实验二 传输层 TCP 协议实验

刘星雨 无 08 2020010850

目录

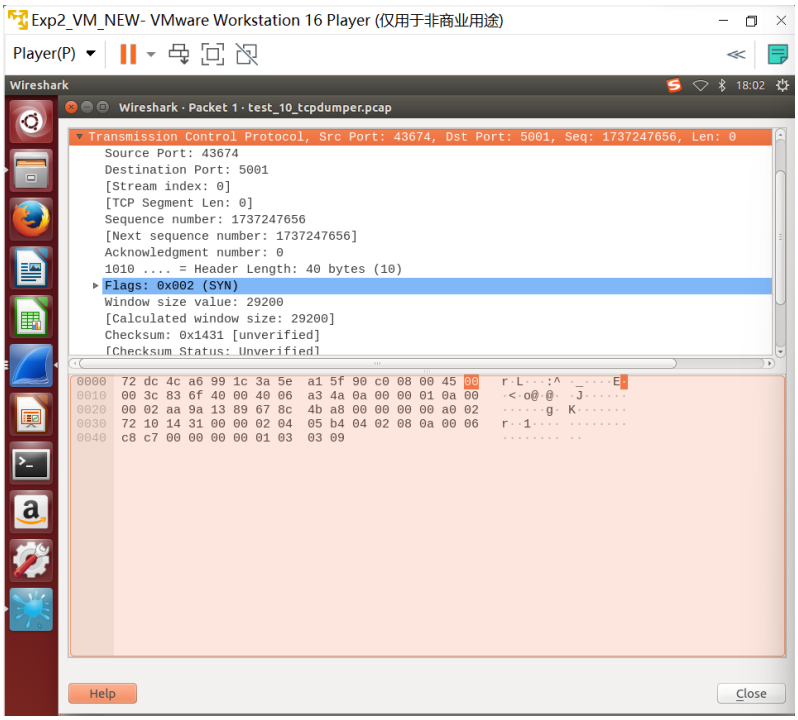
一、实验目的

1. 理解掌握 TCP 连接中三次握手建立连接和拆除连接的过程
2. 理解和掌握 TCP 可靠数据传输的实现原理和方法
3. 理解和掌握 TCP 流量控制的实现原理和方法
4. 理解和掌握 TCP 拥塞控制的实现原理和方法
5. 学习和掌握通过编程和抓包分析工具验证和分析协议运行过程

