第 3 章

TCP/IP

3.1 TCP

TCP 是传输层最重要的协议，提供了可靠、有序的数据传输，是多个广泛使用的表示层协议的运行基础，理解了 TCP 的基本原理，有助于更好地掌握仓颉语言网络编程。

3.1.1 TCP 报文格式

1981 年 IETF 的 RFC 793 定义了 TCP，该协议的报文格式如图 3-1 所示。

图 3-1　RFC 793 TCP 报文格式

2022 年 8 月 IETF 公布了 RFC 9293，该协议替代了 RFC 793，两者的差异很小，在 RFC 9293 中，保留位变成了 4 位，也就是最后两个保留位被使用了，作为控制位的前两位，协议报文格式如图 3-2 所示。

TCP 报文包括首部和数据两部分，其中首部包括 20 字节的固定部分和变长的选项，数据部分也是变长的。TCP 的功能体现在首部各字段上，下面分别介绍这些字段的作用。

1. 源端口和目的端口

每个主机上都运行着多个应用，当发送端主机发送的报文经过网络到达目的主机时，目的主机需要知道这个报文应该传递给哪一个具体的应用，这个区分是通过端口号实现的。IP 地址标识了一台主机，端口号标识了一个具体的应用；借用寄送快递来类比，IP 地址就相当于一个具体的物理地址，如山东省青岛市崂山区海尔路 1 号×××大厦，但是，这个××

图 3-2　RFC 9293 TCP 报文格式

×大厦里面有几百人，光凭这个地址还不能顺利送达，还需要具体的收件人姓名，例如张磊，这个姓名就相当于端口号，在这个地址里是唯一的。

源端口是发送端主机的端口号，目的端口是接收端主机的端口号，每个端口号用 16 位表示，占用 2 字节，由此可以推算出，每个主机最多有 65 536 个端口（在实际应用中端口号 0 不直接使用，一般用来表示所有端口或者动态端口）。

2. 序号和确认号

序号和确认号都分别占用 32 位，也就是 4 字节，它们是 TCP 实现可靠传输的关键。假如有 10 000 字节需要通过 TCP 进行传输，每次传输的数据量是 1000 字节，那么，可以对这 10 000 字节进行编号，从第 1 字节的 0 到最后一个的 9999，如图 3-3 所示。

	第1个报文段数据				第2个报文段数据				第N个报文段数据	第10个报文段数据			
序号	0	1	…	999	1000	1001	…	1999	…	9000	9001	…	9999

图 3-3　报文序号

当第 1 次传输数据时，序号为第 1 个报文段的开始序号，也就是 0（这个 0 是逻辑上的，实际上可能是 0～2^{32}的一个数字），确认号表示已经收到了报文，期望发送方继续发送下一个报文段的第 1 字节数据的编号，这里因为是第 1 次发送报文，所以确认号是 0。当接收方接收到发送方的报文时，就回复一个确认报文，这时确认报文的序号还是 0，因为这个序号表示接收方第 1 个报文段的开始序号，但是，确认号变成了 1000，表示接收方已经收到了序号 1000 以前的所有报文，希望发送方开始发送从序号 1000 开始的下一个报文段（这里假设接收方只确认数据包，不发送其他数据，也就是确认报文的数据部分的长度为 0）。发送方收到该确认报文后，继续发送第 2 个报文段，此时序号变成了 1000，确认号变成了 1，这样一直下去，直到发送完所有数据，假如每次都可以顺利发送和接收，前 4 次收发报文的序号和确认号如表 3-1 所示。

表 3-1　前 4 次收发报文的序号和确认号

次　序	方　向	序　号	确 认 号
1	发送方到接收方	0	0
2	接收方到发送方	0	1000

续表

次　序	方　向	序　号	确 认 号
3	发送方到接收方	1000	1
4	接收方到发送方	1	2000

3. 首部长度

表示首部占用多少个 32 位，也就是多少个 4 字节，换句话说，首部的字节数量必须是 4 的整数倍数。因为 4 位最多表示 15，所以首部最多 60 字节，去掉首部固定部分的 20 字节，首部选项部分最多占用 40 字节。

