. 服务器提取 TCP SYN 报文段，向客户 TCP 发送允许连接的 SYNACK 报文段（SYN = 1,ACK = 客户 TCP Seq+1,）。服务器随机选择初始序列号（Seq = 0）。

   c. 接收到 SYNACK 报文段后，客户给该 TCP 连接分配缓存和变量。客户主机向服务器主机发送报文段（SYN = 0,ACK = 服务器 TCP Seq+1）。

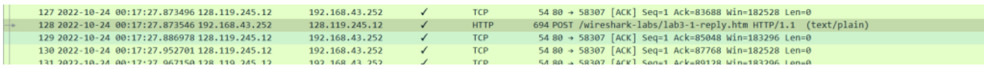
   观察到过程如下，其中我们可以观察到客户 TCP 向服务器发送了多个报文段，同时收到了多个回复。

   我们观察到在完成第三次握手后，TCP 连接进入 ESTABLISHED 状态，随后客户和服务器开始传送数据。可以观察到一个携带应用数据的 PSH 包。



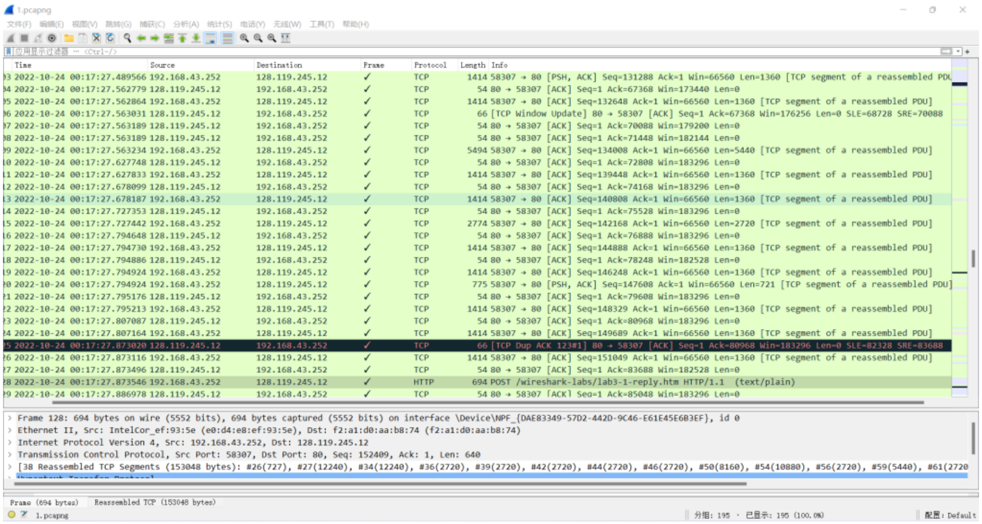
（图 1-3 三次握手过程）

3） 可以观察到 HTTP POST 报文：



（图 1-4 HTTP POST 报文）

4） 同样也可以观察到大量带有“[TCP segment of a reassembled PDU]”信息的报文段。（在基于 TCP 的传输中，如果应用层消息过大（如超过 MSS）TCP Segment 不能一次包含全部的应用层 PDU，而要把一个完整消息分成多个段，就会将除了最后一个报文段（segment）的所有其他报文段都打上“TCP segment of a reassembled PDU”。）同时也可以看到服务器返回的大量确认报文。

