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	年　　级:	2023 级
实验编号:	Lab 06	实验名称:	TCP
姓　　名:	王海生	学　　号:	10235101559

目录

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

2.1.1 捕获 TCP 报文

1. 以 http://old.ecnu.edu.cn/site/xiaoli/2016.jpg 为例，用 wget 确认该链接有效。
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 old.ecnu.edu.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。

2.1.2 分析 TCP 报文

选择一个 TCP 帧，观察其协议层：

1. 根据你的理解，绘制 TCP 报文段的结构图（包括头部各字段的位置及大小）。

2.1.3 TCP 连接的建立和释放

TCP 的三次握手协议主要分为以下三个步骤：

1. 客户端发送一个 SYN 包给服务器，然后等待应答。
2. 服务器端回应给客户端一个 ACK=1、SYN=1 的 TCP 数据段。
3. 客户必须再次回应服务器端一个 ACK 确认数据段。

在你捕获到的结果中，找到设置了 SYN 标志的 TCP 段及其后的数据包，完成以下问题：

1. 绘制三次握手的时序图，直到并包括建立连接后计算机发送的第一个数据包（HTTP GET 请求），包括：
   - 每个数据段的序列号和 Ack 标号；
   - 本地计算机发送或接收每个数据段的时间（以毫秒为单位）；
   - 本地计算机从发送 SYN 段到接收到 SYN-ACK 段的往返时间。
2. SYN 数据包上携带哪些 TCP 选项？

TCP 的四次挥手主要分为以下四个步骤：

1. 客户端进程发出断开连接指令，这将导致客户端的 TCP 程序创建一个特殊的 TCP 报文段，发送到服务器。这个报文段的 FIN 字段被置为 1，表示这是一条断开连接的报文；
2. 服务器接收到客户端发来的断开连接报文，向客户端回送这个报文的确认报文（ACK 字段为 1），告诉服务器已经接收到 FIN 报文，并允许断开连接；
3. 服务器发送完确认报文后，服务器的 TCP 程序创建一条自己的断开连接报文，此报文的 FIN 字段被置为 1，然后发往客户端；
4. 客户端接收到服务器发来的 FIN 报文段，则产生一条确认报文（ACK 为 1），发送给服务器，告知服务器已经接收到了它的断开报文。服务器接收到这条 ACK 报文段后，释放 TCP 连接相关的资源（缓存和变量），而客户端等待一段时间后（半分钟、一分钟或两分钟），也释放处于客户端的缓存和变量。

回答问题：

1. 传输完成后，TCP 连接会以四次挥手或一端发送 RST 数据包的方式断开，同 1 一样，绘制 TCP 连接释放的时序图（从发出第一个 FIN 或 RST 到连接断开为止）。

2.1.4 TCP 数据传输

在“统计”菜单下，选择“IO 图表”，以查看数据包速率。

- 调整过滤器为“tcp.srcport==80”仅查看下载数据包，重新绘图；
- 调整过滤器为“tcp.dstport==80”仅查看上传数据包，重新绘图；

通过你对数据传输的理解，回答以下问题：

1） 实验中下载的大概速率为多少？（以 packets/s 和 bits/s 为单位）
2） 下载内容（即 TCP 有效负载）占下载率的百分比是多少？
3） 实验中上传的大概速率为多少？（以 packets/s 和 bits/s 为单位）
4） 如果最近从服务器收到的 TCP 数据段的序列号是 X，那么下一个发送的 ACK 是多少？

2.1.5 问题讨论

1. 探索 TCP 的拥塞控制和经典 AIMD 策略。
2. 更深入地探索 TCP 的可靠性机制。捕获包括段丢失的 TCP 连接，查看什么触发重新传输以及何时触发，另外查看往返时间估算工具。
3. 查看包括 SACK 在内的选项的使用以了解详细信息。
4. TCP 是 Web 的基础传输层。可以通过设置并发连接来查看浏览器如何使用 TCP。

2.2 实验步骤

1. 以 http://old.ecnu.edu.cn/site/xiaoli/2016.jpg 为例，用 wget 确认该链接有效

```
PS> wget http://old.ecnu.edu.cn/site/xiaoli/2016.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。
5. 分析 TCP 报文

3 实验环境

- 操作系统：Windows 11 家庭中文版 23H2 22631.2715
- 网络适配器：Killer(R) Wi-Fi 6 AX1650i 160MHz Wireless Network Adapter(201NGW)
- Wireshark：Version 4.2.0 (v4.2.0-0-g54eedfc63953)
- wget：GNU Wget 1.21.4 built on mingw32

4 实验结果与分析

4.1 捕获 TCP 报文

首先，我们使用 wget 命令，发现链接已经失效。

```
PS C:\Users\hswan\Desktop> wget http://old.ecnu.edu.cn/site/xiaoli/2016.jpg
wget : 未能解析此远程名称： 'old.ecnu.edu.cn'
所在位置 行:1 字符: 1
+ wget http://old.ecnu.edu.cn/site/xiaoli/2016.jpg
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebExce
   ption
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

PS C:\Users\hswan\Desktop>
```

图 1: 链接失效

接着尝试http://img.zcool.cn/community/01dcd059117b12a801216a3e9c4fd5.jpg，发现链接有效。

```
wget http://img.zcool.cn/community/01dcd059117b12a801216a3e9c4fd5.jpg
--2024-12-16 09:04:31--  http://img.zcool.cn/community/01
    dcd059117b12a801216a3e9c4fd5.jpg
Resolving img.zcool.cn (img.zcool.cn)... 116.211.153.215, 150.139.142.248,
    182.40.32.241, ...
Connecting to img.zcool.cn (img.zcool.cn)|116.211.153.215|:80 ... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://img.zcool.cn/community/01dcd059117b12a801216a3e9c4fd5.jpg
    [following]
```

```
--2024-12-16 09:04:31--  https://img.zcool.cn/community/01dcd059117b12a801216a3e9c4fd5.jpg
Connecting to img.zcool.cn (img.zcool.cn)|116.211.153.215|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1647322 (1.6M) [image/jpeg]
Saving to: '01dcd059117b12a801216a3e9c4fd5.jpg.1'