二、实验结果分析

2.1 TCP 连接管理，建立时的三次握手

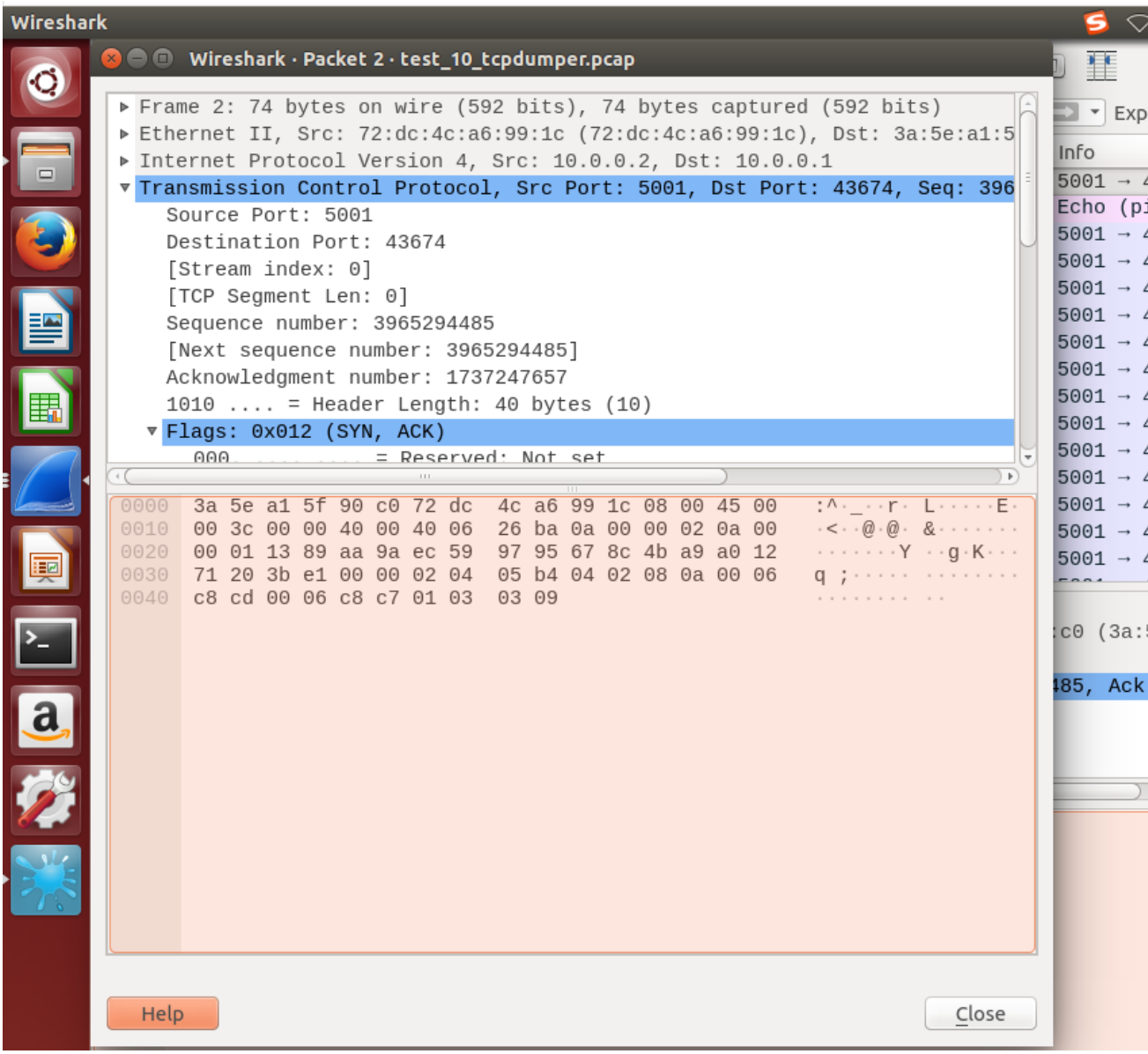
1. H1 发送的报文段



源端口号:43674	目的端口号：5001
序号	Seq number ： 1737247656

确认号	Acknowledgement number ： 0						
首部长度	保留位用	URG: Not set	ACK: Not set	PSH: Not set	RST: Not set	SYN: Set 1	FIN: Not set

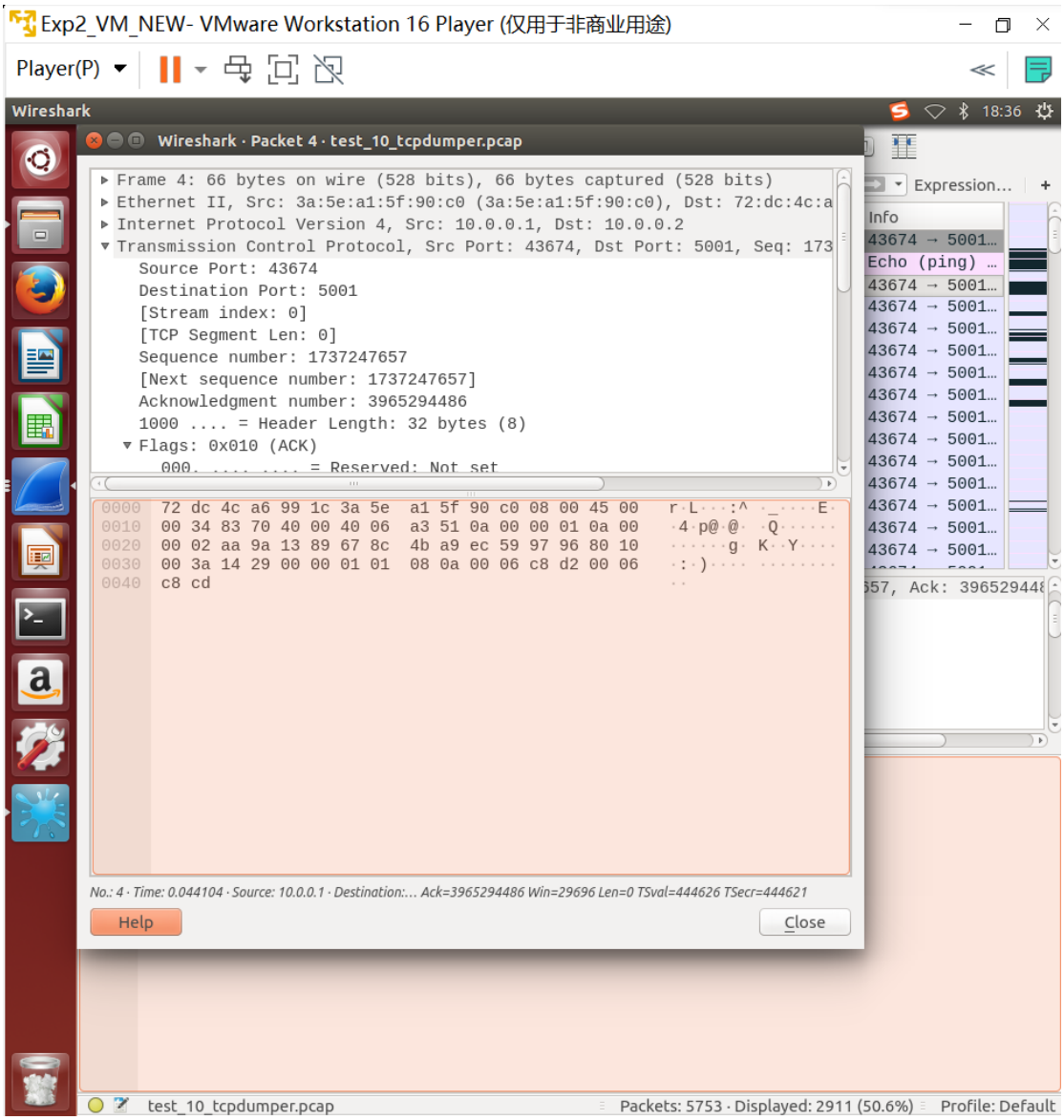
2. H2 发送的报文段



源端口号:5001	目的端口号：43674
序号	Seq number ： 3965294485

确认号	Acknowledgement number ： 1737247657						
首部长度	保留位用	URG: Not set	ACK: Set 1	PSH: Not set	RST: Not set	SYN: Set 1	FIN: Not set

