学生选课序号	
实验项目序号	**4**

桂 林 电 子 科 技 大 学

计算机网络 实 验 报 告

实 验 名 称 实验四 TCP/UDP 协议分析

计算机与信息安全学院 学院

网络空间安全 专 业

姓名 白楚榆

学号 2200350101

实 验 日 期 2024 年 5 月 25 日

评语：

成绩：_____

指导教师签名：

一．实验目的

1. 加深理解 TCP 报文结构
2. 领会 TCP 协议通信机制
3. 通过跟踪 TCP 应用通信，能结合报文对整个通信过程进行分析。
4. 熟悉 UDP 协议报文结构
5. 领会 UDP 协议通信机制
6. 对比 TCP、UDP 协议主要特点

二．实验环境

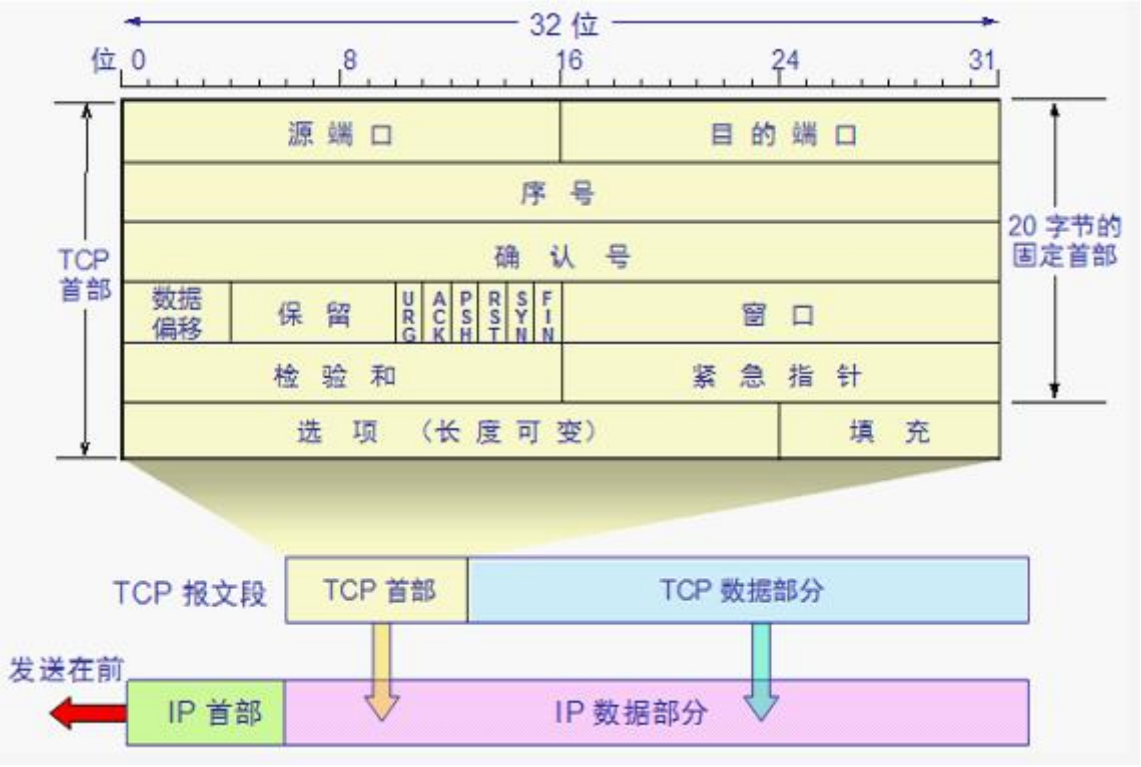
1、头歌基于 Linux 的虚拟机桌面系统

2、网络报文分析工具 wireshark

3、浏览器 firefox

三．相关原理或知识点

TCP 首部格式



源端口(16 位)：通信发送方使用的端口号

目标端口(16 位)：通信接收方使用的端口号

序列号(32 位)：用来确保数据可靠传输的唯一值

确认号(32 位)：接收方在响应时发送的数值

数据偏移(4 位)：标志数据包开始的位置，TCP 头部的长度

SYN：(同步)发起连接的数据包：同步 SYN=1 表示这是一个连接请求或连接接受报文。

ACK：(确认)确认收到的数据包：只有当 ACK=1 时，确认号字段才有效；当 ACK=0 时，确认号无效。

RST：(重置)之前尝试的连接被关闭，(信号差，信号拥挤)：当 RST=1 时，表明 TCP 连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。

FIN：(结束)连接成功，传输完毕之后，连接正在断开：用来释放一个连接，FIN=1 表明此报文段的发送端的数据已发送完毕，并要求释放运输连接。

PSH：(推送)数据包直接发送给应用，而不是缓存起来：接收 TCP 收到 PSH=1 的报文段，就尽快地交付接收应用进程，而不再等到整个缓存都填满了后再向上交付。

URG：(紧急)数据包中承载的内容应该立即由 TCP 协议栈立即进行处理：当 URG=1 时，表明紧急指针字段有效。它告诉系统此报文段中有紧急数据，应尽快传送(相当于高优先级的数据)。

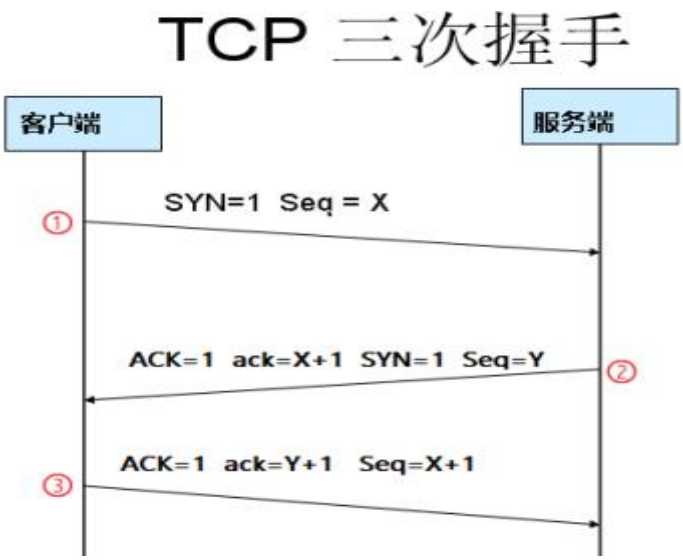
窗口大小(16 位)：匹配缓存区的大小

校验和(16 位)：确认 TCP 数据段中的内容是否发送了变化

紧急指针(16 位)：明确显示数据之前的 16 进制序列号

TCP 连接建立过程

TCP 连接建立过程，双方会产生三个报文，俗称三次握手。



①第一次握手：客户端将标志位 SYN 置为 1 ，随机产生一个值 SEQ = X，并将该数据包发送给服务器，等待服务器确认；

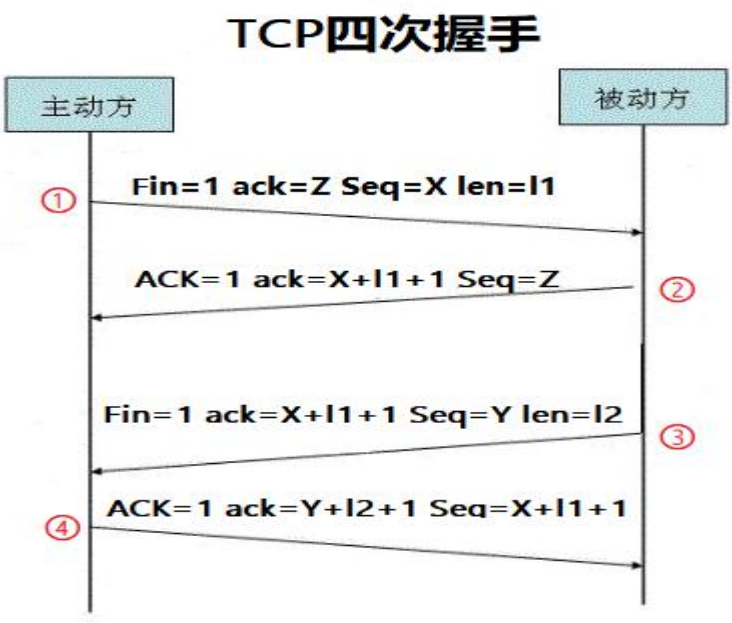
②第二次握手：服务器收到数据包后由标志位 SYN = 1，知道客户端请求建立连接，服务器将标志位 SYN 和 ACK 都置为 1 ，ACK = X + 1，随机产生一个值 SEQ = Y，并将该数据包发送给客户端以确认连接请求；