01dcd059117b12a801216a3e9c4fd5 100%[=====================================>]  1.57M --.-KB/s    in 0.1s

2024-12-16 09:04:31 (11.9 MB/s) - '01dcd059117b12a801216a3e9c4fd5.jpg.1' saved [1647322/1647322]
```

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`。我们主要观察客户端与服务器之间的 tcp 流。

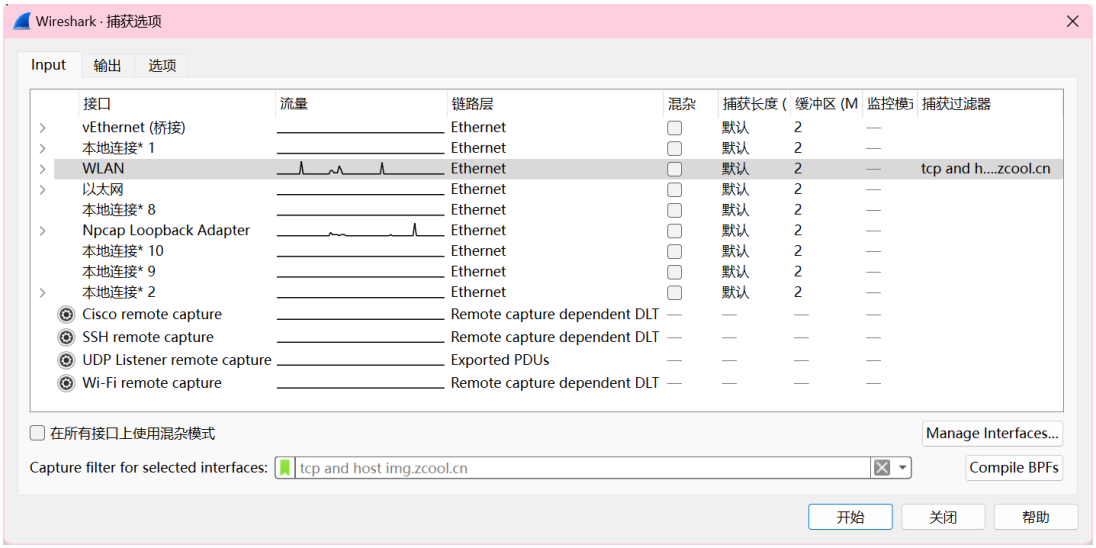
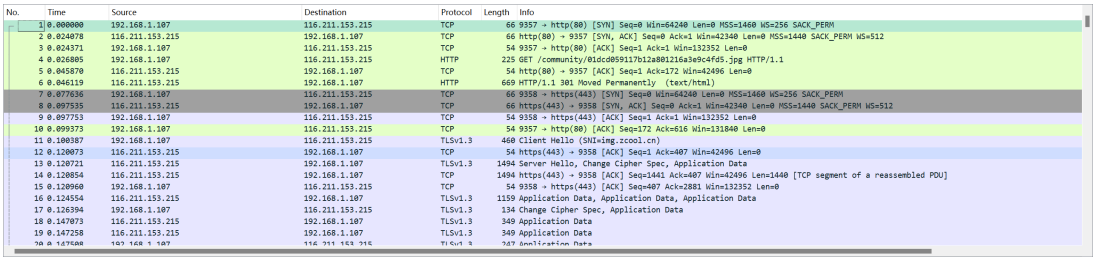


图 2: 设置捕获过滤器

捕获开始后，重复第一步，重新发送 wget 请求。

当 wget 命令结束后，停止 Wireshark 捕获。捕获结果如下：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.107	116.211.153.215	TCP	66	9357 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.024078	116.211.153.215	192.168.1.107	TCP	66	http(80) → 9357 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1440 SACK_PERM WS=512
3	0.024371	192.168.1.107	116.211.153.215	TCP	54	9357 → http(80) [ACK] Seq=1 Ack=1 Win=132352 Len=0
4	0.026805	192.168.1.107	116.211.153.215	HTTP	225	GET /community/01dcd059117b12a801216a3e9c4fd5.jpg HTTP/1.1
5	0.045870	116.211.153.215	192.168.1.107	TCP	54	http(80) → 9357 [ACK] Seq=1 Ack=172 Win=42496 Len=0
6	0.046119	116.211.153.215	192.168.1.107	HTTP	669	HTTP/1.1 301 Moved Permanently (text/html)
