实验三、传输层TCP协议分析实验报告

组号：	3-3-1				
姓名：	李云广	学号：	2193712575	班级：	计算机93
姓名：	李怀邦	学号：	2193712530	班级：	计算机93

一、 实验目的

1. 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。
2. 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。
3. 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。
4. 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。

二、 实验内容

1. 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。
2. 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。
3. 观察和估计客户机到服务器的RTT，双方各自的MSS，计算丢包率及重传的流量。
4. 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，观察到两个过程即可】。

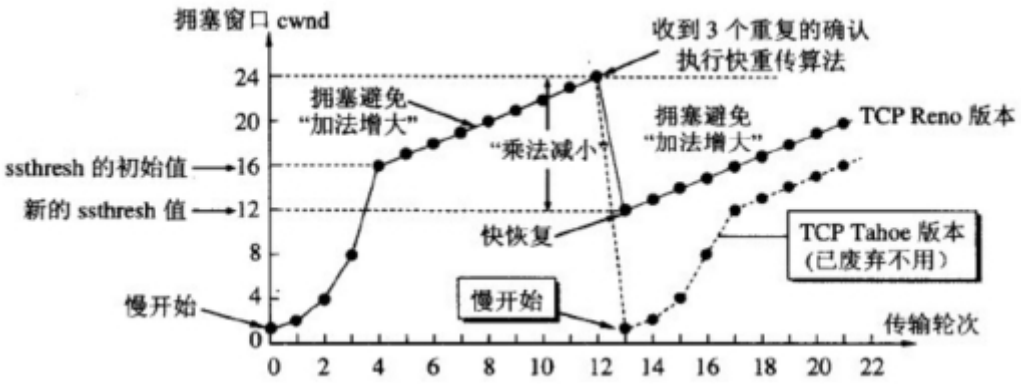


图4-0 典型的TCP 拥塞控制过程图例

5. *（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。观察有没有Delay ACK的应答模式，注意不同操作系统的差异。

【可以增加题目规定以外的分析】

三、 实验环境与分组

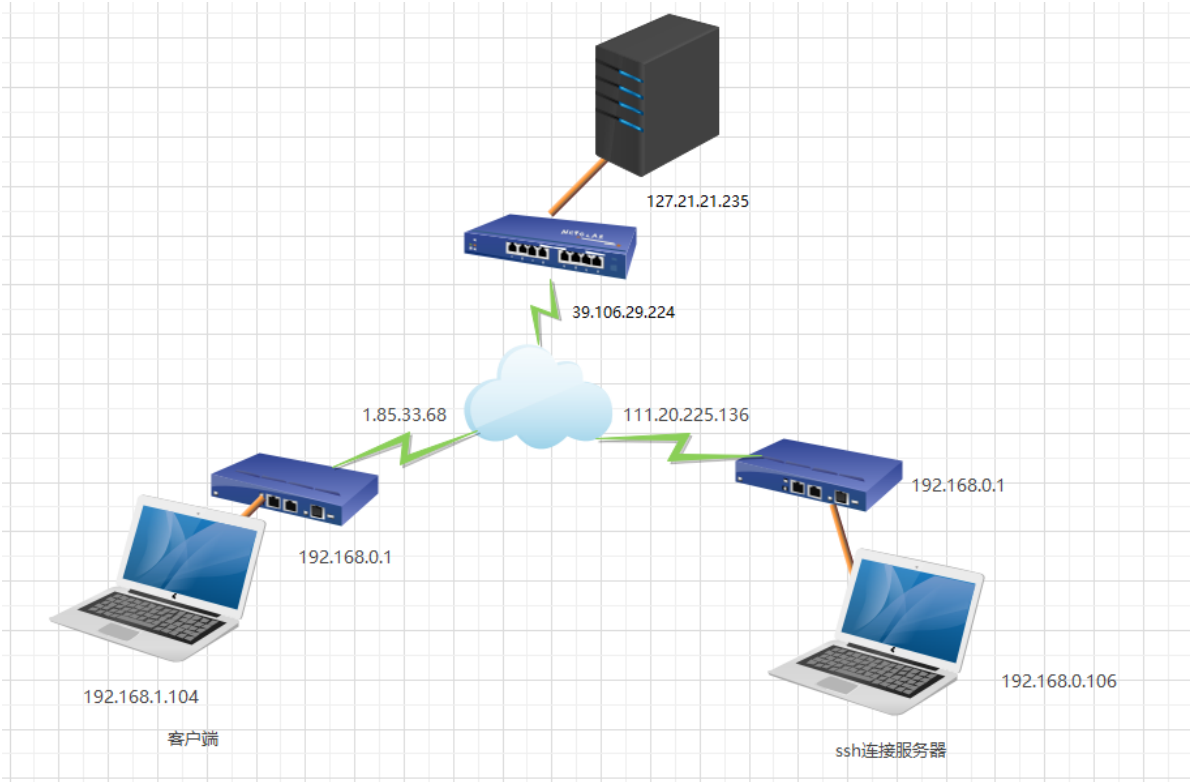
1）云服务器一台，启动Apache2服务（或其他服务器程序）。

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。】

四、 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。



五、 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通过ssh登录到服务器Z上，在云服务器上启动合适的服务器程序。