③第三次握手：客户端收到确认后，检查 ACK 是否为 X + 1，ACK 是否为 1 ，如果正确则将标志位 ACK 置为 1 ，ACK = Y + 1，并将该数据包发送给服务器，服务器检查 ACK 是否为 1 ，如果正确则连接建立成功。

注：图中 Seq 代表 tcp 报文的序号，ack 代表 tcp 报文的确认号。当 ACK 为 1 时，ack 值有效。

3、TCP 连接关闭过程

TCP 连接关闭过程，双方通常会产生四个报文，俗称四次握手。有时 TCP 连接关闭时，只能得到三个报文，这是因为第二次、第三次握手合并到一个报文里了。



①第一次握手：主动关闭方发送一个 FIN ，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：我已经不会再给你发数据了(当然，在 FIN 包之前发送出去的数据，如果没有收到对应的 ACK 确认报文，主动关闭方依然会重发这些数据)，但此时主动关闭方还可以接受数据。

②第二次握手：被动关闭方收到 FIN 包后，发送一个 ACK 给对方，确认序号为:收到报文序号 Seq +收到报文所携

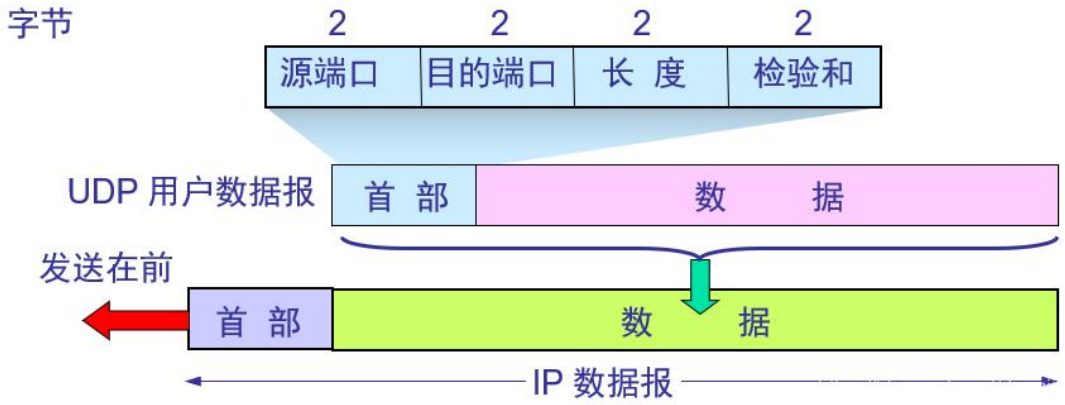
带数据长度 len+ 1 。

上一个报文可能“捎带”了主动关闭方发送的最后一块数据,其长度用字段 len 来表示。

③第三次握手：被动关闭方发送一个 FIN ，用来关闭被动关闭方到主动关闭方的数据传送，也就是告诉主动关闭方，我的数据也发送完了，不会再给你发数据了。

④第四次握手：主动关闭方收到 FIN 后，发送一个 ACK 给被动关闭方，确认序号为：收到报文的序号 Seq + 收到报文所携带数据长度 len+ 1 ，至此，完成四次挥手。

1、UDP 的首部格式



UDP 首部有 8 个字节，由 4 个字段构成，每个字段都是两个字节：

源端口： 源端口号，需要对方回信时选用，不需要时全部置 0 ；

目的端口：目的端口号，在终点交付报文的时候需要用到；

长度：UDP 的数据报的长度（包括首部和数据）其最小值为 8（只有首部）。字段记录了该 UDP 数据包的总长度（以字节为单位），包括 8 字节的 UDP 头和其后的数据部分。最小值是 8（报文头的长度），最大值为 65535 字节；

校验和：检测 UDP 数据报在传输中是否有错，有错则丢弃。它的值是通过计算 UDP 数据报及一个伪包头而得到的。校验和的计算方法与通用的一样，都是累加求和。

注：端口是用来指明数据的来源（应用程序）以及数据发往的目的地（同样是应用程序）。字段包含了 16 比特的 UDP 协议端口号，它使得多个应用程序可以多路复用同一个传输层协议及 UDP 协议，仅通过端口号来区分不同的应用程序。

四．实验任务

1、结合具体的报文数据，分析三次握手的过程。

2、结合具体的报文数据，分析四次握手的过程。

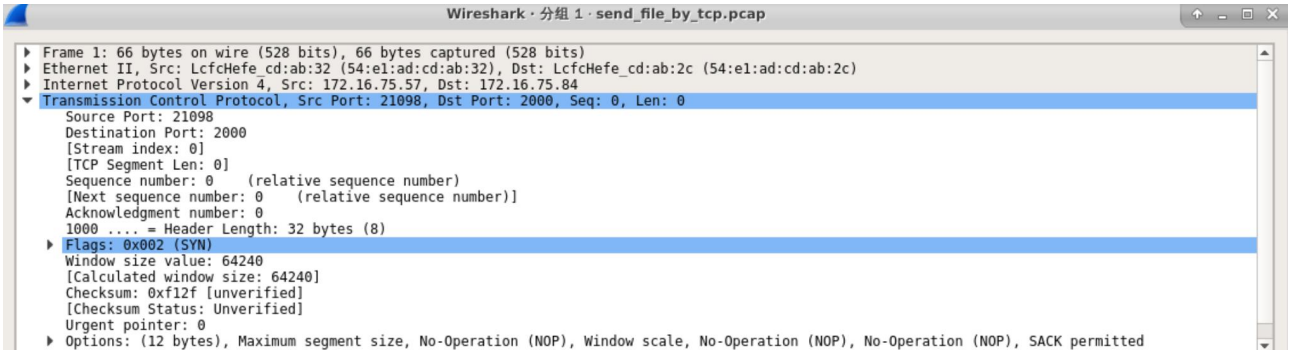
3、结合具体的报文数据，分析各字段的含义。

4、完成相关数据的计算统计。

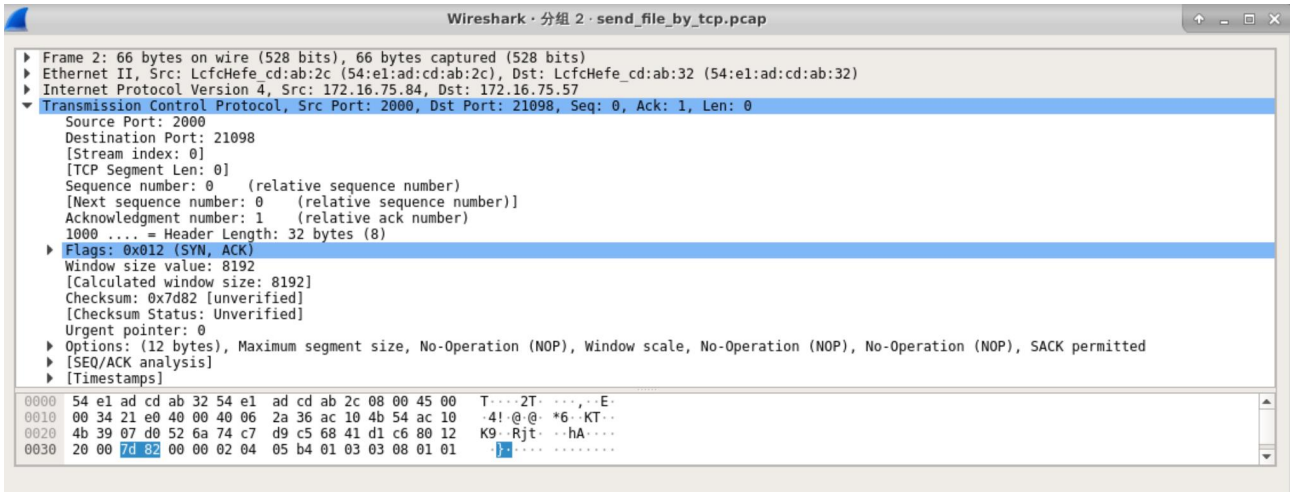
5、在实验报告内，计算双方通信所用的时间、计算 A 方发送的所有的帧的长度之和、计算 A->B 连接的通信吞吐率是多少。

