华东师范大学软件学院实验报告

实验课程：计算机网络实践	年级：2023 级本科	实验成绩：
实验名称：Lab 6 TCP	姓名：张梓卫	
实验编号：（6）	学号：10235101526	实验日期：2024/12/27
指导老师：刘献忠	组号：	实验时间：2 课时

目录

一 实验目的

该实验是课程《计算机网络实践》的第六次实验，全名《TCP》，目标如下：

- 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

二 实验内容与实验步骤

1 实验内容

- Step 1: Capture a Trace, 使用 wget 确认 URL 有效

- Step 2: Inspect the Trace, 捕获并检查获得的包
- Step 3: TCP Segment Structure, 分析 TCP 段结构
- Step 4: TCP Connection Setup and Teardown, 分析 TCP 连接建立和释放过程
- Step 5: TCP Data Transfer, 分析 TCP 数据传输过程

2 实验步骤

- 以 http://old.ecnu.edu.cn/site/xiaoli/2017.jpg为例，用 wget 确认该链接有效。或者 wget www.baidu.com
- 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 port 80，我们主要观察客户端与服务器之间的 tcp 流
- 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

三 实验环境

使用 Wireshark v4.2.5, Windows 11 Pro, Wget Tools 进行实验。
实验报告使用 LaTeX 进行撰写，使用 Vim 编辑器进行文本编辑。

四 实验过程与分析

1 Step 1: Capture a Trace

1 .1 配置 Wireshark

启动 Wireshark，按照实验要求将捕获选项设置为：已连接的以太网，设置捕获过滤器为 tcp port 80，将混杂模式设为关闭，勾选 enable network name resolution。

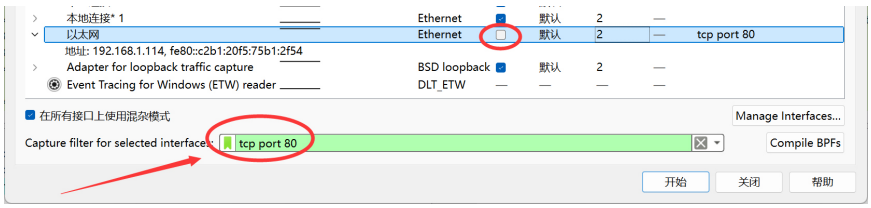


图 1: Wireshark Filter

设置 rename 选项为开启：

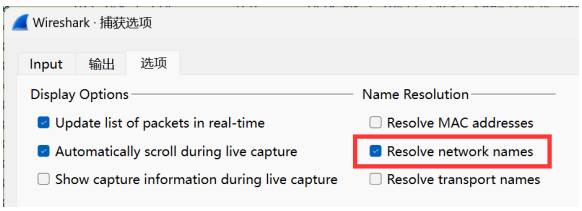


图 2: Wireshark Rename

1 .2 使用 Wget 确认 Url 有效

在命令行输入 `wget www.baidu.com`。

```
26421 ~ master ≠ ?87 wget www.baidu.com
SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = D:\GnuWin32/etc/wgetrc
--2024-12-18 16:45:37--  http://www.baidu.com/
正在解析主机 www.baidu.com... 180.101.50.188, 180.101.50.242
Connecting to www.baidu.com|180.101.50.188|:80... 已连接。
已发出 HTTP 请求，正在等待回应 ... 200 OK
长度: 2381 (2.3K) [text/html]
Saving to: `index.html.2'

100%[======================================================================================>] 2,381

2024-12-18 16:45:37 (51.7 MB/s) - `index.html.2' saved [2381/2381]
```

