**实验三、传输层TCP协议分析实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 组号： | 3-1 |  |  |  |  |
| 姓名： | 陈实 | 学号： | 2215015058 | 班级： | 计算机2101 |
| 姓名： | 罗凯 | 学号： | 2204313382 | 班级： | 计算机2101 |
|  |  |  |  |  |  |

1. 实验目的
2. 理解TCP报文首部格式和字段的作用，TCP连接的建立和释放过程，TCP数据传输中的编号与确认的过程。
3. 理解TCP的错误恢复的工作原理和字节流的传输模式，分析错误恢复机制中TCP双方的交互情况。
4. 理解TCP的流量控制的工作原理，分析流量控制中TCP双方的交互情况。
5. 理解TCP的拥塞控制的工作原理，分析拥塞控制中TCP双方的交互情况。
6. 实验内容
7. 使用基于TCP的应用程序（如浏览器下载文件）传输文件，在**客户端和服务器均要捕获TCP报文**。
8. 分析TCP报文首部信息、TCP连接的建立和释放过程、TCP数据的编号与确认机制。观察几个典型的TCP选项：MSS、SACK、Window Scale、Timestamp等，查资料说明其用途。
9. 观察和估计客户机到服务器的RTT，双方各自的MSS。
10. 观察TCP的流量控制过程，和拥塞控制中的慢启动、快速重传、拥塞避免，快速恢复等过程【观察拥塞控制的难度较大，观察到前两个过程即可】。

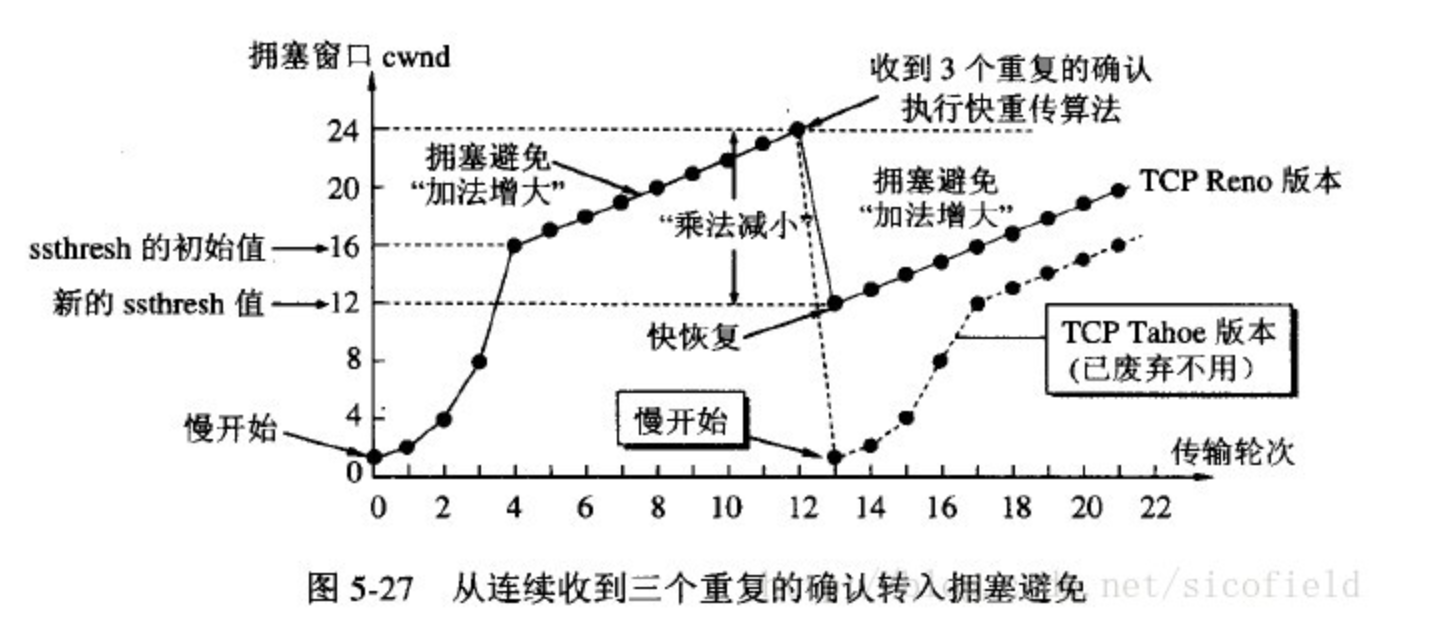


图4-0 典型的TCP 拥塞控制过程图例

1. \*（可选）注意观察初始的cwnd是多少，看看不同的操作系统初始cwnd的差别。

【可以增加题目规定以外的分析】

1. 实验环境与分组

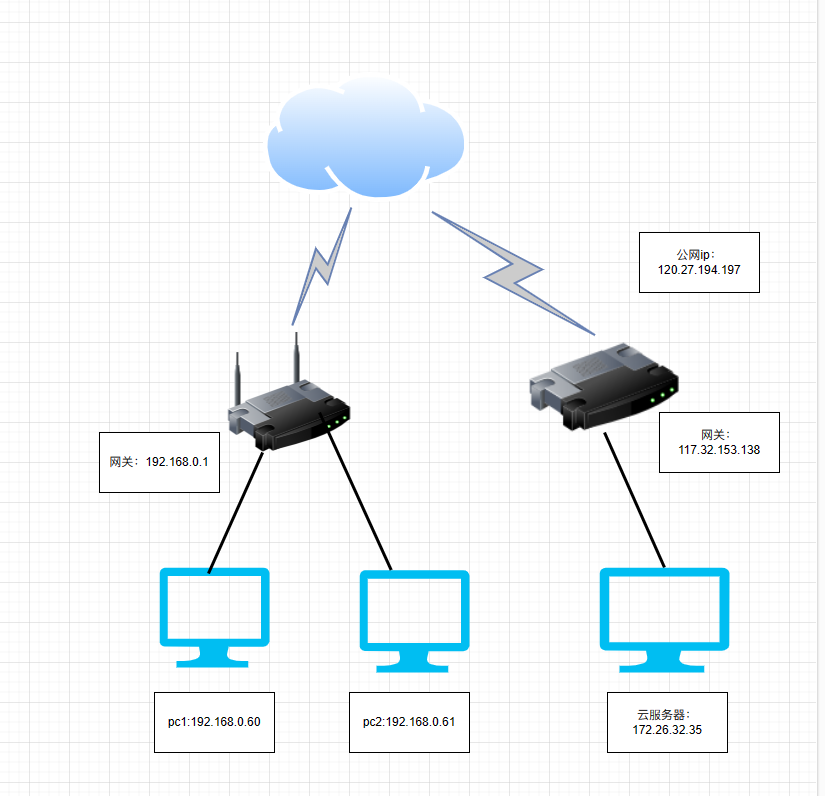
1）云服务器一台，启动Apache2服务（或其他服务器程序）。

2）每2名同学一组，各自在电脑上运行客户端程序（浏览器或其他客户端程序）。

3）使用客户端程序下载数据，运行Wireshark软件捕获报文。【注意：可以关闭Wireshark的HTTP协议分析，专注在TCP协议上，关闭方法是：菜单‘分析’—>‘启用的协议’中，取消‘HTTP’的选择。】