五．TCP 协议分析

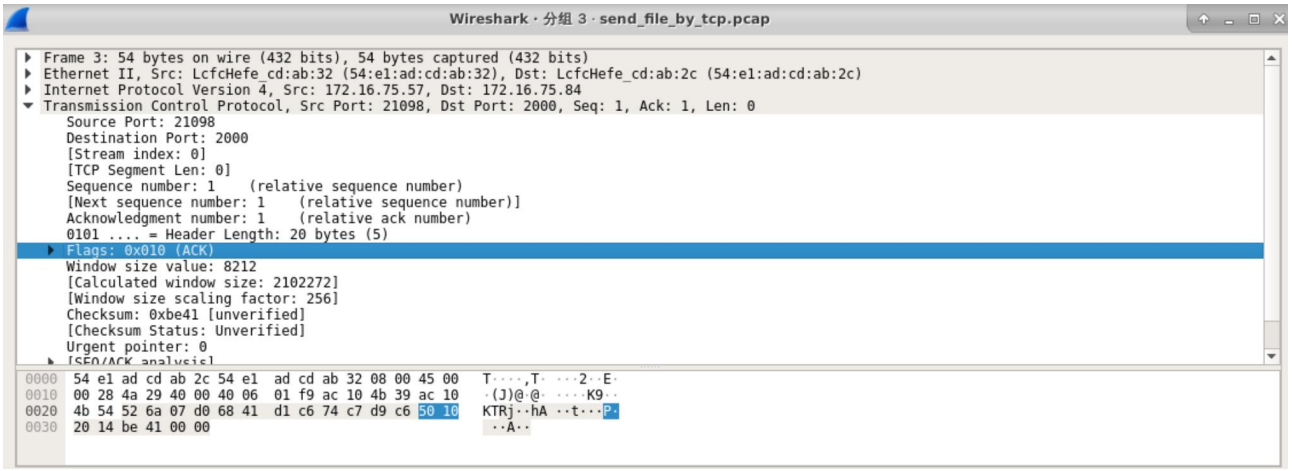
1、TCP 连接建立过程分析



第一次握手：客户端将标志位 SYN 置为 1 ，随机产生一个值 SEQ = X = 0，并将该数据包发送给服务器，等待服务器确认；

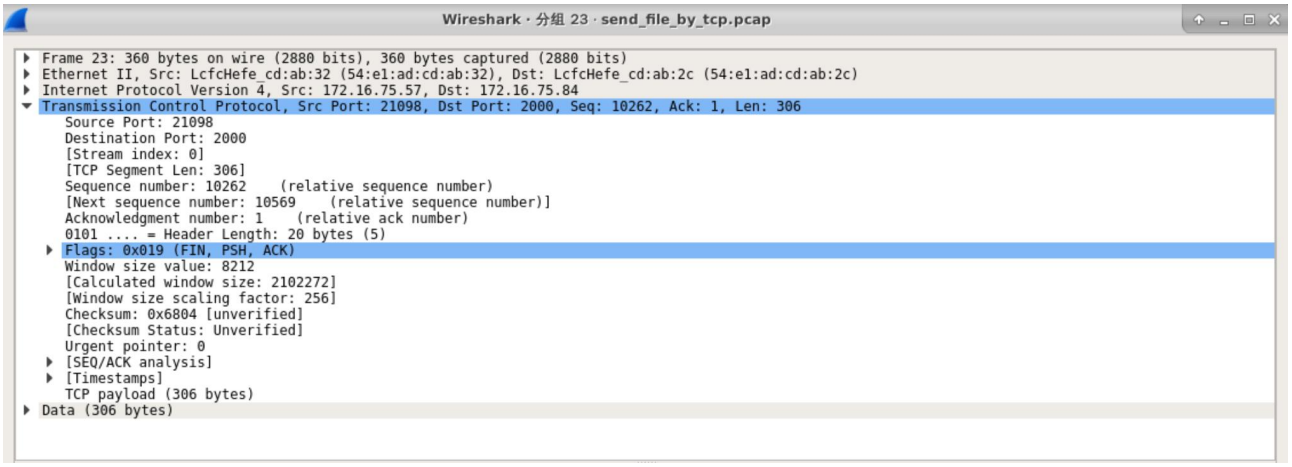


第二次握手：服务器收到数据包后由标志位 SYN = 1，直到客户端请求建立连接，服务器将标志位 SYN 和 ACK 都置为 1 ，ACK = X + 1 = 1，随机产生一个值 SEQ = Y = 0，并将该数据包发送给客户端以确认连接请求；

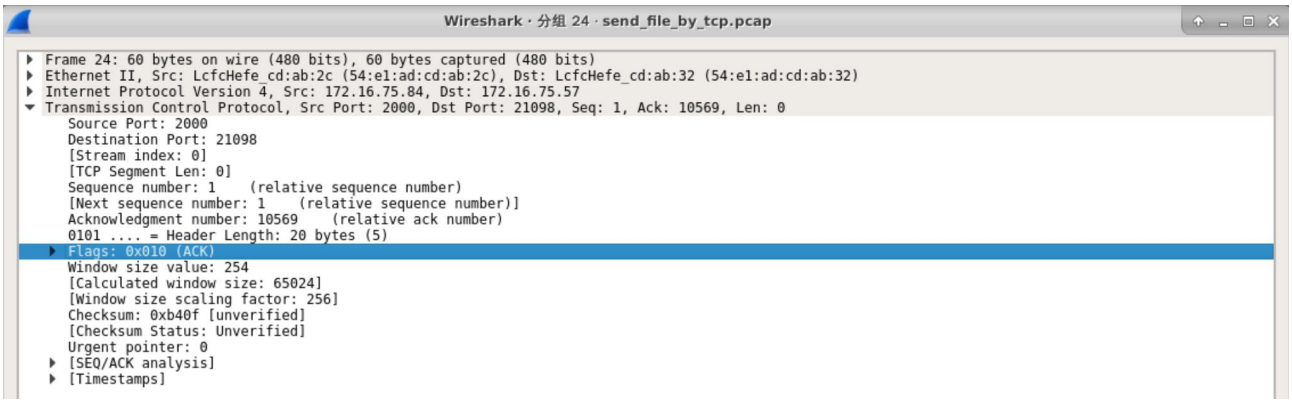


第三次握手：客户端收到确认后，检查 ACK 是否为 X + 1= 1，如果正确则将标志位 ACK 置为 1 ，SEQ = 1,ACK = Y + 1 = 1，并将该数据包发送给服务器，服务器检查 ACK 是否为 1 ，如果正确则连接建立成功,后续开始传输数据。

