华东师范大学软件工程学院实验报告

实验课程：计算机网络实践	**年级**：2023 级	**实验成绩**：
实验名称：TCP	**姓名**：顾翌炜	
实验编号：06	**学号**：10235101527	**实验日期**：2024/12/27
指导教师：王廷	**组号**：01	**实验时间**：2024/12/27

1 实验目的

1) 学会通过 Wireshark 获取 TCP 消息
2) 掌握 TCP 数据包结构
3) 掌握 TCP 数据包各字段的含义
4) 掌握 TCP 连接建立和释放的步骤
5) 掌握 TCP 数据传输阶段的过程

2 实验内容与实验步骤

2.1 实验内容

1) 捕获并检查获取到的包
2) 分析 TCP 结构
3) 分析 TCP 连接建立和释放的过程
4) 分析 TCP 数据传输的过程

2.2 实验步骤

1) 输入以下指令，用 wget 确认链接有效。

```
C:\User\GHOST> wget http://img.zcool.cn/community/01
    dcd059117b12a801216a3e9c4fd5.jpg
```

2) 启动 Wireshark，在菜单栏的捕获 → 选项中进行设置，选择已连接的以太网，设置捕获过滤器为 tcp and host img.zcool.cn。我们主要观察客户端与服务器之间的 tcp 流。
3) 捕获开始后，重复第一步，重新发送请求。
4) 当 wget 命令结束后，停止 Wireshark 捕获。

3 实验环境

- 操作系统：Windows 11 家庭中文版 23H2 22631.4460
- 网络适配器：Killer(R)Wi-Fi 6E AX1675i 160MHz Wireless Network Adapter(211NGW)
- Wireshark：Version 4.4.1
- wget：GNU Wget 1.21.4 built on mingw32

4 实验过程与分析

4.1 捕获 TCP 报文

首先，我们使用 wget 确认链接有效。

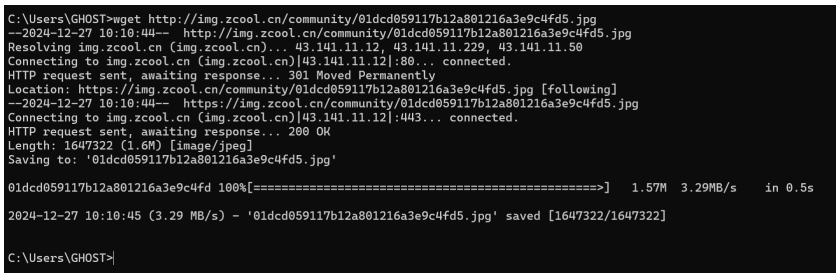

图 1: 使用 wget 确认链接有效

然后，我们启动 Wireshark，在菜单栏的捕获 → 选项中进行设置，选择已连接的以太网，设置捕获过滤器为 tcp and host img.zcool.cn。