1. 实验组网

下图是本实验的组网图，图中参数请根据实际情况标注。

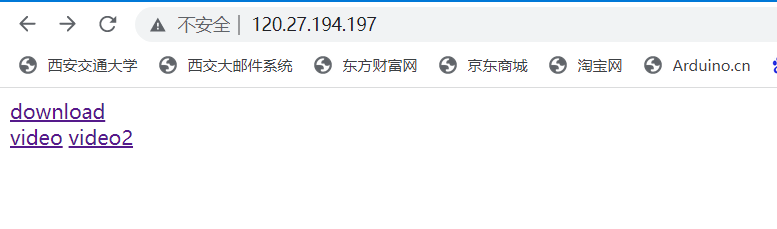


1. 实验过程及结果分析

【过程记录应当详尽，截图并加以说明。以下过程和表格仅供参考。】

步骤1：PC2登录到服务器Z上，在云服务器Z上启动web服务器（Apache、IIS等）。

在云服务器上编辑好index.html文件，放置三个链接，分别指向服务器上放的视频和图片，设置好后，访问服务器公有IP地址120.27.194.197，看到如下的网页界面：

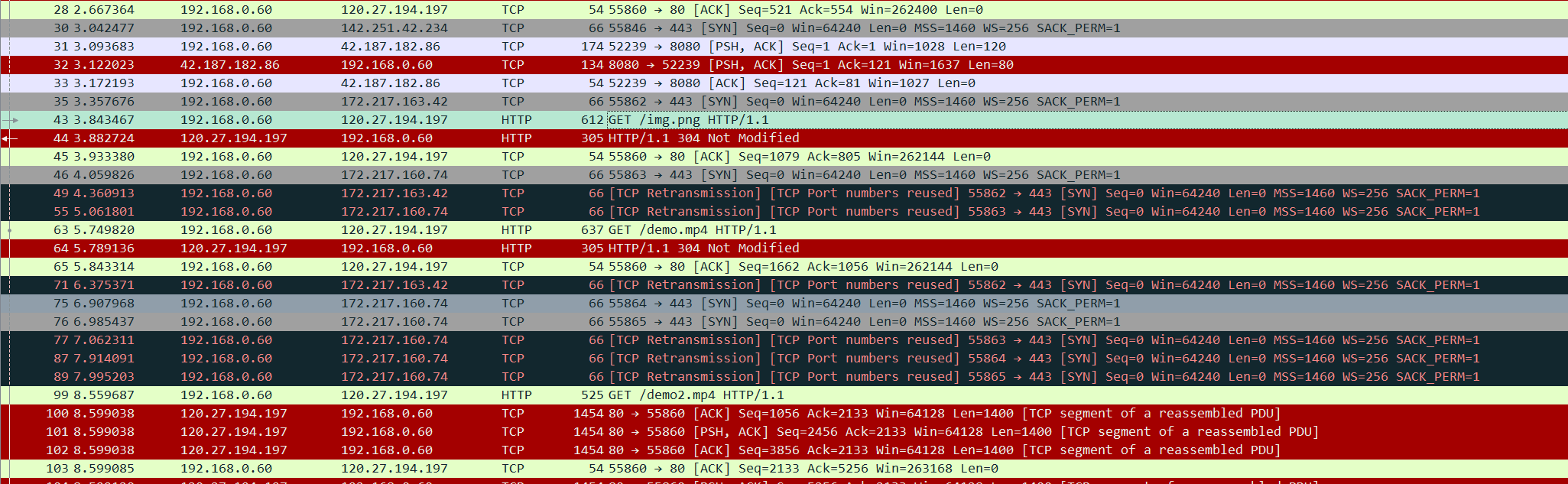


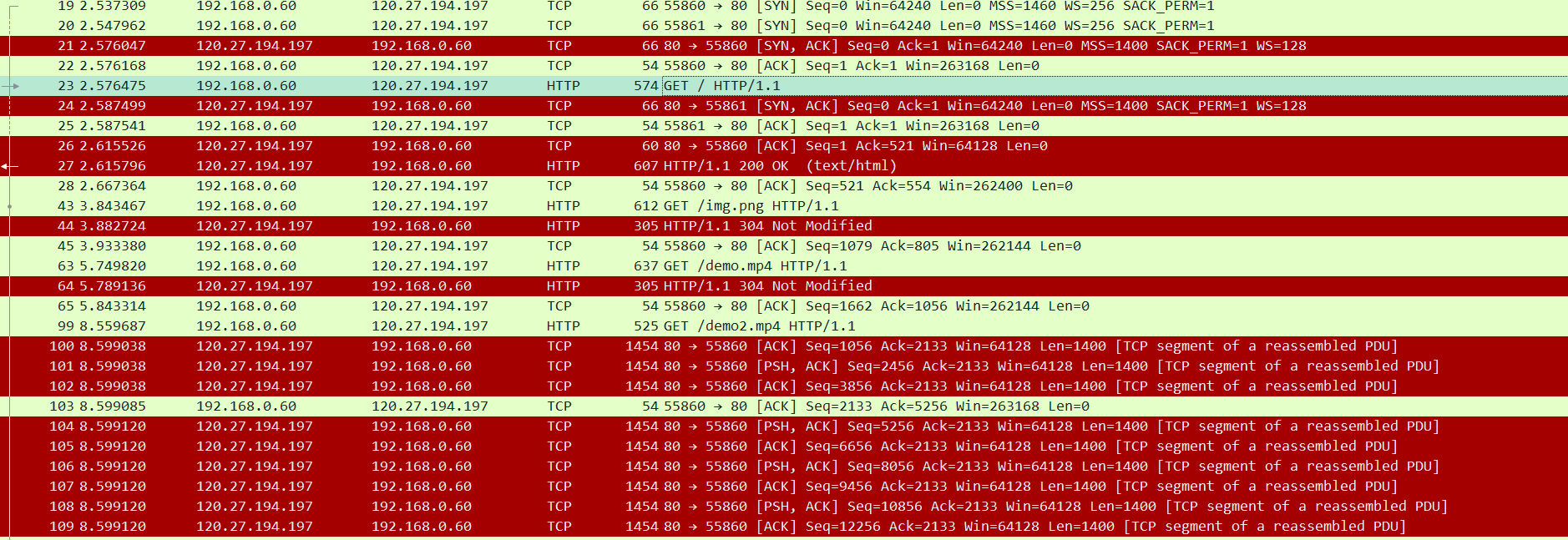
步骤2：在PC1和Z上启动报文捕获软件，开始截获报文（注意抓包一定要包括建立连接的报文）。