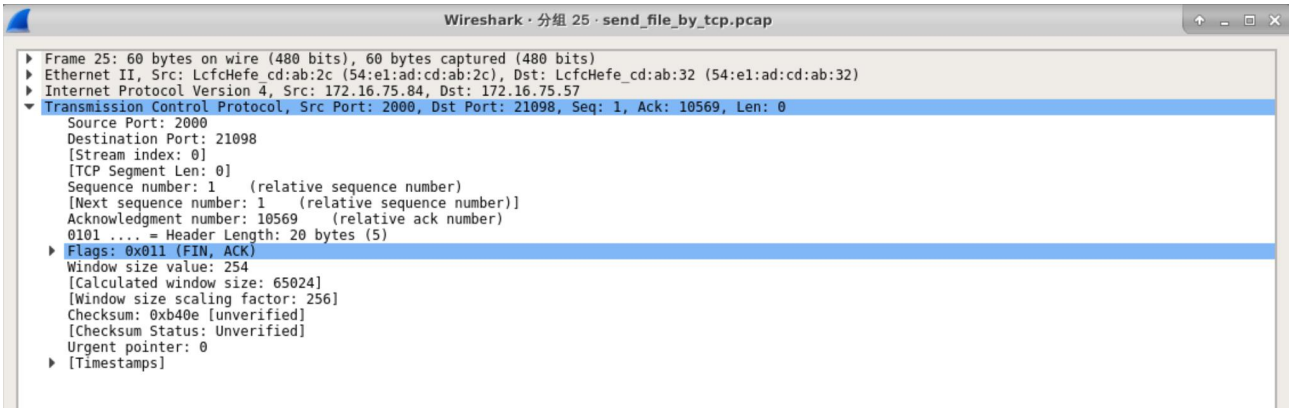
2、TCP 关闭连接过程分析



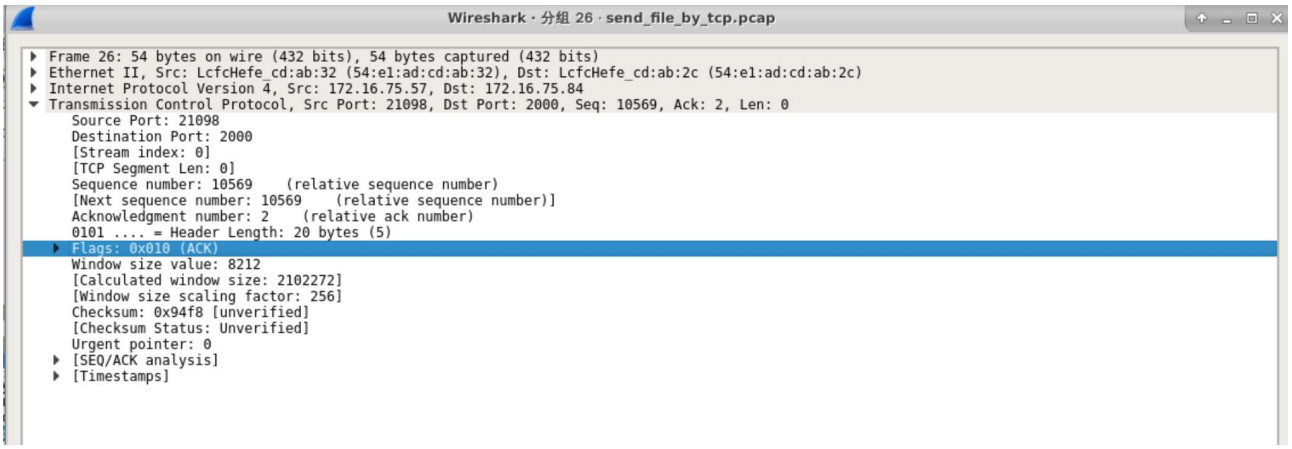
第一次挥手：主动关闭方发送一个 FIN = 1 ，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：不会再给对方发送数据了；



第二次挥手：被动关闭方收到 FIN = 1 包后，发送一个 ACK = 1 给对方，确认序号为:收到报文序号 Seq +收到报文所携带数据长度 len+ 1=1 。上一个报文可能“捎带”了主动关闭方发送的最后一块数据,其长度用字段 len 来表示。

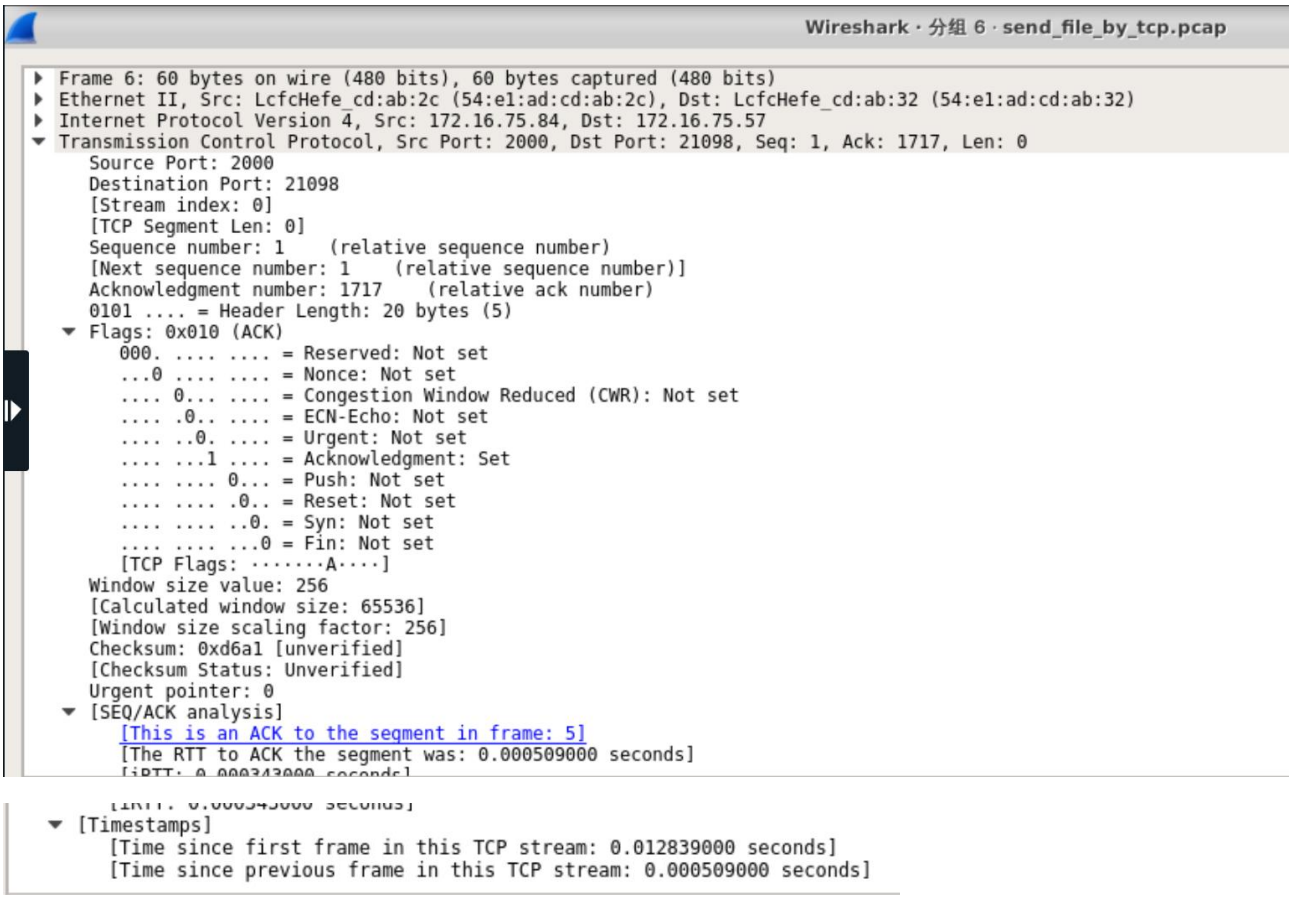


第三次挥手：被动关闭方发送一个 FIN = 1 ，同时 ACK=1，用来关闭被动关闭方到主动关闭方的数据传送，也就是告诉主动关闭方，我的数据也发送完了，不会再给你发送数据了；



第四次挥手：主动关闭方收到 FIN 后，发送一个 ACK 给被动关闭方，确认序号为：收到报文的序号 Seq + 收到报文所携带数据长度 len+ 1 ，至此，完成四次挥手，双向连接关闭。

3、6 号报文分析



Source Port: 2000 //请求端端口：2000

Destination Port: 21098 //服务器端端口：21098

[Stream index: 0] //tcp 流序号：0（wireshark 中对源于同一 tcp 流的包的标记）

[TCP Segment Len: 0] //tcp 报文长度：0

Sequence Number: 1(relative sequence number) //报文序列号：1（相对序号）

Sequence Number (raw): 1959254470 //报文序列号：1959254470（绝对序号）

[Next Sequence Number: 1(relative sequence number)] //下一序列号：1

Acknowledgment Number: 1717(relative ack number) //确认号：1717（相对序号）

Acknowledgment number (raw): 1749145722 //确认号：1749145722（绝对序号）

0101 … = Header Length: 20 bytes (5) //报文头部长度（数据偏移）：20bytes

Flags : 0x010 (ACK) //报文类型：ACK

000… = Reserved: Not set //保留字段

…0… = Nonce: Not set//随机数：无效（随机数(Nonce)是任意的或非重复的值，它包括在经过一个协议的数据交换中，通常为保证活跃度以及避免受重复攻击）