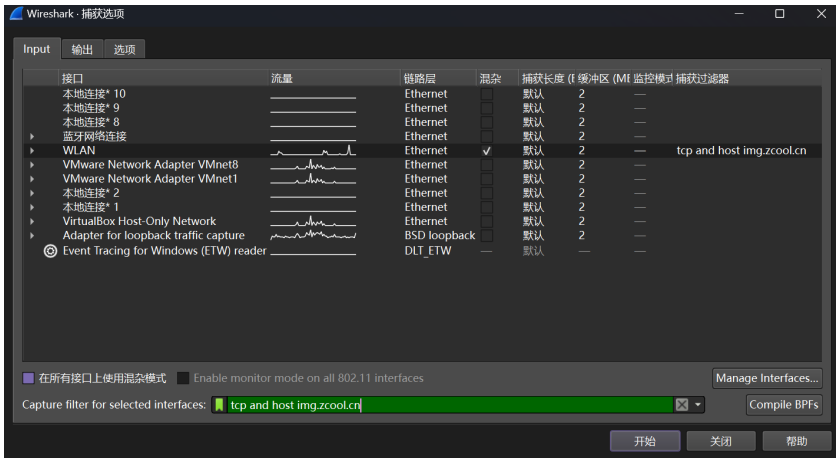

图 2: 设置捕获

捕获开始后，重复第一步，重新发送请求。

当 wget 命令结束后，停止 Wireshark 捕获。

捕获结果如下：

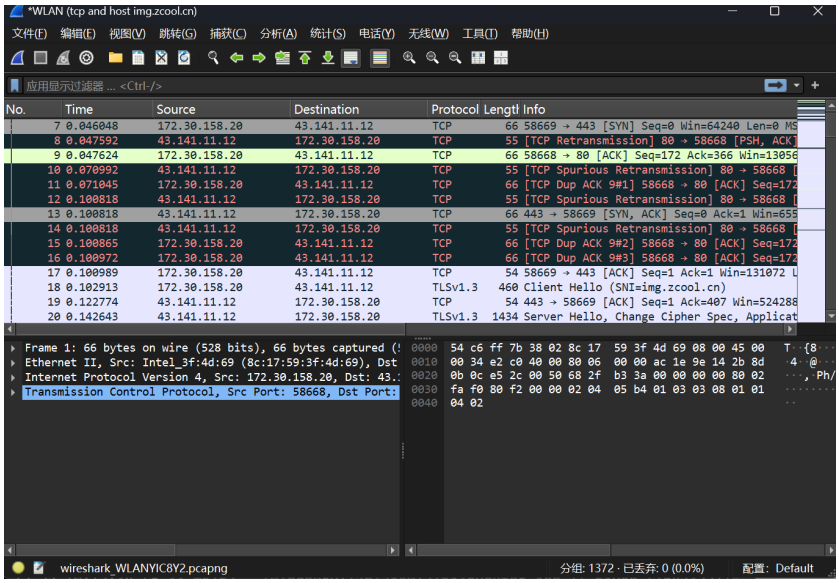

图 3: 捕获结果

4.2 分析 TCP 报文

选择一个 TCP 数据包，如下所示：

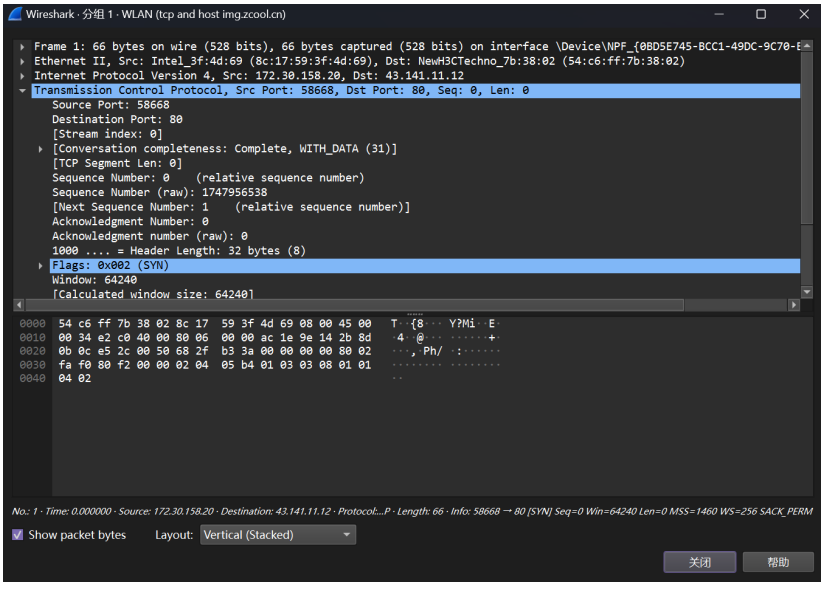

图 4: TCP 数据包

具体来看每一个部分的信息：

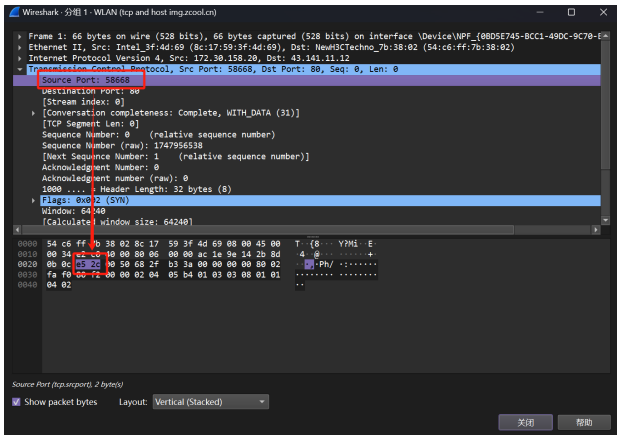

图 5: Source Port

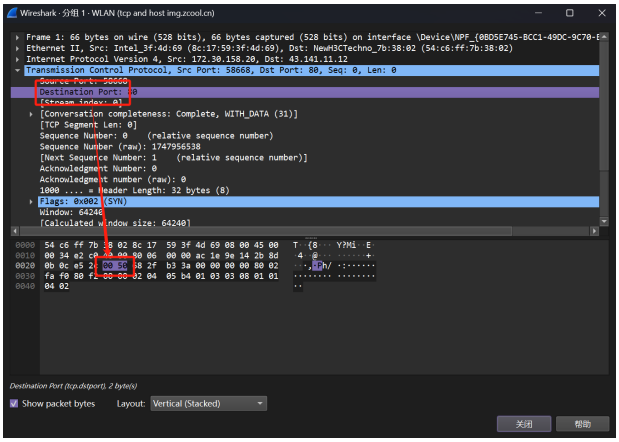

图 6: Destination Port

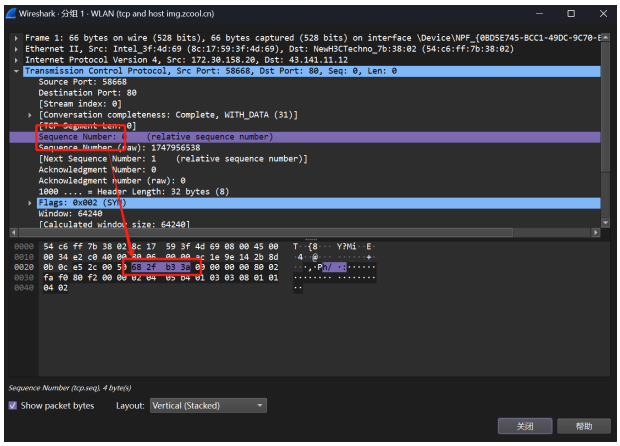

图 7: Sequence Number

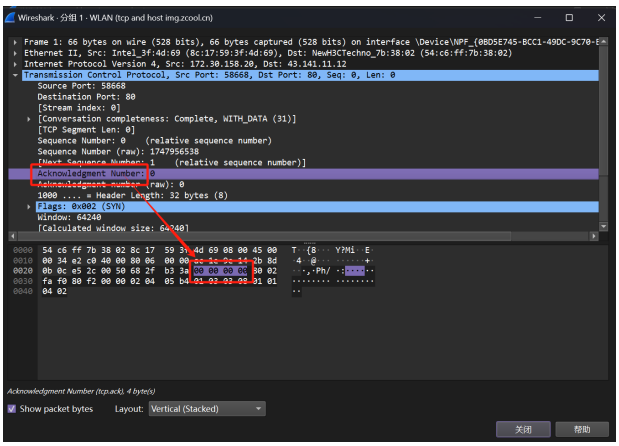

图 8: Acknowledge Number

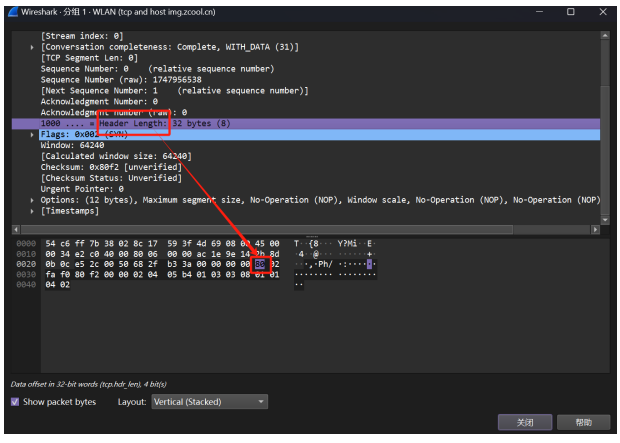

图 9: Header Length

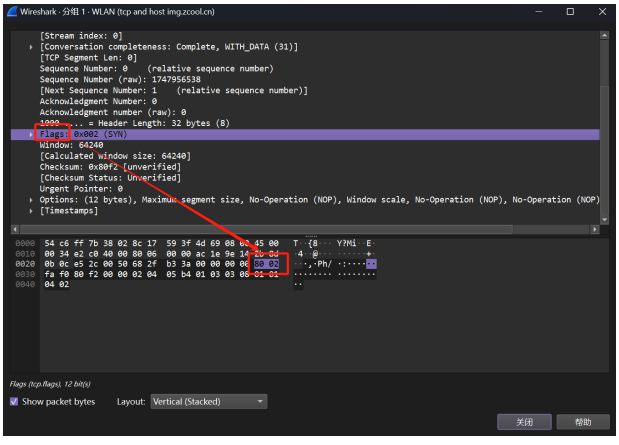

图 10: Flags

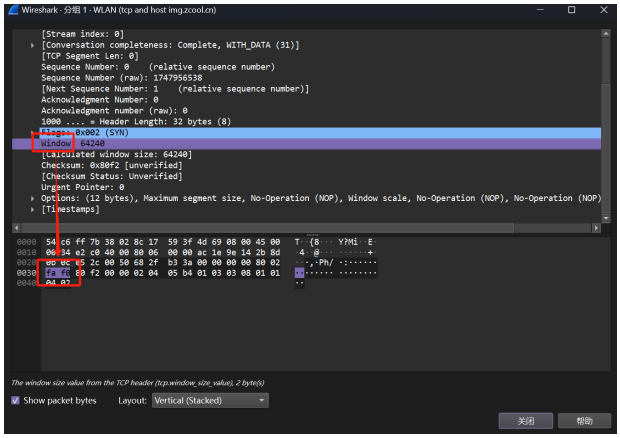

图 11: window

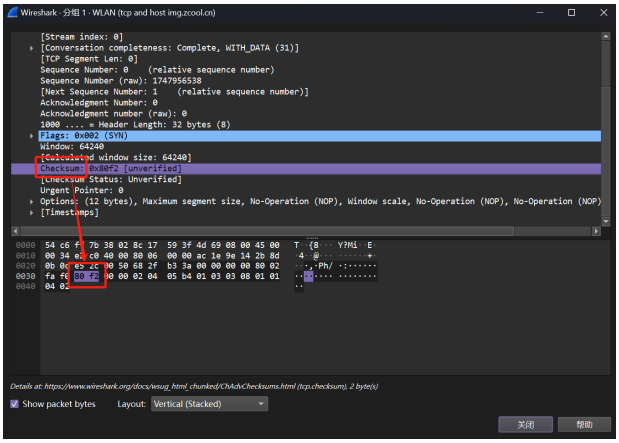