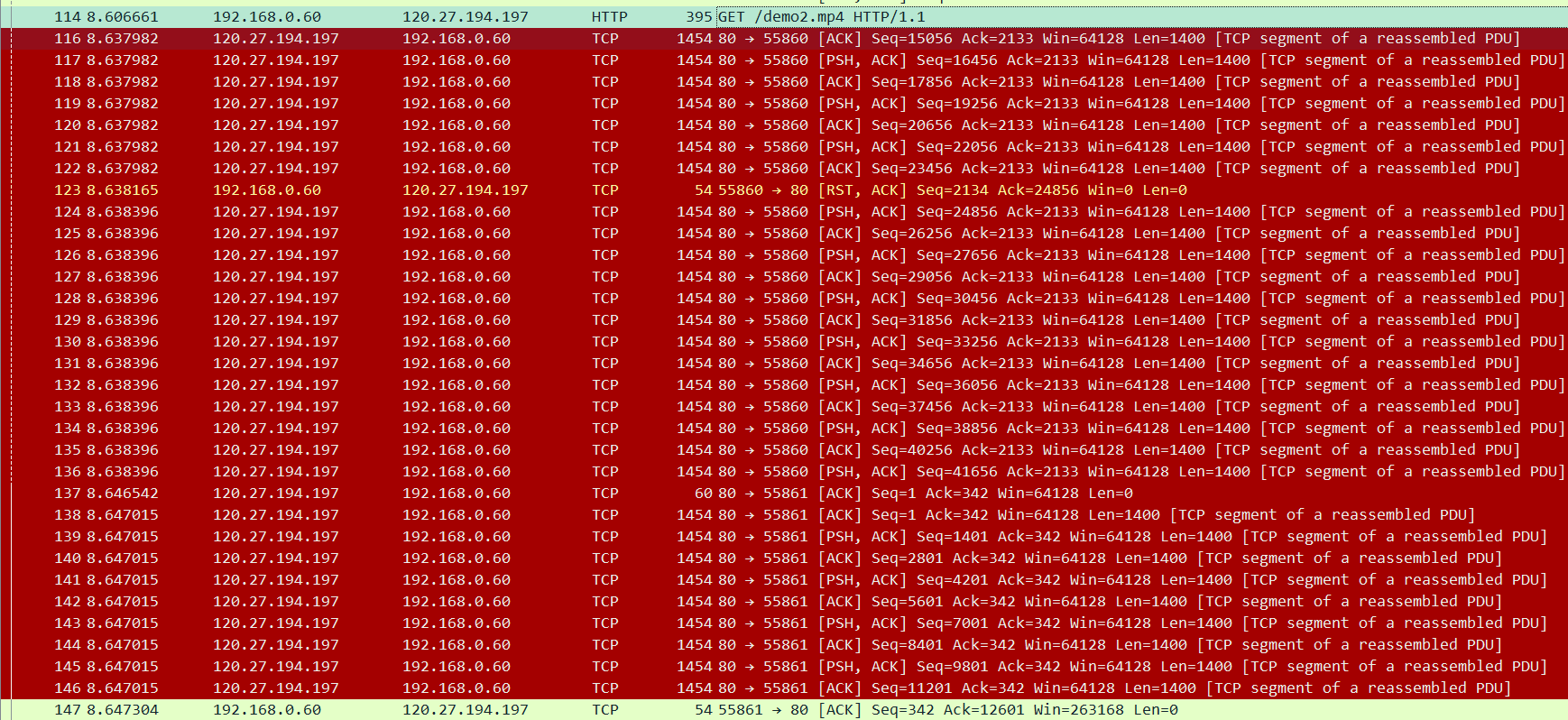
在PC1和服务器端同时运行wireshark，然后PC1开始准备浏览服务器网站首页。