7	0.077636	192.168.1.107	116.211.153.215	TCP	66	9358 → https(443) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
8	0.097535	116.211.153.215	192.168.1.107	TCP	66	https(443) → 9358 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1440 SACK_PERM WS=512
9	0.097753	192.168.1.107	116.211.153.215	TCP	54	9358 → https(443) [ACK] Seq=1 Ack=1 Win=132352 Len=0
10	0.099373	192.168.1.107	116.211.153.215	TCP	54	9357 → http(80) [ACK] Seq=172 Ack=616 Win=131840 Len=0
11	0.100387	192.168.1.107	116.211.153.215	TLSv1.3	460	Client Hello (SNI=img.zcool.cn)
12	0.120073	116.211.153.215	192.168.1.107	TCP	54	https(443) → 9358 [ACK] Seq=1 Ack=407 Win=42496 Len=0
13	0.120721	116.211.153.215	192.168.1.107	TLSv1.3	1494	Server Hello, Change Cipher Spec, Application Data
14	0.120854	116.211.153.215	192.168.1.107	TCP	1494	https(443) → 9358 [ACK] Seq=1441 Ack=407 Win=42496 Len=1440 [TCP segment of a reassembled PDU]
15	0.120960	192.168.1.107	116.211.153.215	TCP	54	9358 → https(443) [ACK] Seq=407 Ack=2881 Win=132352 Len=0