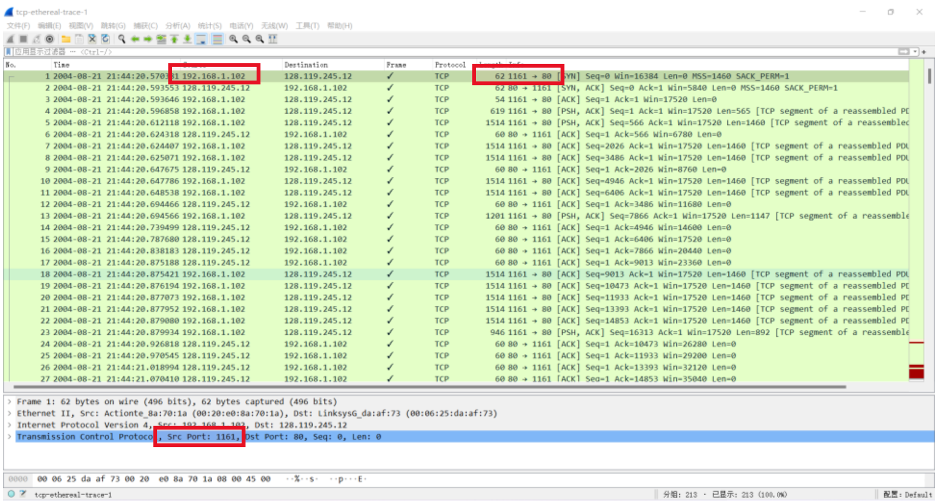


（图 1-5 [TCP segment of a reassembled PDU]and[ACK]）

3. **Q&A(以下内容利用所给 trace 和本机 trace 分别分析)：**

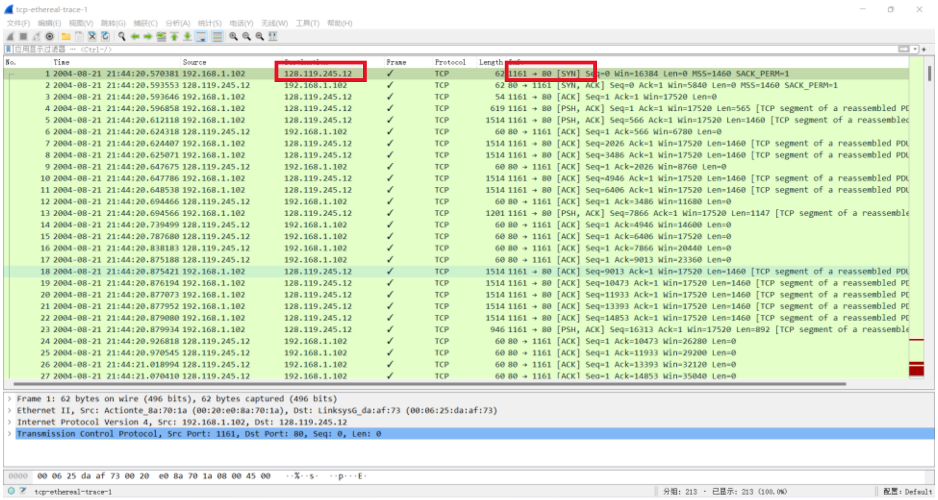
   1） **What is the IP address and TCP port number used by the client computer (source)that is transferring the file to gaia.cs.umass.edu?**

   Ans： IP address：192.168.1.102；TCP port number：1161；



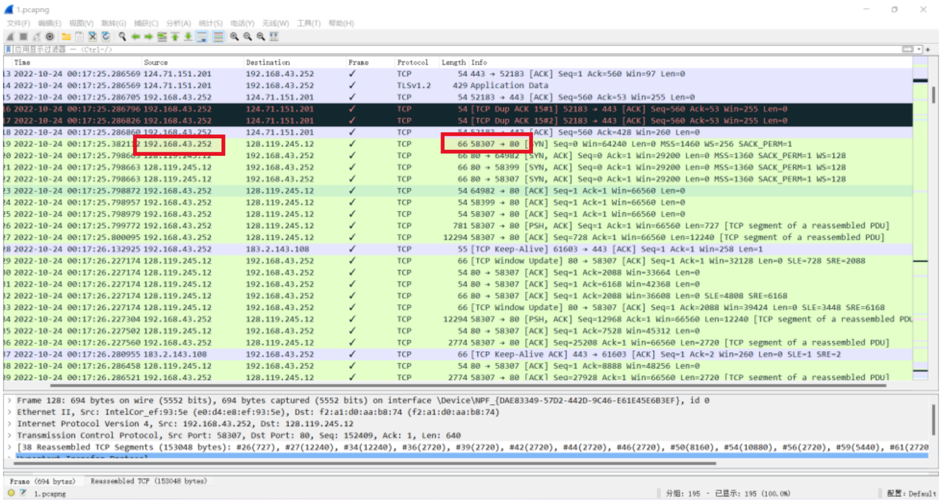
   2） **What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?**