步骤2：

在PC1和Z上启动报文捕获软件，开始截获报文【注意加过滤器，比如host w.x.y.z；不熟悉tcpdump的可以用 tcpdump -n -s 500 tcp and host A.B.C.D and port P -w server.pcap选项，把报文记录到文件中，传输到客户端用Wireshark分析。其中A.B.C.D是客户端的公网地址，**P**是服务端口，如**80**】。

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。

步骤4：

对比观察客户端和服务器截获的报文，分析TCP协议的建立过程的三个报文并填写表3-1。分析TCP连接的释放过程，选择TCP连接撤销的四个报文并填写表3-2。

PC1在服务器端输入：

```
1 tcpdump -n -s 500 tcp -w server.pcap
```

在客户端PC2也启动wireshark抓包，立即使用xftp连接向服务器发送大约500kb的文件。

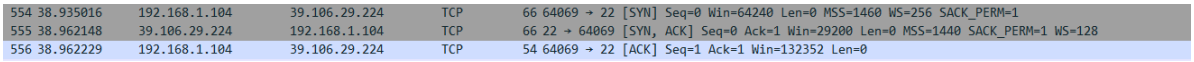
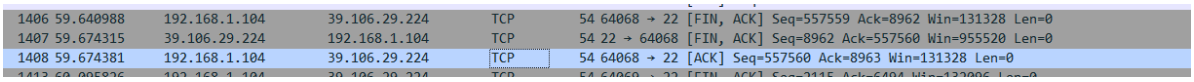


表3-1 TCP连接建立过程的三个报文信息【如果有多条，全部列出】

字段名称	第**1**条报文	第**2**条报文	第**3**条报文
报文序号NO.	554	555	556
Seq #	0	0	1
Ack #	0	1	1
ACK Flag	0	1	1
SYN Flag	1	1	0

表3-2 TCP连接撤销的四个报文信息



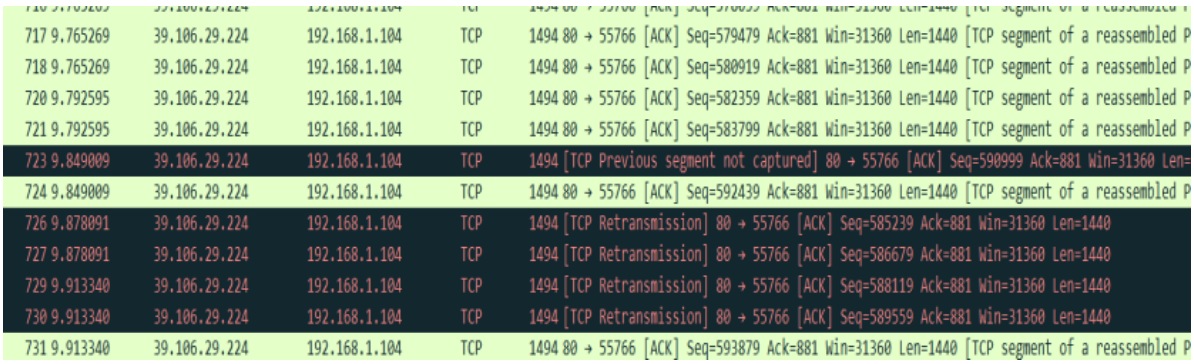
字段名称	首条报文	二条报文	三条报文	四条报文
报文捕获序号NO.	1406	1407	1408	
Seq #	557559	8962	557560	
Ack #	8962	557560	8963	
ACK	1	1	1	
FIN	1	1	0	

图中填写的均为相对序号。可以看到TCP连接撤销的四个报文信息仅仅抓到了三个，这是因为服务器返回客户端的ACK报文和服务器主动提出的FIN报文合为一个报文发送。

步骤**5:**

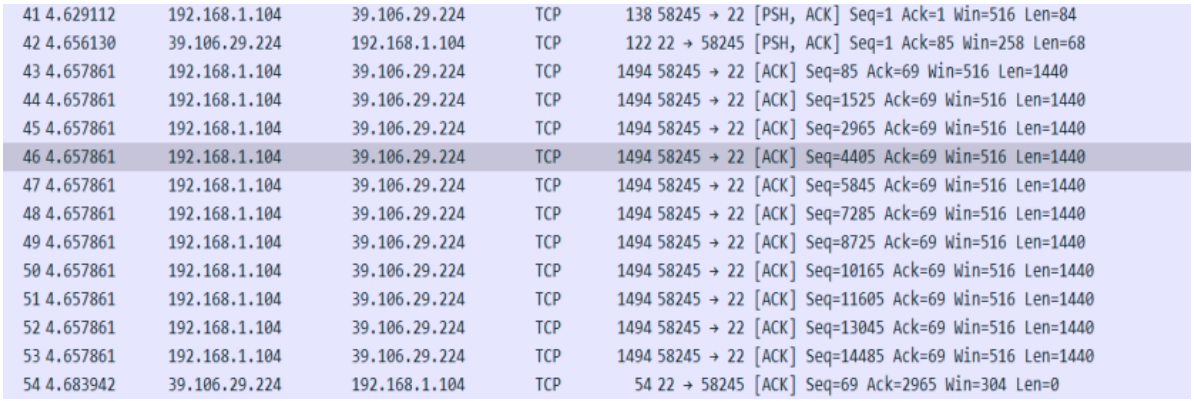
分析TCP数据传送阶段的报文，分析其错误恢复和流量控制机制，并填表。【注：出现明显的流量控制的地方，Wireshark会有[TCP Window Full]标记。如果没有观察到明显的流量控制过程，可以再单独设计实验测试。比如编程设计接收端缓慢接收数据。】