步骤3：在PC1上运行客户端软件，发送和接收一个约500KB的文件。文件传输完成后，停止报文截获。

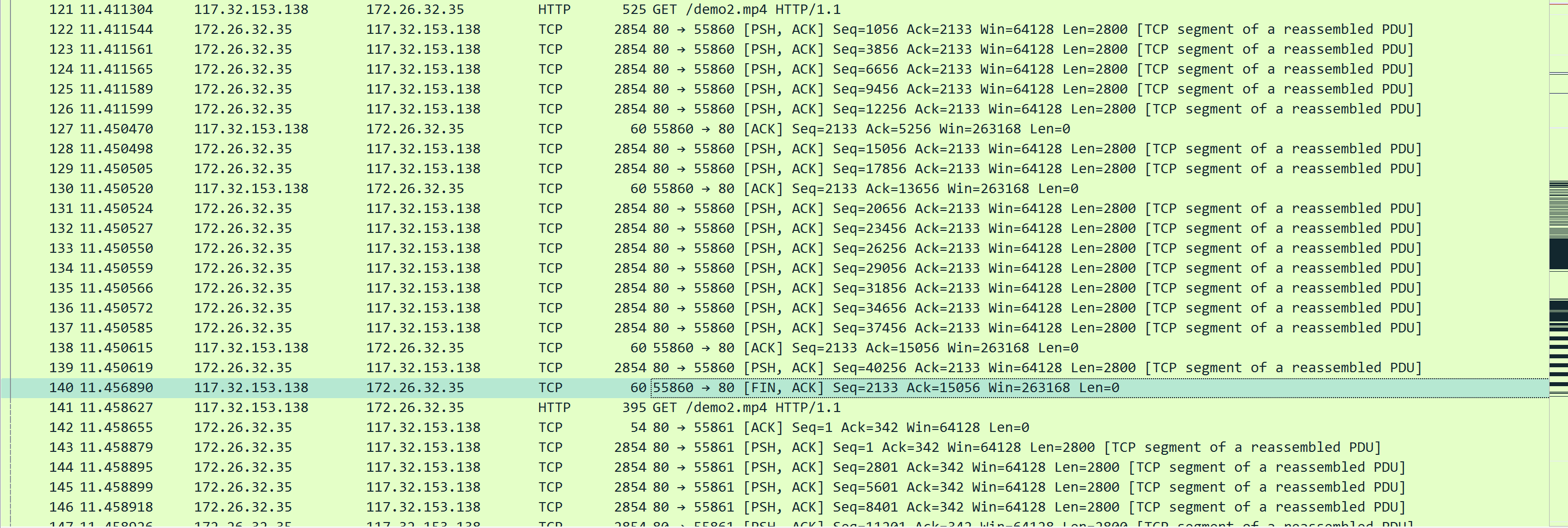
PC1客户端截获报文如下：





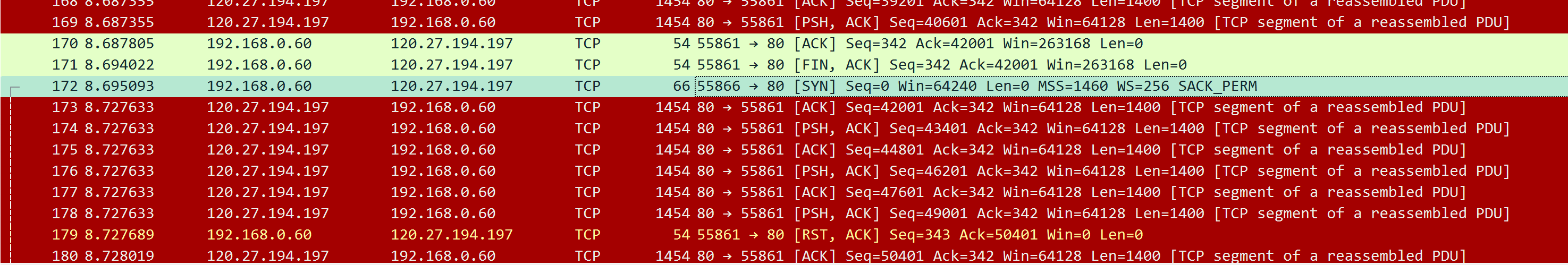


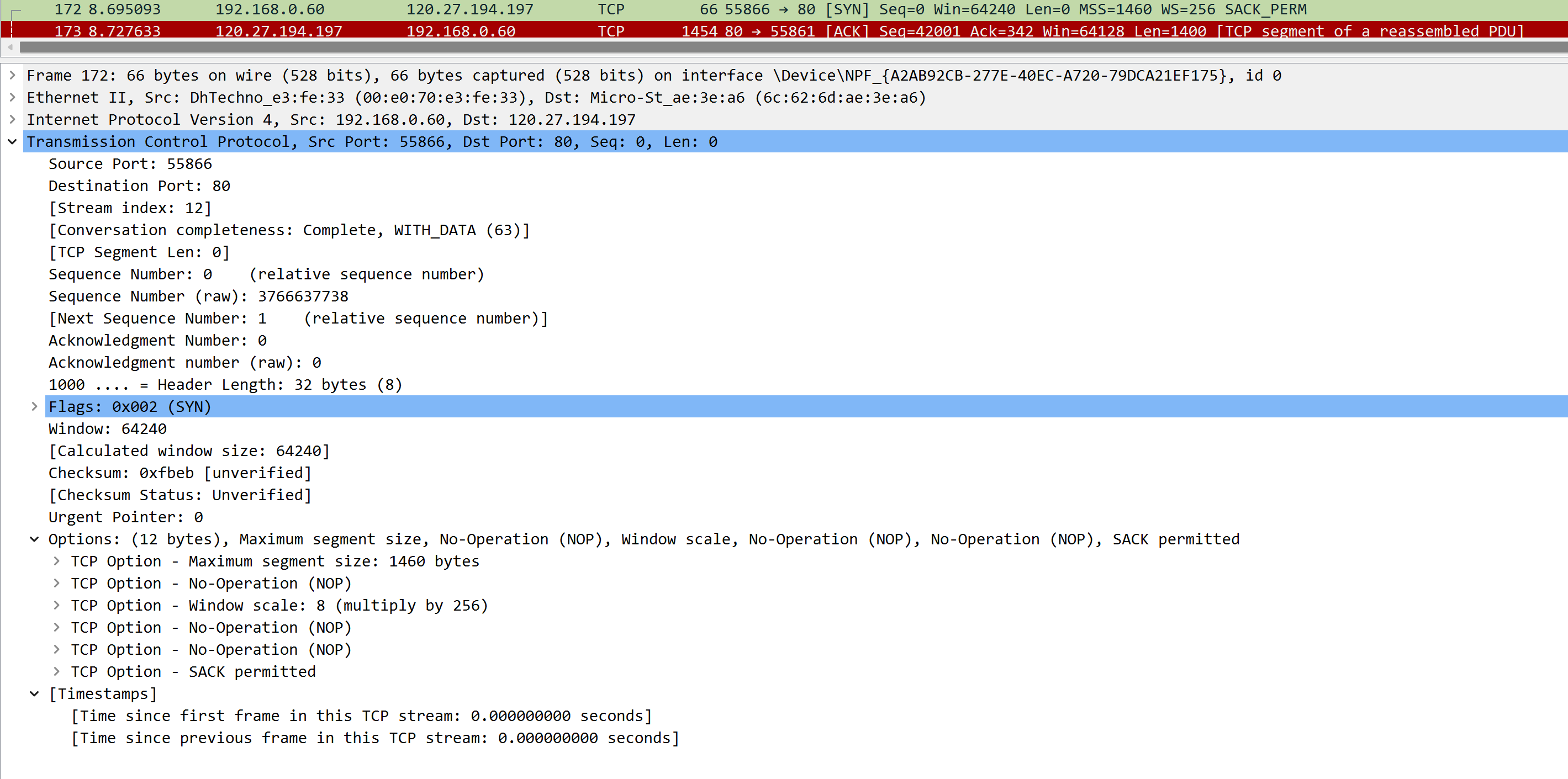
服务器端截获报文如下：



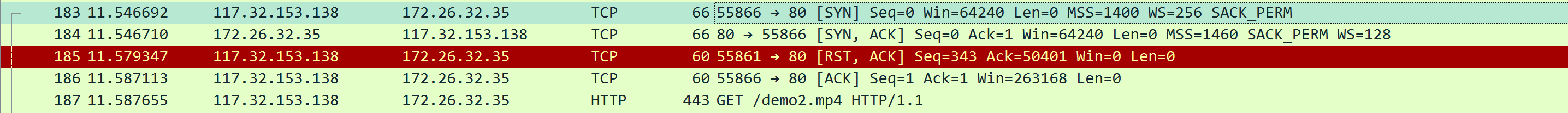
步骤4：对比观察客户端和服务器截获的报文，分析TCP协议的选项：MSS、SACK、Window Scale、Timestamp等，查资料说明其用途。