3. H3 发送的报文

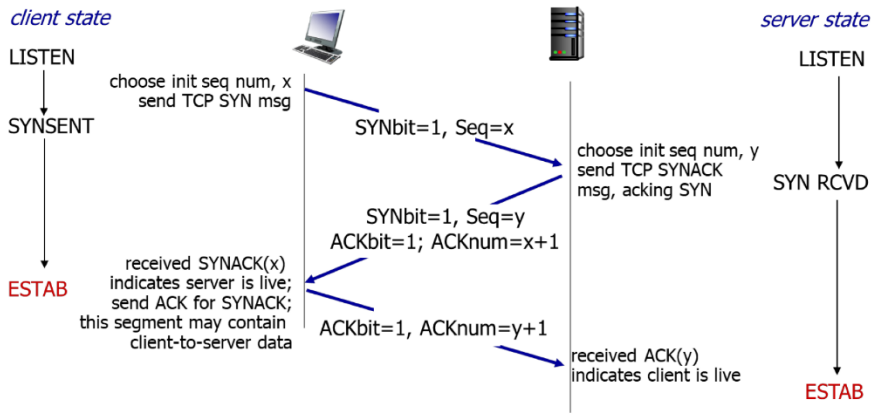


源端口号:43674	目的端口号： 5001
序号	Seq number ： 1737247657

确认号	Acknowledgement number ： 3965294486						
首部长度	保留位用	URG: Not set	ACK: Set 1	PSH: Not set	RST: Not set	SYN: Not set	FIN: Not set

可以看出来显然满足下图所示的 TCP 连接时三次握手的参数性质:

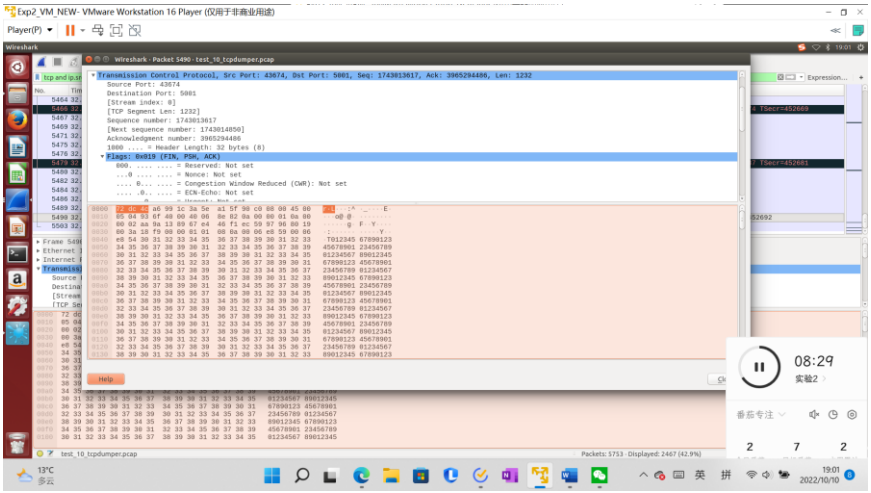
- **TCP连接建立**
  - 客户发送的第一个字段是SYN字段，用于序号同步。
  - 服务器发送第二个SYN +ACK字段，用于反向同步和确认。
  - 客户发送第三个字段，确认收到第二个字段。



TCP三次握手

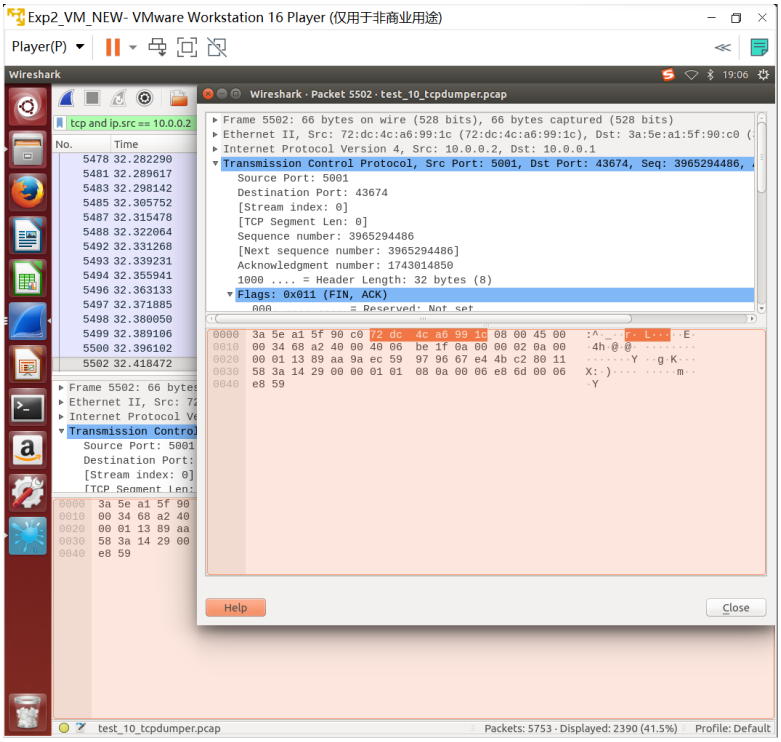
2.2 TCP 连接终止过程

1. H1 发送的报文



源端口号:43674				目的端口号：5001			
序号	Seq number ： 1743014850						
确认号	Acknowledgement number ： 3965294486						
首部长度	保留位用	URG: Not set	ACK: Set 1	PSH: Set 1	RST: Not set	SYN: Not set	FIN: Set 1

2. H2 发送的报文：



源端口号:5001				目的端口号：43674			
序号	Seq number ： 3965294486						
确认号	Acknowledgement number ： 1743014850						
首部长度	保留位用	URG: Not set	ACK: Set 1	PSH: Not set	RST: Not set	SYN: Not set	FIN: Set 1