图 12: Checksum

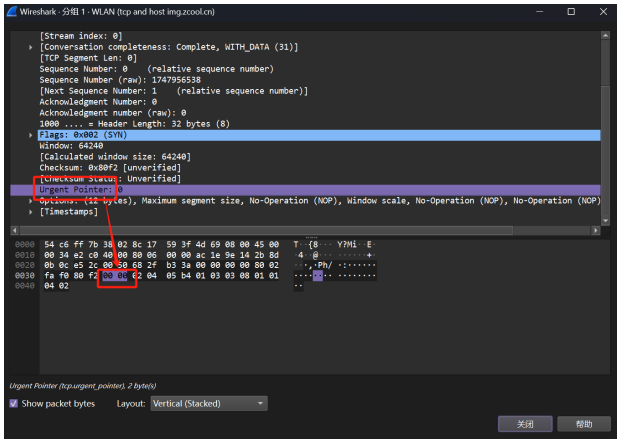

图 13: Urgent Number

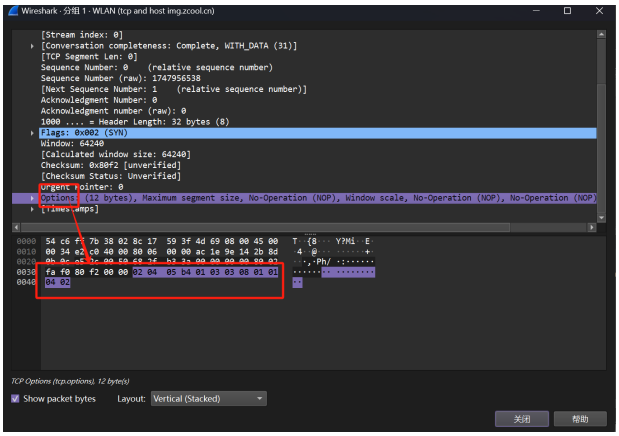

图 14: Options

可以画出 TCP 包的结构如下：

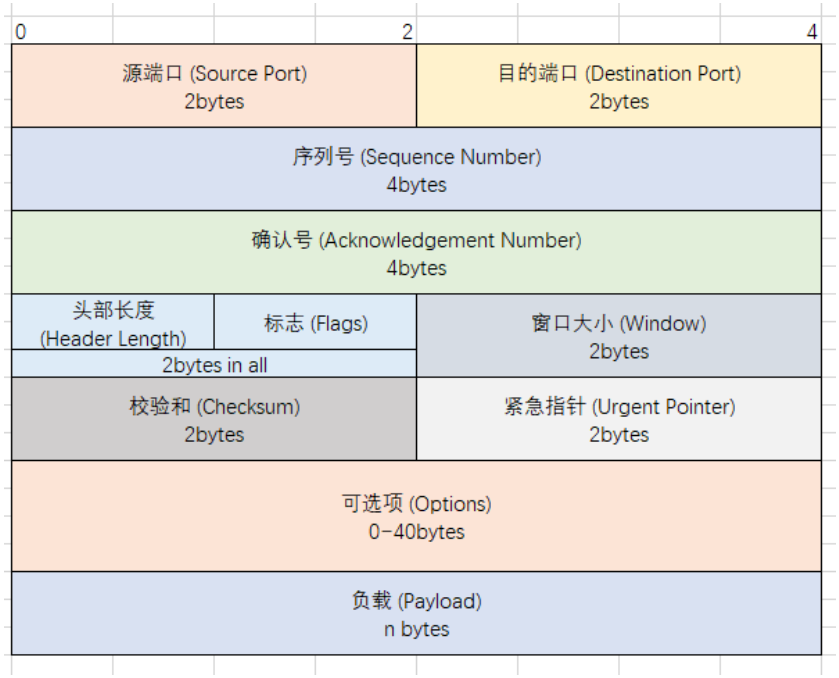

图 15: TCP 包结构

其中 Flags 字段包括 Reserved, Accurate ECN, Congestion Window Reduced, ECN-Echo, Urgent, Acknowledgment, Push, Reset, Syn, Fin。如下图所示：

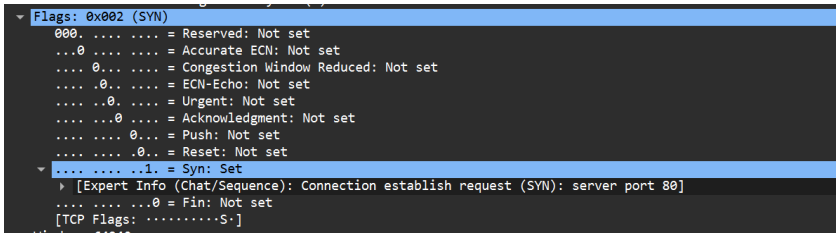

图 16: Flags

TCP 数据包头部各字段的含义如下：

- 源端口：源端口，占 2 字节，用于标识源主机的应用程序进程。
- 目的端口：目的端口，占 2 字节，用于标识目的主机的应用程序进程。
- 序列号：占 4 字节，用于标识从 TCP 源端向目的端发送的字节流，发送方对此进行标记。
- 确认号：占 4 字节，只有当 ACK 标志位为 1 时，确认号字段才有效，确认号等于上次接收到的字节序号加 1。
- 头部长度：占 1 字节，指示 TCP 头部的长度。
- 标志：与头部长度合占 2 字节。各标志位的含义如下：
  - Reserved：保留位。

  - Accurate ECN：显式拥塞通知。