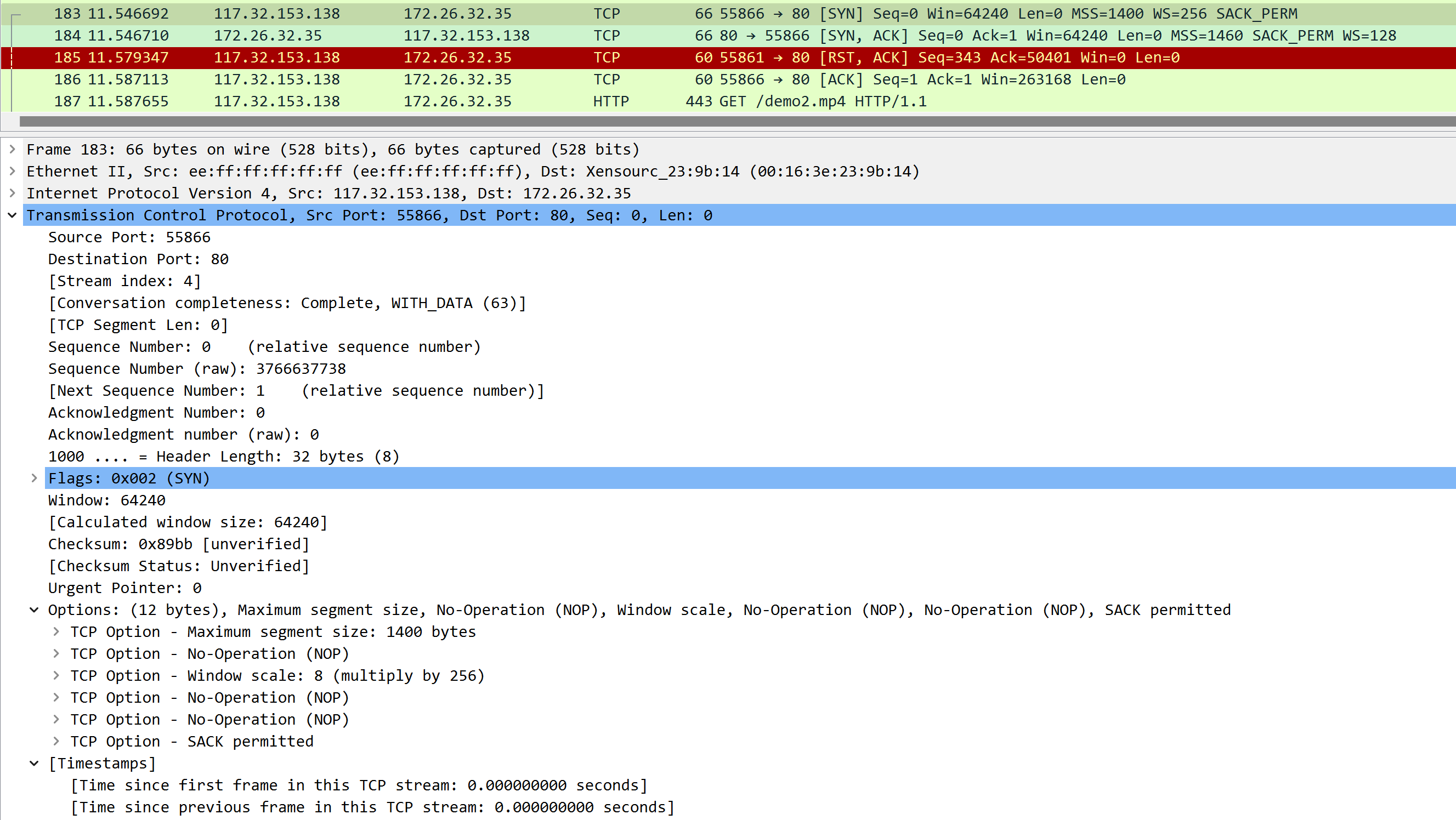
客户端：





服务端：





MSS，SACK，Window Scale，Timestamp都是在TCP建立连接的三次握手过程中建立的。MSS，SACK，Window Scale都包含在TCP报文的Options选项中。

（1）MSS（Maximum Segment Size）：最大段长，最大报文段尺寸。指的是TCP连接中允许发送的最大报文段大小。MSS由双方协商确定，以确保报文段不会因太大而无法在某个链路上传输。

图中客户端发给服务器的数据包MSS为1460bytes，服务端发给客户端的MSS为1400bytes，双方在建立TCP连接时协商选择两个中的较小值，也就是1400bytes为单位发送数据包。

（2）SACK（Selective Acknowledgment）：选择性确认，指的是TCP接收方可以对接收到的报文段进行选择性确认，而不是像传统的ACK那样只确认最后一个收到的报文段。SACK可以提高TCP连接的效率，因为它允许接收方更快地通知发送方哪些报文段已经被正确接收，而哪些报文段需要重传。

图中客户端发给服务端的数据包为SACK permitted，表示客户端允许服务端选择性确认。

（3）Window Scale：窗口缩放，指的是TCP连接中可以使用的最大窗口大小。Window Scale可以扩展TCP连接的窗口大小，从而提高连接的吞吐量。

图中数据包window scale为8（Multiply by 256），表示窗口大小扩展因子为8，即允许扩展为256倍。

（4）Timestamp：时间戳，指的是TCP报文段中的一个字段，用于指示报文段发送的时间。Timestamp可以用于提高TCP连接的可靠性，因为它可以帮助接收方判断报文段是否已经过时。

Timestamps包含两个字段：

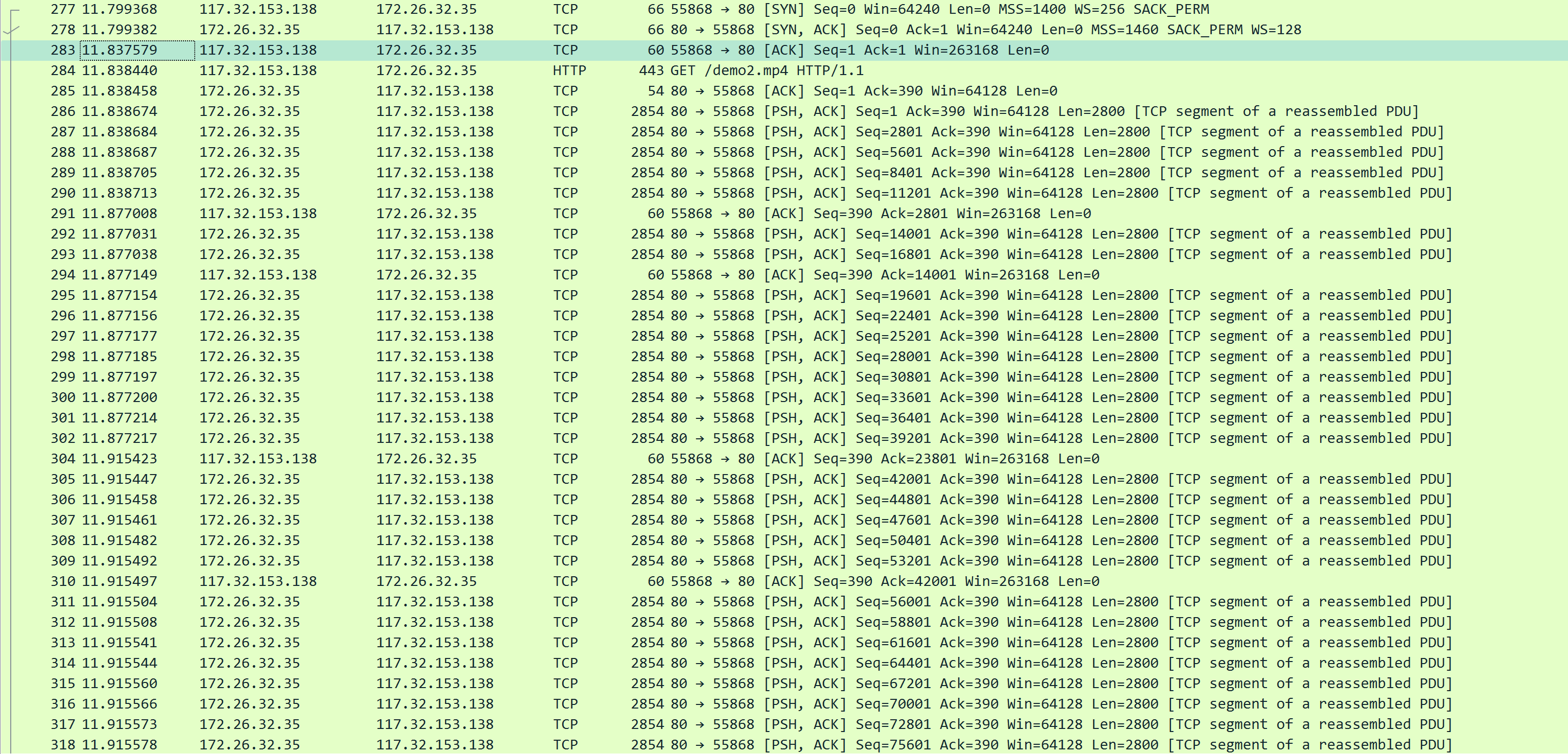
[Time since first frame in this TCP stream: 0.000000000 seconds]