分析TCP数据传送阶段的错误恢复机制：



可以看到序号为723的报文的seq号为590999，然而服务器端连585239都没有收到，所以说进行了重传。

尝试许多次，始终无法抓到[window full]报文，于是仅仅对于普通的流量控制过程进行简单分析。



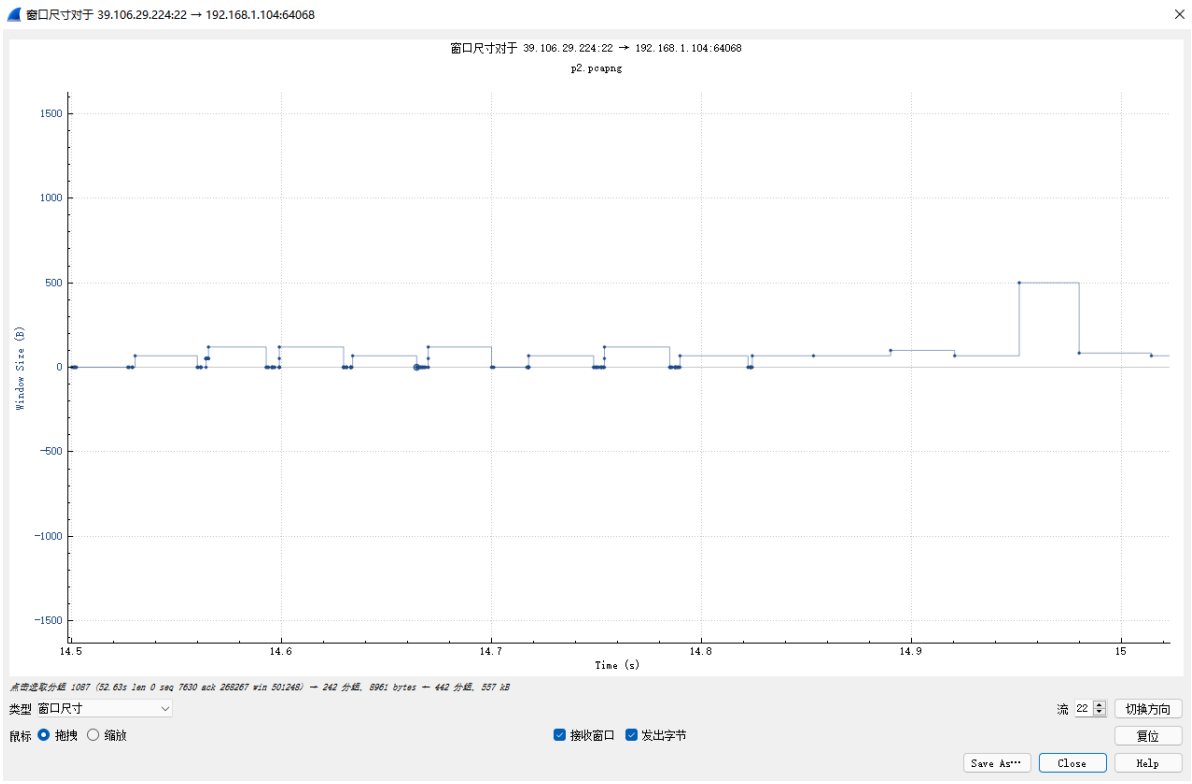
41 4.629112	192.168.1.104	39.106.29.224	TCP	138 58245 → 22 [PSH, ACK] Seq=1 Ack=1 Win=516 Len=84
42 4.656130	39.106.29.224	192.168.1.104	TCP	122 22 → 58245 [PSH, ACK] Seq=1 Ack=85 Win=258 Len=68
43 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=85 Ack=69 Win=516 Len=1440
44 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=1525 Ack=69 Win=516 Len=1440
45 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=2965 Ack=69 Win=516 Len=1440
46 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=4405 Ack=69 Win=516 Len=1440
47 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=5845 Ack=69 Win=516 Len=1440
48 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=7285 Ack=69 Win=516 Len=1440
49 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=8725 Ack=69 Win=516 Len=1440
50 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=10165 Ack=69 Win=516 Len=1440
51 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=11605 Ack=69 Win=516 Len=1440
52 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=13045 Ack=69 Win=516 Len=1440
53 4.657861	192.168.1.104	39.106.29.224	TCP	1494 58245 → 22 [ACK] Seq=14485 Ack=69 Win=516 Len=1440
54 4.683942	39.106.29.224	192.168.1.104	TCP	54 22 → 58245 [ACK] Seq=69 Ack=2965 Win=304 Len=0

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
41	数据	1	1	84		516
42	确认	1	85	68	41	258
43	数据	85	69	1440	42	516
44	数据	1525	69	1440	42	516
45	数据	2965	69	1440	42	516
46	数据	4405	69	1440	42	516
47	数据	5845	69	1440	42	516
48	数据	7285	69	1440	42	516
49	数据	8725	69	1440	42	516
50	数据	10165	69	1440	42	516
51	数据	11605	69	1440	42	516
54	确认	69	2965	0	44	304