  - Congestion Window Reduced：拥塞窗口减小。
  - ECN-Echo：显式拥塞通知回显。
  - URG：紧急指针（urgent pointer）有效。
  - ACK：确认号有效。
  - PSH：接收方应尽快将此报文段交付给应用层。
  - RST：重置连接。
  - SYN：发起一个新连接。
  - FIN：释放一个连接。

- 窗口大小：占 2 字节，表示发送方在收到确认前允许发送的字节数。
- 校验和：占 2 字节，用于检验 TCP 首部和 TCP 数据的正确性。
- 紧急指针：占 2 字节，仅当 URG 标志为 1 时有效。紧急指针指出紧急数据在报文段中的位置。
- 可选项：占 0-40 字节，可选项可以有 0 个或多个，用于一些可选的设置。

4.3 TCP 连接的建立和释放

4.3.1 三次握手

1) 第一次握手 TCP 客户进程也是先创建传输控制块 TCB，然后向服务器发出连接请求报文，这是报文首部中的同部位 SYN=1，同时选择一个初始序列号 seq=x ，此时，TCP 客户端进程进入了 SYN-SENT 同步已发送状态
2) 第二次握手 TCP 服务器收到请求报文后，如果同意连接，则会向客户端发出确认报文。确认报文中应该 ACK=1，SYN=1，确认号是 ack=x+1，同时也要为自己初始化一个序列号 seq=y，此时，TCP 服务器进程进入了 SYN-RCVD 同步收到状态
3) 第三次握手 TCP 客户端收到确认后，还要向服务器给出确认。确认报文的 ACK=1，ack=y+1，自己的序列号 seq=x+1，此时，TCP 连接建立，客户端进入 ESTABLISHED 已建立连接状态触发三次握手