16	0.124554	116.211.153.215	192.168.1.107	TLSv1.3	1159	Application Data, Application Data, Application Data
17	0.126394	192.168.1.107	116.211.153.215	TLSv1.3	134	Change Cipher Spec, Application Data
18	0.147073	116.211.153.215	192.168.1.107	TLSv1.3	349	Application Data
19	0.147258	116.211.153.215	192.168.1.107	TLSv1.3	349	Application Data

图 3: 捕获结果

4.2 TCP 报文段的结构

选择一个 TCP 数据包，如下所示：

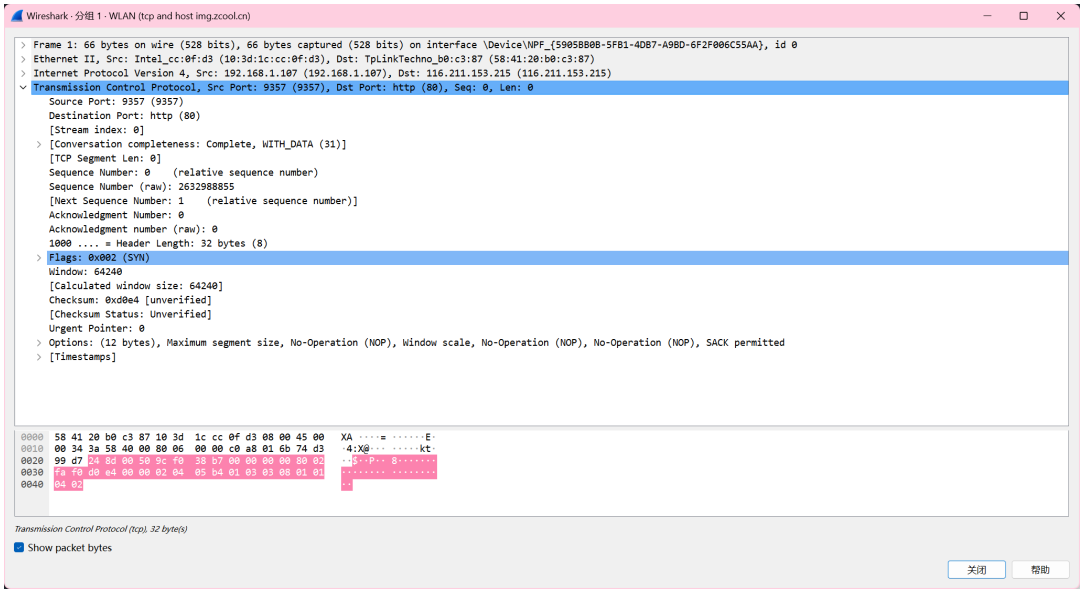


图 4: 一个 TCP 数据包

可以画出 TCP 包的结构如下：

<table>
<tr><td colspan="2">源端口 Source Port

2 bytes</td><td colspan="2">目的端口 Destination Port

2 bytes</td></tr>
<tr><td colspan="4">序列号 Sequence Number

4 bytes</td></tr>
<tr><td colspan="4">确认号 Acknowledgment Number

4 bytes</td></tr>
<tr><td>头部长度 Header Length</td><td>标志 Flags</td><td colspan="2">窗口大小 Window

2 bytes</td></tr>
<tr><td colspan="2">共 2 bytes</td><td colspan="2"></td></tr>
<tr><td colspan="2">校验和 Checksum

2 bytes</td><td colspan="2">紧急指针 Urgent Pointer

2 bytes</td></tr>
<tr><td colspan="4">可选项 Options

0-40 bytes</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">负载 Payload

n bytes</td></tr>
</table>

表 1: TCP 报文段的结构图