…0… = Congestion window Reduced (CWR): Not set

//拥塞窗口减少戳：无标记（TCP 拥塞控制）

…0… =ECN-Echo: Not set //显式拥塞戳:无效（ECN-Echo 与 TCP 拥塞控制）

…0… = Urgent: Not set //紧急指针戳：无效

…1… = Acknowledgment: Set //确认号戳：有效

…0… = Push: Not set //推送戳:无效

…0… = Reset: Not set //复位戳：无效

…0. = Syn: Not set //同步戳:无效

…0 = Fin: Not set //结束戳:无效

[TCP Flags …… A……]

window: 256 //窗口大小：256（TCP 接收者缓冲的字节大小）

[Calculated window size: 65536] //计算出的窗口大小（窗口大小单位*窗口大小）

[window size scaling factor: 256] //窗口大小换算系数

Checksum: 0xd6a1 [unverified] //校验和

[Checksum Status : Unverified] //校验状态

Urgent Pointer: 0（如果设置了 URG 位，这个域将被检查作为额外的指令，告诉 CPU 从数据包的哪里开始读取数据）

[SEQ/ACK analysis]//序列号及确认号的分析结果，当且仅当数据中含有 ACK 时，才有此项！

[This is an ACK to the segment in frame: 5]//这是对 5 号报文的回应

[The RTT to ACK the segment was: 0.00050900e seconds]//往返时延

[iRTT: 8.000343000 seconds]//互联网往返时延

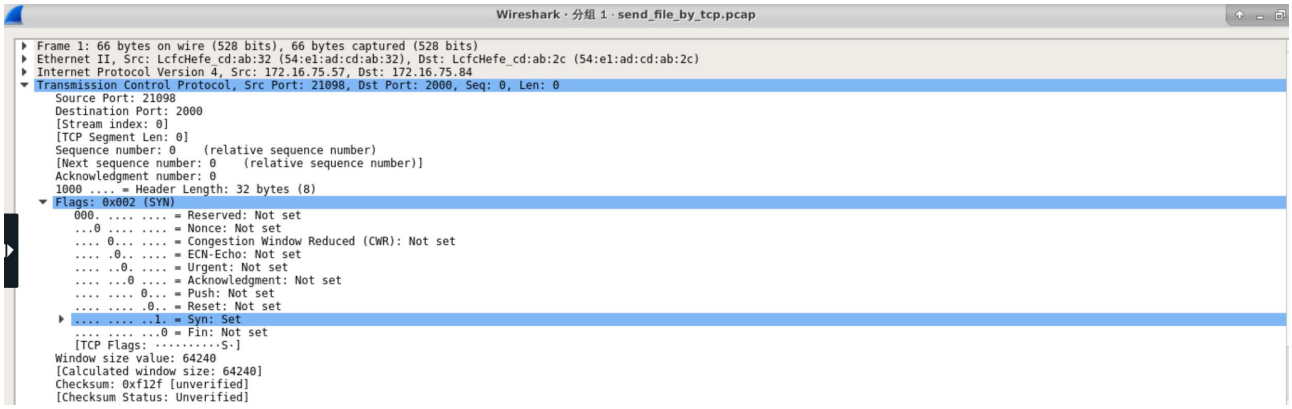
[Timestamps]//时间戳

[Time since first frame in this TCP stream: 0.012839000 seconds]// 从此 TCP 流中的第一帧开始的时间：0.012839000 秒

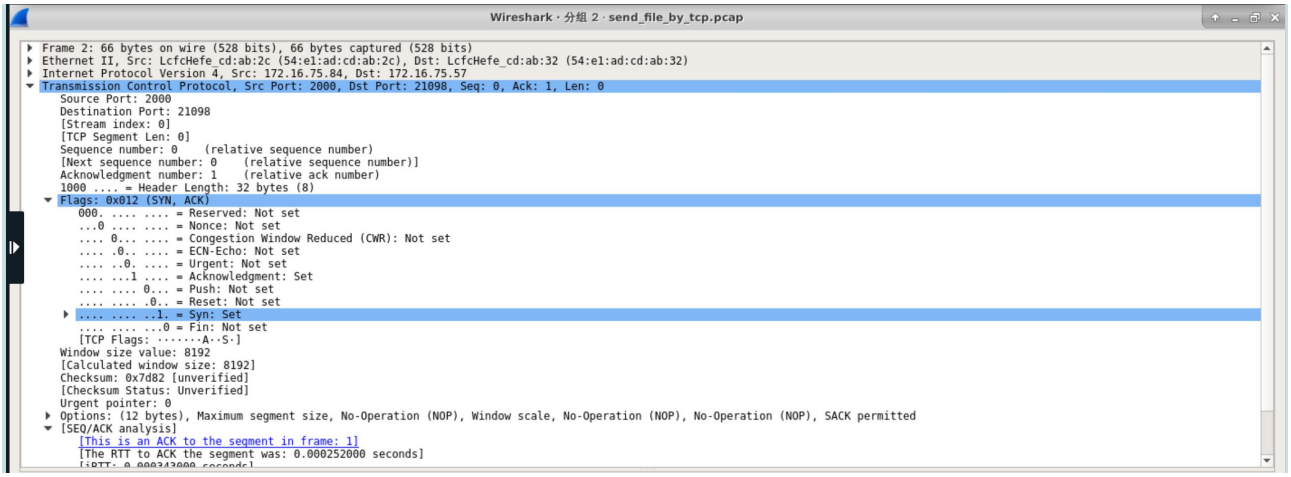
[Time since previous frame in this TCP stream: 0.000509000 seconds]// 从此 TCP 流中的上一帧开始的时间：0.000509000 秒

