TCP 数据段捕获与分析报告

李伟

1711350 计算机科学与技术一班

更新：*November 24, 2019*

摘　　要

本实验通过 WireShark 捕捉实时网络数据包，并根据网络协议分析流程分析 TCP 的连接建立、数据传输、连接关闭的全过程，并对 TCP 的重要数据段进行分析，深入理解 TCP 的传输机制，同时在此实验中对 SYN，FIN，ACK，RST，PUSH 等 TCP 字段的作用进行了详细分析，此外在实验中对 TCP keep-alive 机制和快速关闭连接的机制进行了初步探索。

关键词： TCP，WireShark，数据包捕获，数据包分析

1　实验要求

通过 HTTP 访问某个网页，使用 Wireshark 对整个过程中的数据段进行捕获，分析 TCP 连接建立、数据传输、连接关闭的全过程，至少对其中 5 个典型的 TCP 数据段进行详细分析，给出界面截图，并同时提交捕获文件。

2　实验环境

- 操作系统：windows10 专业版
- 捕获工具：WireShark

3　TCP 数据传输机制分析

TCP 是一种面向连接的、可靠的、基于字节流的传输层通信协议，由 IETF 的 RFC 793 定义。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能。用户数据报协议（UDP）是同一层内另一个重要的传输协议。相较于 UDP，TCP 的强大优势在于可靠性，TCP 为上层的应用层提供了超时重传、差错检验、流量控制、拥塞控制等可靠机制，能够最大程度上保证数据端到端的可靠性。

TCP 可靠性的基础很大程度上来自于 TCP 面向连接并维护连接状态实现的，因此了解和分析 TCP 建立连接以及维护连接状态的机制是十分重要的，本次实验的的主要目的就是重点分析 TCP 面向连接的这种机制。

TCP 协议中数据的传输过程主要可以看做三个部分，分别是三次握手建立连接，数据的可靠传输，四次挥手拆除连接（部分情况下使用 RESET 快速拆除连接），下面主要分析这三个部分的具体细节。

3.1 捕获数据包情况

打开 WireShark 之后点击相应的网卡设备即可针对该设备捕获数据包，默认情况下会捕获该网卡上的所有数据包，可以通过设置过滤器过滤掉非目标数据包，方便找到分析的数据包。

在本次分析中，使用的数据包保存在学院网站.pcapng文件中，数据包产生于访问学院网站的过程中,本次分析主要围绕两个 TCP 连接进行,第一条 TCP 连接的过滤器设置为ip.addr == 222.30.45.190 &&tcp.port==4
第二条 TCP 连接的过滤器为ip.addr == 54.223.158.33 && tcp.port==80，通过筛选之后可以分别看到**图 1**，**图 2**所示的数据包捕获界面。

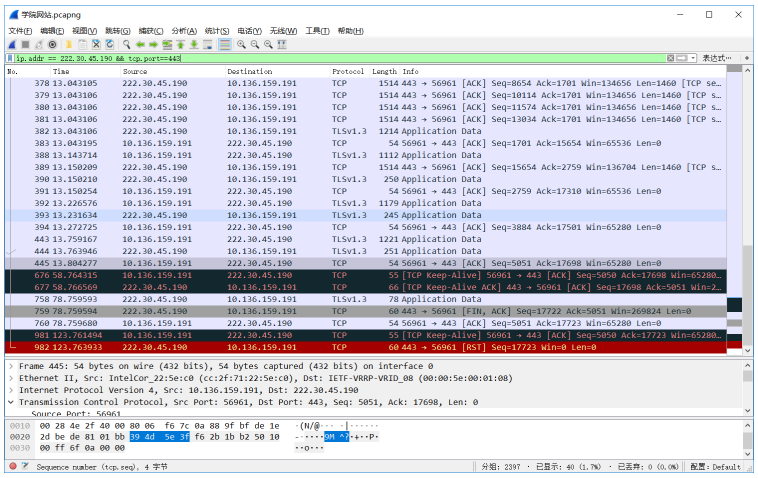

图 1: TCP 连接 1

其中由于 TCP 连接 1 中出现的关于 TCP 相关的特殊情况较多，所以在下面的分析主要以 TCP 连接 1 作为分析对象，但由于 TCP 连接 1 中的连接拆除部分使用的是 RESET 直接拆除，而不是普遍的四次挥手形式，所以在链接拆除部分四次挥手将以 TCP 连接 2 作为分析对象。

3.2 三次握手建立连接

三次握手的详细分析基于 TCP 连接 1 的数据包进行，三次握手的过程简单描述一下就是客户端首先发起一个连接请求，其中标志位置位 SYN 位，服务器端接收到连接请求之后会回送一个 ACK|SYN 报文确认接收到连接请求并同意建立连接，之后客户端回送一个 ACK 确认报文，确认连接建立并通知服务器端链路连接正常。三次握手的数据包捕获截图如**图 3**所示：