   Ans： IP address：128.119.245.12；TCP port number：80；



3） **What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?**

Ans： IP address：192.168.43.252；TCP port number：58307；



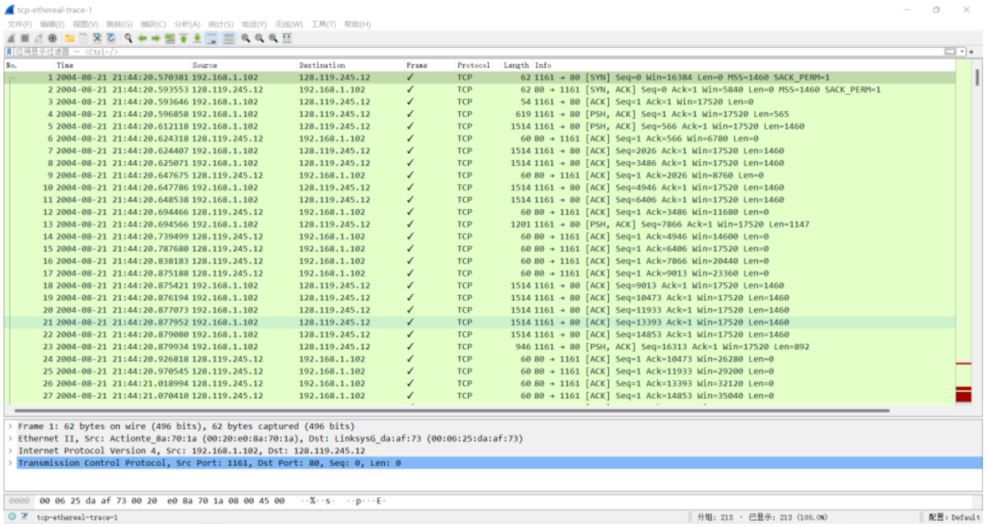
Part 3：TCP Basics

1. **实验内容：**

   分析 TCP 数据流。

2. **实验流程：**

   选择分析-关闭 HTTP 协议，关闭后 Wireshark 界面如图所示：

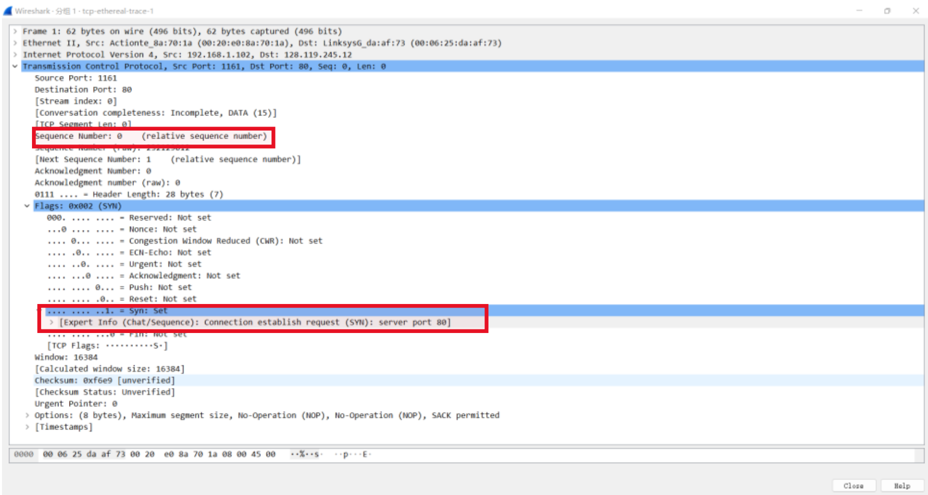


（图 1-6 在分析中关闭 TCP 协议后界面）

3. Q&A(以下内容利用所给 trace 分析)：

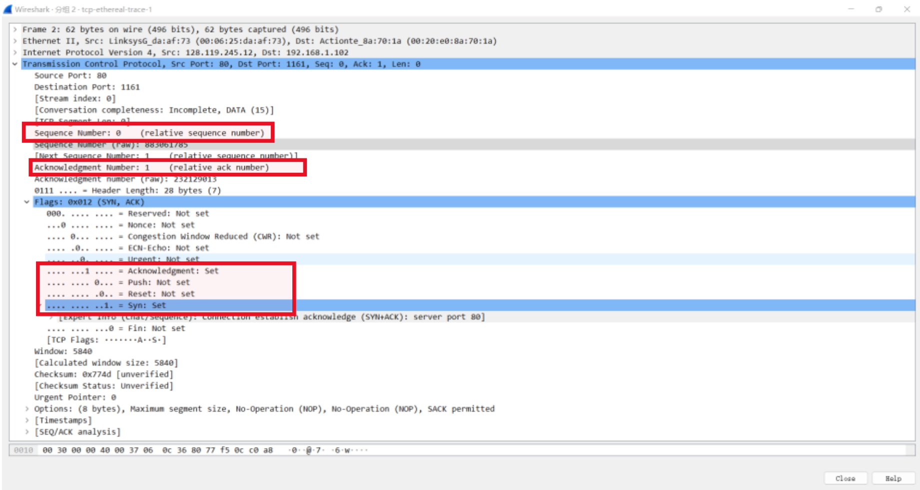
4） What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

Ans：Sequence number：0；报文段的 SYN 标志被置 1 及“Connection establish request(SYN):sever port 80”；



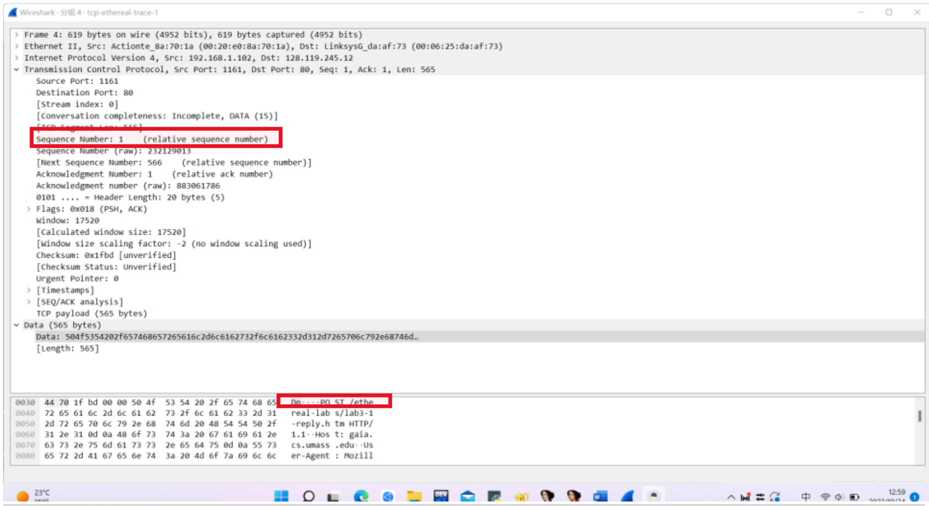
5） What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Ans： Sequence number： 0； Value of the Acknowledgement field： 1； 服务器通过接收到的 SYN 报文中 Seq + 1 得到 ACK 的值； 通过 ACK 和 SYN 均被置 1 证明这是一个 SYNACK 报文段。



6) What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

Ans: Sequence number: 1;



7) Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3, page 242 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 242 for all subsequent segments.