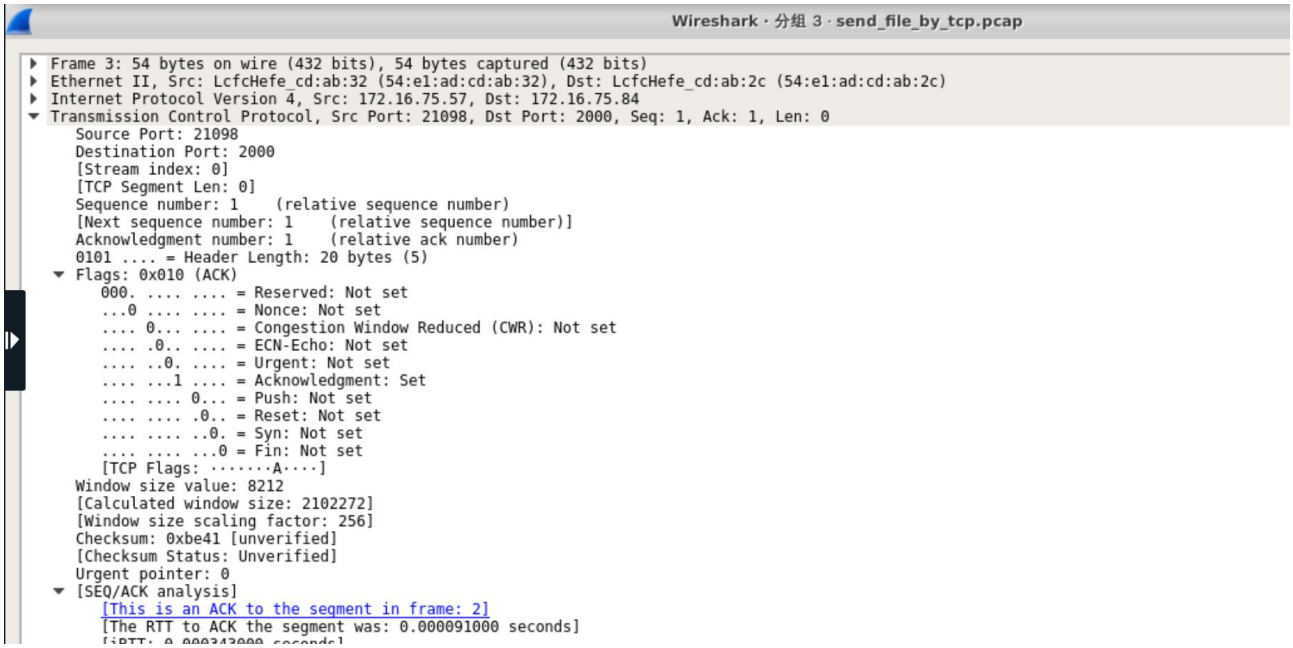
4、A 方 TCP 报文序列号分析

（1）A 方第 1 个报文的序列号（相对值）是 0

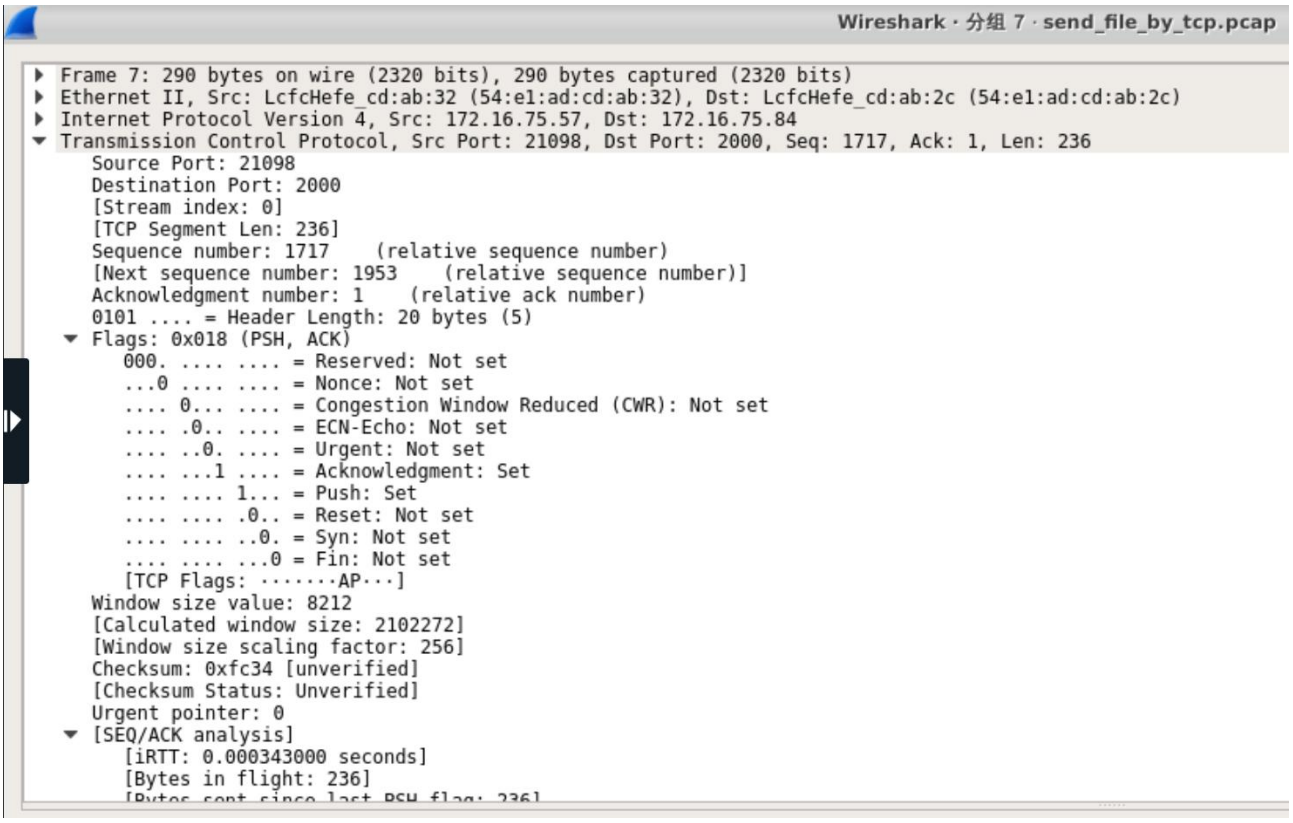


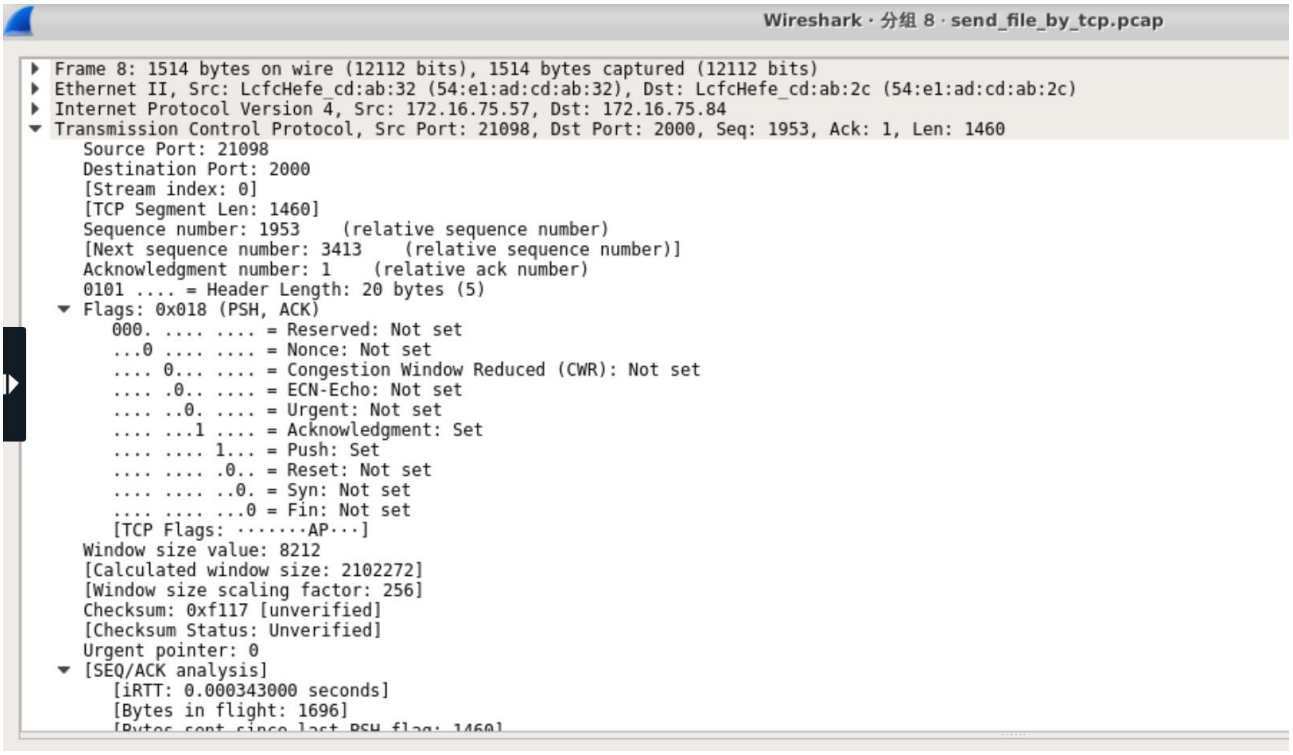
（2）因为在 TCP 建立连接的第二次握手中的确认号为 ack=X+1=1，A 方第 2 个报文作为 TCP 建立连接的第三次握手，其序列号 seq=X+1=1