在 Wireshark 中，我们来观察三次握手，可以看到三次握手的过程如下：

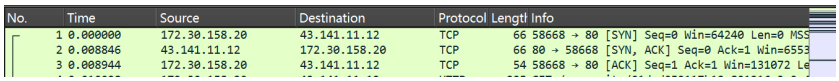

图 17: 三次握手

画出三次握手协议的步骤图如下：

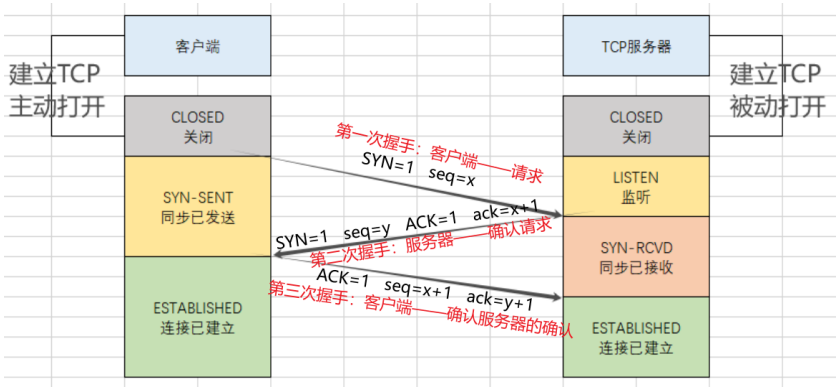

图 18: 三次握手过程

观察 SYN package，如下图所示：

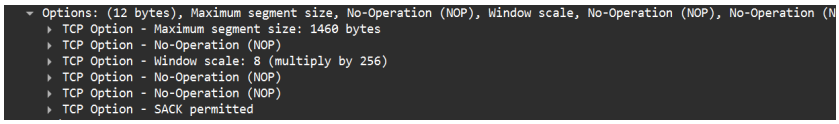

图 19: TCP SYN Options

Question

What TCP Options are carried on the SYN packets for your trace?

Answer

1) **Maximum segment size**：指定最大段的大小
2) **No-Operation**：无操作（可用于填充字节）
3) **Window scale**：指定了窗口大小的倍数，用于调整窗口大小的限制
4) **SACK permitted**：允许使用 SACK 选择确认（选择性确认）

此外，还可以观察到一些其他的信息，如 SYN 包中包含的 End of Option List、Timestamp（记录时间信息）等选项。在这个包中，Wireshark 将时间戳放置在 Options 之外，并用中括号标记，查看时可以关注 first frame 和 previous frame，均为 0。

4.3.2 四次挥手

TCP 的四次挥手过程如下：

1. 客户端发送关闭连接的报文段，设置 FIN 标志为 1，表示请求关闭连接，并停止发送数据。序列号设为 $seq = x$（为之前发送的所有数据的最后一个字节序号加 1），客户端进入 FIN-WAIT-1 状态，等待服务器的确认报文。

2. 服务器收到 FIN 报文后，发送确认报文，设置 ACK 为 1，且 ack $= x+1$，带上自己的序列号 $seq = y$。此后，服务器进入 CLOSE-WAIT 状态，并通知上层应用程序该连接已释放，但仍然可以接收数据。
3. 客户端收到服务器的 ACK 报文后，进入 FIN-WAIT-2 状态，此时仍能够接收来自服务器的数据。
4. 服务器在发送完所有数据后，会发送自己的 FIN 报文，并等待客户端的 ACK 确认。此时服务器进入 LAST-ACK 状态。
5. 客户端收到服务器的 FIN 报文后，会反馈一个 ACK 报文，告知服务器已接收，进入 TIME-WAIT 状态，等待 $2MSL$（2 倍报文段最大存活时间）。此时网络中可能存留最长时间的报文为 30 秒、1 分钟或 2 分钟。如果没有特殊情况，客户端最终会进入 CLOSED 状态。
6. 服务器收到客户端的 ACK 报文后，进入 CLOSED 状态，结束连接过程。

在 Wireshark 中，我们来观察四次挥手，可以看到四次挥手的过程如下：

图 20: 四次挥手

画出四次挥手协议的步骤图如下：

图 21: 四次挥手过程

回答问题：

Question

为什么建立连接时使用三次握手，而释放连接时四次挥手？

Answer

当一方接收到另一方发送的 `FIN` 包时，可能还有一些数据包尚未发送。因此，它会首先回复一个 `ACK` 包，待所有的数据包处理完毕并发送完后，随后再发送 `FIN ACK` 包以确认关闭连接。在此过程中，对方收到 `FIN ACK` 包后，会再回复一个 `ACK` 包。而在连接的建立过程中不会出现这样的情况，因此只需三次握手便可完成连接的建立。

4.4 TCP 数据传输

在 Static 统计菜单下，选择“IO”图表，以查看数据包的传输速率。
为了方便观察，将间隔改为 100ms，Y 轴改为 Bits，

1) 调整过滤器为“tcp.srcport==80”，只查看下载数据包，重新绘图
2) 调整过滤器为“tcp.dstport==80”，只查看上传数据包，重新绘图

观察 `Wireshark` 生成的 `IO` 图表, 如下所示：

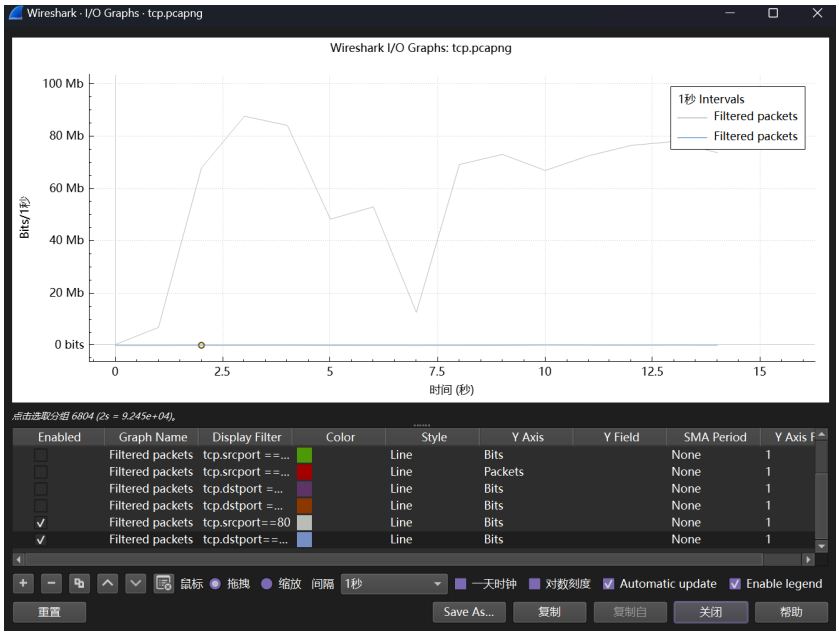

图 22: TCP Data Transfer

Question-1

What is the rough data rate in the download direction in packets/second and bits/second once the TCP connection is running well?