3. H1 发送的报文

源端口号: 43674				目的端口号：5001			
序号	Seq number ： 1743014850						
确认号	Acknowledgement number ： 3965294487						
首部长度	保留位用	URG: Not set	ACK: Set 1	PSH: Not set	RST: Not set	SYN: Not set	FIN: Not set

2.3 TCP 可靠传输

1. h2 发送给 h1 的重复 ACK 包：

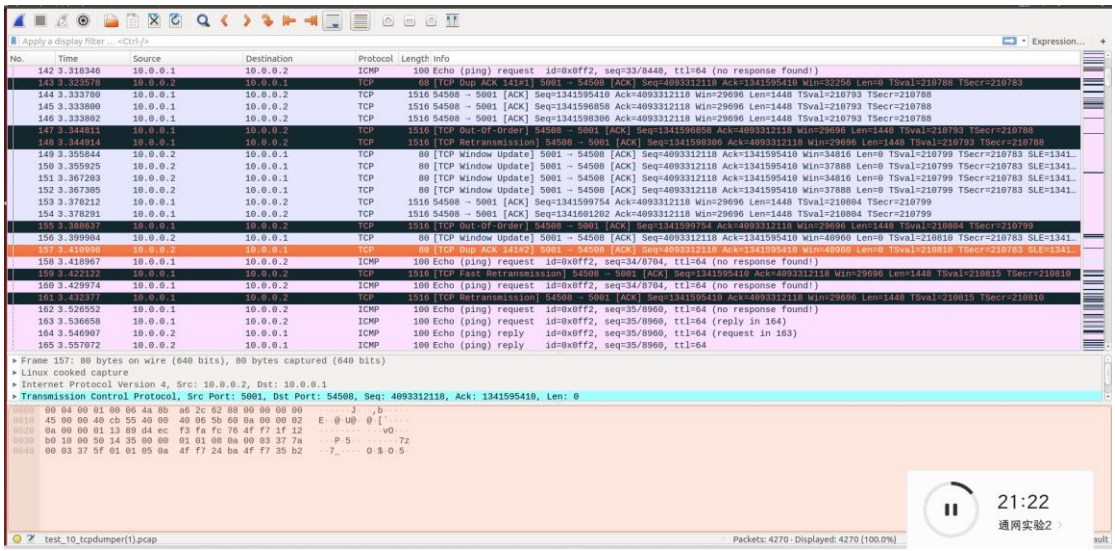
H2 给 h1 发送重复的 ACK 的**原因**是连续发送的数据包中有一个或多个缺失，并且在超时间隔 RTO 内发送端收到 3 次重复的 ACK，这种情况出现概率较低。在助教的提示下修改代码后，现在捕捉到了相关的重复 ACK 包【TCP Dup ACK】