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

TCP 数据包头部各字段的含义如下：

- 源端口：源端口号，占 2 字节，用来标识源主机的应用程序进程。
- 目的端口：目的端口号，占 2 字节，用来标识目的主机的应用程序进程。
- 序列号：占 4 字节，用来标识从 TCP 源端向目的端发送的字节流，发起方发送数据时对此进行标记。
- 确认号：占 4 字节，只有 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：确认序号有效。
  - PSH：接收方应该尽快将这个报文交给应用层。
  - 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，窗口大小是指发送方在收到确认前允许发送的字节数。
- 校验和：占 2 字节，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，紧急指针才有效。紧急指针告诉系统此报文段中有紧急数据，紧急数据的字节数由紧急指针指出。
- 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 三次握手的时序图

我们知道，三次握手的过程如下：

1. 客户端发送一个 `SYN` 包给服务器，然后等待应答。
2. 服务器端回应给客户端一个 `ACK=1`、`SYN=1` 的 `TCP` 数据段。
3. 客户必须再次回应服务器端一个 `ACK` 确认数据段。

在 `Wireshark` 中，我们可以看到三次握手的过程如下：

1	0.000000	192.168.1.107	116.211.153.215	TCP	66 9357 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.024078	116.211.153.215	192.168.1.107	TCP	66 http(80) → 9357 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1440 SACK_PERM WS=512
3	0.024371	192.168.1.107	116.211.153.215	TCP	54 9357 → http(80) [ACK] Seq=1 Ack=1 Win=132352 Len=0

图 5: 三次握手