图 3: Wget

1 .3 在 Wireshark 中查看 Trace

回到 Wireshark 中点击停止捕获，此时可以看到和刚刚通过 Wget 命令获取的 IP （180.101.50.188）地址相关的 TCP 包。

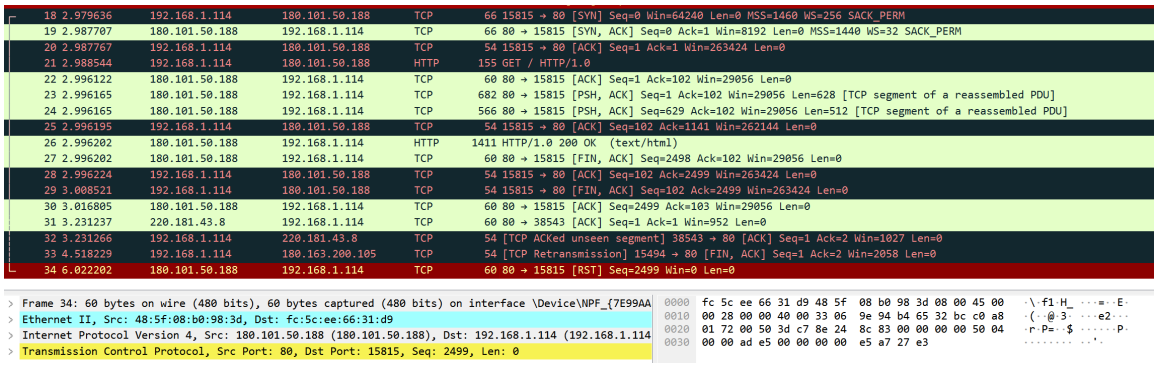


图 4: Wireshark Capture

可以发现很多红了，查阅资料发现，捕获到的 TCP 包和 IP 包中的校验和 (Checksum) 都显示错误的原因很可能是由于硬件校验和卸载 (Checksum Offloading) 功能导致的。这并不表示网络通信真的存在问题，而是因为抓包工具捕获到的数据包在经过网卡时尚未完成校验和计算。

网络适配器（网卡）通常具有硬件加速功能，它可以在发送数据包时自动计算校验和，而不是由操作系统完成。这种优化称为 Checksum Offloading。

2 Step 2: Inspect the Trace

成功获取 TCP 包后，我们可以对其进行分析，在 Wireshark 中点击帧，查看详细的具体信息，整合后的结果如下：

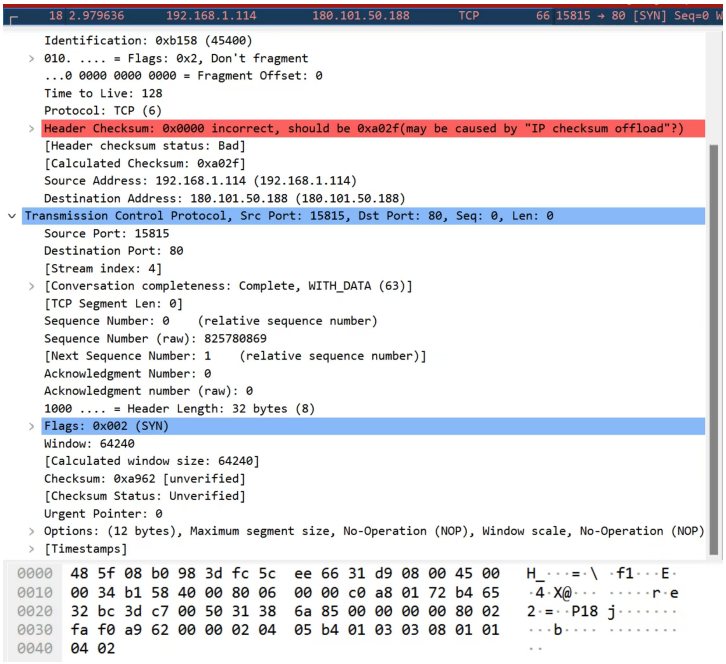


图 5: TCP Segment

可以看到在一个 TCP 数据包中有以下的信息：

- Source Port 源端口
- Destination Port 目标端口
- Sequence Number 序号
- Acknowledgment Number 确认号
- Header Length 头长度
- Flags 标记位
- Options 选项位