4. 保留位

在 RFC 793 中占用 6 位，在 RFC 9293 中占用 4 位，保留以后使用，目前都置为 0。

5. 控制位

在 RFC 793 中占用 6 位，在 RFC 9293 中占用 8 位，功能如下所示。

1) CWR

CWR(Congestion Window Reduced)标志与后面的 ECE(ECN-Echo)标志都用于 IP 首部的 ECN 字段，如果 ECE 标志为 1，则通知对方已将拥塞窗口缩小。

2) ECE

若其值为 1，则会通知对方，从对方到本地的网络有阻塞。在收到数据包的 IP 首部中当 ECN 为 1 时将 TCP 首部中的 ECE 设为 1。

3) URG

将 URG(Urgent Pointer Field is Significant)设为 1，表示包中有需要紧急处理的数据，对于需要紧急处理的数据，与后面的紧急指针有关。

4) ACK

将 ACK(Acknowledgement Field is Significant)设为 1，确认应答的字段有效，TCP 规定除了最初建立连接时的 SYN 包之外该位必须设为 1。

5) PSH

将 PSH(Push Function)设为 1，表示需要将收到的数据立刻传给上层应用协议，若设为 0，则先对数据进行缓存。

6) RST

将 RST(Reset the Connection)设为 1，表示 TCP 连接出现异常必须强制断开连接。

7) SYN(Synchronize Sequence Numbers)

用于建立连接，将该位设为 1，表示这是一个连接请求或连接接受报文。

8) FIN

将 FIN(No more Data from Sender)设为 1，表明此报文段的发送端的数据已发送完毕，并要求释放连接。

6. 窗口大小

窗口大小字段用来控制对方发送的数据量，单位为字节。TCP 连接的一端根据设置的缓存空间大小确定自己的接收窗口大小，然后通知对方以确定对方的发送窗口的上限。窗口大小是一个 16 位的字段，所以窗口最大为 65 535。

7. 校验和

长度为 16 位，由发送端填充，接收端对 TCP 报文段执行 CRC 校验以验证 TCP 报文段在传输过程中是否损坏(校验过程中包括一个伪首部，伪首部的数据是从 IP 数据报头获取的，其目的是检测 TCP 数据段是否已经正确到达)，如果校验失败，则 TCP 直接丢弃这个报文段，这是 TCP 可靠传输的一个重要保障。

8. 紧急指针

长度为 16 位，它只在 URG 标志被设置为 1 时生效。

3.1.2 三次握手

TCP 传输的可靠性是建立在 TCP 连接的基础上的，TCP 的传输是双工的，也就是说报文可以同时在两个方向上传输，为了建立稳定的连接，传输的双方需要分别确认自己和对方都具备数据发送和接收能力，为了验证这种能力，就需要双方进行三次报文发送和接收，简称三次握手。下面通过一个实际的示例进行演示，客户端 IP 地址为 10.191.42.206，端口为 19516，服务器端 IP 地址为 1.15.152.31，端口为 6379，为了直观地展示握手过程，使用 Wireshark 捕获了双方通信的数据包，如图 3-4 所示。

No.	Time	Source	Destination	Protocol	Length	Info	
47	1.813091	10.191.42.206	1.15.152.31	TCP	66	19516 → 6379 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	第 1 次握手
48	1.837924	1.15.152.31	10.191.42.206	TCP	66	6379 → 19516 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1424 SACK_PERM WS=128	第 2 次握手
49	1.837991	10.191.42.206	1.15.152.31	TCP	54	19516 → 6379 [ACK] Seq=1 Ack=1 Win=132352 Len=0	第 3 次握手

图 3-4 三次握手

1. 第 1 次握手

客户端首先向服务器端发起连接请求，重点关注请求报文中的两个字段：一个是 SYN 标志，要置为 1；另一个是序号，会随机生成一个序号作为起始序号，第 1 次握手的报文如图 3-5 所示，图中封包字节区中的高亮部分即为 TCP 协议的首部。

