山东大学 计算机 学院

计算机网络 课程实验报告

学号：202000130143	姓名： 郑凯饶	班级： 2020 级 1 班
实验题目：TCP		
实验学时：2	实验日期： 2022-4-12	

实验目的：

了解 TCP 如何提供可靠的数据传输，TCP 的拥塞控制算法（慢启动、拥塞避免），接收方的流量控制机制以及电脑如何与服务器建立 TCP 连接。

硬件环境：

Dell Latitude 5411

Intel(R) Core(TM) i5-10400H CPU @ 2.60GHz(8GPUs),~2.6GHz

软件环境：

Windows 10 家庭中文版 64 位（10.0，版本 18363）

Wireshark-win64-3.6.2

实验步骤与内容：

1. 问题：

（1） 客户端的 IP 及端口号？

（2） 服务端的 IP 及端口号？

（3） 自己实验的情况？

（4） 用于初始化 TCP 连接的 TCP SYN 包的序列号？什么信息将其标识为 SYN 包？

（5） SYNACK 的序列号？Acknowledgement 字段的值？gaia.cs.umass.edu 如何确认该值？什么信息将其标识为 SYNACK 包？

（6） 包含 HTTP POST 命令的 TCP 段的序列号？

（7） 前 6 个段的序列号？*发送时间？* 是否接收到 ACK？分析每个段的不同，ACK 何时接收到，它们的 RTT 是多少？EstimatedRTT 在接收到每个 ACK 之后的值是多少?

（8） 前 6 个段的长度？

（9） 整个追踪过程中建议的最小接收缓存空间？其缺乏是否会限制发送者？

（10）是否有重发的段？

（11）在 ACK 包中接收方通常确认多少数据？如何识别接收方正在 ACK 一个接收到的段？

（12）TCP 连接的吞吐量（单位时间内的传输量）。解释。

（13）观察时间序列图，是否可以判断慢启动的起止时间，分析和理想 TCP 的不同。

（14）用自己机器实验完成（13）

2. 阐述基本方法

（1）主要参考课本 TCP 的相关章节。

（2）使用 WireShark 过滤器命令筛选相关报文:

tcp.analysis.ack_rtt： 选择与 RTT 相关的报文，之中包含 RTT 信息

ip.src == 192.168.1.102： 通过源 IP 进行过滤

ip.addr == 128.119.245.12 && tcp： 使用逻辑运算符

（3）使用 WireShark 统计工具绘图，可视化数据

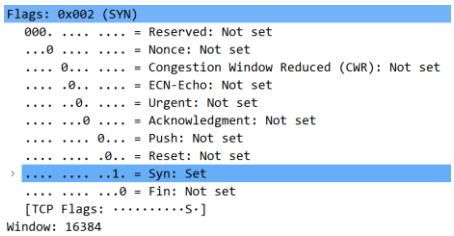
3. 实验结果展示与分析

（1） Source Address: 192.168.1.102 Source Port: 1161

（2） Destination Address: 128.119.245.12 Destination Port: 80

（3） Source Address: 172.25.161.130 Source Port: 56230

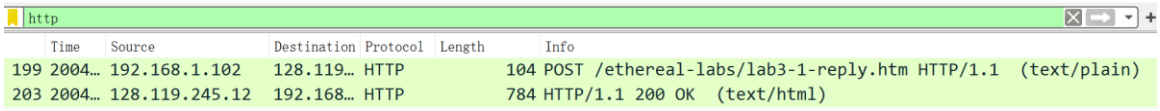
（4） Sequence Number (raw): 232129012，标识字段中的 1 bit 标识是否为 SYN 包