其中 Flag 字段每一位标记位都有其具体的含义，具体如下所示：

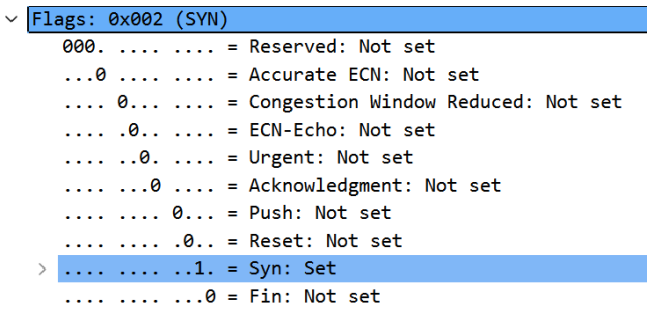


图 6: TCP Flags

3 Step 3: TCP Segment Structure

依次点击每一个字段，查看 TCP 协议中，每一个字段的长度分别是多少：

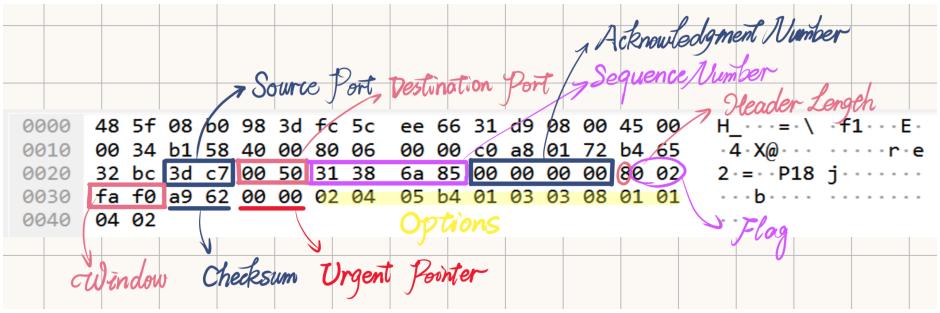


图 7: TCP Field Length

按照实验手册中的要求：

> *Do not break down the Flags field, or any Options field, and if you find that some TCP fields share a byte then group them.*

我们在图中不分解 Flag 字段和 Options 字段，故绘制成图表大致应如下所示：

Source Port = 2 bytes	**Destination Port = 2 bytes**
Sequence Number = 4 bytes	
Acknowledgement Number = 4 bytes	**Header Length = 0.5 bytes**
Flags = 1.5 bytes	**Window Size = 2 bytes**
Checksum = 2 bytes	**Urgent Pointer = 2 bytes**
Options = Varied Length	**Padding = Varied Length**