Ans: Sequence numbers: 1, 566, 2026, 3486, 4946, 6406;

Send times: 2004-08-21 21:44:20.596858,

2004-08-21 21:44:20.612118,

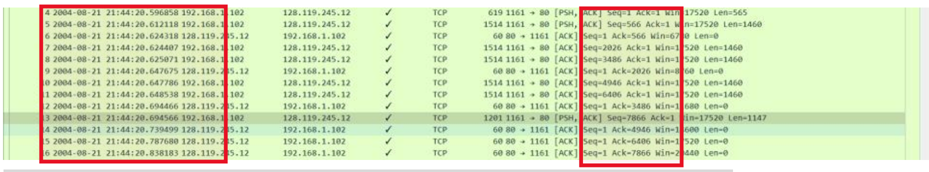
2004-08-21 21:44:20.624407,

2004-08-21 21:44:20.625071,
2004-08-21 21:44:20.647786,
2004-08-21 21:44:20.648538;

Receive times：第 1 个包：2004-08-21 21:44:20.624318,
第 2 个包：2004-08-21 21:44:20.647675,
第 3 个包：2004-08-21 21:44:20.694466,
第 4 个包：2004-08-21 21:44:20.739499,
第 5 个包：2004-08-21 21:44:20.787680,
第 6 个包：2004-08-21 21:44:20.838183;

RTT value for each：第 1 个包：RTT=0.02746,
第 2 个包：RTT= 0.035557,
第 3 个包：RTT=0.070059,
第 4 个包：RTT=0.114428,
第 5 个包：RTT=0.139894,
第 6 个包：RTT=0.189645;

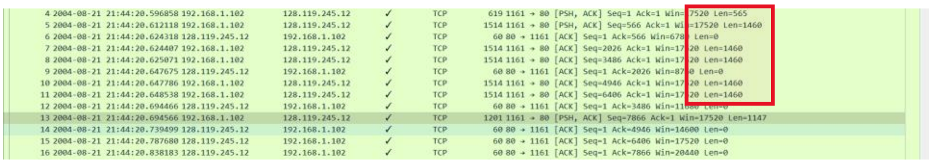
EstimatedRTT value：计算公式为
EstimatedRTT =0.875·EstimatedRTT+0.125 SampleRTT
（这里假设初始 EstimatedRTT 与第一次得到的 SampleRTT 相等）
第 1 个包：EstimatedRTT =0.02746,
第 2 个包：EstimatedRTT = 0.028472125,
第 3 个包：EstimatedRTT =0.0336704844,
第 4 个包：EstimatedRTT =0.0437651738,
第 5 个包：EstimatedRTT =0.0557812771,
第 6 个包：EstimatedRTT =0.0725142425;



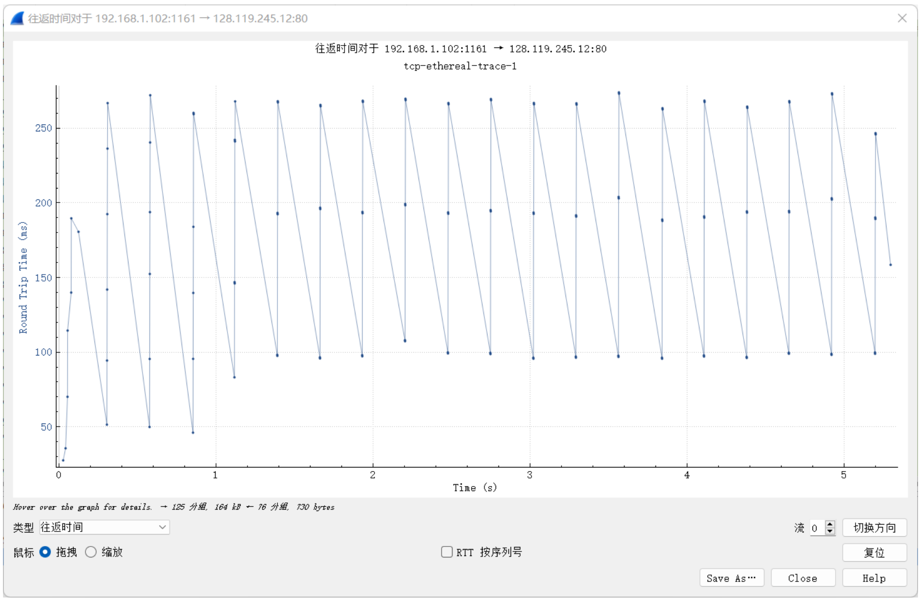
4	2004-08-21 21:44:20.596858	192.168.1.102	128.119.245.12	✓	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565
5	2004-08-21 21:44:20.612118	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460
6	2004-08-21 21:44:20.624318	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	2004-08-21 21:44:20.624407	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460
8	2004-08-21 21:44:20.625071	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460
9	2004-08-21 21:44:20.647675	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	2004-08-21 21:44:20.647786	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460
11	2004-08-21 21:44:20.648538	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460
12	2004-08-21 21:44:20.694466	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	2004-08-21 21:44:20.694566	192.168.1.102	128.119.245.12	✓	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147
14	2004-08-21 21:44:20.739499	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	2004-08-21 21:44:20.787680	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	2004-08-21 21:44:20.838183	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0

8） **What is the length of each of the first six TCP segments?**

Ans： Length：565，1460，1460，1460，1460，1460； 58307；



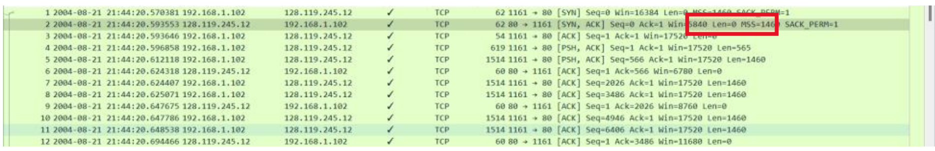
4	2004-08-21 21:44:20.596858	192.168.1.102	128.119.245.12	✓	TCP	619	1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565
5	2004-08-21 21:44:20.612118	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460
6	2004-08-21 21:44:20.624318	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
7	2004-08-21 21:44:20.624407	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460
8	2004-08-21 21:44:20.625071	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460
9	2004-08-21 21:44:20.647675	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
10	2004-08-21 21:44:20.647786	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460
11	2004-08-21 21:44:20.648538	192.168.1.102	128.119.245.12	✓	TCP	1514	1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460
12	2004-08-21 21:44:20.694466	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
13	2004-08-21 21:44:20.694566	192.168.1.102	128.119.245.12	✓	TCP	1201	1161 → 80 [PSH, ACK] Seq=7866 Ack=1 Win=17520 Len=1147
14	2004-08-21 21:44:20.739499	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
15	2004-08-21 21:44:20.787680	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
16	2004-08-21 21:44:20.838183	128.119.245.12	192.168.1.102	✓	TCP	60	80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0