画出三次握手协议的步骤图如下：

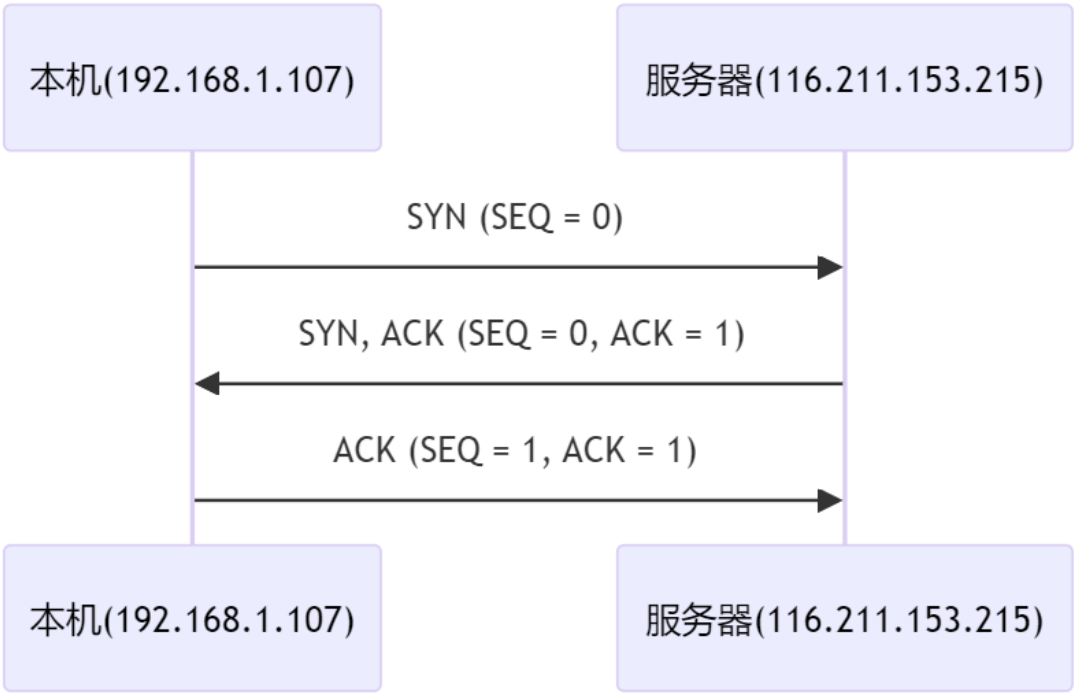


图 6: 三次握手协议的步骤图

4.3.2 SYN 数据包上携带的 TCP 选项

在 TCP 连接建立过程中，SYN 数据包用于初始化连接，并携带多种 TCP 选项：

1. Maximum Segment Size (MSS)

   定义了在这个连接上发送的 TCP 段中最大的数据量（不包括 TCP 头部）。接收端通过这个选项告诉发送端它能接受的最大段大小，从而避免分片。

2. Window Scale Option

   允许窗口大小超过 65,535 字节。它通过指定一个移位计数来扩大窗口尺寸，这使得 TCP 能够在高速、高延迟的网络上有效地工作。

3. SACK Permitted

   Selective Acknowledgment 允许接收方通知发送方哪些数据已经成功接收，哪些数据丢失需要重传。这样可以提高效率，因为不需要重传所有未确认的数据。

4. Timestamps

   时间戳选项提供了一种测量往返时间的方法，并且有助于处理旧的数据包（即防止老的重复数据包被错误地认为是新的）。每个数据包都包含发送时的时间戳，以及最近收到的数据包的时间戳回显。

5. No Operation (NOP)

   NOP 选项没有实际作用，通常用来填充其他选项之间的空间，确保选项字段对齐到 32 比特边界，便于解析。

6. User Timeout Option (UTO)

   它允许应用程序设置一个超时值，当超过该时间还未收到对方的响应，则关闭连接。

4.3.3 四次挥手的时序图

TCP 的四次挥手主要分为以下四个步骤：

1. 客户端进程发出断开连接指令，这将导致客户端的 TCP 程序创建一个特殊的 TCP 报文段，发送到服务器。这个报文段的 FIN 字段被置为 1，表示这是一条断开连接的报文；
2. 服务器接收到客户端发来的断开连接报文，向客户端回送这个报文的确认报文（ACK 字段为 1），告诉服务器已经接收到 FIN 报文，并允许断开连接；
3. 服务器发送完确认报文后，服务器的 TCP 程序创建一条自己的断开连接报文，此报文的 FIN 字段被置为 1，然后发往客户端；
4. 客户端接收到服务器发来的 FIN 报文段，则产生一条确认报文（ACK 为 1），发送给服务器，告知服务器已经接收到了它的断开报文。服务器接收到这条 ACK 报文段后，释放 TCP 连接相关的资源（缓存和变量），而客户端等待一段时间后（半分钟、一分钟或两分钟），也释放处于客户端的缓存和变量。