表 1: TCP Header Fields

4 Step 4: TCP Connection Setup and Teardown

在过滤器中可以输入 `tcp.flags.syn==1` 来找到第一帧，即 SYN 包。

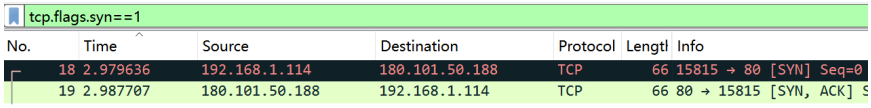


图 8: TCP SYN

4 .1 TCP Three-Way Handshake

TCP 协议的连接建立过程，由三次握手完成，即客户端发送 SYN 包，服务器发送 SYN+ACK 包，客户端再发送 ACK 包。

我们先在 Wireshark 中找到 TCP 的三次握手包，再分别点击进入其字段，寻找 Seq、Ack、Win、Len 字段等的大小。

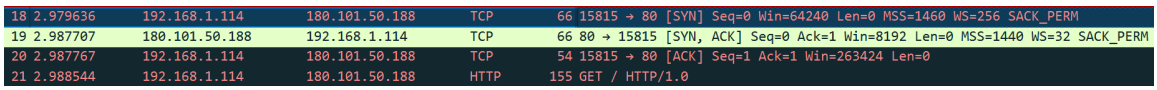
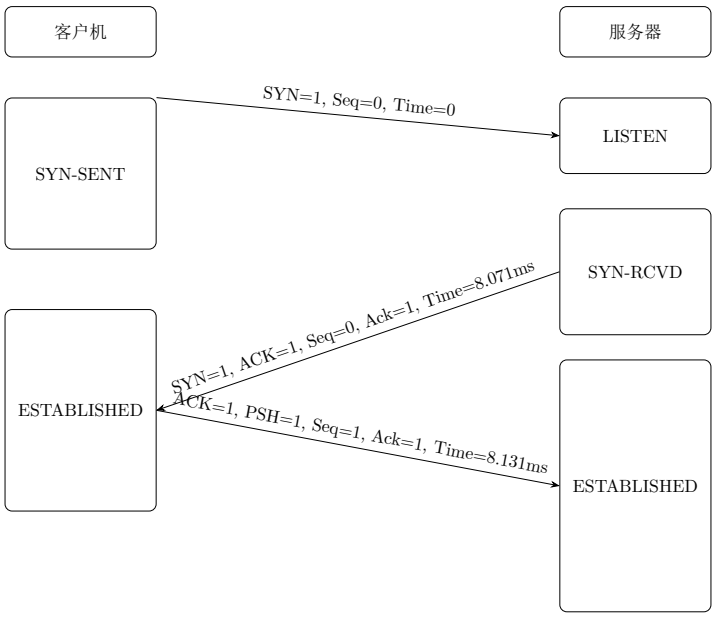


图 9: TCP Three-Way Handshake

由此，我们将客户机放在左侧，服务器放置于右侧，可以绘制出 TCP 三次握手的流程图：

其中，第一次发送 SYN 时，$Time = 2.979636$，到了 [SYN, ACK] 的包时 $Time = 2.987707$，到 [ACK] 包时，时间是 $Time = 2.987767$。通过简单的减法就可以得到，往返时间 $= 0.008131s = 8.131ms$。



Question

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

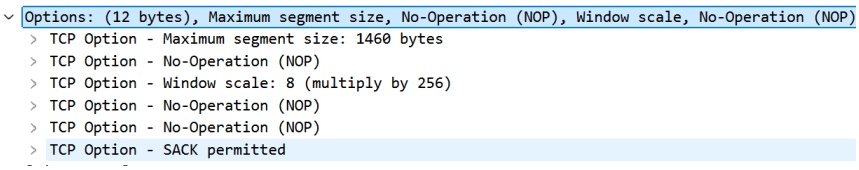


图 10: TCP SYN Options

解答 四 .1: Options

根据 Wireshark 抓包获得的 Options 字段
它含有 Maximum segment size, No-Operation, Window scale, SACK permitted.

- Maximum segment size: 指定最大段的大小
- No-Operation: 无功能（可以用于字节填充）
- Windows scale: 指定了窗口乘以的倍数，用于解决窗口大小的限制
- SACK permitted: 允许 SACK 选择性确认（乱序传输）

查阅资料得知，还有一些 SYN 包中会包含 End of Option List, Timestamp (记录时间信息) 等字段。
在这个包中，Wireshark 将 Timestamps 放置于 Options 之外，且用中括号标记，点开查看 first frame 和 previous frame 都为 0.

4 .2 TCP Four-Way Handshake

TCP 协议的连接释放过程，由四次握手完成，即客户端发送 FIN 包，服务器发送 ACK 包，服务器发送 FIN+ACK 包，客户端再发送 ACK 包。

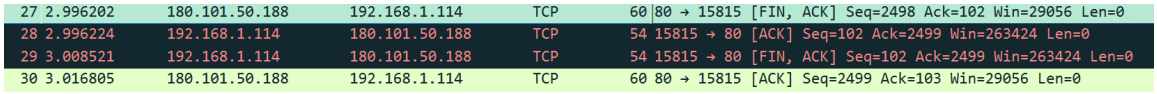
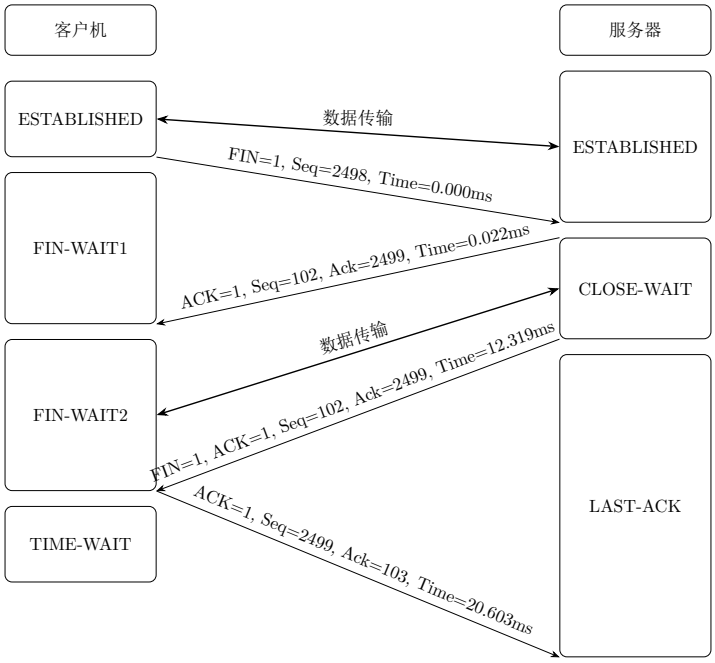