表示从TCP第一帧开始计时的时间。

[Time since previous frame in this TCP stream: 0.000000000 seconds]

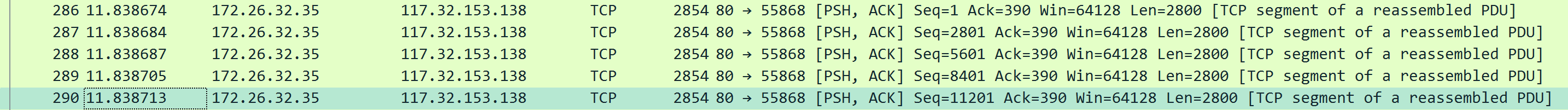
表示上一帧间隔的时间。

步骤5：分析TCP数据传送阶段的报文，分析其错误恢复和流量控制机制，并填表。【注：出现明显的流量控制的地方，Wireshark会有应答窗口的变化，乃至[TCP Window Full]或[TCP Zero Window]标记的报文出现。如果没有观察到明显的流量控制过程，可以再单独设计实验测试。比如编程设计接收端缓慢接收数据。】



由图中第二个和第三个报文（即TCP三次握手的后两次报文）计算RTT，得到RTT为0.038s。

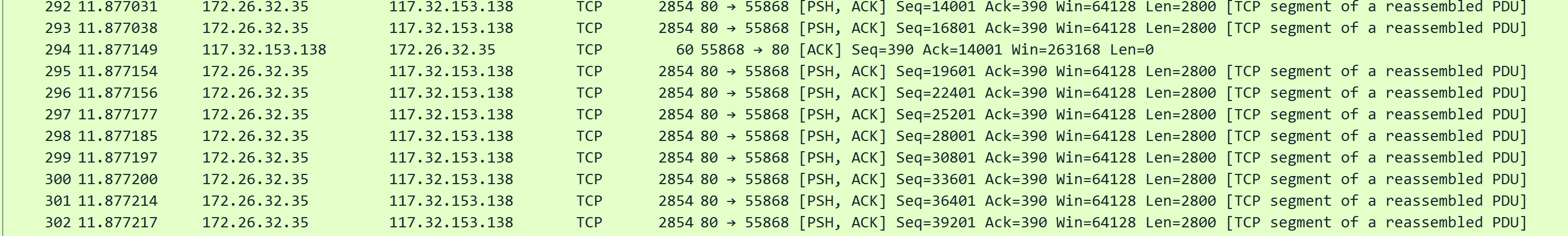
图中第四个报文是客户端发给服务端的HTTP请求，第五个报文是服务端对客户端的响应，len=0。