（图 1-7 TCP RTT 流图形）

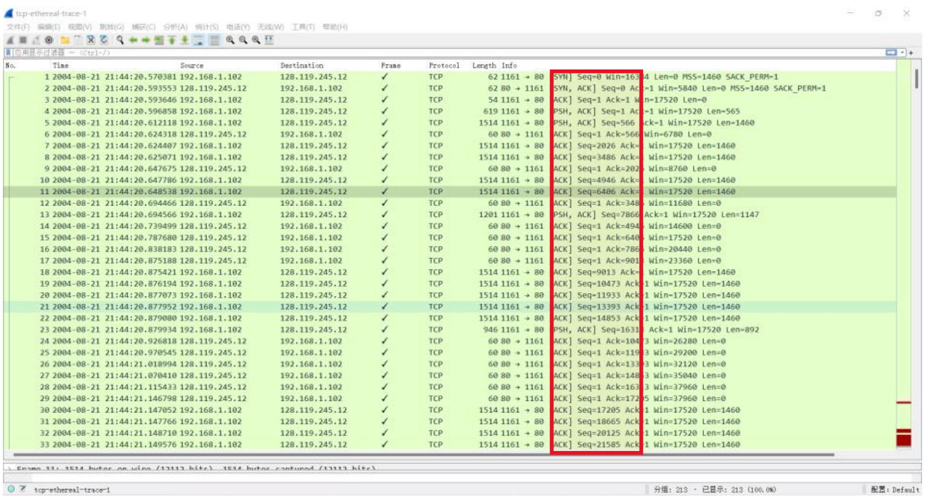
9） **What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?**

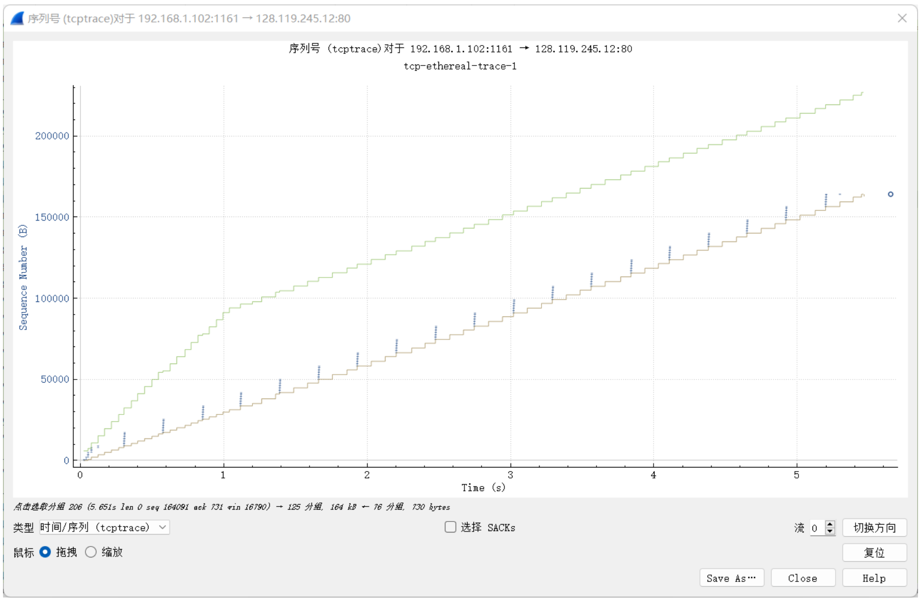
Ans： Minimum amount of available buffer：5840；由于 TCP 的流量控制机制，当接收窗口大小较小时发送方的传输速率会下降，但在上述过程中，接收窗口始终大于 MSS，故不会限制发送方传输速率。



10） **Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?**

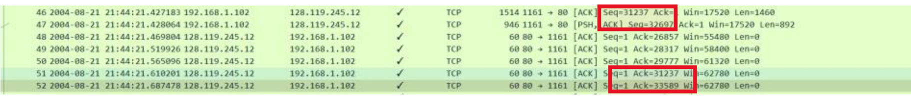
Ans： 可以观察到 TCP 流中相应的序列号一直增大，并未出现重复的情况，因此可知并没有出现重传。同样也可以利用 TCP 序列号流图形得到上述结论。





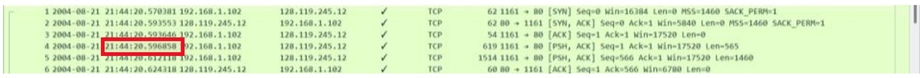
11) How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 250 in the text)

