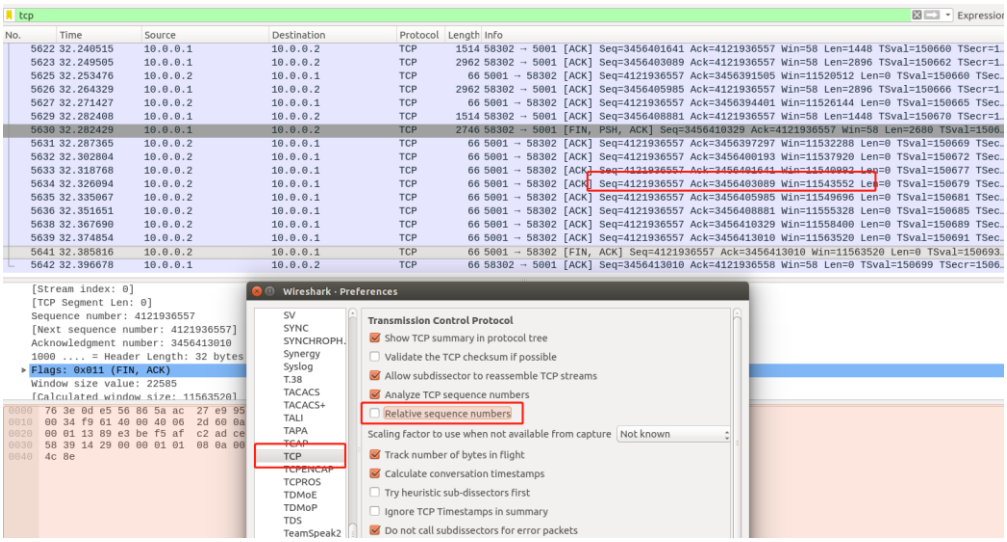
实验二 传输层 TCP 协议实验 答疑和补充

（1） Wireshark 中显示的 Seq 和 Ack 是 Relative value。

解答：

Relative value 指的是相对第一个数据包的值。

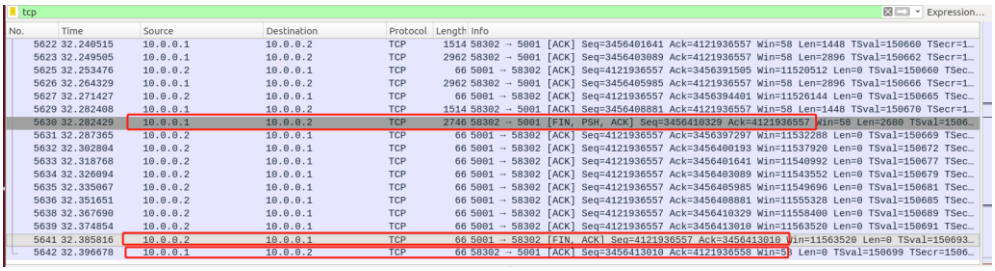
选择 Wireshark 工具栏“Edit-Preferences-Protocols-TCP”，取消勾选 Relative sequence numbers，即可显示绝对值，如下图所示。



（2） 在 TCP 连接终止过程中只观察到类似“三次握手”的数据包，即 FIN 包、ACK+FIN 包和 ACK 包，而不是课件中的四次数据包：FIN 包、ACK 包、FIN 包和 ACK 包。

解答：

实验中，服务器端（10.0.0.2）发送的 ACK 包和 FIN 包组成了 1 个数据包 ACK+FIN 包，因此只观察到 3 个数据包。如下图所示。



实验表格中对应的 4 个表格只需要填 3 个即可，ACK+FIN 包填写一个表格。

（3） 在 TCP 流量控制中观察到接收窗口大小一直增长，而没有减小。

解答：

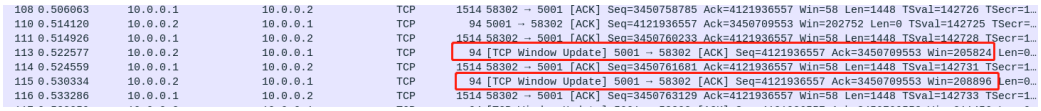
接收窗口大小是由接收端操作系统根据上层应用对接收包的处理速率调节的。iperf 并不对接收到的包进行复杂处理，瓶颈较小，因此窗口会不断增大。在实际应用中，接收窗口会根据应用处理能力确定，感兴趣的同学可以尝试使用 Wireshark 抓取通过互联网的其他 TCP 流量，对接收窗口的变化进行进一步研究。实验表格填写 calculated window size 或 window size