图 11: TCP Four-Way Handshake

主动关闭方（180.101.50.188）发送 FIN，请求关闭连接，同时带上 ACK 确认之前的数据包。这种行为在**第一次挥手**时很常见。

根据 Wireshark 抓包获得的 Seq、Ack 字段，我们可以绘制出 TCP 四次挥手的流程图：

注意，此处的 [Time] 均为时间戳。



Question

总结：为什么建立连接时使用三次握手，而释放连接时四次挥手？

TCP 是全双工通信，双方的关闭是独立的。

在建立连接时，如果是两次握手，可能会导致“已失效的连接请求报文段”的问题。例如，客户端发出第一个 SYN 后，若网络中断，客户端超时重试，但服务器后续接收到这个过时的 SYN，并建立错误的连接，可能造成资源浪费或通信失败。

而在释放连接时，客户端和服务器都需要各自发送 FIN 和确认 ACK，因此需要四次报文。当客户端发送 FIN 时，仅表示“客户端完成了数据发送”，而服务器可能还在发送数据，因此服务器不能立即发送 FIN，必须等数据发送完成后才能关闭连接。如果是三次挥手，可能导致服务器未发送完数据就关闭连接，丢失重要数据。

5 Step 5: TCP Data Transfer

5 .1 获取统计信息

在”Static” 统计菜单下，选择”IO 图表”，以查看数据包的传输速率。

按照实验手册的说明，将间隔改为 100ms，将 Y 轴的坐标改为 Bits，

- 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；

此时获得的图像如下图所示：

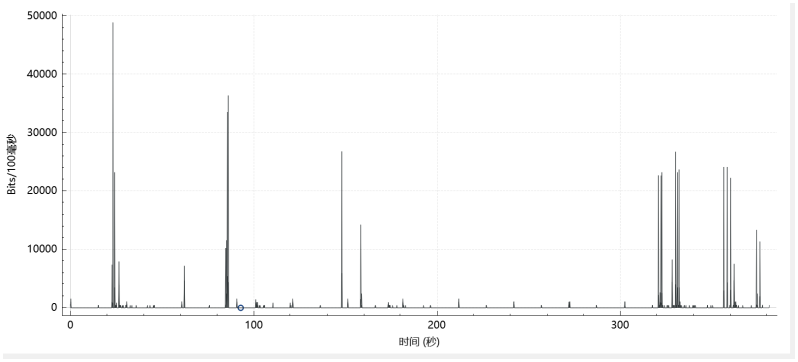


图 12: TCP Data Transfer

5 .2 调整图像显示

此时图像较为不平稳，故我找了另一个大概在 100 MB 左右的文件，用于测试：

```
26421 ~ master ≠ ?101 wget http://speedtest.london.linode.com/100MB-london.bin in cmd at 11:30:08
SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = D:\GnuWin32/etc/wgetrc
--2024-12-19 11:30:33--  http://speedtest.london.linode.com/100MB-london.bin
正在解析主机 speedtest.london.linode.com... 176.58.107.39
Connecting to speedtest.london.linode.com|176.58.107.39|:80... 已连接。
已发出 HTTP 请求，正在等待回应 ... 200 OK
长度: 104857600 (100M) [application/octet-stream]
Saving to: `100MB-london.bin.1'

100%[=======================================================================================>] 104,857,600 11.5M/s   in 10s

2024-12-19 11:30:44 (9.67 MB/s) - `100MB-london.bin.1' saved [104857600/104857600]
```