在 Wireshark 中，我们可以看到四次挥手的过程如下：

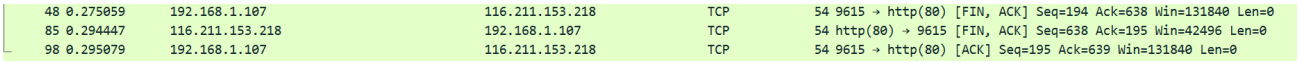


图 7: 四次挥手

四次挥手协议的时序图如下：

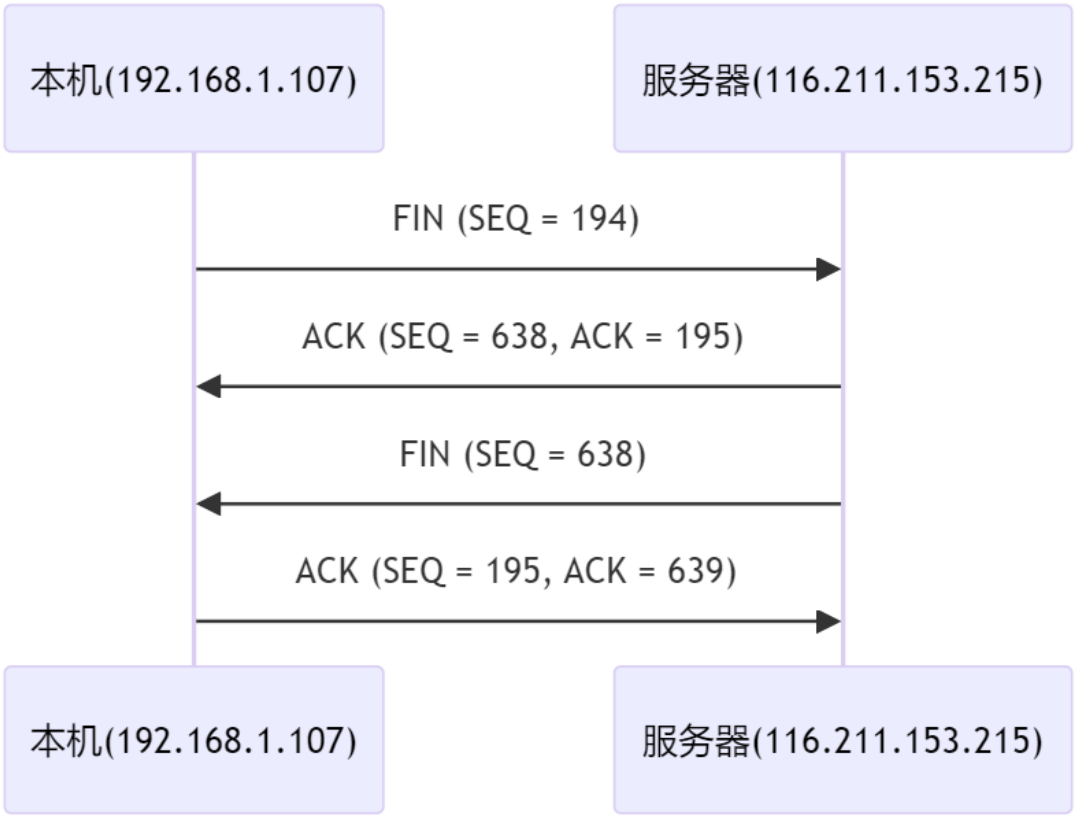


图 8: 四次挥手协议的步骤图

4.4 TCP 数据传输

通过你对数据传输的理解，回答以下问题：

1. 实验中下载的大概速率为多少？（以 `packets/s` 和 `bits/s` 为单位）

   答：观察 `Wireshark` 生成的 `IO` （下载）图表, 如下所示：

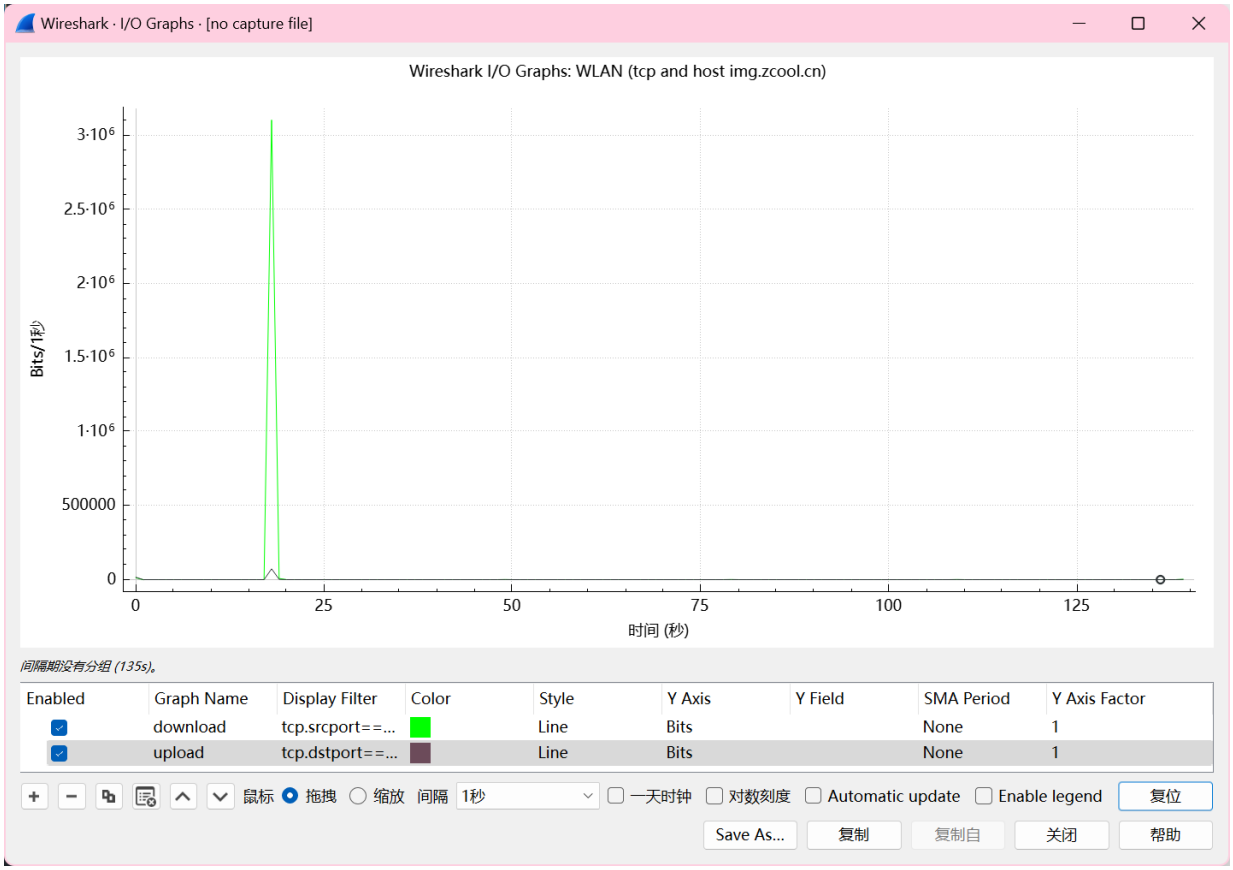


图 9: IO （下载）图表

所以，下载速度为 3107000 bits/s。

2. 下载内容（即 TCP 有效负载）占下载率的百分比是多少？

   答：使用 tcp.srcport==80 进行过滤，在中间位置选择一个较大的包，分析数据如下：总的数据量为 1374B，TCP 携带的内容为 1320B，所以内容占下载率的 96.10

3. 实验中上传的大概速率为多少？（以 packets/s 和 bits/s 为单位）

   答：观察 Wireshark 生成的 IO （上传）图表, 如下所示：

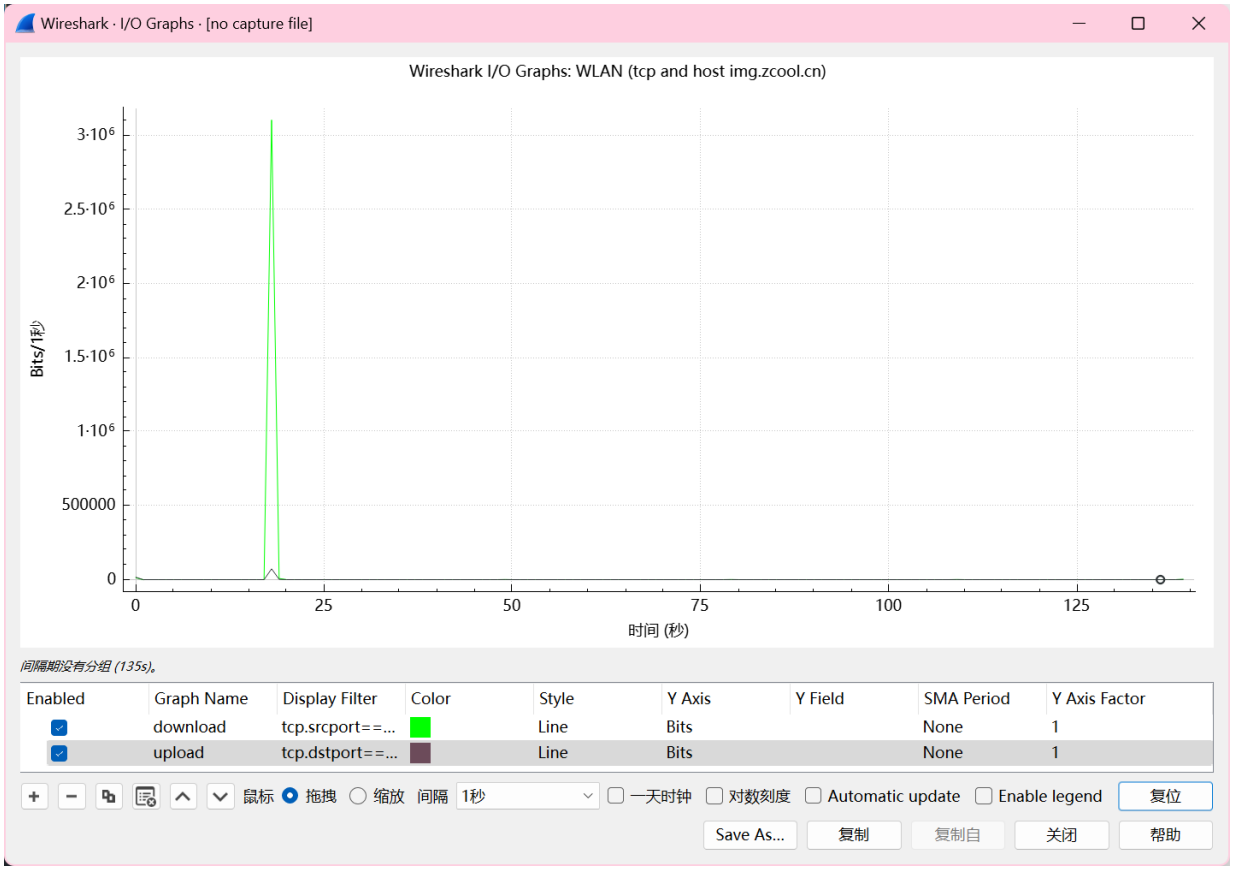


图 10: IO （上传）图表

所以，上传速度为 732900 bits/s。

4. 如果最近从服务器收到的 TCP 数据段的序列号是 X，那么下一个发送的 ACK 是多少？

   答：下一个发送的 ACK 是 X + Segment Len

4.5 继续探索 TCP 协议

下载一个大文件，观察 TCP 流图表，如下所示：

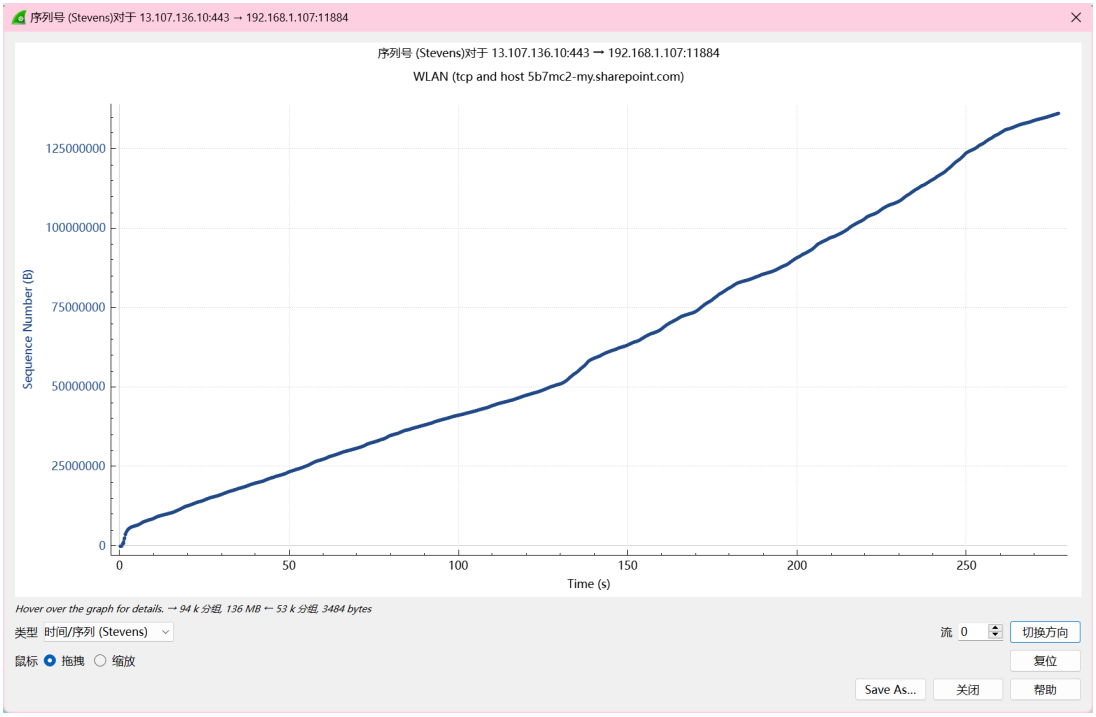


图 11: 序列号

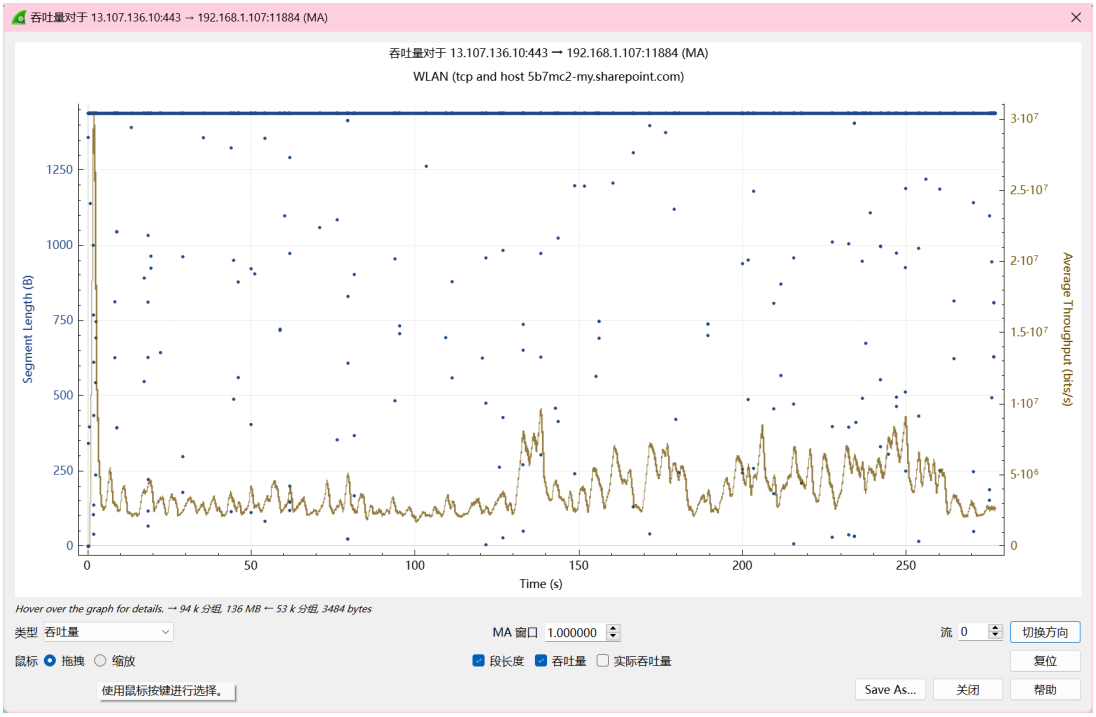


图 12: 吞吐量

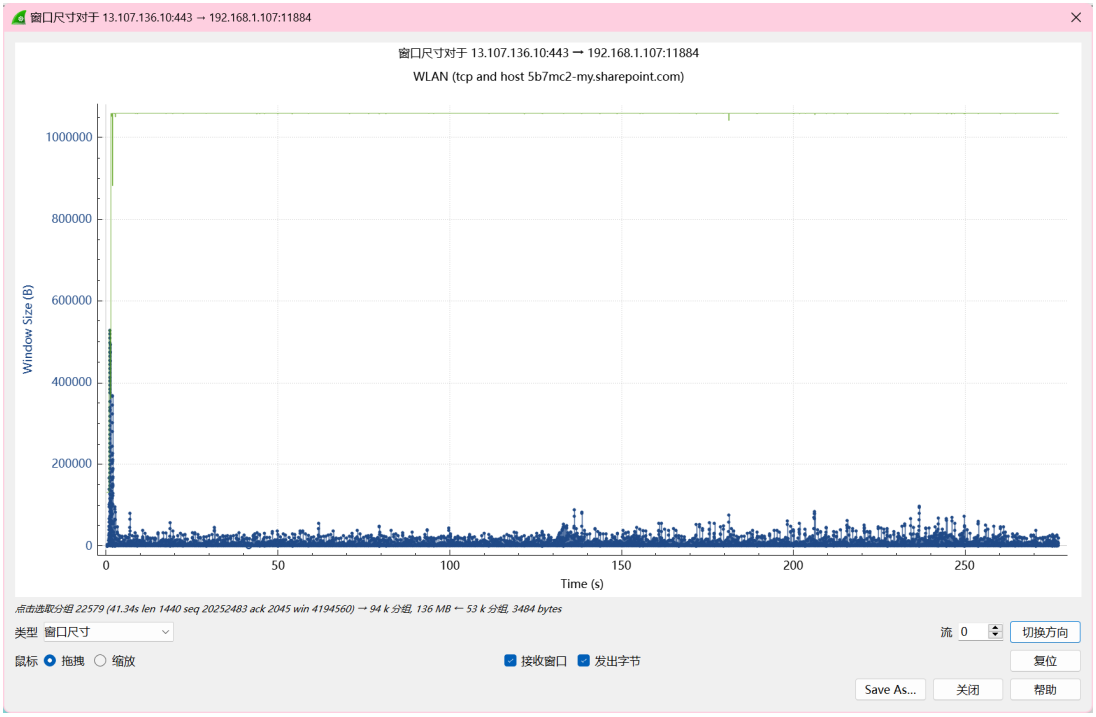


图 13: 窗口尺寸

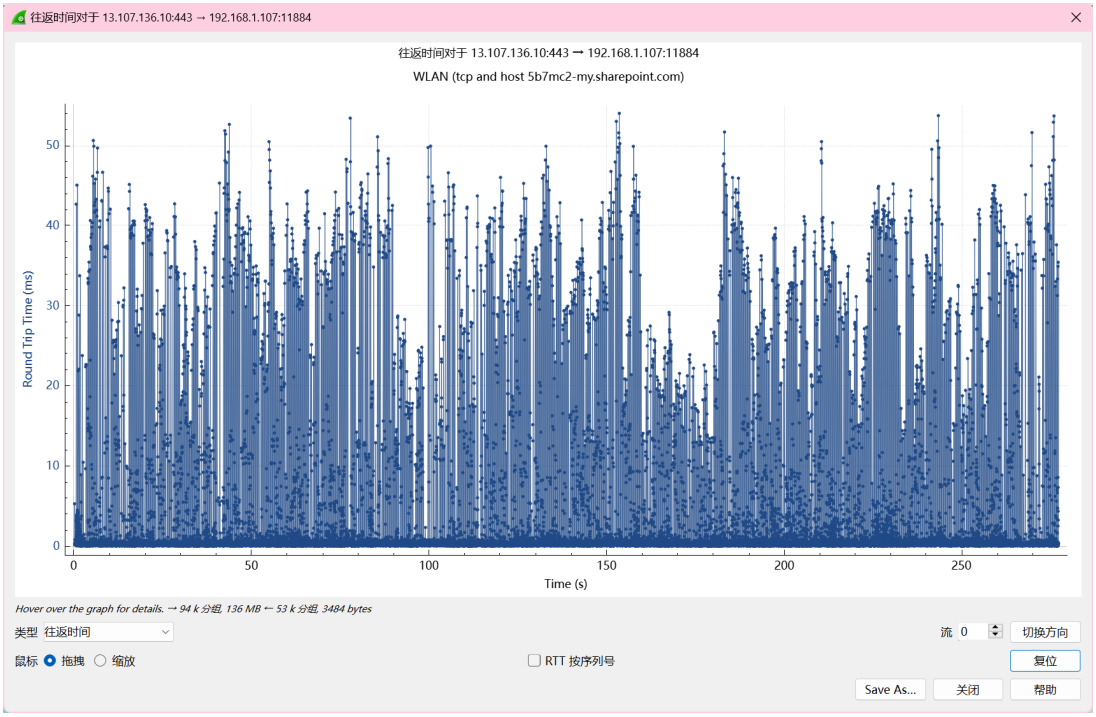


图 14: 往返时间

回答下列问题：