均可（二者相差一个 scaling 系数，系数在连接建立时协商确定）。可以参考以下链接中“吞吐受到接收 window size 限制”与“吞吐受到网络质量限制”两部分进一步理解：
https://mp.weixin.qq.com/s/45o0psfjNahhk7O9O0WJ-A

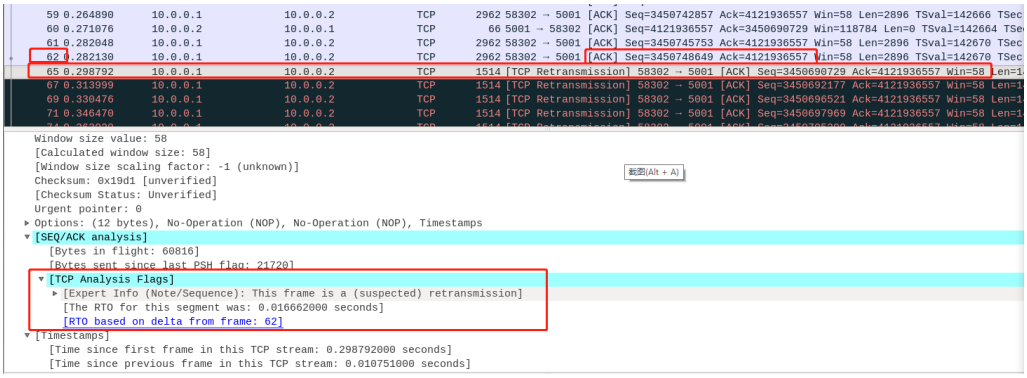
（4） 实验中在 Wireshark 中观察到的[TCP Window Update]和[TCP Retransmission]的含义。

解答：

[TCP Window Update]对应的是 TCP 窗口更新，当接收方的接收窗口发生突变时，接收方通过[TCP Window Update]消息告知对方当前的接收窗口大小，如下图所示 Win 字段在不断更新，不是 ACK 数据包，因此不是快速重传的重复 ACK 数据包。



[TCP Retransmission]对应的是 TCP 重传。Wireshark 判断一个数据包时[TCP Retransmission]的机制是：当抓到 2 次同一包数据时且没有抓到第一个数据包的反馈 ACK，Wireshark 判定重传有效，标记为[TCP Retransmission]。具体地，基于上述判定机制，在实验中，当客户端 10.0.0.1 初次发送一个数据包，并启动定时器，在预设超时间隔 RTO（Retransmission TimeOut，重传超时时间）内并没有收到服务器端 10.0.0.2 的反馈 ACK，超时触发客户端重传。如下图，红线的 TCP 包为重传包，Wireshark 为该包添加了重传原因，是由于 RTO 超时导致，以及第一个数据包序号为 62，并给出了当前的 RTO 为 0.0166s。