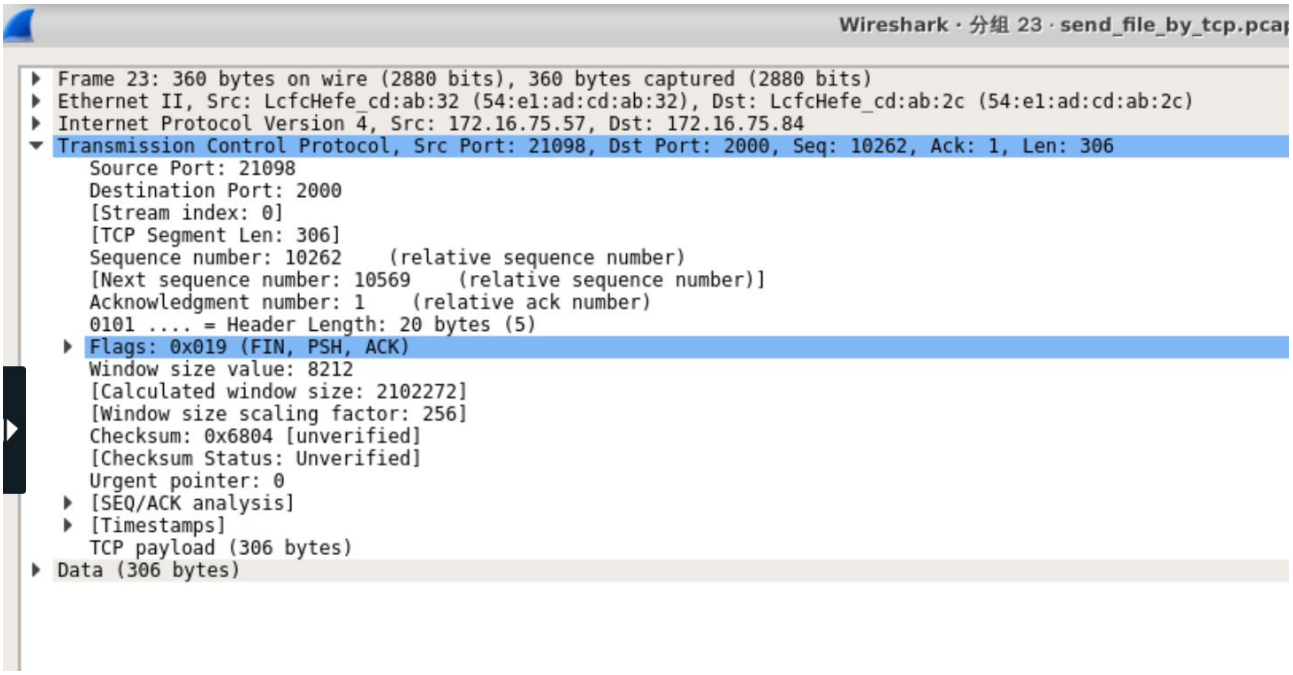


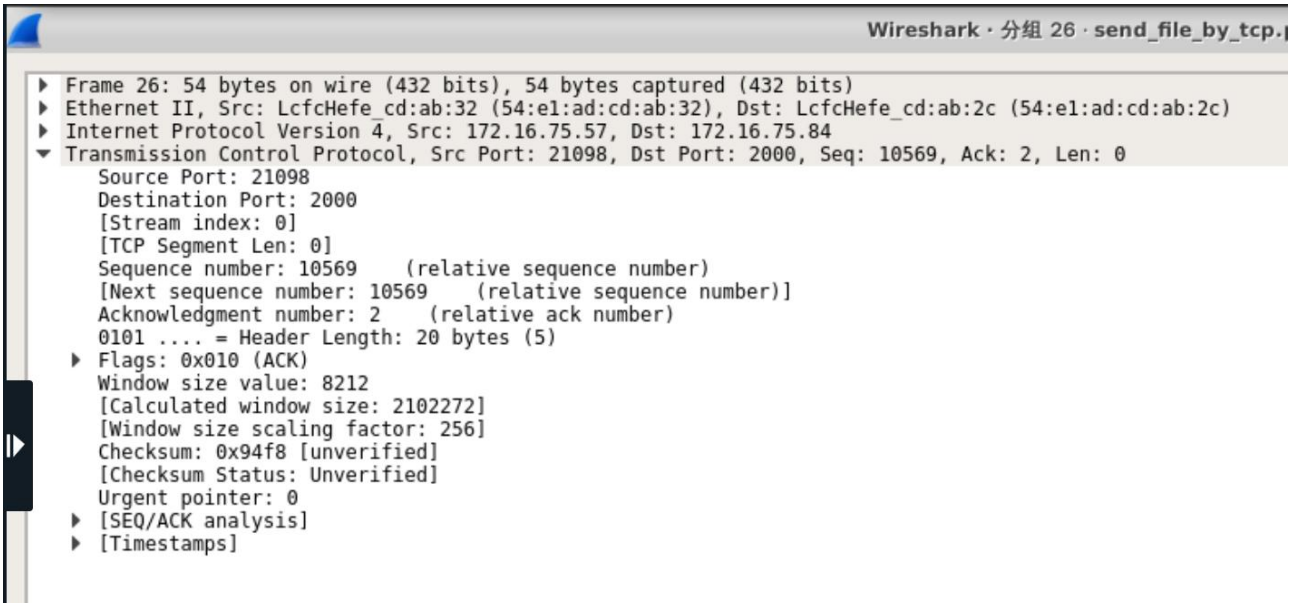
（3）下一个报文序列号 = 其前一报文序列号 + 其前一报文所携带数据长度



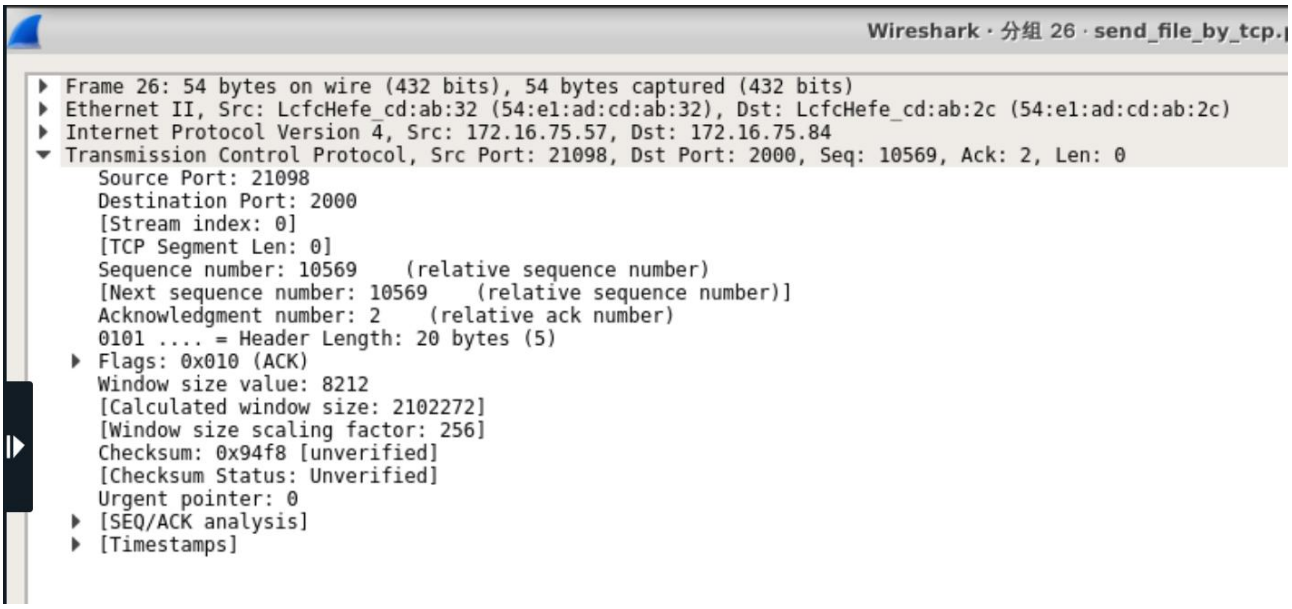


（4）在最后的 TCP 连接关闭时（第四次挥手），主动关闭方收到 FIN 后，会额外发送一个 ACK 给被动关闭方（这就是该 26 号报文的产生原因），此时 26 号报文（A 方最后一个报文的序号）+1 即 10568+1 得到最终的报文序列号为 10569.