Answer-1

根据表格可以看出，下载速率到达 3000kb+/100ms，也就是 $3*10^8 bits/s$，下载速率到达大约 $1.8*10^8 bits/s$

Question-2

What percentage of this download rate is content? Show your calculation. To find out, look at a typical download packet; there should be many similar, large download packets. You can see how long it is, and how many bytes of TCP payload it contains.

Answer-2

选中一个数据包观察：

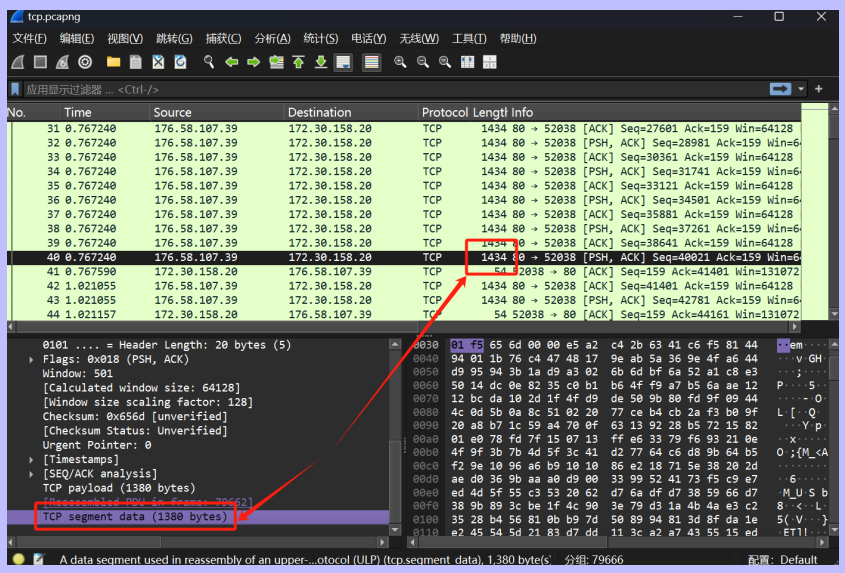

图 23: TCP Packet

可以看到其总大小为 1434 字节，其中 TCP 负载部分为 1380 字节，占比约为 96.23%.

Question-3

What is the rough data rate in the upload direction in packets/second and bits/second due to the ACK packets?

Answer-3

观察可得，平均值约为 1.0×10^5 Bits/s, 最大值达到了 2.0×10^5 Bits/s 。

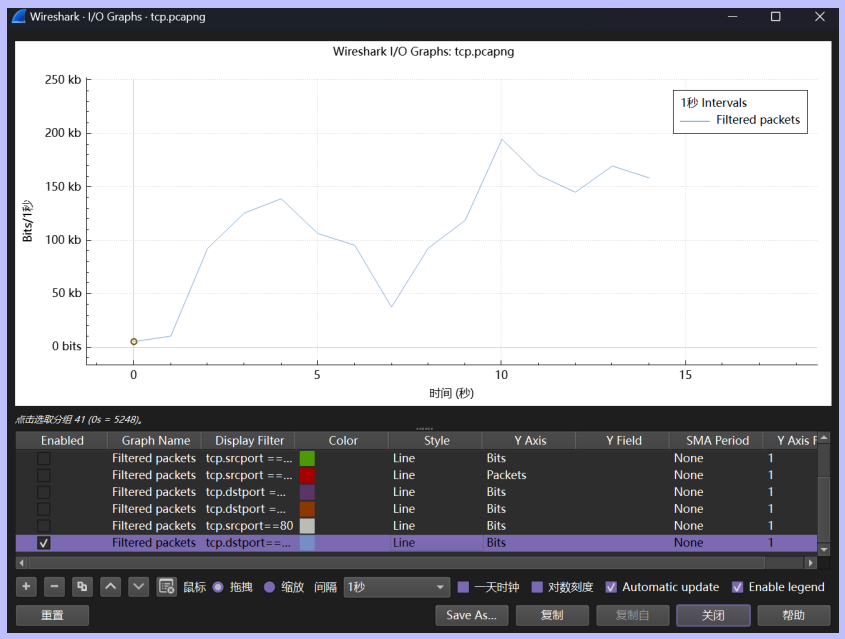

图 24: ACK Rough data rate

Question-4

If the most recently received TCP segment from the server has a sequence number of X, then what ACK number does the next transmitted TCP segment carry?

Answer-4

Ack 告诉下一个期望的序列号，因此下一个发送的 ACK 是 X + Segment Len

4.5 问题讨论

因为想要更好的展现数据，所以这里通过以下命令，选择一个 100MB 的照片来下载，重新 wget:

```
1 C:\User\GHOST> wget http://speedtest.london.linode.com/100MB-london.bin
```