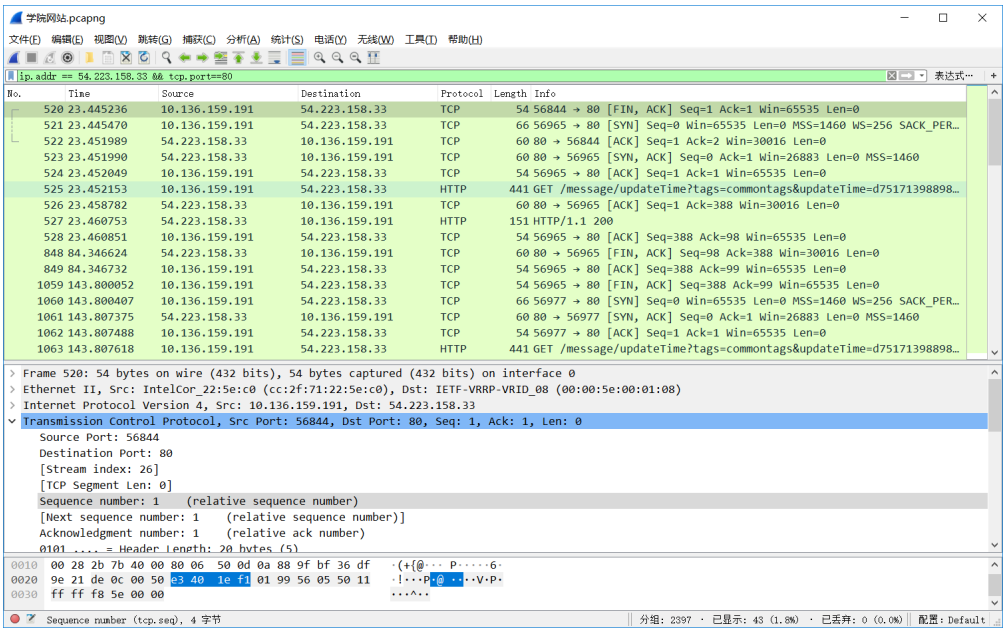

图 2: TCP 连接 2

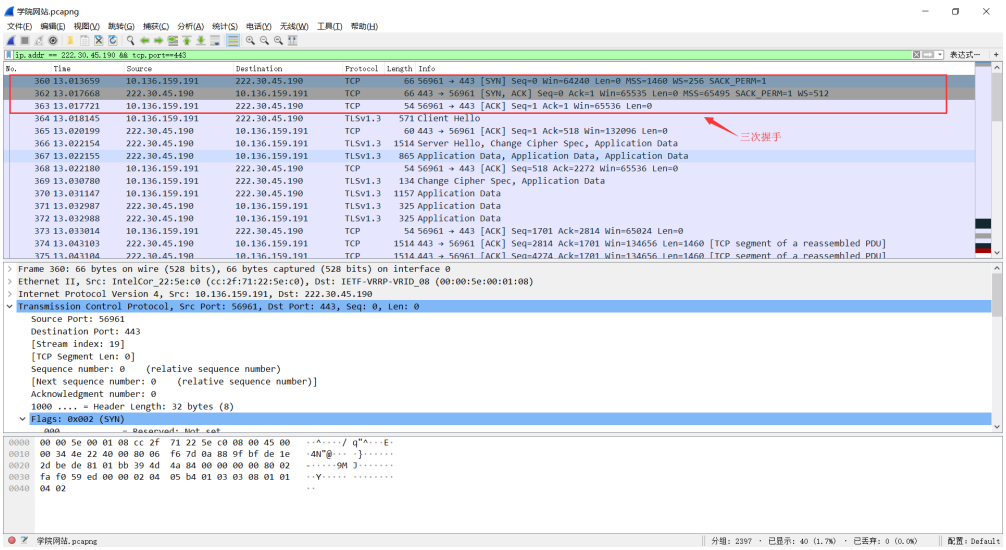

图 3: 三次握手建立连接数据包

之所以要经过三次握手才建立连接，目的在于充分保证 TCP 连接的可靠性，为了防止已经失效的连接请求报文突然又传送到了服务器，导致产生错误，因此单次握手建立连接不可靠，第一次握手确认了从 A 到 B 的发送可靠性，第二次握手确定了从 B 到 A 的发送可靠性以及 B 的接收可靠性，第三次握手确定了 A 的接收可靠性并在此确定了 TCP 连接的稳定性，所以通过三次连接能够很好地检验 TCP 连接的稳定性和可靠性。TCP 建立连接的三次握手示意图如**图 4**。

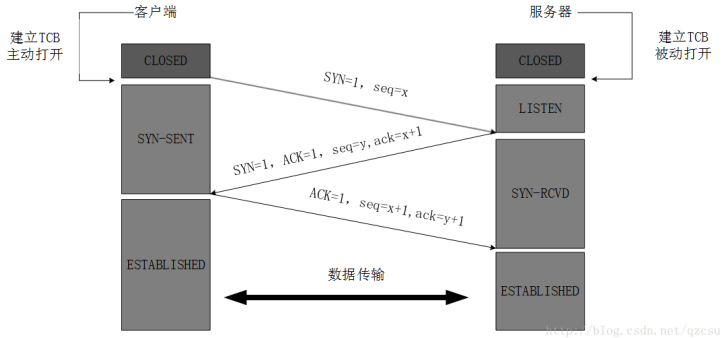

图 4: 三次握手建立连接示意图

3.2.1 第一次握手

第一次握手是客户端发送一条 SYN 标志的报文给服务端，数据包的具体信息如**图 5**所示：

在第一次握手过程中，客户端会首先发送一条不带数据体的 TCP 报文，并选择自己的发送序列号，在这里，选取了默认的 0，同时会将 flags 标志位中的 SYN 位置位为 1，说明发起连接的请求，同时在 TCP 头部中我们还可以看到整个 TCP 头部的长度为 32 个字节，一般来说 TCP 头部为 20 个字节，超过 20 字节说明包含选项。在**图 5**中我们还可以看到校验和位，以及选项 options 的设置，在这条连接请求中，包含了一个 MSS 的确定（MSS=1460），同时还设置了窗口的规模，即 window scale 设置，以及确定使用 SACK 选择确认的机制。

初次之外，可以看到在 WireShark 中能够看到时间戳的记录，这部分的作用在后面会说到。

3.2.2 第二次握手

第二次握手是服务器接收到客户端的连接请求后发送一条 SYN|SCK 置位的空数据报文给客户端，如**图 6**所示：

从截图中不难看出第二次握手中，服务器端也确定了自己的发送基础序号，这里仍然使用的是默认的 0，同时确认序号为 1（即第一次握手报文的发送序号加 1），同时也能够看到 Flags 中 SYN 与 SCK 为均置为 1，表明是对连接请求的回复，说明了服务器端同意建立连接。同样的，在条报文中，服务器也报告了自己的接收窗口大小，以及在选项中确定了自己的 window scale，MSS，明确了使用 SACK 的选择确认机制。