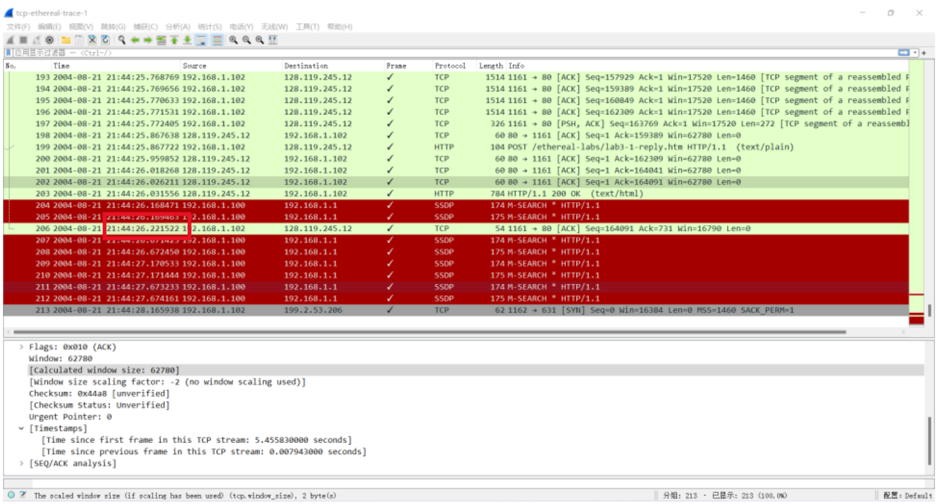
Ans: 大部分确认为 1460 bytes；如第 52 个报文段就是间隔确认的。

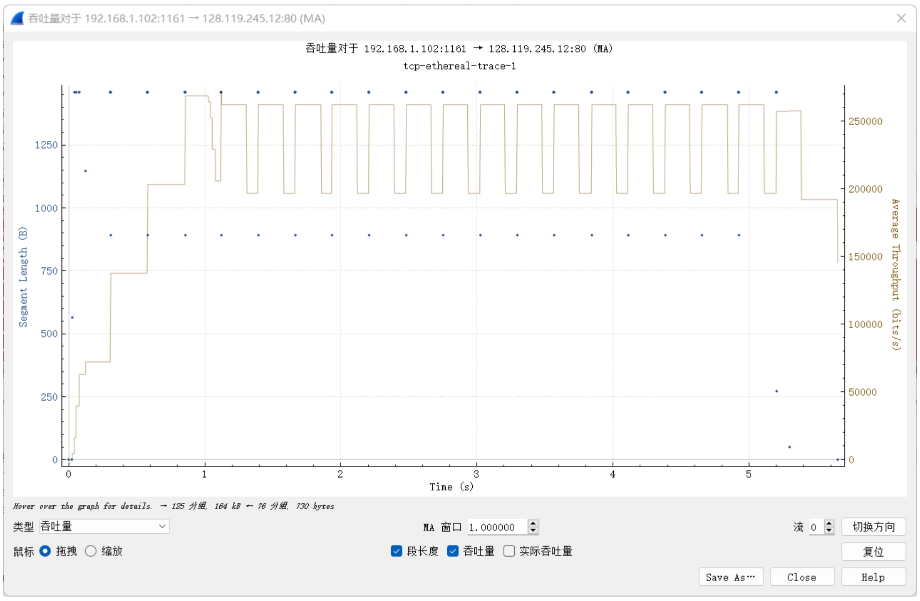


12) What is the throughput (bytes transferred per unit time) for the TCP connection?Explain how you calculated this value

Ans： 传输数据总量为164091 bytes，传输总时间为 T= 26.026211-20.596858=5.429353s。$\text{Throughput} = \frac{164091 bytes}{5.429353s} \approx$ 30222.94Byte/s。也可以利用图表直接得出。