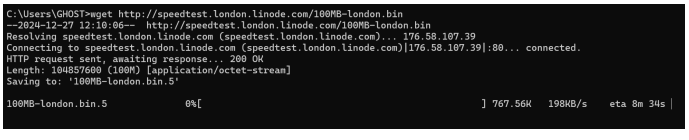

图 25: Get 100MB File

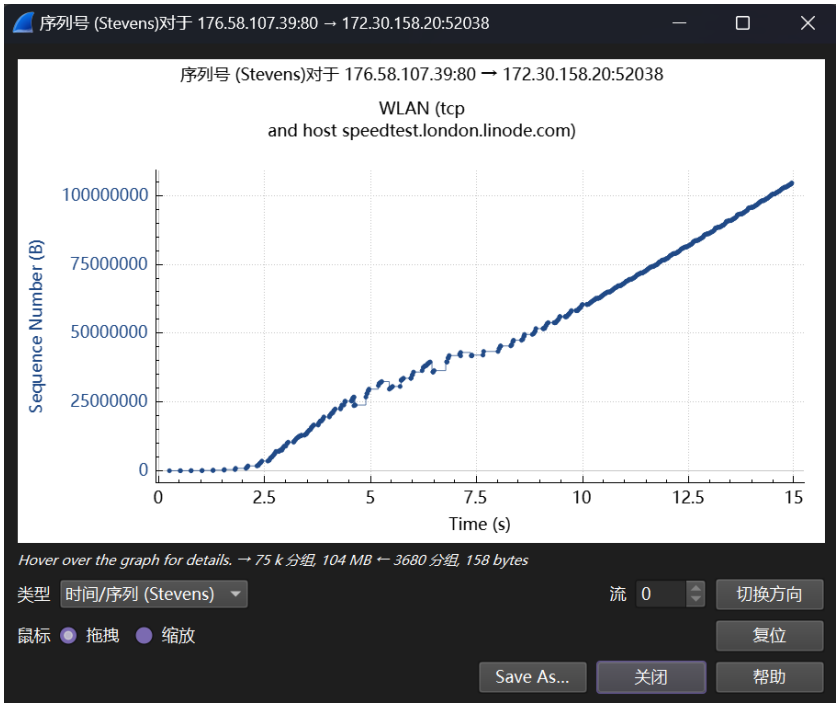

图 26: 序列号

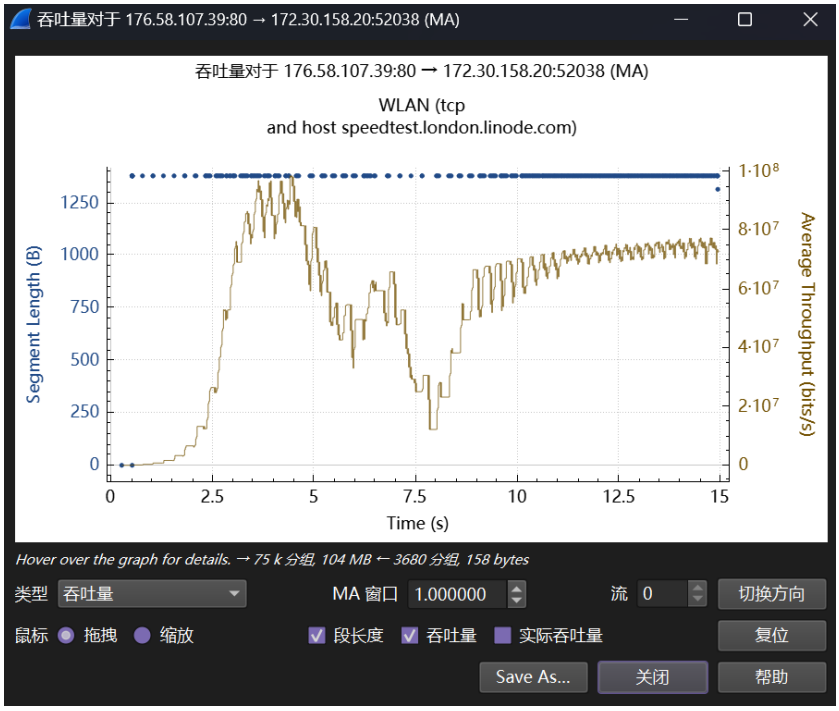

图 27: 吞吐量

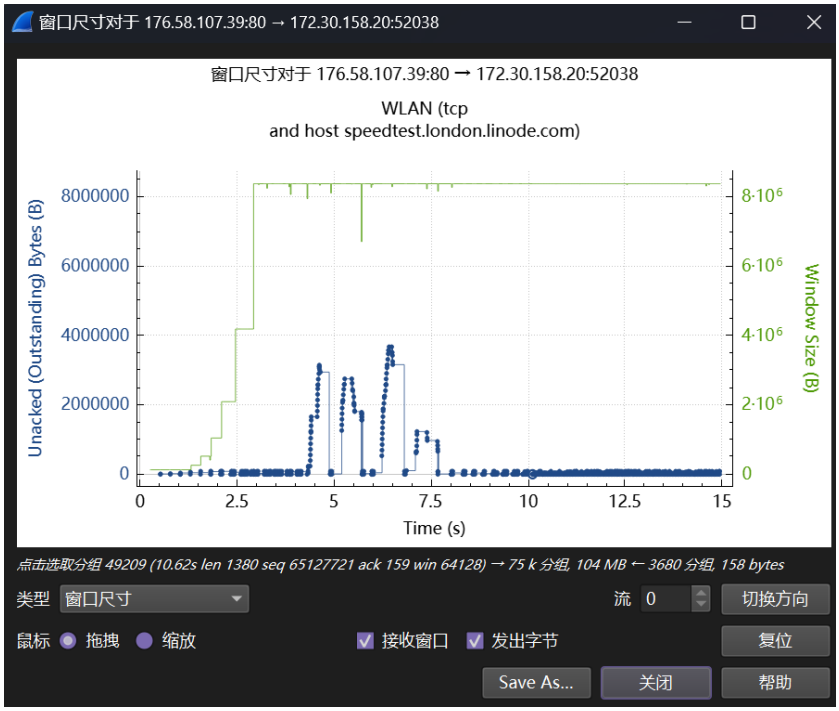

图 28: 窗口尺寸

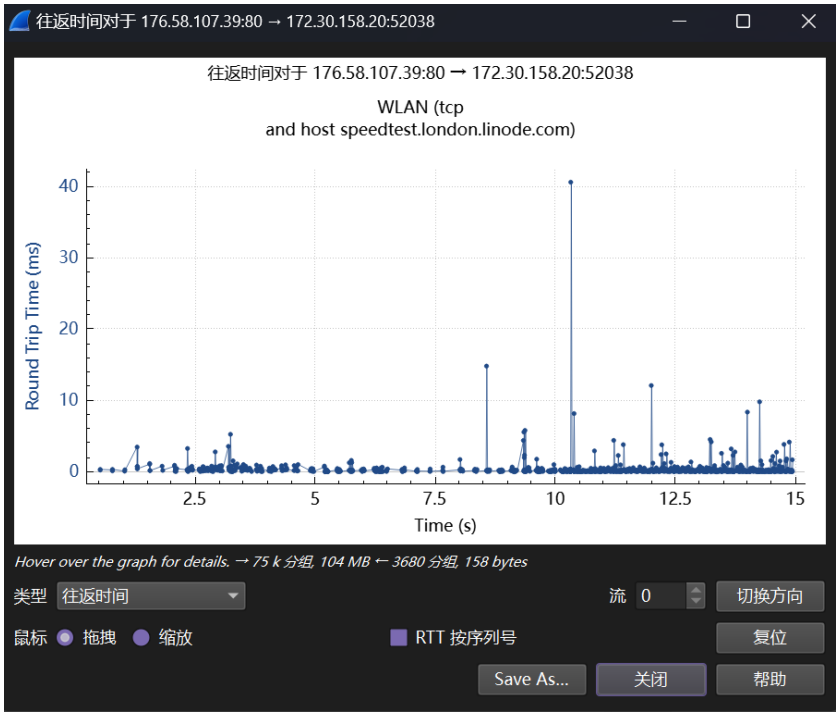

图 29: 往返时间

回答以下问题：

Question-1

Explore the congestion control and the classic AIMD behavior of TCP. To do this, you will likely want to capture a trace while you are sending (not receiving) a moderate amount of data on a TCP connection. You can then use the "TCP Stream Graph" tools as well as other analysis to observe how the congestion window changes over time.

Answer-1

从吞吐量表中可以观察到，吞吐量随着时间的推移而上升，但在特定时间点会经历急剧下降并出现波动。这种波动是由于网络拥堵事件导致的，此时传输控制协议（TCP）会调整其拥塞窗口大小，随后逐渐恢复，这一动态调整过程即为拥塞管理机制。

Question-2