注意到，在最后，由于发送和接收数据包，所以客户端就能够通过这两次握手确定 RTT 的实际大小，从而估算 RTO，用来实现超时重传机制。

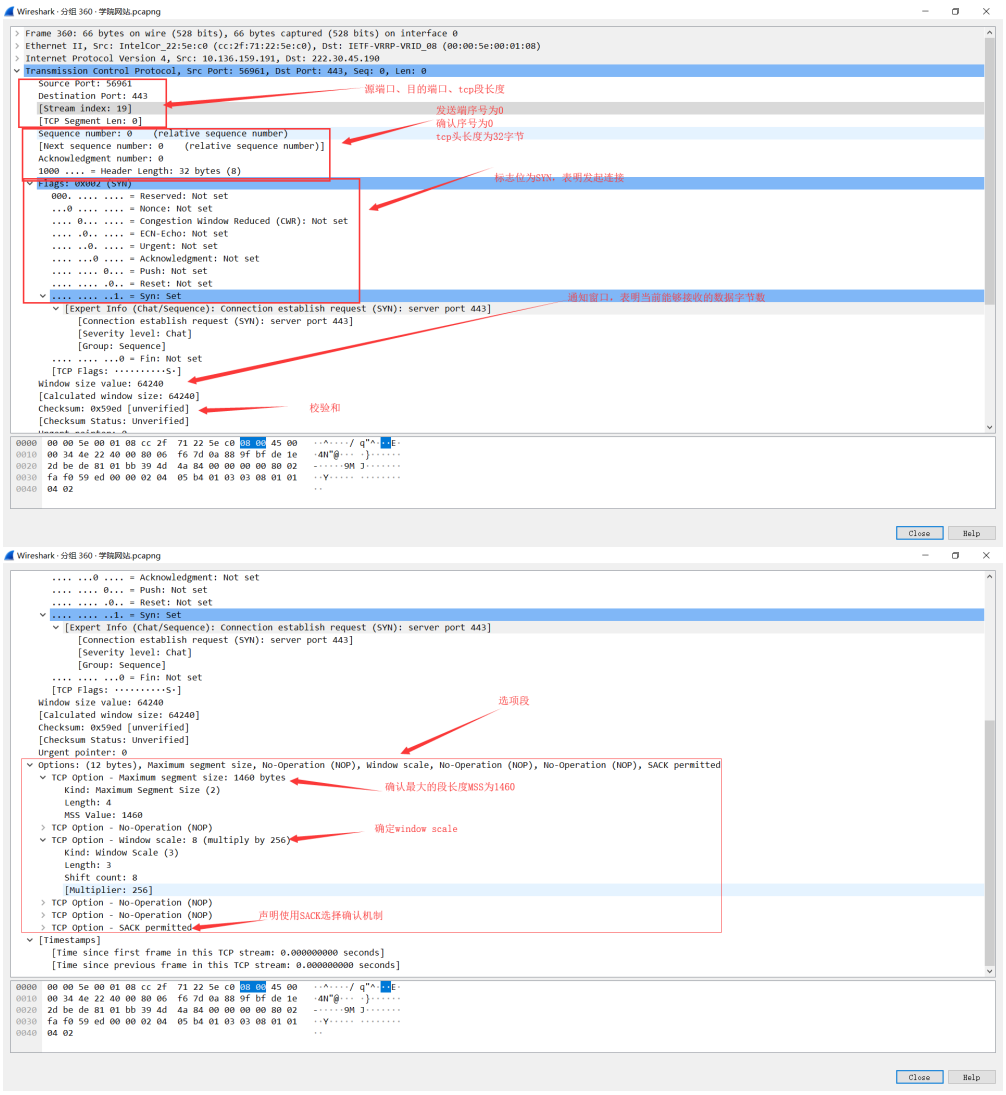

图 5: 第一次握手数据包分析截图

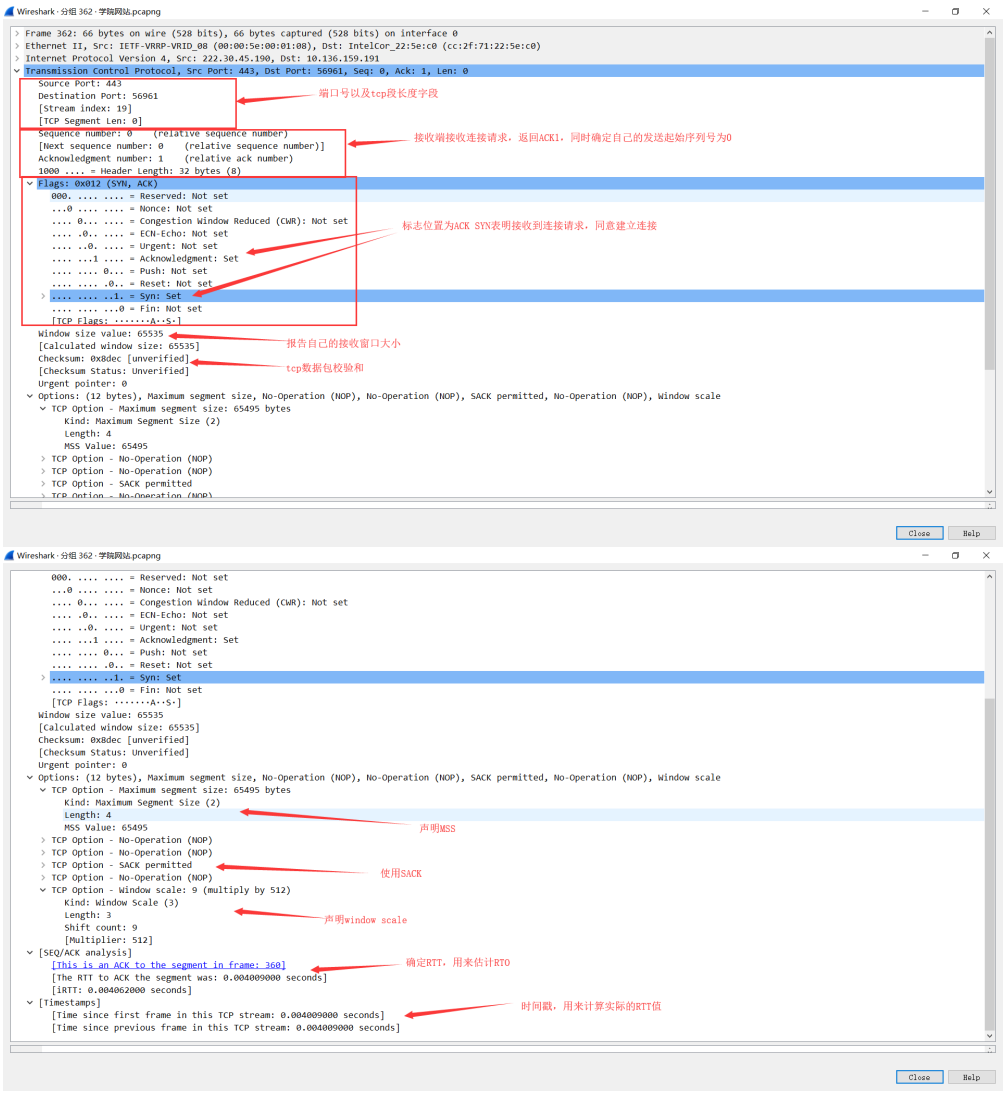

图 6: 第二次握手数据包分析截图

3.2.3 第三次握手

第三次的握手为客户端发送一条 ACK 给服务器，说明自己接收到了服务器的回复，数据包的解析如**图 7**：

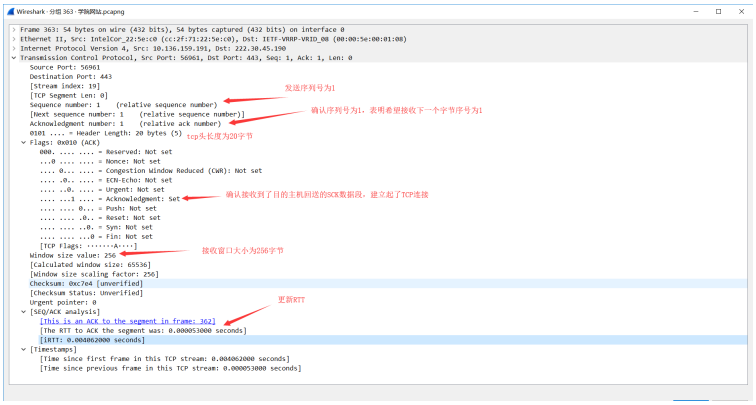

图 7: 第三次握手数据包分析截图

从报文中可以看到，回复的确认序列号为服务器发送序列号 +1，同时自己的发送序列号为第一次发送序列号 +1，Flags 中 ACK 置位为 1，说明为 ACK 回复报文。

特别的，在最后可以看到经过后两次握手之后，服务器端也获得了一个完整的收发过程，能够得到第一次收发的 RTT 数据，并估计下一次的 RTT 和 RTO，实现超时重传的机制。

3.3 数据传输过程

TCP 连接建立之后，客户端和服务器端通过彼此的收发数据包进行数据交互，严格按照序列号的顺序以及窗口大小的约束进行数据的发送与接收，在这里主要说明两种情况下的数据收发，用来体现型的 TCP 数据流情况, 如**图 8**。