上表可以看到服务器的Window字段发生了变化，由开始的win = 258变到win = 304，这就是服务器端的流量控制的过程。

通过wireshark的统计分析，可以看到实际的流量控制非常复杂，接收窗口在不断变化：



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，重传率和重传流量，平均传输速度等参数。

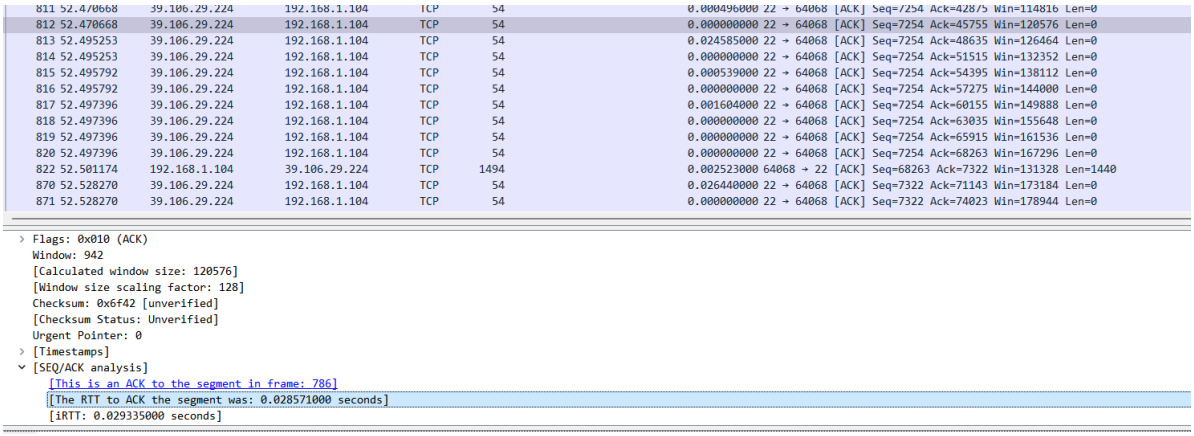
RTT即是一个报文从发出到接收到他的应答报文所用的总时间，可以直接使用wireshark进行分析。