1. 探索 TCP 的拥塞控制和经典 AIMD 策略。

   答：从吞吐量图表中可以看出，吞吐量随着时间的增加而增加，但是在某些时刻会突然下降，不断波动，这是因为在这些时刻发生了拥塞，TCP 会减小拥塞窗口，然后再慢慢增加，这就是拥塞控制的过程。

2. 更深入地探索 TCP 的可靠性机制。捕获包括段丢失的 TCP 连接，查看什么触发重新传输以及何时触发，另外查看往返时间估算工具。

   答：可以看到，当接受到的数据包顺序错误时（可能是之前的数据包丢失了），会触发重传机制，重传丢失的数据包。

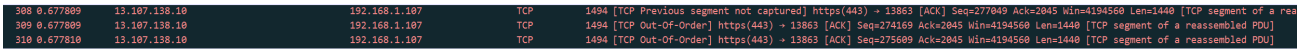


图 15: 错误重传

   往返时间图表已在上方列出。

3. 查看包括 SACK 在内的选项的使用以了解详细信息。

   观察一个 Options 中包含了 SACK 的数据包，如下图所示：

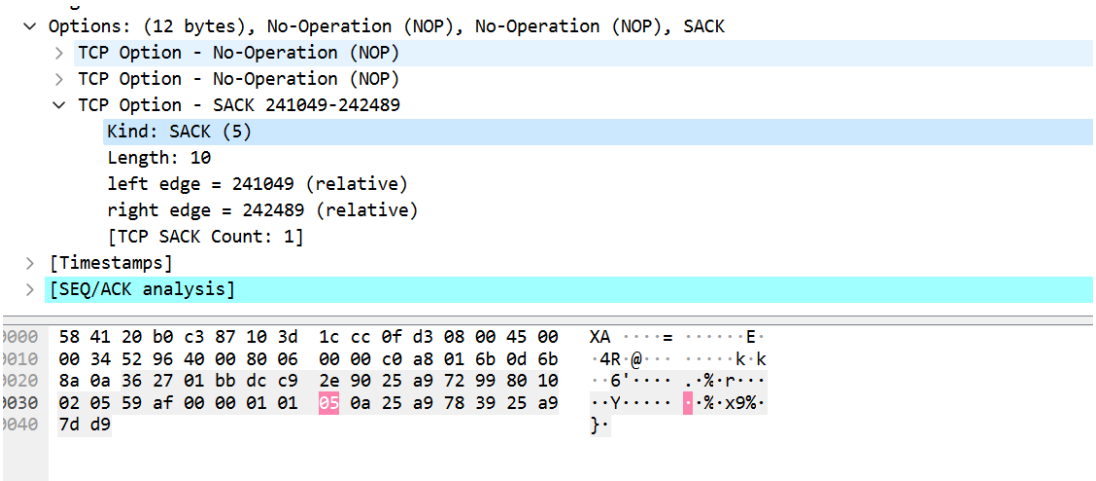


图 16: SACK

   可以看到，SACK 中包含了丢失的数据包的序列号范围。

4. TCP 是 Web 的基础传输层。可以通过设置并发连接来查看浏览器如何使用 TCP。

   答：浏览器处理用户请求时，会建立多个 TCP 连接，并发地处理用户请求。

5 实验结果总结

本次计算机网络实验围绕 TCP 展开，让我对其有了深入且全面的理解。

在实验过程中，我运用 Wireshark 和 wget 工具，成功捕获并分析了 TCP 报文，清晰绘制出 TCP 报文段结构，明确各字段含义，这使我对 TCP 数据包的构成有了透彻认知。通过观察 TCP 连接建立的三次握手和释放的四次挥手过程，我不仅绘制出详细时序图，还深入了解到每个步骤中数据包的变化及作用，尤其是 SYN 数据包携带的多种重要选项，如 MSS、Window Scale Option 等，它们在优化连接性能方面起着关键作用。

在数据传输阶段，我学会从 Wireshark 的 IO 图表分析下载和上传速率，了解到 TCP 有效负载在下载率中的占比情况，还掌握了根据服务器接收的序列号确定下一个 ACK 值的方法。此外，通过进一步探索 TCP 协议，我对其拥塞控制的 AIMD 策略、可靠性机制（如段丢失触发重传）以及 SACK 选项的使用有了直观认识，并且明白了浏览器在处理 Web 请求时如何利用 TCP 建立并发连接。

此次实验让我深刻认识到 TCP 在网络通信中的核心地位和重要性，它确保了数据的可靠传输与高效交互。

6 附录

无