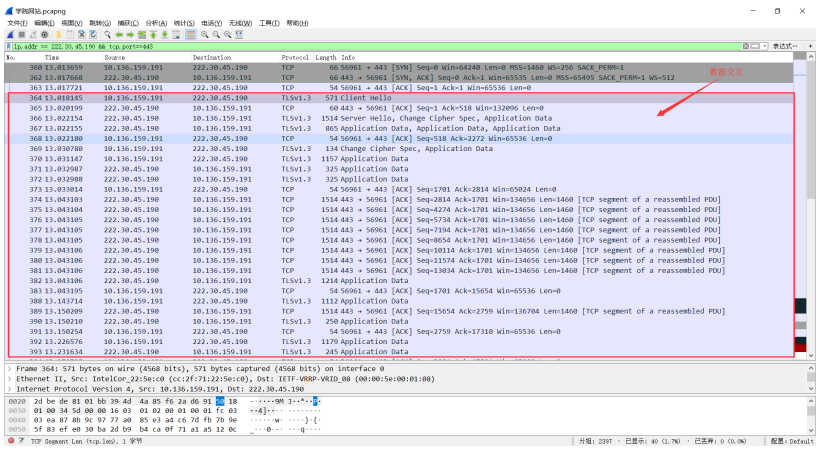

图 8: 数据传输过程

第一种为单个数据包的收发，即客户端发送单一的数据包，服务器端进行响应，这个过程不会

涉及数据包的乱序问题，比较简单，数据收发的过程如图 9、图 10所示：

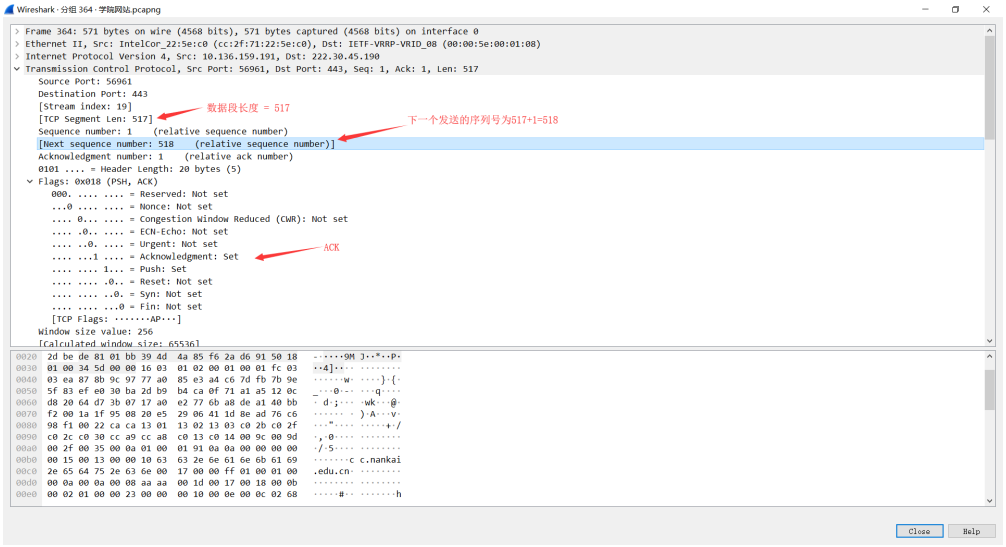

图 9: 客户端发送数据包

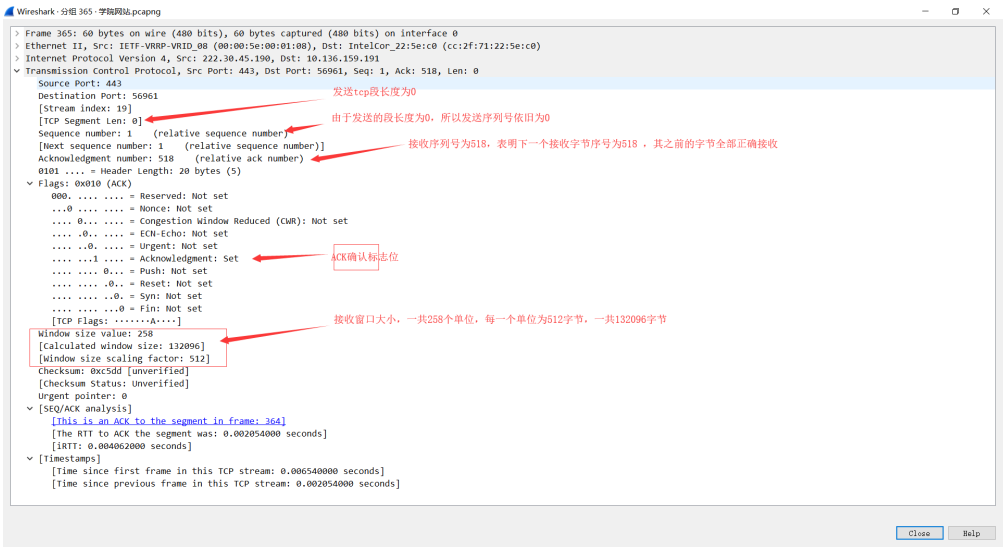

图 10: 服务器确认数据包

从图 9可以看到，数据段长度为 517 字节，发送序列号为 1，同时确认了第三次握手的服务器回复数据包，在图 10中可以看到回复数据包的数据段长度为 0，ack 序号为 518，为 517+1，由于长度为 0，所以发送序列号仍然为 1。

值得说明的是，这条数据交互基于TLS handshack protocol，TLS握手协议，TLS 握手协议在客户和服务器进程之间协商它们在安全信道中要使用的安全参数，这些参数包括要采用的协议版本、加密算法和密钥。另外客户要认证服务器，服务器则可以选择认证或不认证客户。握手协议工作过程如下：

- 客户端发送 Client Hello 报文给服务器端，服务器回答 Server Hello。
- 服务器端发送 Certificate 报文传送数字证书和公钥。

- 服务器端请求客户端证书时，客户端要返回证书或返回没有证书的指示。
- 服务器此时要返回改变加密规范协议报文和 Encrypted Handshake 报文，以示完整的握手消息交换已经全部完成。
- 握手协议完成后，客户端即可与服务器端传输应用加密数据，应用数据加密一般是用第 步密钥协商时确定的对称加密密钥，如 DES、3DE 等。非对称密钥一般为 RSA，用于数字证书的验证。

本次数据包即为客户端发送给服务器端的 Client Hello 报文，在此报文中设定了注入加密算法，协议版本、压缩算法之类的安全参数，截图如**图 11**所示：

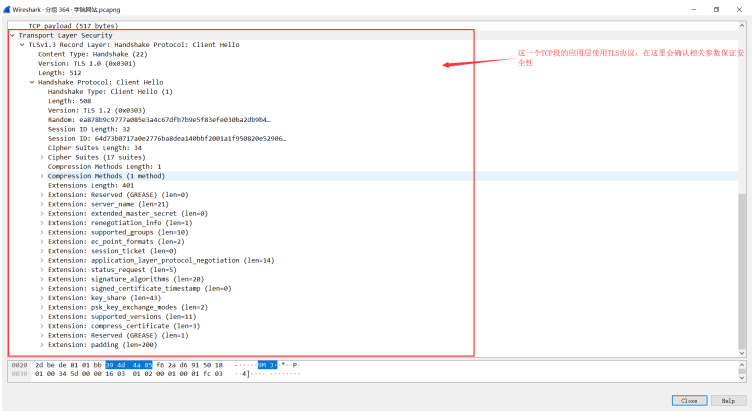

图 11: TLS 握手协议

第二种数据收发即为发送端的数据包连续发送，从**图 12**中可以看到，服务器端连续向客户端发送了多条数据包，这种情况一般出现在数据包大小太大，而发送的端最大长度限制为 1460 字节，所以需要分多个数据包进行发送。