根据RTT可以判断，从第6个报文开始到第10个报文，这5个报文是连续的五个包，总大小是2800×5=14000=10\*MSS。

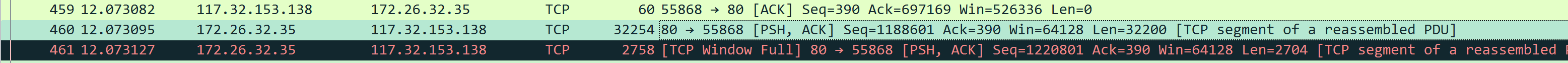


上图是客户端对服务端的确认信号，表示前5个包都收到了。



服务端收到确认后，将窗口大小翻倍，窗口大小变成了20MSS。

如此翻倍进行下去，直到出现了Window Full的数据包。



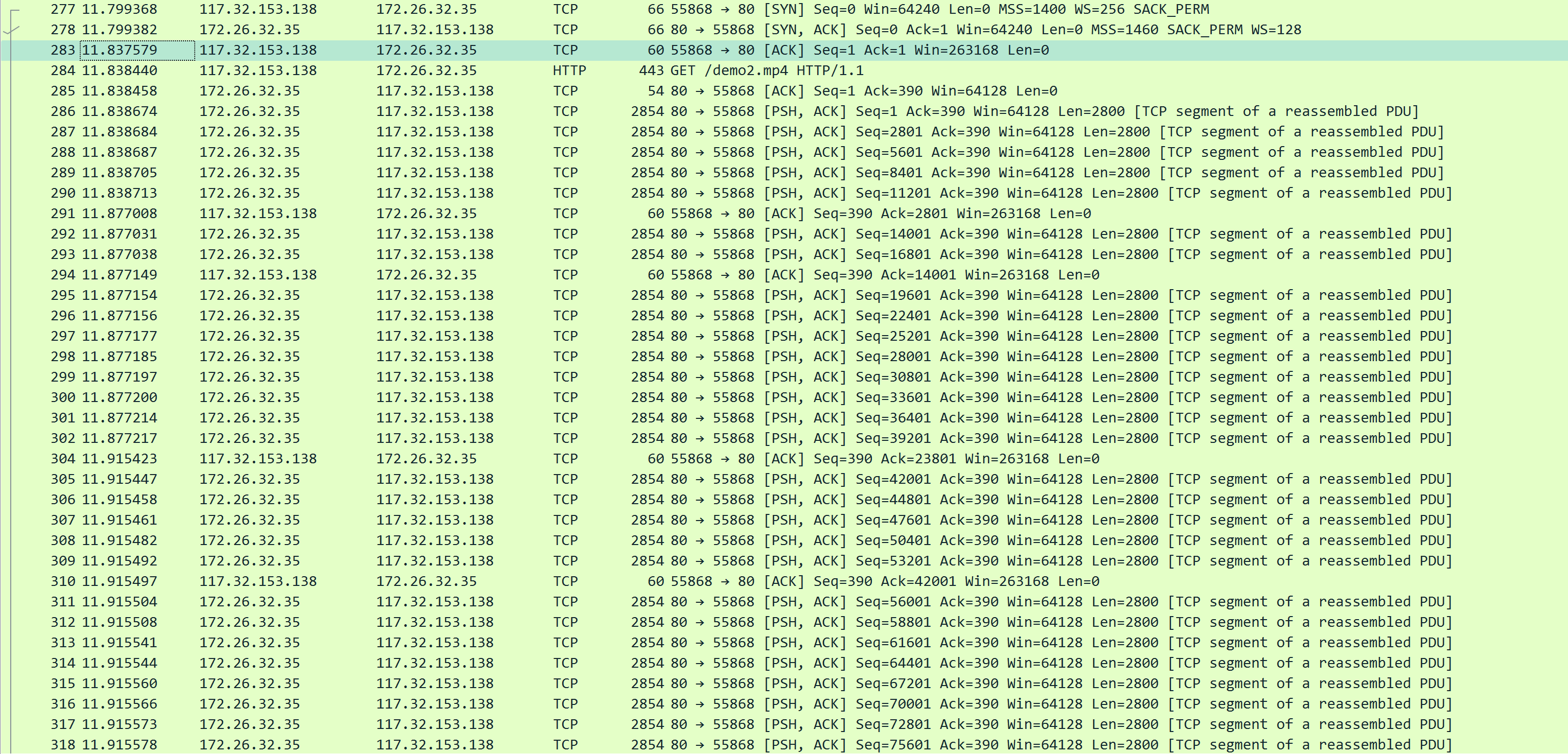
服务端发送的报文序列号seq为1188601，报文长度为32200，收到客户端的上一个确认报文的ACK为697169，所以计算得到服务端已发送未确认的报文大小为1188601+32200-697169=523632，所以剩下的窗口大小为526336-523632=2704，所以出现window full现象，发送的报文大小为2704。

表3-3 记录TCP数据传送阶段的报文

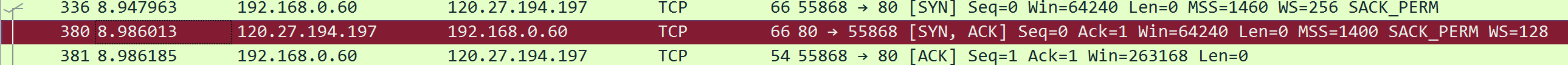
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 报文序号 | 报文种类 (数据/确认) | 序号字段Seq Number | 确认号Ack Number | 数据长度 | 确认到哪条报文（填序号） | 窗口大小 |
| 277 | SYN | 0 | 无 | 0 | 无 | 64240 |
| 278 | SYN,ACK | 0 | 1 | 0 | 277 | 64240 |
| 283 | ACK | 1 | 1 | 0 | 278 | 263168 |
| 286 | 数据 | 1 | 390 | 2800 | 无 |  |
| … |  |  |  |  |  |  |
| 290 | 数据 | 11201 | 390 | 2800 | 无 | 64128 |
| 291 | 确认 | 390 | 2801 | 0 | 287 | 263168 |
| … |  |  |  |  |  |  |
| 419 | 数据 | 686001 | 390 | 11168 | 无 | 64128 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

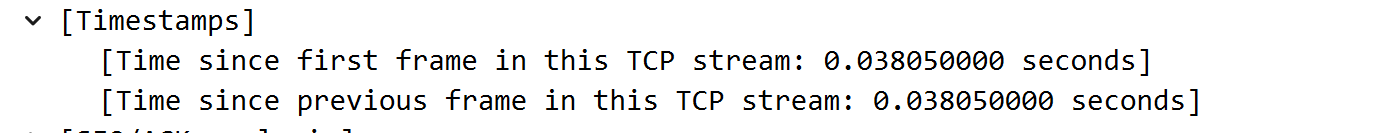
步骤6、分析客户机和服务器两边各自捕获到的分组，分析整个TCP流，估计双方的RTT。