在过滤器中加入一句话:

```
&& tcp.analysis.ack_rtt
```

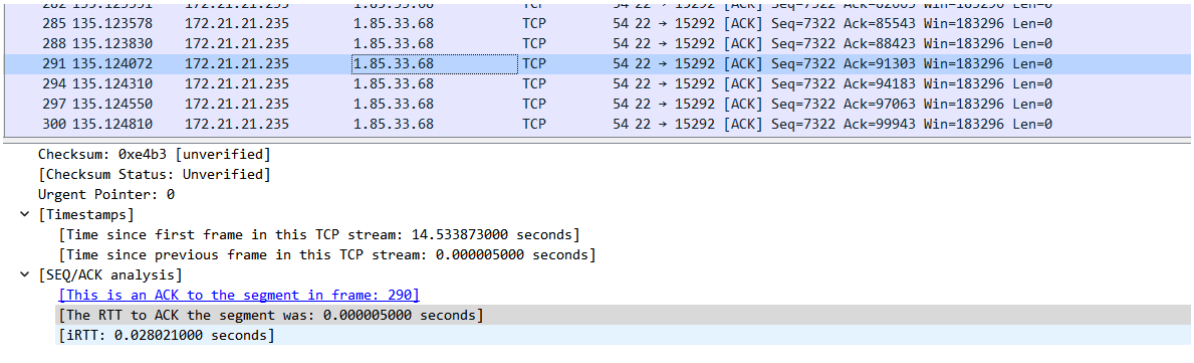
即可以在报文中自动出现计算好的rtt的值。

在客户端我们分析：



发现客户端对服务器发送消息的rtt的值均为0.027s左右。

在服务器端我们分析

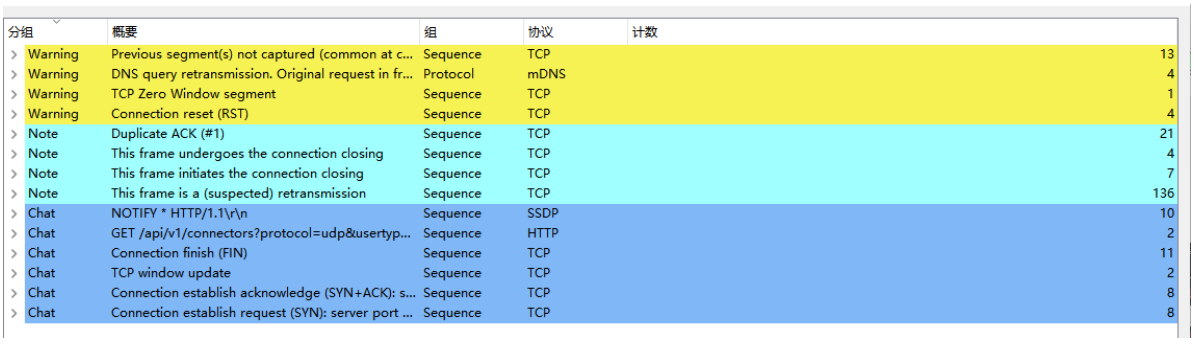


285	135.123578	172.21.21.235	1.85.33.68	TCP	54 22 → 15292 [ACK] Seq=7322 Ack=85543 Win=183296 Len=0
288	135.123830	172.21.21.235	1.85.33.68	TCP	54 22 → 15292 [ACK] Seq=7322 Ack=88423 Win=183296 Len=0
291	135.124072	172.21.21.235	1.85.33.68	TCP	54 22 → 15292 [ACK] Seq=7322 Ack=91303 Win=183296 Len=0
294	135.124310	172.21.21.235	1.85.33.68	TCP	54 22 → 15292 [ACK] Seq=7322 Ack=94183 Win=183296 Len=0
297	135.124550	172.21.21.235	1.85.33.68	TCP	54 22 → 15292 [ACK] Seq=7322 Ack=97063 Win=183296 Len=0
300	135.124810	172.21.21.235	1.85.33.68	TCP	54 22 → 15292 [ACK] Seq=7322 Ack=99943 Win=183296 Len=0

```
Checksum: 0xe4b3 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
▾ [Timestamps]
    [Time since first frame in this TCP stream: 14.533873000 seconds]
    [Time since previous frame in this TCP stream: 0.000005000 seconds]
▾ [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 290]
    [The RTT to ACK the segment was: 0.000005000 seconds]
    [iRTT: 0.028021000 seconds]
```

发现服务器对客户端回复消息的rtt值为0,000005s左右。

此外发现本次传输文件过程中没有任何重传现象，于是尝试再重新传一个文件进行抓包处理：



分组	概要	组	协议	计数
Warning	Previous segment(s) not captured (common at c...	Sequence	TCP	13
Warning	DNS query retransmission. Original request in fr...	Protocol	mDNS	4
Warning	TCP Zero Window segment	Sequence	TCP	1
Warning	Connection reset (RST)	Sequence	TCP	4
Note	Duplicate ACK (#1)	Sequence	TCP	21
Note	This frame undergoes the connection closing	Sequence	TCP	4
Note	This frame initiates the connection closing	Sequence	TCP	7
Note	This frame is a (suspected) retransmission	Sequence	TCP	136
Chat	NOTIFY * HTTP/1.1\r\n	Sequence	SSDP	10
Chat	GET /api/v1/connectors?protocol=udp&usertyp...	Sequence	HTTP	2
Chat	Connection finish (FIN)	Sequence	TCP	11
Chat	TCP window update	Sequence	TCP	2
Chat	Connection establish acknowledge (SYN+ACK): s...	Sequence	TCP	8
Chat	Connection establish request (SYN): server port ...	Sequence	TCP	8

发现有许多重传数据包。

尝试计算总数据包的数量：

$$\text{数据包数量} = \frac{\text{文件大小}}{MSS\text{大小}} = \frac{533144B}{1440B} = 370.23\text{个} = 371\text{个}$$

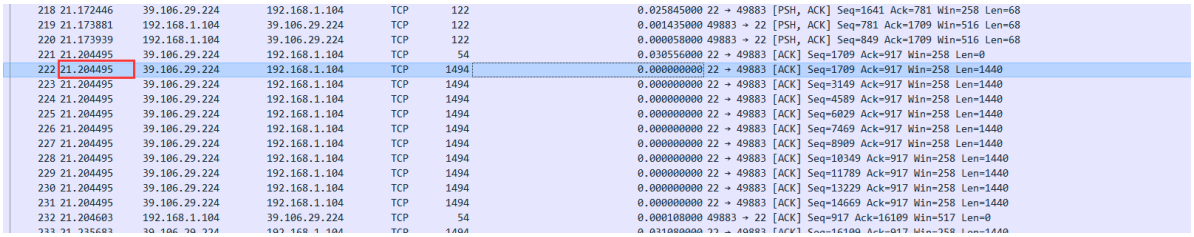
总共有371个数据包传送，重传数据包有136个，所以重传率为：

$$\text{重传率} = \frac{136}{371} = 36.66\%$$

重传流量为：

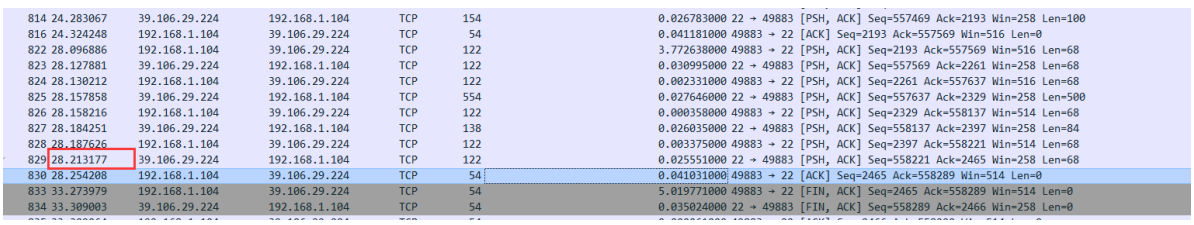
$$\text{重传流量} = 136 \times 1440B = 191.25KB$$