Wireshark · 分组 23 · send_file_by_tcp.pcap
Frame 23: 360 bytes on wire (2880 bits), 360 bytes captured (2880 bits)
Ethernet II, Src: LcfcHefe_cd:ab:32 (54:e1:ad:cd:ab:32), Dst: LcfcHefe_cd:ab:2c (54:e1:ad:cd:ab:2c)
Internet Protocol Version 4, Src: 172.16.75.57, Dst: 172.16.75.84
Transmission Control Protocol, Src Port: 21098, Dst Port: 2000, Seq: 10262, Ack: 1, Len: 306
Source Port: 21098
Destination Port: 2000
[Stream index: 0]
[TCP Segment Len: 306]
Sequence number: 10262 (relative sequence number)
[Next sequence number: 10569 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x019 (FIN, PSH, ACK)
Window size value: 8212
[Calculated window size: 2102272]
[Window size scaling factor: 256]
Checksum: 0x6804 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[SEQ/ACK analysis]
[Timestamps]
TCP payload (306 bytes)
Data (306 bytes)

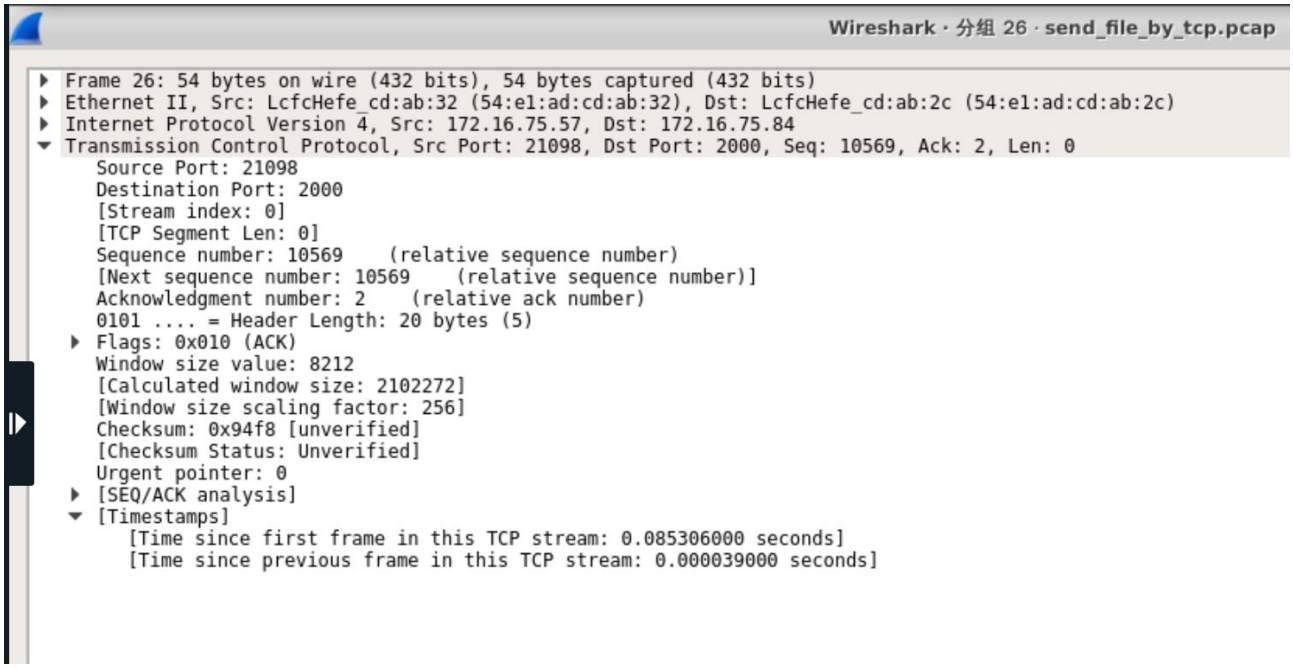

Wireshark · 分组 26 · send_file_by_tcp.
Frame 26: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
Ethernet II, Src: LcfcHefe_cd:ab:32 (54:e1:ad:cd:ab:32), Dst: LcfcHefe_cd:ab:2c (54:e1:ad:cd:ab:2c)
Internet Protocol Version 4, Src: 172.16.75.57, Dst: 172.16.75.84
Transmission Control Protocol, Src Port: 21098, Dst Port: 2000, Seq: 10569, Ack: 2, Len: 0
Source Port: 21098
Destination Port: 2000
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 10569 (relative sequence number)
[Next sequence number: 10569 (relative sequence number)]
Acknowledgment number: 2 (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window size value: 8212
[Calculated window size: 2102272]
[Window size scaling factor: 256]
Checksum: 0x94f8 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[SEQ/ACK analysis]
[Timestamps]

10567 字节



5、计算

1）如图，双方通信所用时间：0.085306s



2)A 方发送的所有的帧的长度之和为过滤后 lengh 的数值总和,A 方发送的所有的帧的长度之和:11497 字节

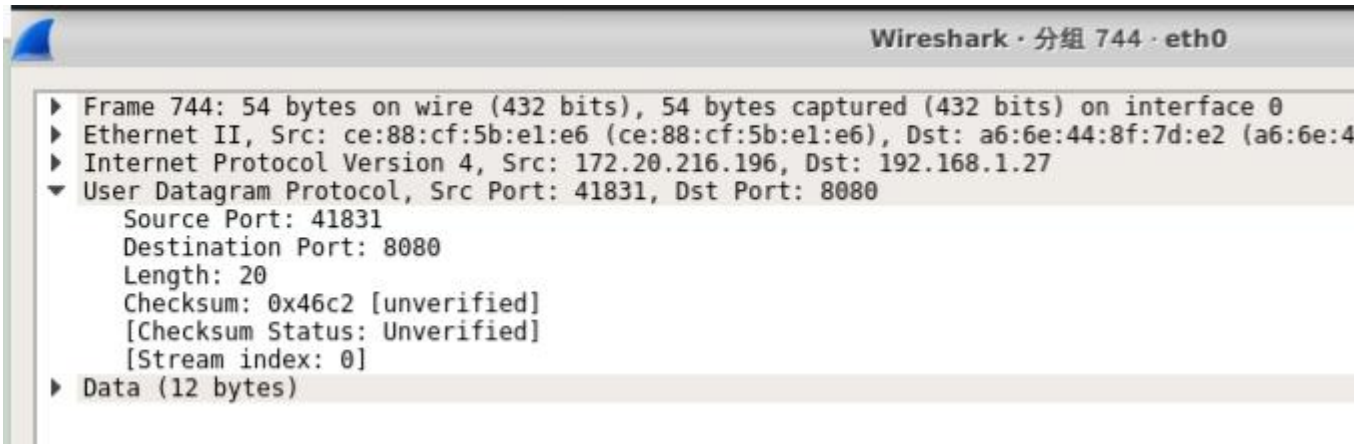
3）A->B 连接的通信吞吐率:

A 方发送的所有的帧的长度之和*8/10^6)Mb/双方通信所用的时间 s

A->B 连接的通信吞吐率:1.078Mbps

六. UDP 协议分析

1、UDP 报文分析



User Datagram Protocol, Src Port: 41831,Dst Port: 8080//UDP 协议
Source Port: 41831 //请求方端口：41831
Destination Port: 8080 //服务器端口：8080
Length: 20 //长度：20
Checksum: 0x46c2 [unverified] //校验和
[checksum Status: Unverified] //校验状态
[Stream index: 0] //流序号：0

2、分析 TCP、UDP 协议主要特点

填写下表：

序号	特性	TCP 协议的特点	UDP 协议的特点
1	连接	面向连接	无连接
2	可靠性	高可靠	较可靠
3	实时性	一般	较好
4	网络开销	较大	较小
5	数据负载边界	面向字节流，以整个待传输的数据为边界，保证数据正确性	基于数据报，以每段报文为边界，不保证传输的数据顺序

七. 实验总结

本次实验作为计算机网络这门课程中很重要的一环：对 TCP 及 UDP 协议的理解、学习和调用协议的过程。是极其有必要升入理解和掌握的，通过此次实验，我认识到传输层的 TCP/UDP 协议与网络层的 IP 协议是 Internet 最基本的协议，是 Internet 国际互联网络的基础。通过在 wireshark 中对数据报的构造和解析、跟踪 TCP 应用通信，我加深了对 TCP 报文结构的理解，并能领会 TCP 协议通信机制：三次握手以及四次挥手的建立连接/关闭连接的机制，结合报文对整个通信过程进行分析，包括建立连接、传输数据、关闭连接三个重要阶段的主要判别特征以及其作用。而通过对 TCP/UDP 报文的各个字段的分析，我进一步加深了他们在通信中通过自身标志位从而对每次传输的调节与控制以及整个通信的流量控制和服务质量。我认识到：IP 层并不保证数据报一定被正确地递交到接收方，也不指示数据报的发送速度有多快。是由 TCP 负责快速地发送数据报，以便使用网络容量，同时不会引起网络拥塞：在 TCP 超时后，要重传没有递交的数据报（即使被正确递交的数据报，也可能存在错序的问题），TCP 把接收到的数据报重新装配成正确的顺序。从而达到可靠的通信服务。而 UDP 在牺牲了部分通信质量的情况下，获得了相比于 TCP 更快、更高效的数据传输，所以经常被用以传输文件进程中。