服务端：



由图中第二个和第三个报文（即TCP三次握手的后两次报文）计算RTT，得到服务端RTT为0.038s。

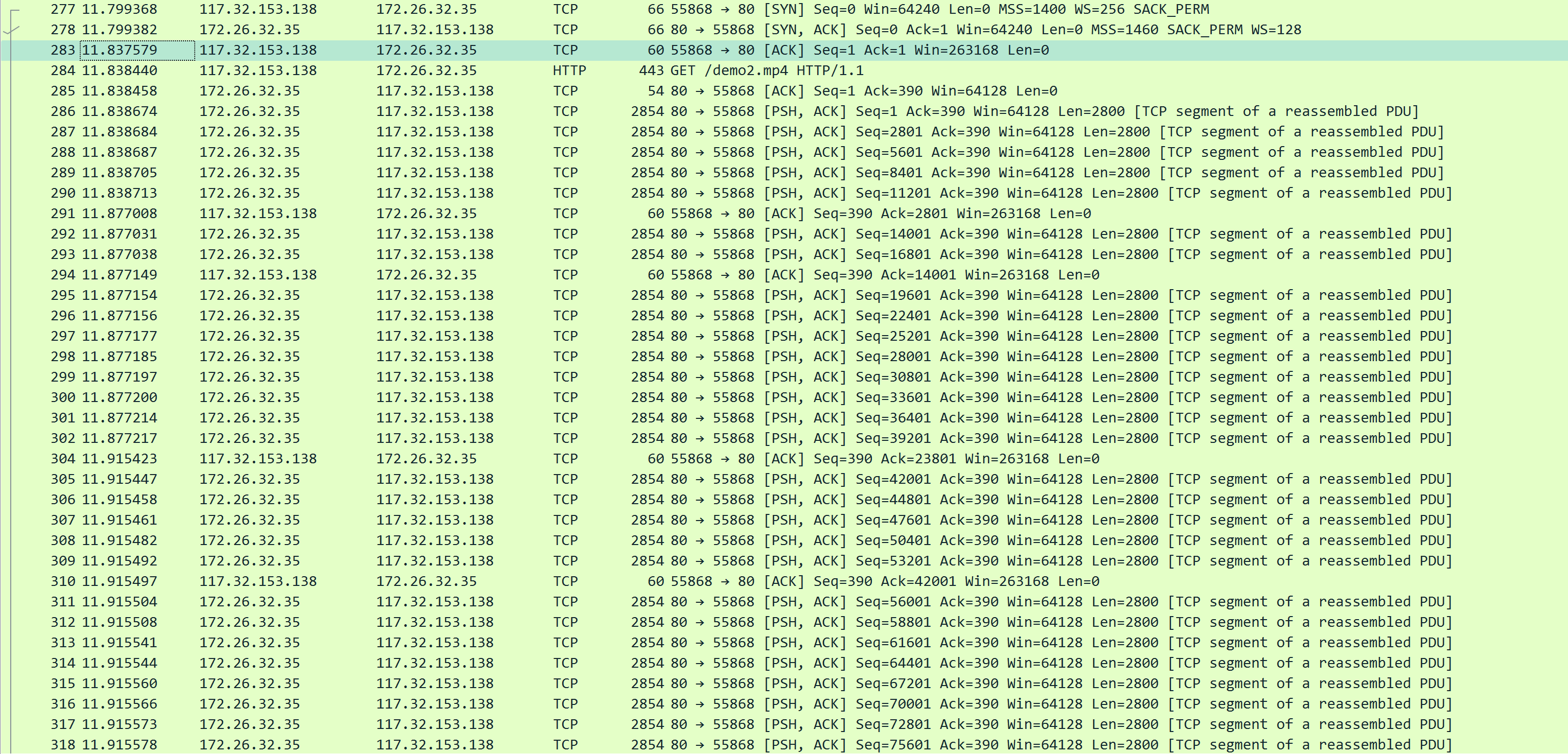




图中第二个报文中Timestamps里显示距离上一个报文0.038s，因此客户端的RTT也是0.038s。

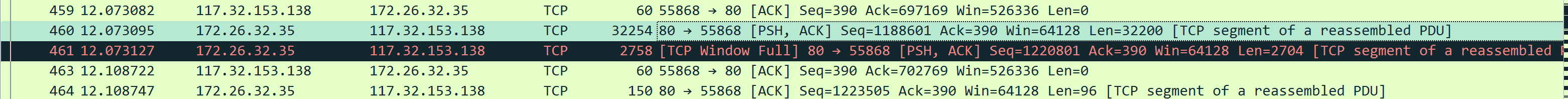
步骤7、分析整个TCP流的拥塞控制，找到拥塞控制的几个典型过程（即慢启动、快速重传等）。

慢启动：



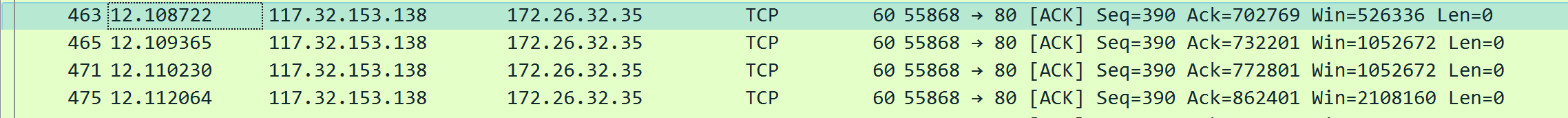
根据RTT可知，一次性发送的数据包大小为10MSS~20MSS~40MSS，每个RTT之间数据包大小翻倍，此过程为慢启动过程。

发生拥塞和拥塞控制：



Window Full现象发生。

快速重传：



发生拥塞后的快速重传阶段，窗口大小从526336变成1052672变成2108160，一直翻倍。

步骤8、如果拥塞控制的相关过程不明显，请设计合适的方法再次测试。

步骤9、完成其他可选的实验步骤。

1. 互动讨论主题

1）TCP的流量控制和拥塞控制有什么不同？

TCP 的流量控制和拥塞控制是两种不同的控制机制，它们分别针对网络通信中的不同问题。

流量控制解决的是发送方和接收方速率不匹配的问题。这是在通信的两端之间进行的控制，通常由接收方向发送方发送窗口大小的通告来实现，发送方根据接收方通告的窗口大小来控制发送数据的速率，确保不会超出接收方的处理能力。

拥塞控制解决的是避免网络资源被耗尽的问题，在网络中避免拥塞的发生，即防止网络中的数据包过多导致网络性能下降或崩溃。TCP 使用拥塞控制算法来动态调整发送数据的速率，以避免网络拥塞。常见的拥塞控制算法包括慢启动、拥塞避免、快重传和快恢复等。

2）TCP的流量控制是哪一方（接收、发送）来主导的？什么情况下会发生流量控制？

TCP 的流量控制是由接收方来主导的。如果接收方处理数据的速度比发送方发送数据的速度慢，接受方来不及接受而出现接收缓存不够用的情况，就会发生流量控制。

3）讨论传输层与其上下相邻层的关系；

传输层提供了端到端的数据传输服务，负责将应用层的数据分割成适合传输的报文段，并通过网络层进行传输，确保数据的可靠性、顺序性和流量控制。

上层是应用层，传输层为应用层提供了端到端的数据传输服务。应用层协议可以选择使用传输层提供的TCP或UDP协议，根据需要来决定数据传输的可靠性和效率。传输层提供的服务包括流量控制、拥塞控制、错误恢复等，这些服务有助于应用层实现可靠的数据传输。

下层是网络层，传输层向网络层提供了数据传输的服务。传输层将应用层的数据封装成报文段，并通过网络层的IP协议进行传输。

4）讨论TCP协议在传输实时语音流方面的优缺点。

优点：

（1）可靠性：TCP 提供可靠的数据传输，通过序号、确认和重传机制来确保数据的完整性和顺序性。这意味着在传输语音流时，不太可能出现数据丢失或乱序的情况，从而保证了语音的质量和连续性。

（2）流量控制和拥塞控制：TCP 提供了流量控制和拥塞控制机制，可以根据网络的情况动态调整数据传输的速率，避免网络拥塞，从而保证语音流的稳定传输。

（3）适应性强：TCP 协议能够适应各种网络环境，包括低带宽、高延迟、丢包等情况。它的拥塞控制算法能够在网络拥塞时自动调整发送速率，以适应当前的网络状况。

缺点：

（1）延迟较高：TCP 在保证可靠性的同时，会引入一定的延迟，包括连接建立的握手延迟、拥塞控制引入的传输延迟等。对于实时语音流应用而言，即时性是非常重要的，而 TCP 的延迟可能会对语音通话产生一定影响。

（2）不适用于丢包敏感的应用：TCP 的重传机制可以保证数据的可靠性，但在网络丢包率较高的情况下，重传可能会导致较大的延迟和带宽浪费，影响语音通话的实时性和质量。

（3）带宽利用率相对较低：TCP 的流量控制和拥塞控制机制会根据网络情况动态调整发送速率，但在网络质量良好的情况下，可能无法充分利用可用的带宽，导致带宽资源的浪费。