起始时间：



218	21.172446	39.106.29.224	192.168.1.104	TCP	122	0.025845000 22 → 49883 [PSH, ACK] Seq=1641 Ack=781 Win=258 Len=68
219	21.173881	192.168.1.104	39.106.29.224	TCP	122	0.001435000 49883 → 22 [PSH, ACK] Seq=781 Ack=1709 Win=516 Len=68
220	21.173939	192.168.1.104	39.106.29.224	TCP	122	0.000058000 49883 → 22 [PSH, ACK] Seq=849 Ack=1709 Win=516 Len=68
221	21.204495	39.106.29.224	192.168.1.104	TCP	54	0.030556000 22 → 49883 [ACK] Seq=1709 Ack=917 Win=258 Len=0
222	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=1709 Ack=917 Win=258 Len=1440
223	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=3149 Ack=917 Win=258 Len=1440
224	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=4589 Ack=917 Win=258 Len=1440
225	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=6029 Ack=917 Win=258 Len=1440
226	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=7469 Ack=917 Win=258 Len=1440
227	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=8909 Ack=917 Win=258 Len=1440
228	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=10349 Ack=917 Win=258 Len=1440
229	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=11789 Ack=917 Win=258 Len=1440
230	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=13229 Ack=917 Win=258 Len=1440
231	21.204495	39.106.29.224	192.168.1.104	TCP	1494	0.000000000 22 → 49883 [ACK] Seq=14669 Ack=917 Win=258 Len=1440
232	21.204603	192.168.1.104	39.106.29.224	TCP	54	0.000108000 49883 → 22 [ACK] Seq=917 Ack=16109 Win=517 Len=0

结束时间：



814	24.283067	39.106.29.224	192.168.1.104	TCP	154	0.026783000 22 → 49883 [PSH, ACK] Seq=557469 Ack=2193 Win=258 Len=100
816	24.324248	192.168.1.104	39.106.29.224	TCP	54	0.041181000 49883 → 22 [ACK] Seq=2193 Ack=557569 Win=516 Len=0
822	28.096886	192.168.1.104	39.106.29.224	TCP	122	3.772638000 49883 → 22 [PSH, ACK] Seq=2193 Ack=557569 Win=516 Len=68
823	28.127881	39.106.29.224	192.168.1.104	TCP	122	0.030995000 22 → 49883 [PSH, ACK] Seq=557569 Ack=2261 Win=258 Len=68
824	28.130212	192.168.1.104	39.106.29.224	TCP	122	0.002331000 49883 → 22 [PSH, ACK] Seq=2261 Ack=557637 Win=516 Len=68
825	28.157858	39.106.29.224	192.168.1.104	TCP	554	0.027646000 22 → 49883 [PSH, ACK] Seq=557637 Ack=2329 Win=258 Len=500
826	28.158216	192.168.1.104	39.106.29.224	TCP	122	0.000358000 49883 → 22 [PSH, ACK] Seq=2329 Ack=558137 Win=514 Len=68
827	28.184251	39.106.29.224	192.168.1.104	TCP	138	0.026035000 22 → 49883 [PSH, ACK] Seq=558137 Ack=2397 Win=258 Len=84
828	28.187626	192.168.1.104	39.106.29.224	TCP	122	0.003375000 49883 → 22 [PSH, ACK] Seq=2397 Ack=558221 Win=514 Len=68
829	28.213177	39.106.29.224	192.168.1.104	TCP	122	0.025551000 22 → 49883 [PSH, ACK] Seq=558221 Ack=2465 Win=258 Len=68
830	28.254208	192.168.1.104	39.106.29.224	TCP	54	0.041031000 49883 → 22 [ACK] Seq=2465 Ack=558289 Win=514 Len=0
833	33.273979	192.168.1.104	39.106.29.224	TCP	54	5.019771000 49883 → 22 [FIN, ACK] Seq=2465 Ack=558289 Win=514 Len=0
834	33.309003	39.106.29.224	192.168.1.104	TCP	54	0.035024000 22 → 49883 [FIN, ACK] Seq=558289 Ack=2466 Win=258 Len=0

总时间：

$$time = 28.213177 - 21.204995s = 7.008182s$$

平均传输速度为：

$$v = \frac{size}{time} = \frac{533144B}{7.008182s} = 76163.43Bps$$

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、拥塞避免，快速恢复），计算各个时期发送数据平均传输速度。