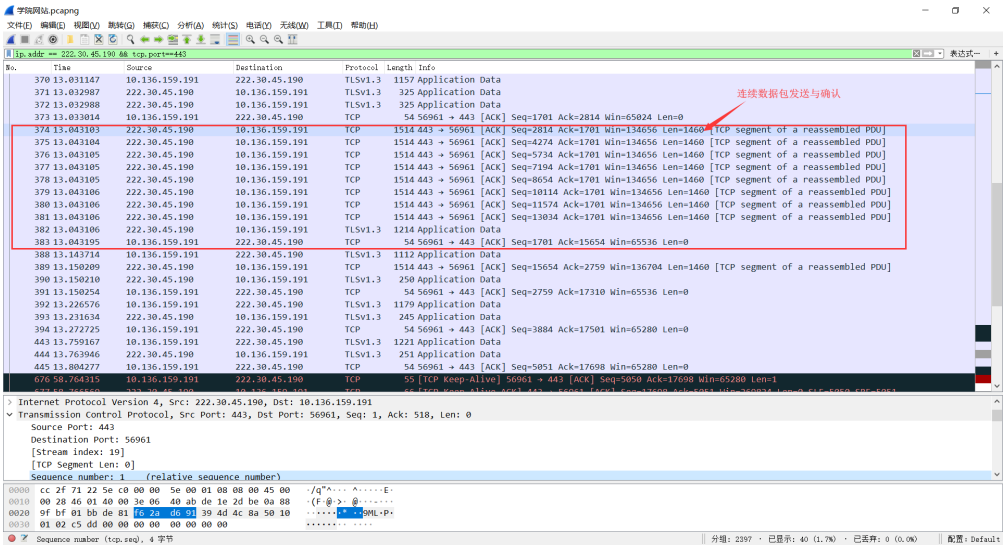

图 12: TLS 握手协议

我们可以看到在服务器端连续发送多个数据包之后，客户端只发送了一条 ACK 报文，这种情

况体现了 TCP 协议中的累计确认机制，我们关注连续发送的最后一条报文如图 13：

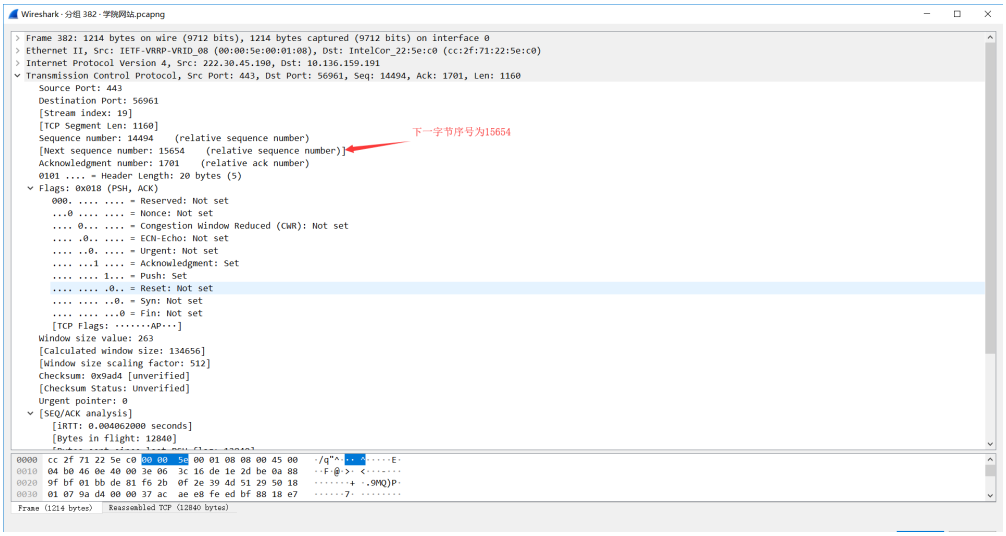

图 13: 服务器连续发送最后一个数据包

可以看到，客户只是在按序接受了所有连续发送的数据包之后才回复了一条 ACK 报文。

3.3.1 PUSH 字段作用

通常来说当接收放接收数据之后，在什么时间将缓冲区哪些数据包发往上层应用是由接收方自行决定的，但是当出现上述的将大文件拆分成多个文件的情况下，让接收方自行决定推送方式可能会导致接收方应用层一次接受的数据不完整，因此对发送方来说，会将分拆打包成 TCP 报文的多个数据包的最后一个 TCP 报文 flags 中的 PUSH 字段置为 1，要求接收方在接收发到包含 PUSH 字段的 TCP 报文时，立即将缓冲区中已经接收到的报文传给上层应用。

除此之外，如果应用层希望自己的数据能够立即发送出去而不是在发送缓冲区中等待，可以添加 PUSH 字段，这样发送端会优先发发送带有 PUSH 字段的报文，当然如果缓冲区满了，TCP 同样会将发送缓冲区中的所有数据打包发送。

在上述分析中的图 13中可以看到 PUSH 字段被置位为 1，原因即是因为该报文为打包发送的最后一个报文，为了快速发送以及在接收方能够尽快传给上层应用，使用了 PUSH 标志位。

3.3.2 TCP Keep-alive 机制

在本次的实验中出现了 TCP keep-alive 机制，这种机制出现在，当服务器客户端经过一定时间未通信之后，客户端会发送一条带有一个字节 0 数据的 keep-alive 报文区确认与服务器之间的连接是否还存在，具体报文可以看图 14:

服务器端会通过选择确认机制确认收到的这一个字节的数据，截图如图 15所示：

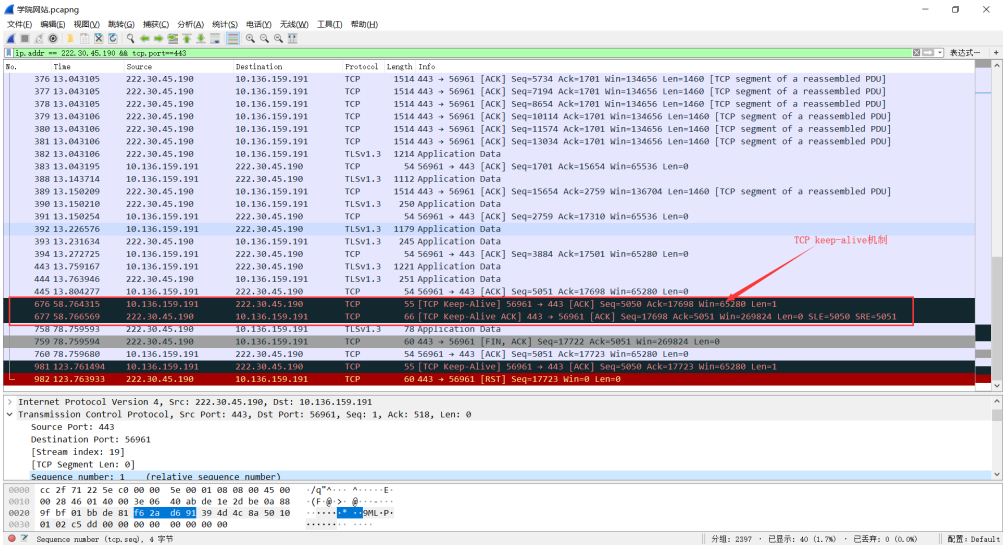

图 **14:** keep-alive 报文

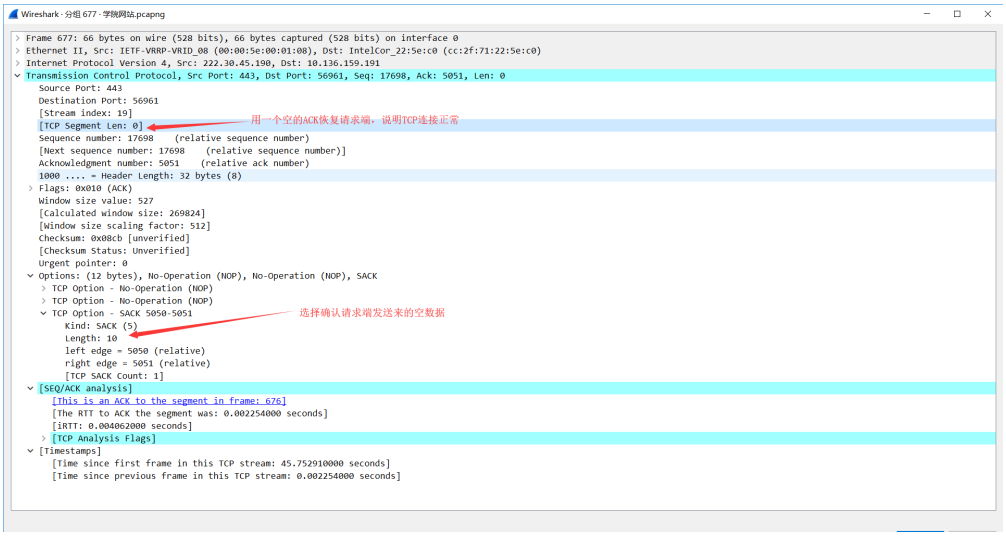

图 **15:** 选择确认机制

3.4 四次挥手拆除连接

数据收发结束之后或者长时间连接未用，则会发起连接关闭请求，关闭连接一共分为四个过程，称为四次挥手拆除连接。拆除的过程示意图如**图 16**所示：

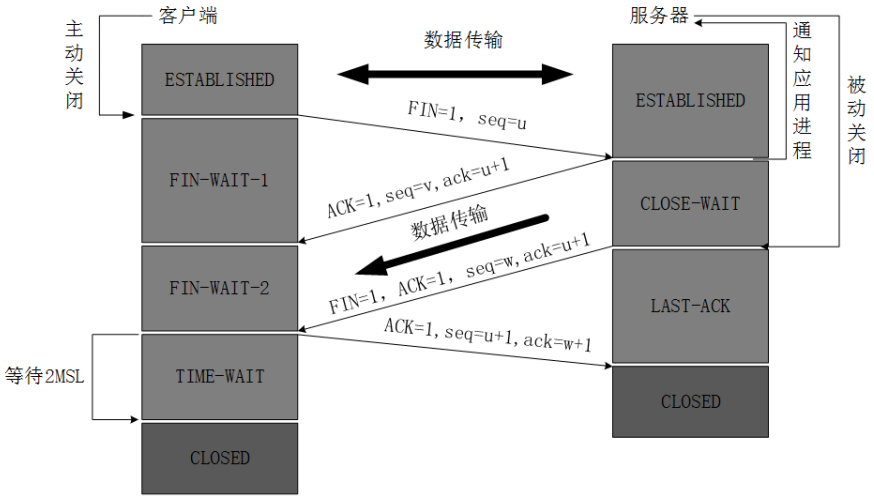

图 16: 四次挥手拆除连接示意图

四次挥手所分析的数据 TCP 连接为 TCP 连接 2，截图如**图 17**:

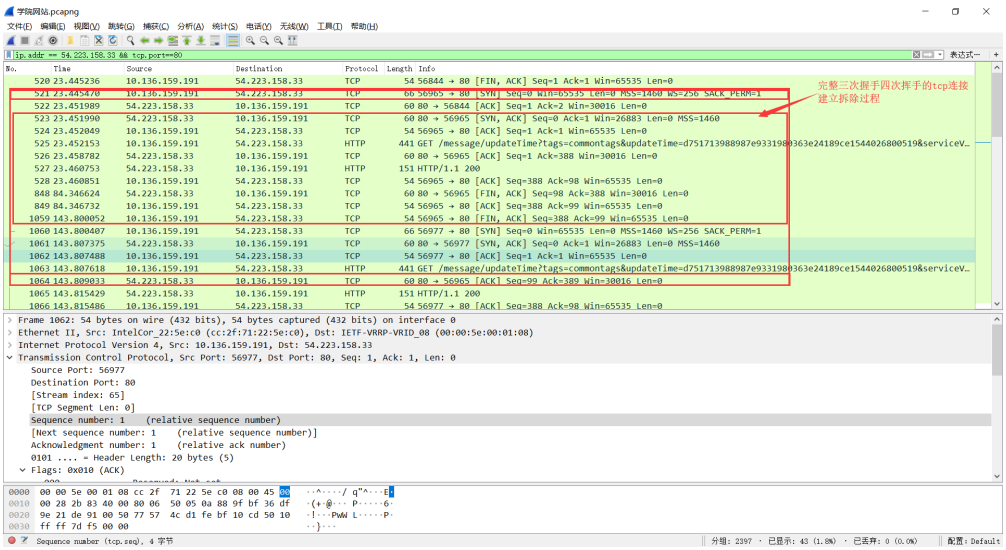

图 17: TCP 连接 2 完整的建立拆除连接截图

3.4.1 第一次挥手

第一次挥手为服务器端发现没有数据收发结束之后，为节省系统资源，会发起关闭连接的请求，报文的特征为 FIN 位置位为 1，发送完这条报文之后，服务器将不能继续发送数据报文，但是能够接收和发送 ACK 报文。从图 **17**可以看到该报文发送序列号为 98，确认序列号为 388。

3.4.2 第二次挥手