图 13: Get 100 MB File

得到的结果较为可观了，接下来我们开始分析。

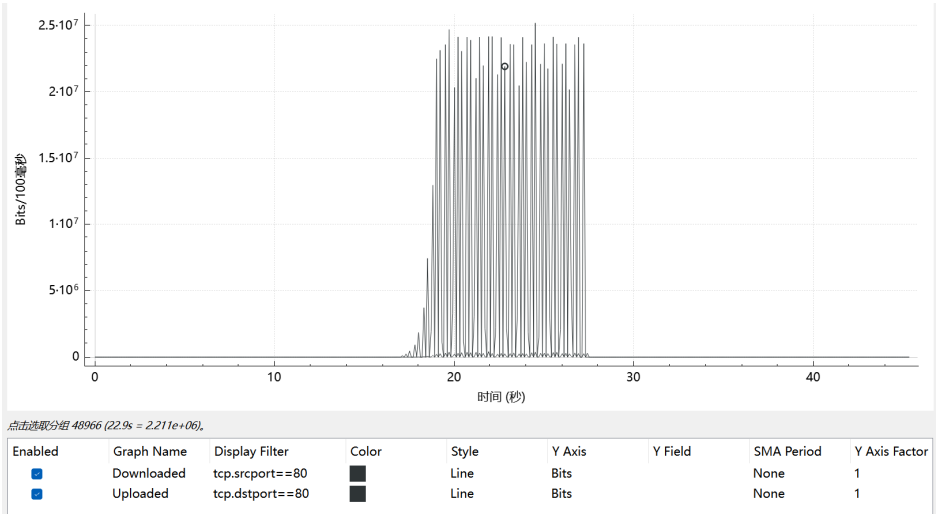


图 14: TCP Data Transfer

5 .3 回答问题

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? 实验中的下载速率是多少？Bits/s & packets/s

使用快捷键 X、Shift + Y 增长 X 轴，缩短 Y 轴，可以得到调整后的图像。

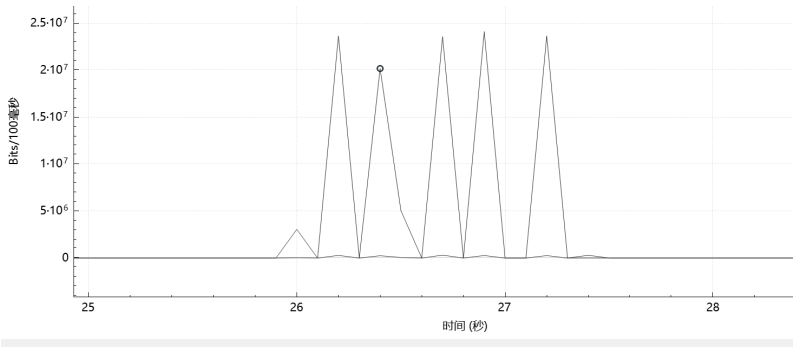


图 15: TCP Rough data rate

此时可以轻松看出，其速率的平均值大致在 $1.5 \times 10^7 bits/0.1s = 1.5 \times 10^8 bits/s$, 最大值达到了 $2.4 \times 10^8 bits/s$。

Question

What is the rough data rate in the upload direction in packets/second and bits/second due to the ACK packets? 实验中的上传速率，即 ACK 消息的发送速率是多少？

重新调整图表，将 Downloaded 方向取消显示。

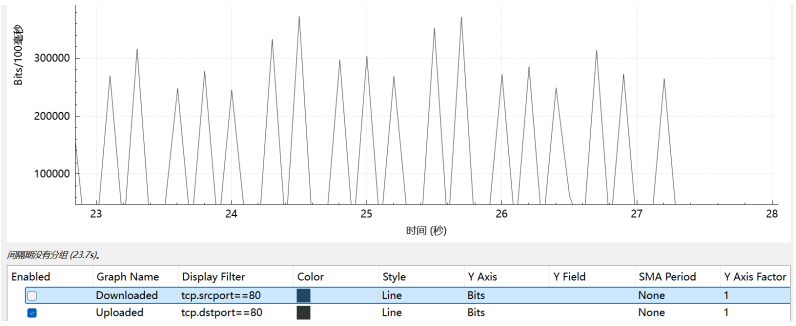


图 16: ACK Rough data rate

观察可得，平均值约为 $2.0 \times 10^6 bits/s$, 最大值达到了 $3.8 \times 10^6 bits/s$。

Question

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.