首先观察一下传输开始阶段的数据包传送个数：

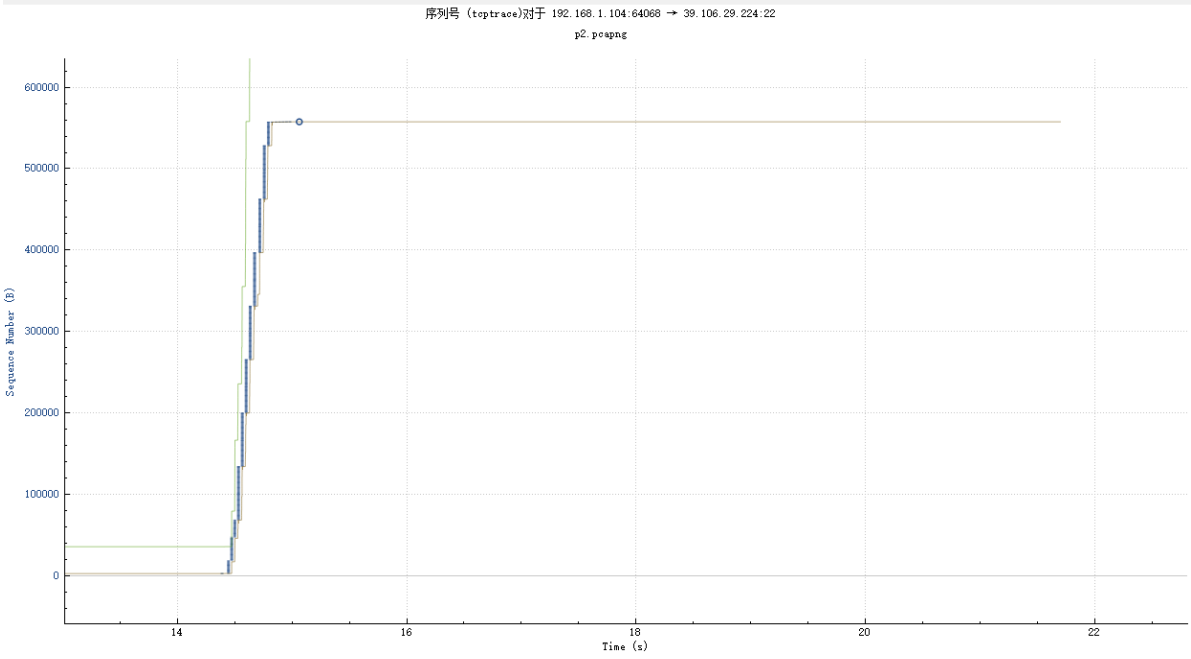
752	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.029554000 64068 → 22 [ACK] Seq=2555 Ack=7254 Win=131328 Len=1440
753	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=3995 Ack=7254 Win=131328 Len=1440
754	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=5435 Ack=7254 Win=131328 Len=1440
755	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=6875 Ack=7254 Win=131328 Len=1440
756	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=8315 Ack=7254 Win=131328 Len=1440
757	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=9755 Ack=7254 Win=131328 Len=1440
758	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=11195 Ack=7254 Win=131328 Len=1440
759	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=12635 Ack=7254 Win=131328 Len=1440
760	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=14075 Ack=7254 Win=131328 Len=1440
761	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=15515 Ack=7254 Win=131328 Len=1440
762	52.414282	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=16955 Ack=7254 Win=131328 Len=1440
763	52.442024	39.106.29.224	192.168.1.104	TCP	54	0.027742000 22 → 64068 [ACK] Seq=7254 Ack=5435 Win=38912 Len=0
764	52.442024	39.106.29.224	192.168.1.104	TCP	54	0.000000000 22 → 64068 [ACK] Seq=7254 Ack=8315 Win=44672 Len=0
765	52.442024	39.106.29.224	192.168.1.104	TCP	54	0.000000000 22 → 64068 [ACK] Seq=7254 Ack=11195 Win=50560 Len=0
766	52.442024	39.106.29.224	192.168.1.104	TCP	54	0.000000000 22 → 64068 [ACK] Seq=7254 Ack=14075 Win=56448 Len=0
767	52.442024	39.106.29.224	192.168.1.104	TCP	54	0.000000000 22 → 64068 [ACK] Seq=7254 Ack=16955 Win=62208 Len=0
768	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000073000 64068 → 22 [ACK] Seq=18395 Ack=7254 Win=131328 Len=1440
769	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=19835 Ack=7254 Win=131328 Len=1440

可以看出初始阶段No752到No762均为客户端传送的数据包，一共11个，但是分析服务器端的ACK报文可以发现，服务器端的ACK在第一次传输阶段ACK序号仅仅到16955，也就是说服务器端仅仅对于No761进行了确认，并没有对No762进行确认，由此我们得出第一个阶段服务器通过拥塞控制接收了10个报文，最后的第11个报文虽然也接收了但是却在下一阶段进行确认。

767	52.442024	39.106.29.224	192.168.1.104	TCP	54	0.000000000 22 → 64068 [ACK] Seq=7254 Ack=16955 Win=62208 Len=0
768	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000073000 64068 → 22 [ACK] Seq=18395 Ack=7254 Win=131328 Len=1440
769	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=19835 Ack=7254 Win=131328 Len=1440
770	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=21275 Ack=7254 Win=131328 Len=1440
771	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=22715 Ack=7254 Win=131328 Len=1440
772	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=24155 Ack=7254 Win=131328 Len=1440
773	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=25595 Ack=7254 Win=131328 Len=1440
774	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=27035 Ack=7254 Win=131328 Len=1440
775	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=28475 Ack=7254 Win=131328 Len=1440
776	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=29915 Ack=7254 Win=131328 Len=1440
777	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=31355 Ack=7254 Win=131328 Len=1440
778	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=32795 Ack=7254 Win=131328 Len=1440
779	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [PSH, ACK] Seq=34235 Ack=7254 Win=131328 Len=1440
780	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=35675 Ack=7254 Win=131328 Len=1440
781	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=37115 Ack=7254 Win=131328 Len=1440
782	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=38555 Ack=7254 Win=131328 Len=1440
783	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=39995 Ack=7254 Win=131328 Len=1440
784	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=41435 Ack=7254 Win=131328 Len=1440
785	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=42875 Ack=7254 Win=131328 Len=1440
786	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=44315 Ack=7254 Win=131328 Len=1440
787	52.442097	192.168.1.104	39.106.29.224	TCP	1494	0.000000000 64068 → 22 [ACK] Seq=45755 Ack=7254 Win=131328 Len=1440
788	52.468616	39.106.29.224	192.168.1.104	TCP	54	0.026519000 22 → 64068 [ACK] Seq=7254 Ack=19835 Win=68096 Len=0
789	52.468616	39.106.29.224	192.168.1.104	TCP	54	0.000000000 22 → 64068 [ACK] Seq=7254 Ack=22715 Win=73856 Len=0