第二次挥手是客户端接收到服务器端发送的关闭连接请求，客户端回送一个 ACK 报文，说明接收到关闭请求，进入被动关闭等待状态，在此期间客户端仍然可以向服务器发送数据，这一点在 TCP 连接 2 的关闭连接过程中可以看出来。从图 **17**可以看到该报文发送序列号为 388，确认序列号为 99。

3.4.3 第三次挥手

第三次挥手为客户端继续发送一条 FIN 段，说明自己进入关闭等待状态，等待服务器端回送 ACK 进行关闭，在此期间客户端无法继续发送数据。从图 **17**可以看到该报文发送序列号为 388，确认序列号为 99，之所以序列号没有改变是由于这两条报文并没有携带数据段。

3.4.4 第四次挥手

当服务器接收到客户端的 ACK 报文之后，进入到关闭等待状态 2，等待接收客户端的 FIN 段。接收到客户端的 FIN 字段报文之后，回送一个 ACK 回复，从图 **17**可以看到该报文发送序列号为 99，确认序列号为 389。开始计时后经过 2 倍的 TCP 生存周期之后关闭 TCP 连接，同时在客户端接收到 ACK 回复之后立即关闭连接。

3.4.5 reset 快速拆除连接

在 TCP 连接 1 中，经过两次挥手之后，在服务器端接收到来自客户端对于关闭连接请求的 ACK 报文之后，直接发送了一条 RESET 命令快速拆除与客户端之间的连接，利用 RST 位进行复位的时候，不必等待缓冲区中的包都发出，而是可以直接发送 RST 段后关闭连接，而客户端接收到 RST 段后也不必回复 ACK，可直接关闭连接。一般来说 RST 段用来关闭异常连接，但是也有的情况会被用来快速关闭连接，减少关闭连接时的交互。