选中一个数据包观察：

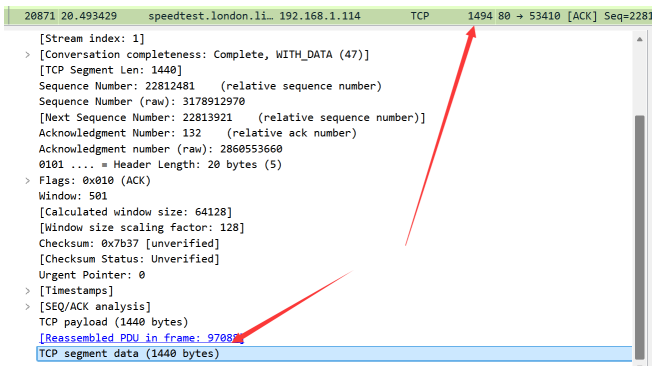


图 17: TCP Packet

可以看到其总大小为 1494 字节，其中 TCP 负载部分为 1440 字节，占比约为 96.38%.

Question

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? 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？

Ack 号告诉下一个期望的序列号，因此它将是 X 加上这一帧的 TCP 载荷的大小。

6 Explore on your own

6 .1 TCP 拥塞与 AIMD 策略

TCP 拥塞控制使用了 AIMD（加性增大、乘性减小）策略：

- 加性增大：每个 RTT 成功确认后，拥塞窗口（Congestion Window，CWND）按线性增加。
- 乘性减小：检测到丢包或网络拥塞时，CWND 会立即减半。

拥塞控制的四个阶段：

- 慢启动（Slow Start）
- 拥塞避免（Congestion Avoidance）
- 快速重传（Fast Retransmit）
- 快速恢复（Fast Recovery）

6 .2 TCP 的可靠性机制

TCP 通过以下机制实现可靠性：

- 序列号（Sequence Number）和确认号（Acknowledgment Number）。
- 超时重传（Timeout Retransmission）。
- 快速重传（Fast Retransmit）。
- 滑动窗口协议。
- RTT 估算（Round Trip Time）：动态调整超时时间。

五 实验结果总结

1 TCP 报文结构的理解

在实验中，我通过 Wireshark 对捕获的 TCP 数据包进行了详细分析，明确了 TCP 报文中的各个字段及其功能。

- 了解了 TCP 报文头部各字段的长度和作用，例如源端口、目的端口、序列号（Sequence Number）、确认号（Acknowledgment Number）、标志位（Flags）等。
- 掌握了 TCP 报文中选项字段的具体内容，包括 MSS、窗口缩放因子（Window Scale）、选择性确认（SACK）等，并理解了它们对数据传输效率的影响。

2 TCP 三次握手与四次挥手

通过实验详细分析了 TCP 的连接建立（Three-Way Handshake）和释放（Four-Way Handshake）过程，明确了：

三次握手的具体步骤和目的，了解了 SYN、SYN+ACK 和 ACK 数据包的字段变化。四次挥手中，客户端和服务器独立关闭连接的过程，解释了为什么需要四次挥手而不是三次。

3 TCP 数据传输过程

在实验中，利用 Wireshark 捕获和分析了 TCP 数据传输的过程，并从以下几个方面得到了结果：

- 计算了 TCP 数据包的传输速率，包括下载方向的速率（bits/s 和 packets/s）以及 ACK 报文的上传速率。
- 分析了 TCP 数据包的负载比例，得出数据包的有效负载占比达 96.38%，进一步理解了 TCP 报文的资源利用效率。

六 附录

参考资料

- 常用测速文件：https://b.lxd.cc/post-323.html