在下一阶段可以看出连续传输了二十条报文，即先以cwnd = 10 起始，然后窗口乘2，因此基本符合拥塞控制中慢启动的特征。

通过统计分析也可以看出慢启动的过程：



可以看到刚开始启动的过程中连续发送的数据包个数较少，慢慢才会增大。

步骤8、

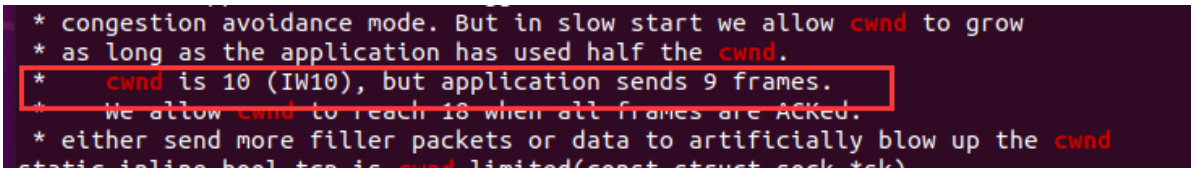
如果拥塞控制的相关过程不明显，请设计合适的方法再次测试。

步骤9、

完成其他可选的实验步骤。

查看linux系统的cwnd可以通过：

```
cat /usr/src/linux-headers-5.13.0-30-generic/include/net/tcp.h
```

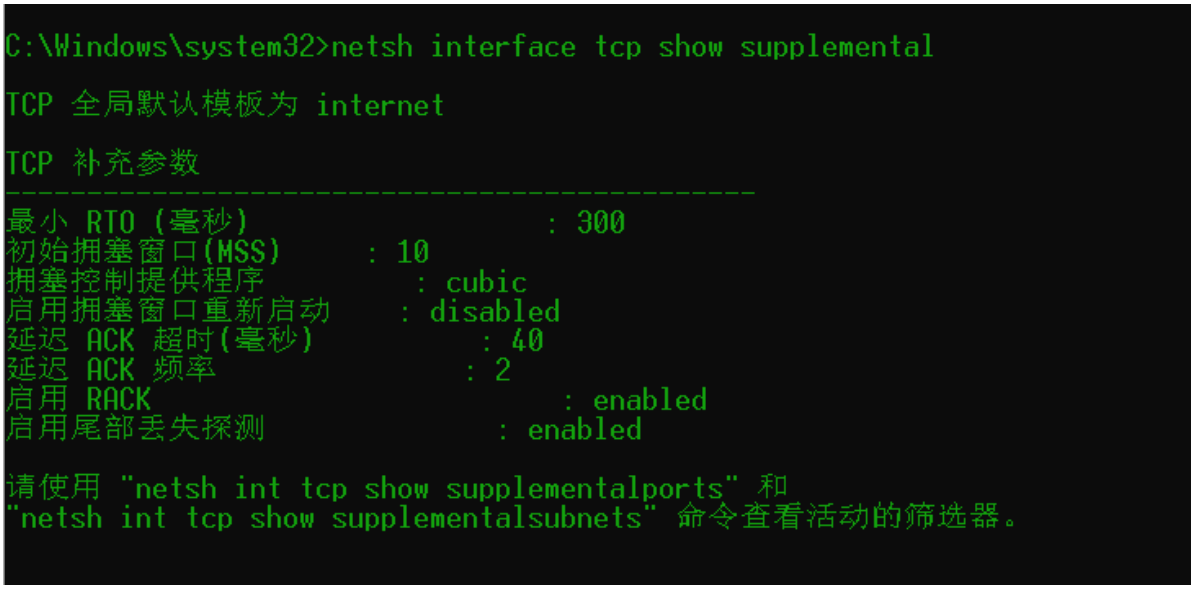


可以看到初始cwnd为10。

经过我们的上述实验也可以证明这个初始**cwnd**。

查看windows系统的初始cwnd的值：

```
netsh interface tcp show supplemental
```



六、 互动讨论主题

1）TCP的流量控制和拥塞控制有什么不同？

流量控制是接收方对发送方的控制，接收方控制报文中window字段的大小来控制接收方发送报文的数量，流量控制是为了防止接收方接收缓冲区不足而存在的机制。

拥塞控制是发送方主动进行的，拥塞控制是为了控制一个RTT内最多发送的数据包数量，是为了防止过多的数据包进入到网络之中。

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？

发送方，流量控制是数据传输过程中时刻进行的，通过接收方回复的报文，使得发送方维护一个接收窗口，随着这个接收窗口的变化来进行控制数据的发送。

3）讨论传输层与其上下相邻层的关系；

传输层是一个端到端的层，传输层的报文要被封装为IP报文进行传输，传输层的主要协议有TCP和UDP协议，TCP给予可靠的面向连接服务，UDP给予不可靠服务，二者都需要封装上源IP地址、目的IP地址进行传输，由网络层进行具体的IP地址转发。

传输层与应用层的关系：传输层为应用层提供服务，应用层可以选择TCP或UDP进行传输，应用层会选择具体的传输层端口号进行传输数据，例如http选择的80端口，应用层的数据也是需要封装为tcp数据进行端到端传输的。

4）讨论TCP协议在传输实时语音流方面的优缺点。

TCP传输数据准确，但是比较UDP来说速度较慢。