Explore the reliability mechanisms of TCP more deeply. Capture a trace of a connection that includes segment loss. See what triggers the retransmissions and when. Also look at the round-trip time estimator.

Answer-2

当检测到数据包未能按预期顺序到达时（这通常是由于早期的数据包在传输过程中丢失），将激活数据重传流程，以确保丢失的数据包得到补充。** 往返时间图表已在上方列出

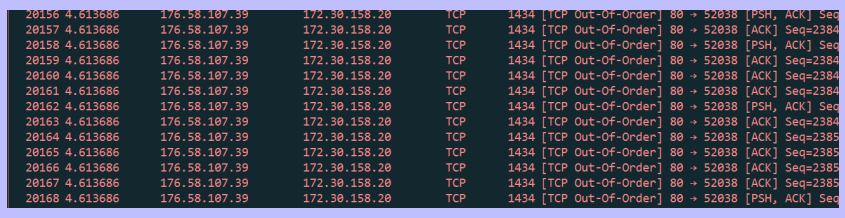

图 30: 错误重传

Question-3

Look at the use of options including SACK to work through the details. You should see information about ranges of received bytes during times of segment loss.

Answer-3

观察一个 Options 中包含 SACK 的数据包，如下图所示：

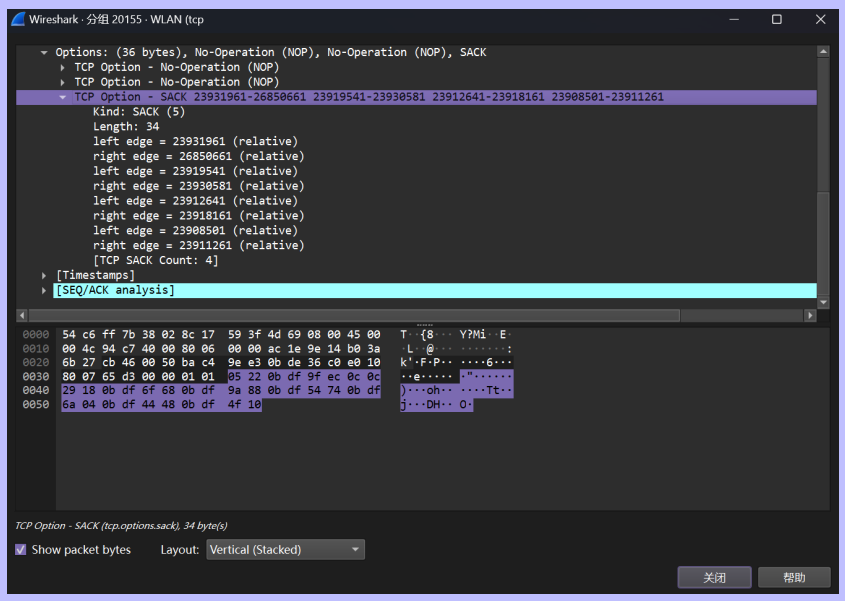

图 31: SACK

可以看到，SACK 中包含了丢失的数据包的序列号范围。

Question-4

TCP is the transport layer underlying the web. You can see how your browser makes use of TCP by setting up concurrent connections.

Answer-4

在处理用户发起的网络请求时，浏览器通常会创建多个 TCP 连接，以便并行处理这些请求。

5 实验结果总结

在本实验环节，我深入探究了传输控制协议（TCP）的数据包构成，并掌握了其连接的建立与终止流程，同时对数据传输机制有了全面的认识。

此外，我还对 TCP 的流量控制原理和确保数据传输可靠性的方法进行了学习。

6 附录

无