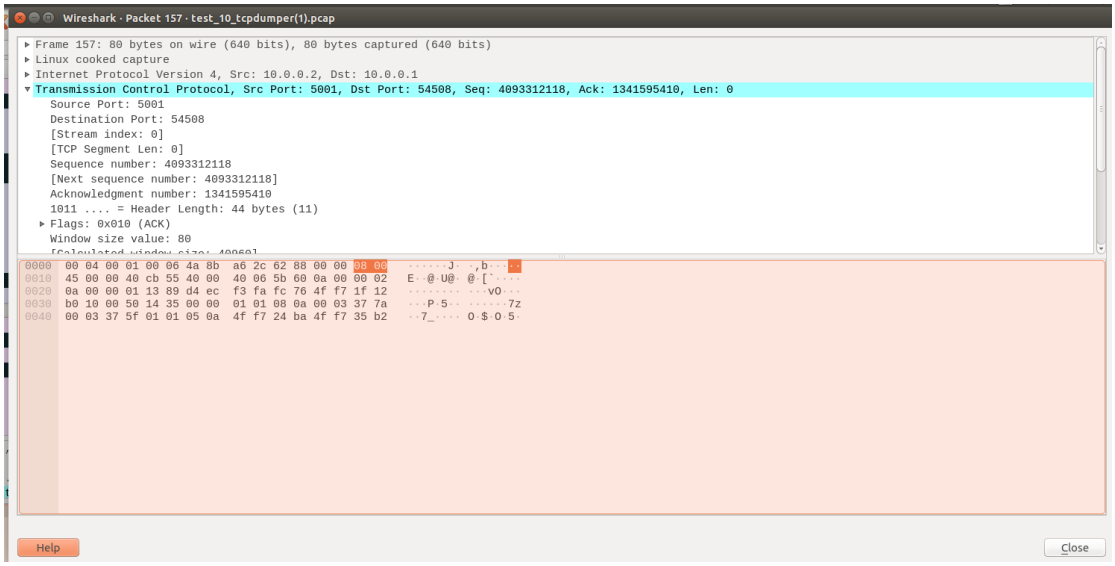
数据包的信息如下图：

源端口：5001　　目的端口：54508

Seq：409332118　　ack：1341595410

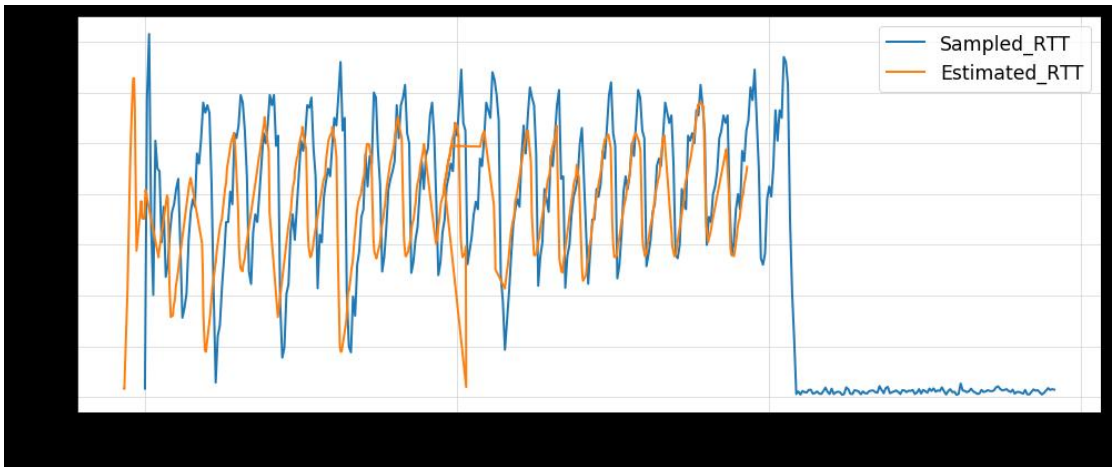
Acknowledgment：set





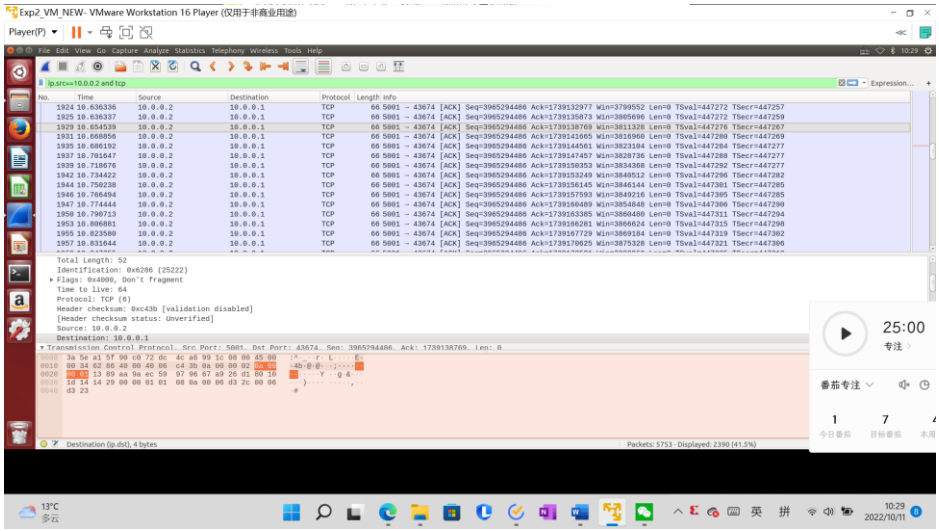
2. 分析 ping 采样的 RTT 和基于 tcpprobe 估计的 RTT 的差别

结果如下图，可以看到，估计的 RTT 平均比样本峰值更小而且在时间上有所提前，结束得也更早。峰值更小是因为 estimatedRTT 是由 srtt 反推得到的，不计算重传的包，而 sampleRTT 是根据 ping 获得的，更加准确一些。时间上的提前是因为 estimatedRTT 的全职由指数衰减的特性。



2.4 TCP 流量控制

记录一段连续时间内 h2 的接收窗口变化并分析变化原因



时间	10.654539	10.668856	10.686192	10.701647	10.718676	10.734422
接收窗口	3811328	3816960	3823104	3828736	3834368	3840512

10.750238	10.766494	10.774444	10.790713	10.806881	10.823580	10.831644
3846144	3849216	3854848	3860480	3866624	3869184	3875328

10.847355	10.863098	10.879985	10.895189	10.913426
3880960	3886592	3892224	3904000	3909632

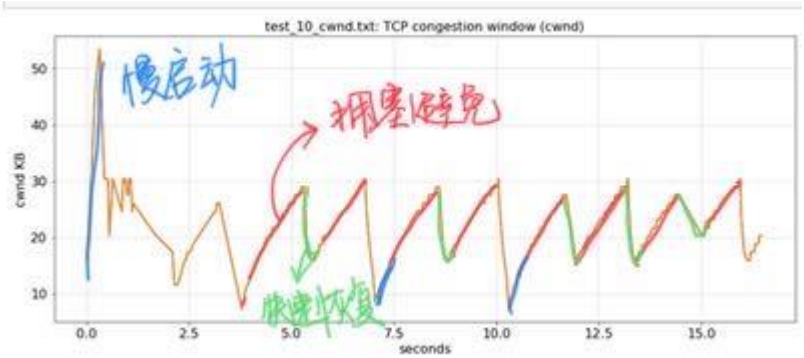
可以看到接收窗口一直增长而没有减小，是因为接收端操作系统会根据上层应用对接收包的处理速率调节，iperf 并不对接收到的包进行