（图 1-8 TCP 吞吐量流图形）

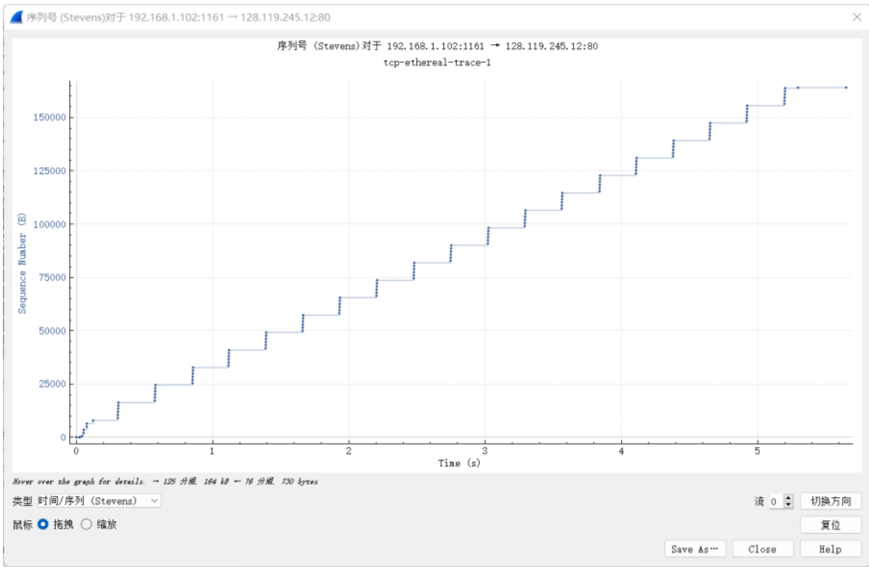
Part 4：TCP congestion control in action

1. 实验内容：

分析 TCP 数据包，观察 TCP 拥塞控制机制。

2. 实验流程：

利用 Wireshark 制图，得到如图所示时间序列：

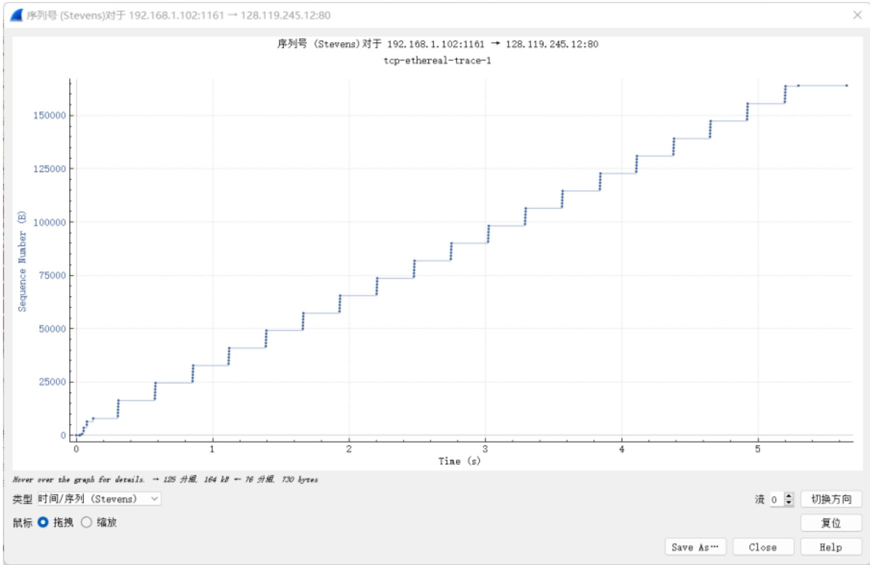


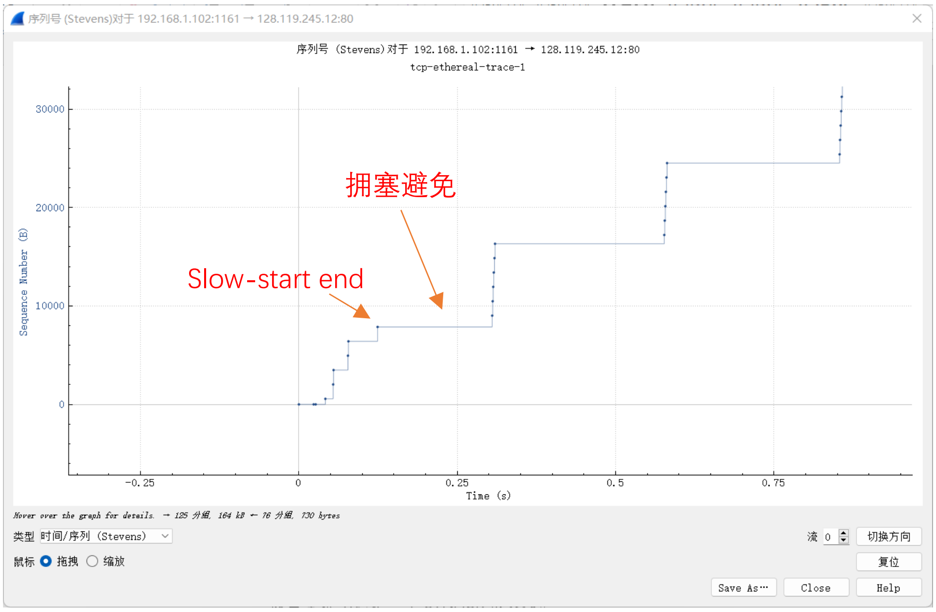
（图 1-9 Stevens TCP 流图形）

3. Q&A(以下内容利用所给 trace 分析)：

13） Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text?

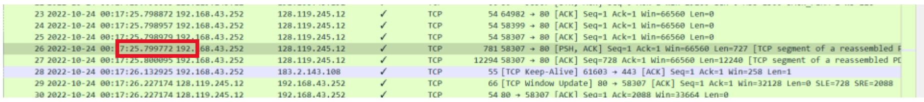
Ans：慢启动从 TCP 连接建立完成后发送第一个分组（分组 5）开始，在第 13 个分组之后不再呈指数增长，慢启动结束，拥塞避免开始。观察可知在后续传输过程中每次均发送六个报文段，说明 cwnd 遇到了 ssthresh，进入 congestion avoidance 阶段， cwnd 不再增加， 但是书上写得理想机制是会线性增加直到遇到超时，可能这里存在不同的控制机制。此外，在慢启动阶段 cwnd 并不是严格按照指数增长的， 与课本上的理想情况存在一定的差别。