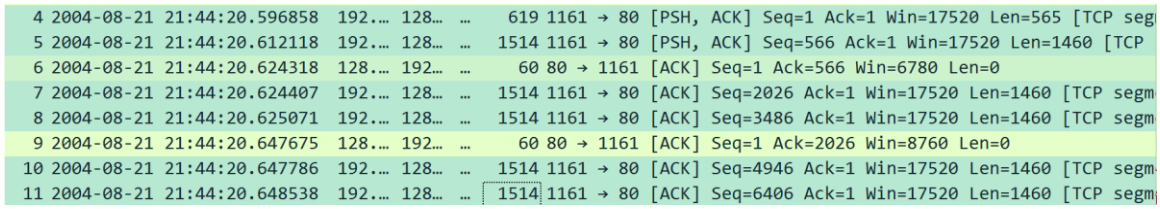
（5） Sequence Number (raw): 883061785 Acknowledgment number (raw): 232129013

可以发现 Acknowledgment number = SYN 中 Sequence Number + 1. SYNACK 同样由标识字段决定。

（6） Sequence Number (raw): 232293053

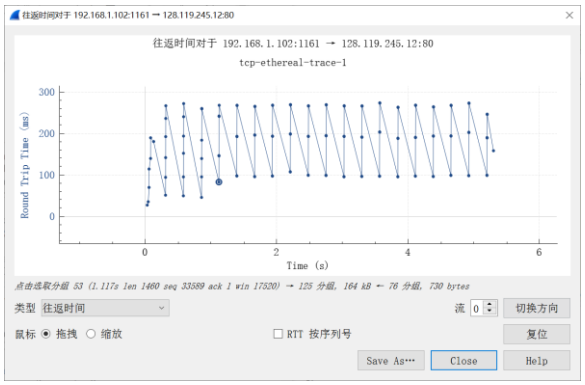


（7） 1, 566, 2026, 3486, 4946, 6406（根据字节流编号），ACK 报文随其后（TCP 是全双工通信，我的主机实时收到服务器的 ACK），而 TCP 中 ACK 的 Acknowledgment number 是其希望接收的下一字节的序号，因此对应 ACK number 为 566, 2026, 3486, 4946，6406，7866.



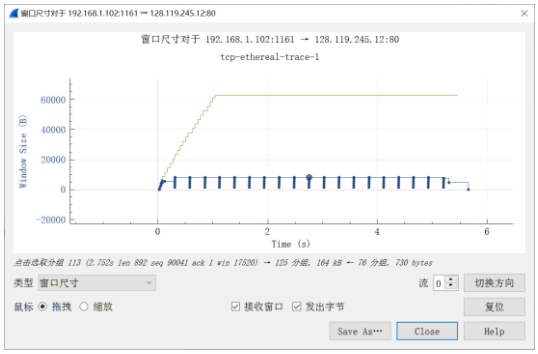
可以观察到前 6 个段的 RTT 分别为 0.02746s，0.03556s，0.07006s，0.11443s，0.13989s，0.18965s.

由公式 EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT 得 RTT 估计值分别为 0.02746s, 0.0284725s, 0.0336709s, 0.0437658s, 0.0557813s, 0.0725149s.