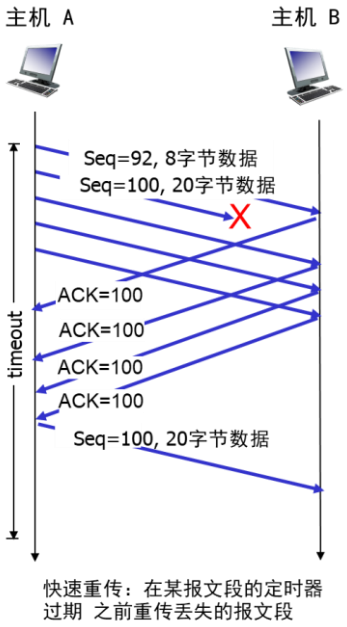


根据理论课讲授知识，由超时触发重传并不会检测到冗余的 ACK 包，与实验观测现象相符。

（5） 实验 5.4 第（1）步的为保证可靠数据传输， h2 发送给 h1 的重复 ACK 包如何找到？

解答：

根据理论课讲授知识，该重复 ACK 是触发快速重传机制才会产生的数据包，而快速重传机制的触发条件较复杂（出现概率较低），如下图所示，当连续发送的数据包中间有一个或多个数据包缺失，且在超时间隔 RTO 内发送端收到 3 次重复 ACK 才会触发。



相比而言，在本实验中，超时触发重传（Wireshark 软件会标识[TCP Retransmission]）大概率是由于数据包在路由器中等待排队（符合先进先出规则），队列满的时候，后到达的数据包被丢包，容易观察到。而快速重传数据包（Wireshark 软件会标识[TCP Fast Retransmission]）的出现概率较低，在 Mininet 仿真环境中，需要人工增加链路丢包率，才容易观察到中间数据包丢包产生的快速重传现象。

因此，我们需要对实验提供代码进行几处修改：

- 将原有 h0 和 s0 之间链路的丢包率置为 20（%）。

```
    # Add links with appropriate bandwidth, delay, and queue size parameters.
    # Set the router queue size using the queue_size argument
    # Set bandwidths/latencies using the bandwidths and minimum RTT given in the network diagram above
    self.addLink(h1, s0, bw=1000, delay='10ms', max_queue_size=queue_size, loss=0)
    self.addLink(h2, s0, bw=1.5, delay='10ms', max_queue_size=queue_size, loss=20)
```

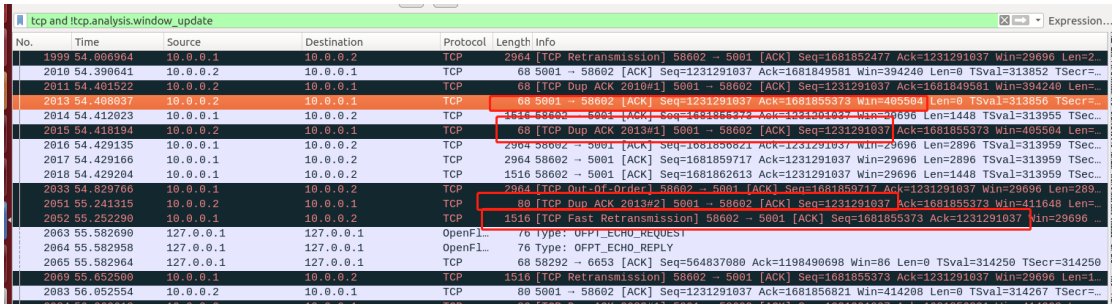
- 将原有 tcpdump 命令更改，使得能够监测网络中所有类型数据包。

```
# Start capturing packets
from subprocess import Popen
experiment_name = 'test_10'    # set experiment name
tcpdumper = Popen("tcpdump -i any -S -w ./{}_tcpdumper.pcap".format(experiment_name), shell=True)
```

- 将实验时间由 30s 适当增加，如改为 60s，以便能够捕捉到快速重传现象。

```
# Start the long lived TCP connections with start_iperf
experiment_time = 60
start_iperf(net, experiment_time)
```

基于上述改动，重新运行代码，将监测的数据包通过 Wireshark 显示，有较大概率可观察到对应的[TCP Fast Retransmission]数据包，并找到其关联的重复 ACK 包（Wireshark 用[TCP Dup ACK]标识），如下图所示。



（部分情况打开该数据包会提示“The capture file appears to be damaged or corrupt”，可直接点左上角“×”关闭提示查看快速重传数据包。）

关于 TCP 重传更多的扩展内容，感兴趣的同学可以了解 TCP 选项（option）的 SACK 等。

https://baike.baidu.com/item/sack/11068287?fr=aladdin

https://zhuanlan.zhihu.com/p/374385428