拆除 TCP 连接 1 的过程即数据包的结构截图如下所示：

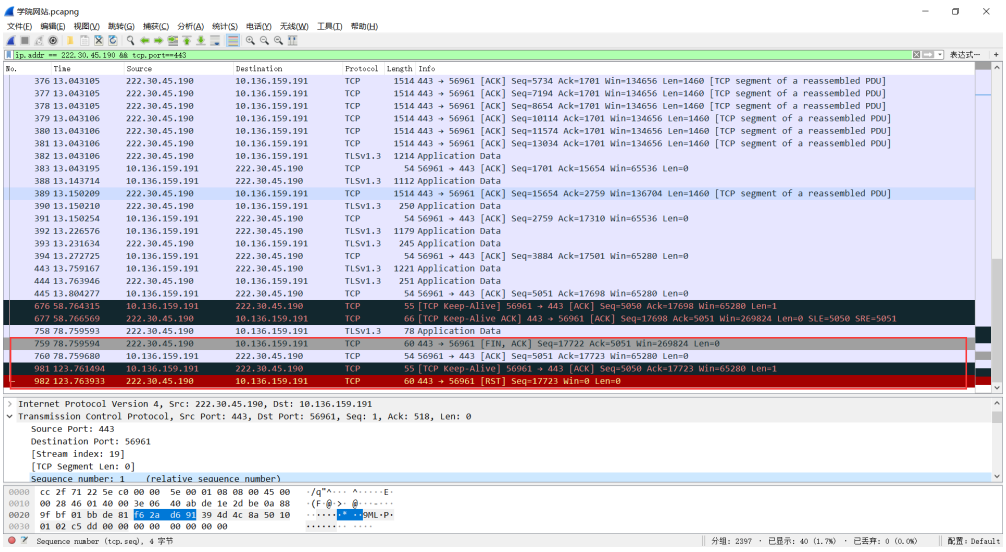

图 18: TCP 连接 1 拆除连接截图

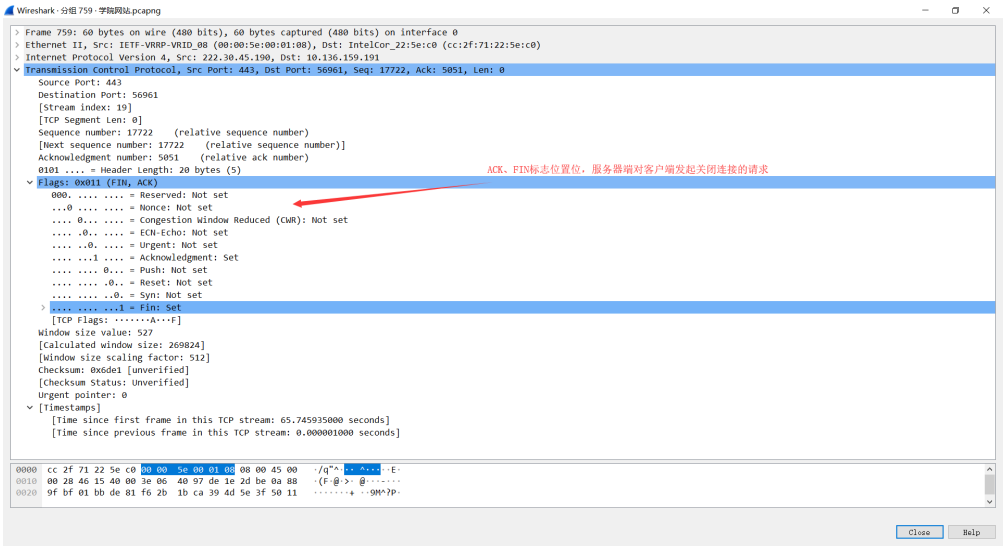

图 19: TCP 连接 1 服务器发出关闭连接报文结构

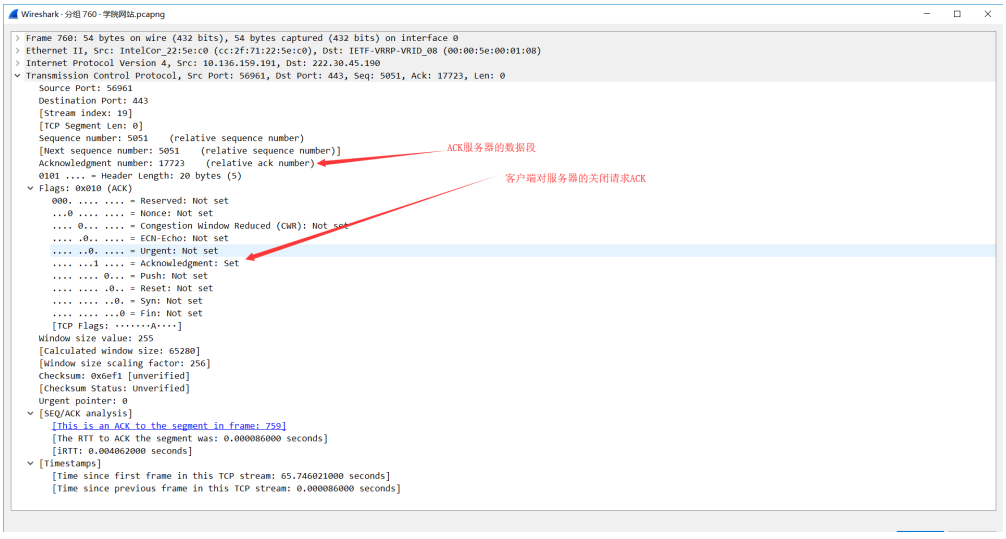

图 20: TCP 连接 1 客户端回复 ACK 报文结构

4 总结与思考

在本次 TCP 数据包分析中发现了在实际情况下，TCP 连接的建立和拆除以及数据传输过程和课堂上所学的还是存在一定的区别，比如在拆除连接的过程中，一般情况下课堂上所介绍的都是四次挥手过程拆除 TCP 连接，但是在这次分析的实际数据保重可以看出，也存在一些情况下使用 RST 段来快速关闭连接，此外在实际的数据传输过程中也有使用 push 字段来强制将报文立即发送的情况，以及使用 keep-alive 机制来保证连接的稳定性。

总而言之，通过这次对于 TCP 数据包的分析和 TCP 连接的建立拆除分析，我对于在课堂上所了解的 TCP 协议有了更深刻的认识，也对于 WireShark 这个工具使用的更加熟练了。

5 参考资料

[1] 维基百科. 传输控制协议.https://zh.wikipedia.org/wiki/%E4%BC%A0%E8%BE%93%E6%8E%A7%E5%88%B6%E5%8D%8F%E8%AE%AE

[2] 百度百科. 握手协议.https://baike.baidu.com/item/%E6%8F%A1%E6%89%8B%E5%8D%8F%E8%AE%AE/4058729?fr=aladdin