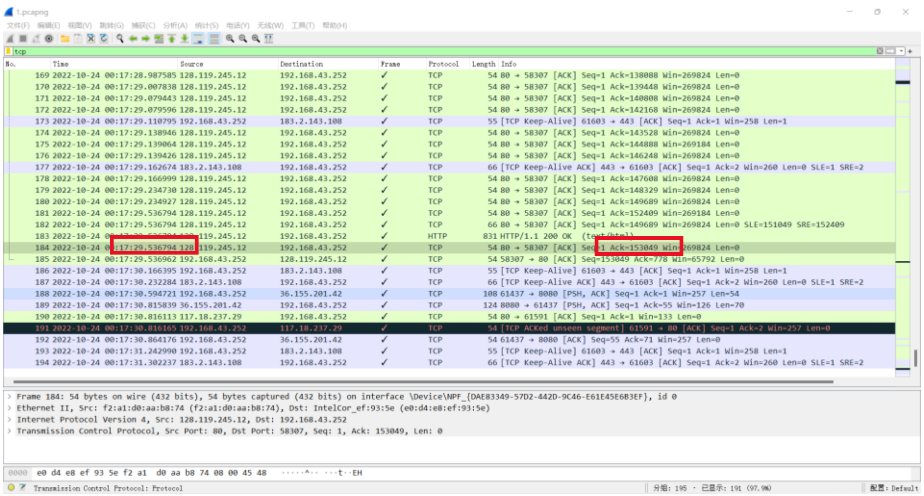


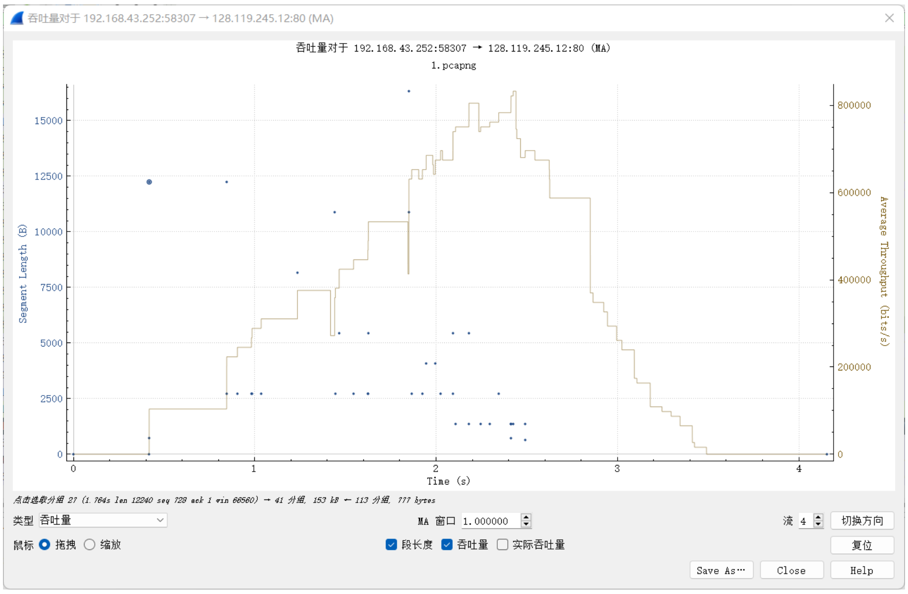


14） Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu

Ans： 传输数据总量为164091bytes，传输总时间为 T= 29.536794-25.799772=3.737022s。Throughput $= \frac{153049bytes}{3.737022s} \approx 40954.80$Byte/s。也可以利用图表直接得出。





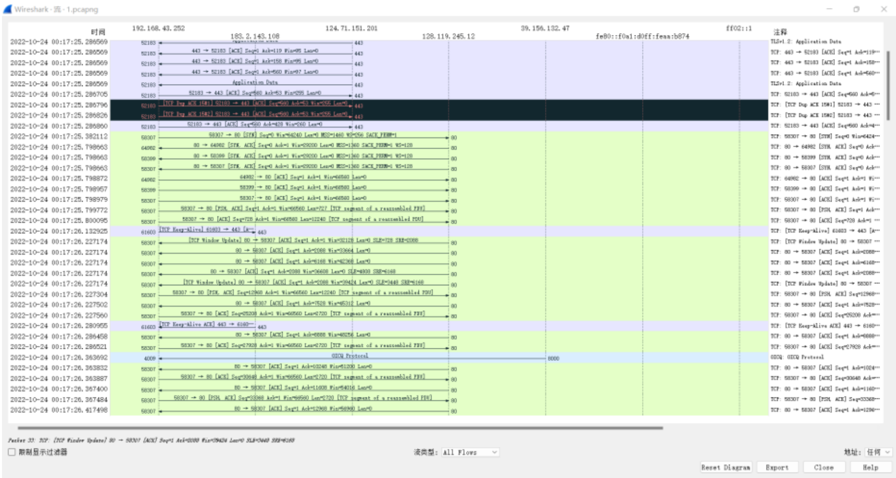


（图 1-10 TCP 吞吐量流图形）

Ans：慢启动从 TCP 连接建立完成后发送第一个分组（分组 27）开始，在第 36 个分组之后不再呈指数增长，慢启动结束，拥塞避免开始。可以比所给 trace 看到更为明显的拥塞控制过程。

补充内容：

利用 Wireshark 我们也可以看到 TCP 流量图，以实验中所得数据为例，可以得到如下流量图：



（图 1-11 TCP 流量图）

除了三次握手，在结束连接时 TCP 也具有四次挥手过程：

由于 TCP 连接是全双工的，因此，每个方向都必须要单独进行关闭。在一方完成数据发送任务后，将会发送一个 FIN 来终止本方连接，但是在这个 TCP 连接上仍然能够发送数据，直到接收方也发送了 FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭。

（1）第一次挥手：客户端发送一个 FIN，用来关闭客户机到服务器的数据传送，客户端进入 FIN_WAIT_1 状态。

（2）第二次挥手：服务器收到 FIN 后，发送一个 ACK 给客户端，确认序号为收到序号+1，服务器进入 CLOSE_WAIT 状态。

（3）第三次挥手：服务器发送一个 FIN，用来关闭服务器到客户端的数据传送，服务器进入 LAST_ACK 状态。

（4）第四次挥手：客户端收入 FIN 后，客户端进入 TIME_WAIT 状态，接着发送一个 ACK 给服务器，确认序号为收到序号+ 1，服务器进入 CLOSED 状态，完成四次挥手。