复杂处理，瓶颈较小，因此接收窗口会不断增大。

2.5 TCP 拥塞控制

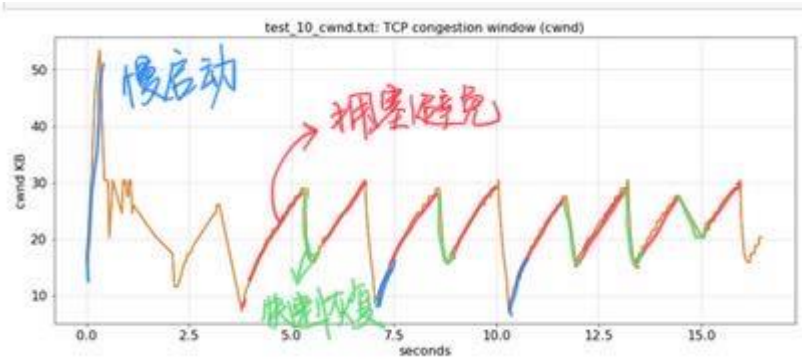
1. 拥塞窗口变化曲线

Queue_size 为 10 时：

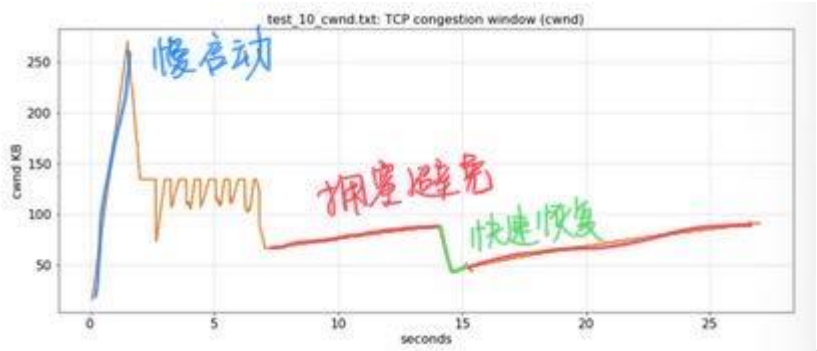


从较低的起点开始以指数形式增长的曲线是慢启动，例如图中蓝色部分；收到 ack 之前以线性增长的是拥塞避免，如图中红色部分；降低窗口大小到原来一半之后增大的是快速恢复，如图中绿色部分。

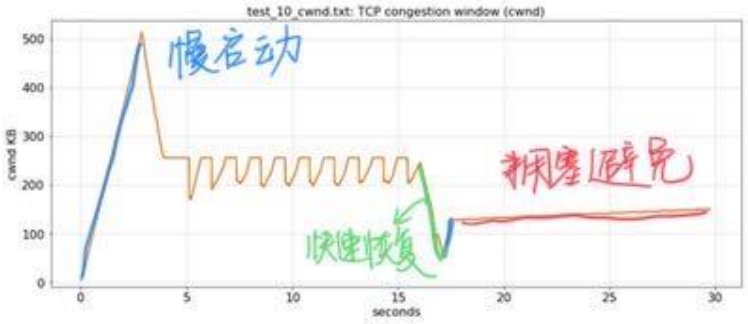
2. 路由器缓存大小为 10、50、100 时的拥塞窗口变化曲线



queue_size==10



Queue_size=50



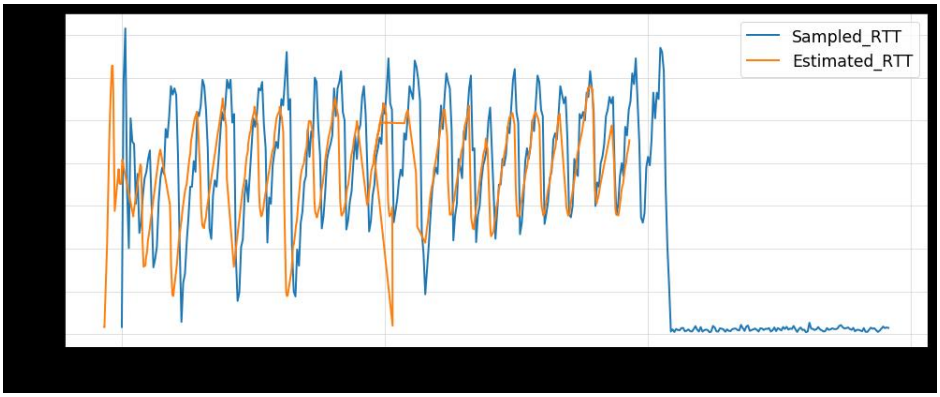
Queue_size =100

可以看到随着最大队列长度的增加，cwnd 的最大值也会增加，曲线看起来很平只是因为纵轴增加了。因为能够缓存的内容变多了，网络又足够的好，拥塞避免的阶段时间也增加了，也就是说不会一会儿就要丢包下降了。