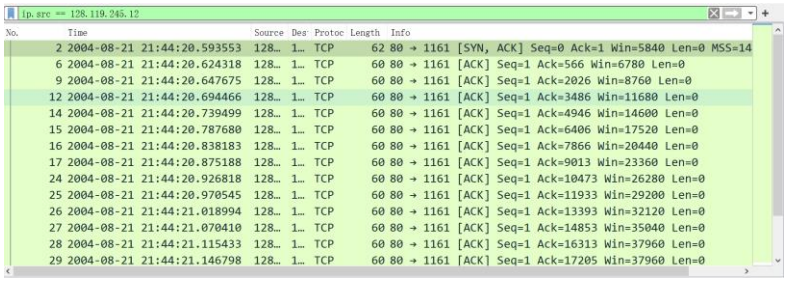


（8） 565，1460 * 5

（9） 5840（第一个 ACK 返回），观察到接收窗口远远大于发送窗口，因此不会触发相关流量控制机制。

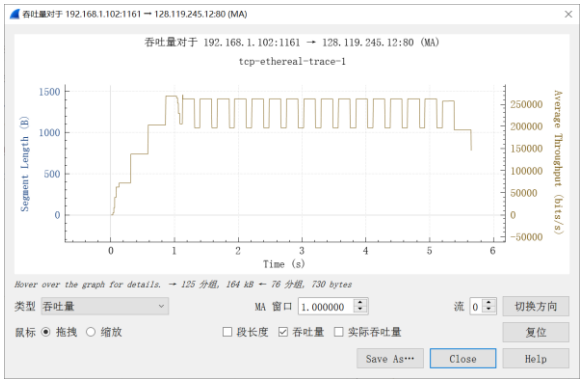


（10）作者提供的 trace 中没有重发。使用 ip.src 进行过滤发现没有重发的包。

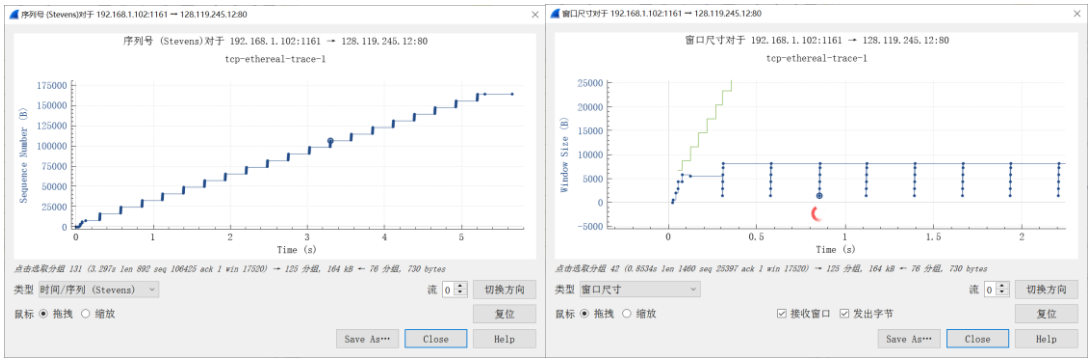


（11）对于每个发送段回复一个 ACK。

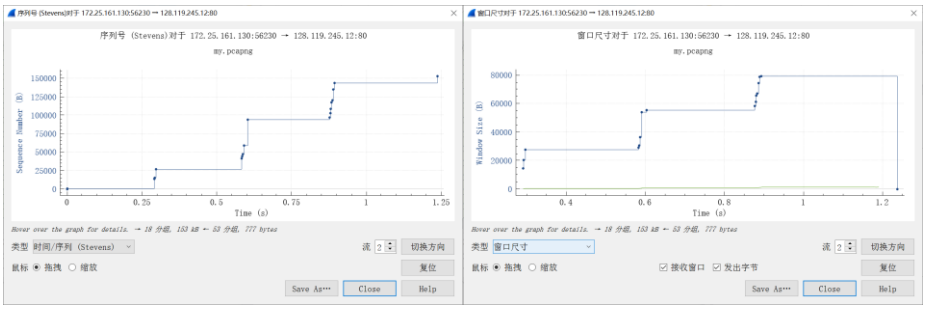
（12）吞吐量 = 每秒收到的比特数（观察右侧纵轴）



（13）通过段的发送连续程度或者直接通过发送窗口尺寸可以看出慢启动阶段（0s 至 0.3s 左右），和理想 TCP 相比发送速率似乎是线性增长而不是指数增长。



（14）自己实验则发现倍增情况，在 1s 左右的时间窗口不断倍增，似乎一直处于 slow start 阶段。



结论分析与体会：

这次实验我了解了 TCP 协议，大致学习了建立连接、流量控制及拥塞控制等等内容。但是由于实验样本有限我很难一窥其全貌，数据传输过程中并没有发生 timeout、duplicate ACK 等等引起连接管理的变化。而且实际的运输协议同教科书上有差异，或者说复杂许多。希望日后的学习能更加深入地学习 TCP。