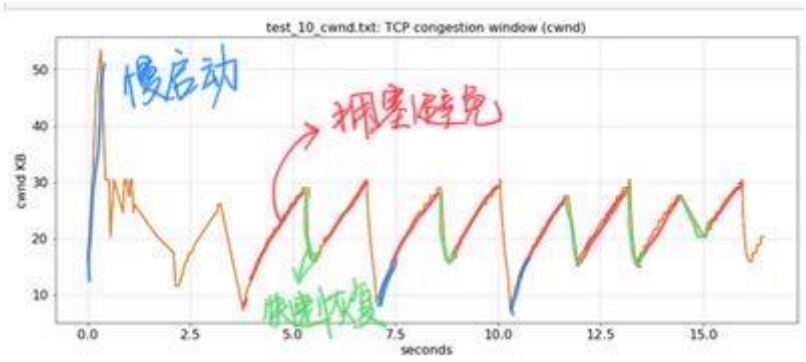
3．选做

【选做：缓存增大为 500/1000 时，当缓存达到某一数值，溢出产生拥塞。根据实验记录的拥塞窗口的和 RTT 变化。

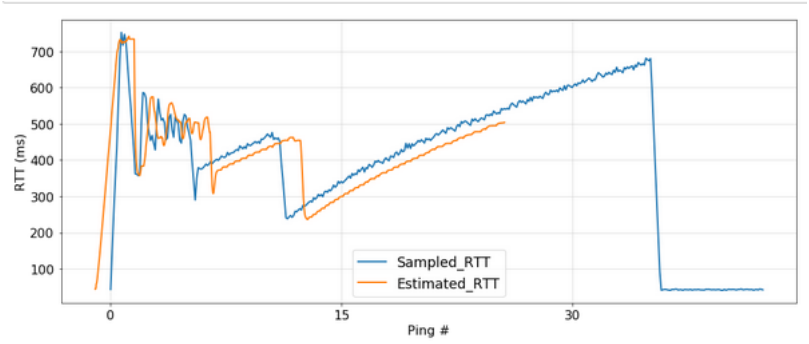
10-RTT:



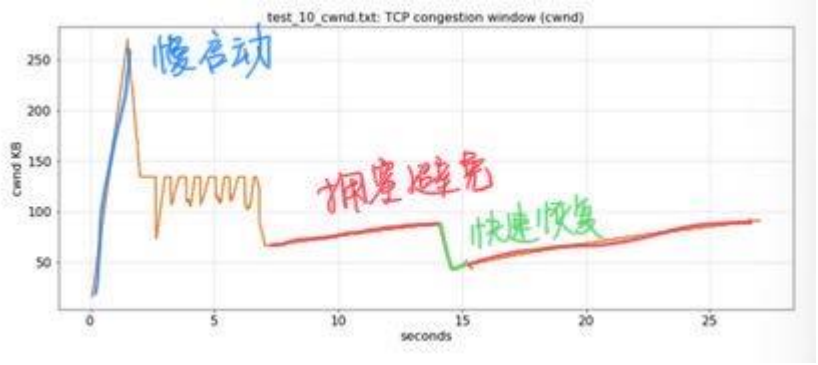
10-cnwd:



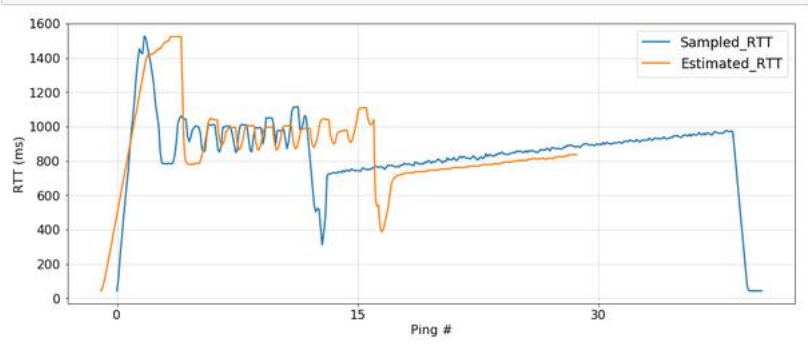
50-RTT:



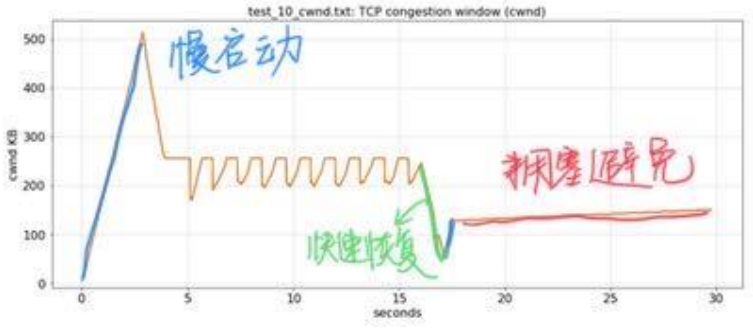
50-cnwd:



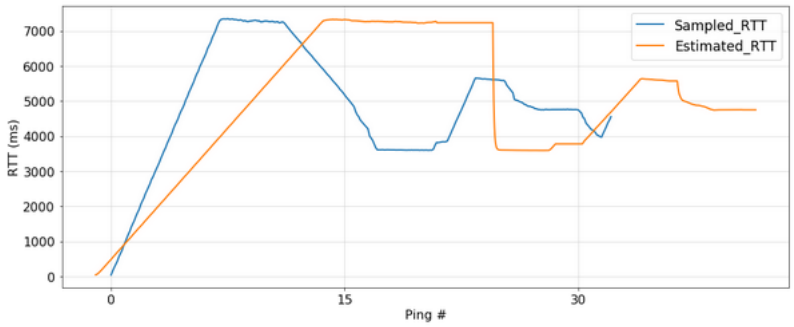
100-RTT



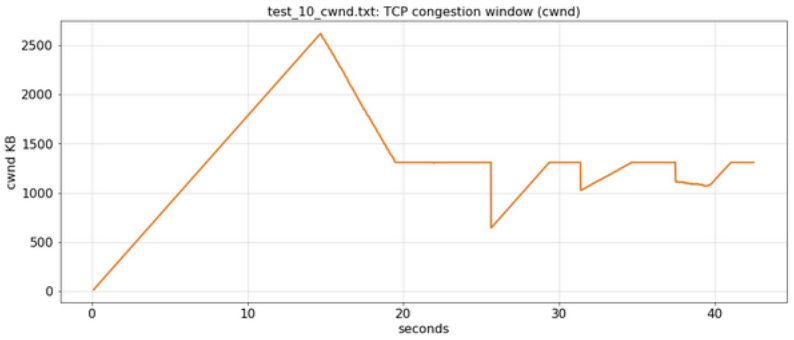
100-cnwd:



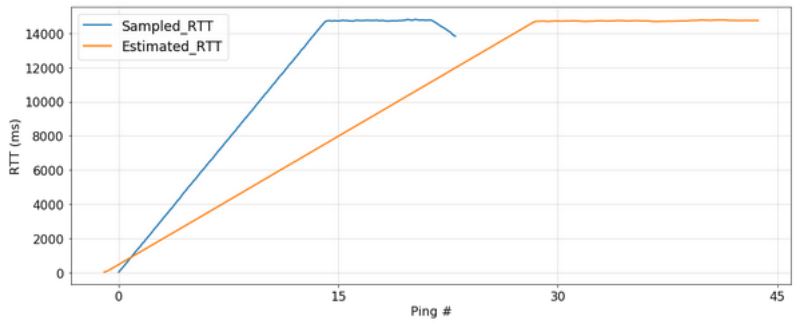
500-RTT-

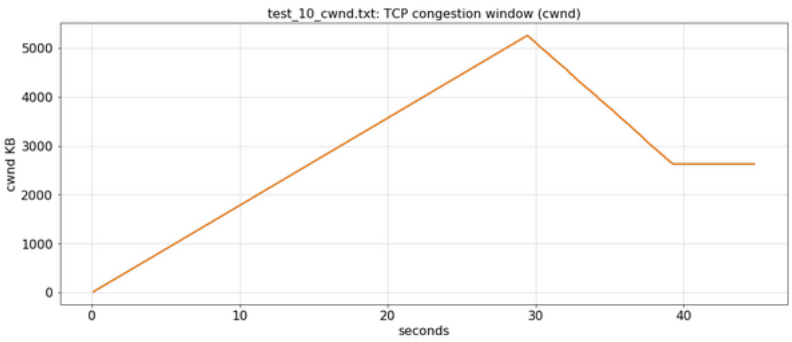


500-cnwd



1000-RTT-





1000-cnwd

缓存大小为 500/1000 数据较大，在最初慢启动的时候发出的包就比价多，网络中运转的包比较多，对网速或许有些影响，使得 RTT 增加；此外服务端需要处理的任务也会增加，花费的时间就增加了。

【多少给点写选做题的辛苦分吧】

三、思考题

（1）基于 wireshark 和抓取到的数据包，分析 h1 的接收窗口变化情况，解释产生这样现象的原因。

No.	Time	Source	Destination	Protocol	Length	Info
428	1.789301	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737573481 Ack=3965294486 Win=29696 Len=1448 TSval=445063 TSecr=445058
431	1.799312	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737574929 Ack=3965294486 Win=29696 Len=1448 TSval=445065 TSecr=445060
433	1.807231	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737576377 Ack=3965294486 Win=29696 Len=1448 TSval=445067 TSecr=445062
434	1.816280	10.0.0.1	10.0.0.2	TCP	2962	43674 → 5001 [ACK] Seq=1737577825 Ack=3965294486 Win=29696 Len=2896 TSval=445069 TSecr=445064
436	1.832033	10.0.0.1	10.0.0.2	TCP	2962	43674 → 5001 [ACK] Seq=1737580721 Ack=3965294486 Win=29696 Len=2896 TSval=445073 TSecr=445068
438	1.848309	10.0.0.1	10.0.0.2	TCP	2962	43674 → 5001 [ACK] Seq=1737583617 Ack=3965294486 Win=29696 Len=2896 TSval=445078 TSecr=445072
441	1.863207	10.0.0.1	10.0.0.2	TCP	2962	43674 → 5001 [ACK] Seq=1737586513 Ack=3965294486 Win=29696 Len=2896 TSval=445081 TSecr=445076
443	1.872511	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737589409 Ack=3965294486 Win=29696 Len=1448 TSval=445084 TSecr=445078
445	1.879956	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737590857 Ack=3965294486 Win=29696 Len=1448 TSval=445085 TSecr=445080
447	1.887679	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737592305 Ack=3965294486 Win=29696 Len=1448 TSval=445087 TSecr=445082
450	1.896159	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737593753 Ack=3965294486 Win=29696 Len=1448 TSval=445090 TSecr=445084
453	1.903709	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737595201 Ack=3965294486 Win=29696 Len=1448 TSval=445091 TSecr=445086
455	1.912220	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737596649 Ack=3965294486 Win=29696 Len=1448 TSval=445094 TSecr=445088
457	1.920780	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [ACK] Seq=1737598097 Ack=3965294486 Win=29696 Len=1448 TSval=445096 TSecr=445090
458	1.929853	10.0.0.1	10.0.0.2	TCP	2962	43674 → 5001 [ACK] Seq=1737599545 Ack=3965294486 Win=29696 Len=2896 TSval=445098 TSecr=445092
460	1.944259	10.0.0.1	10.0.0.2	TCP	1514	43674 → 5001 [PSH, ACK] Seq=1737602441 Ack=3965294486 Win=29696 Len=1448 TSval=445102 TSecr=445096

H1 的接收窗口大小不变，h2 传给 h1 的只有确认的消息，并不附带别的其他信息。

（2）本实验利用 wireshark 分析抓包的过程中，除了本实验重点分析的 TCP 协议数据包，还存在哪些其他的数据包?

还存在 ICMP 的数据包。它是 TCP/IP 协议簇的一个子协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。是一个网络层的协议。