根据 TCP 的格式，标识出序号、确认号和控制位等关键字段所占用的字节，如图 3-6 所示。

这里随机生成的初始序号为 11011101 00110101 11011110 01101100，即十进制的 3 711 295 084；因为是发起连接请求，所以确认号是 00000000 00000000 00000000 00000000，也就是十进制的 0；12 位控制位为 0000 00000010，只有 SYN 控制位被设置为 1，其余均为 0。

2. 第 2 次握手

服务器端收到客户端的连接请求后，如果同意连接就向客户端发送应答包，服务器端将第 1 次握手收到的客户端序号加 1 作为确认号，同时将 ACK 控制位设置为 1，然后生成服务器端的随机序号，把 SYN 控制位也设置为 1，第 2 次握手的报文如图 3-7 所示。

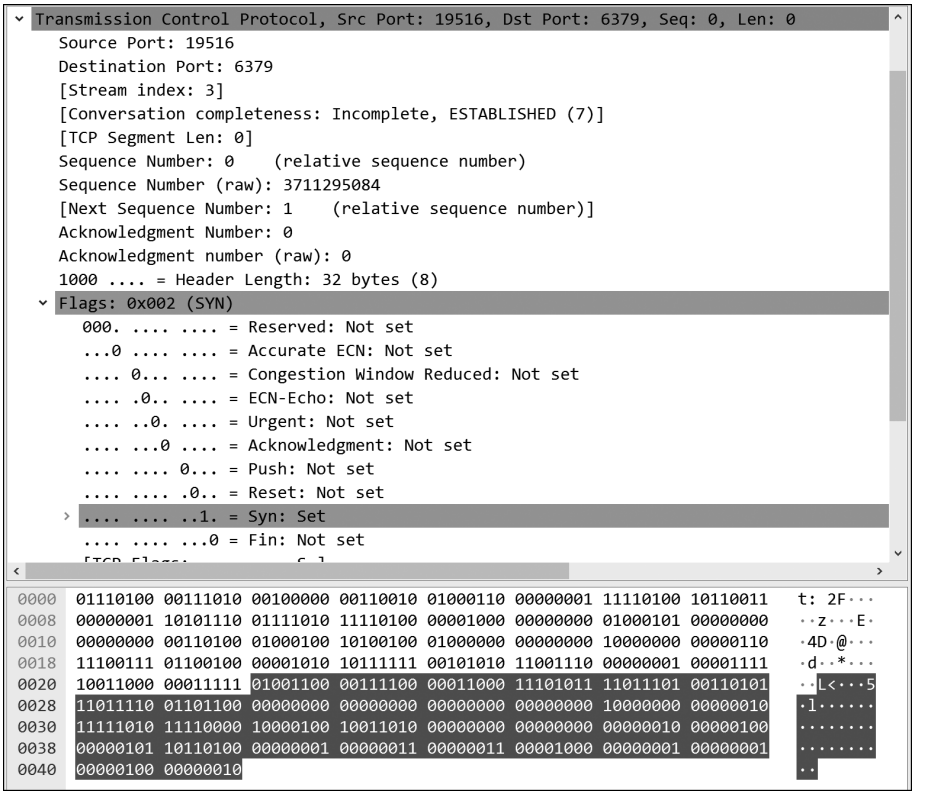

图 3-5 第 1 次握手

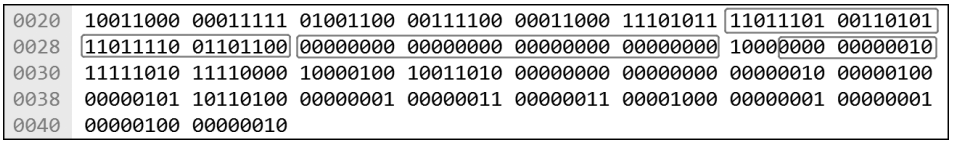

图 3-6 第 1 次握手的关键字段

根据 TCP 的格式，标识出序号、确认号和控制位等关键字段所占用的字节，如图 3-8 所示。

这里随机生成的服务器端初始序号为 11101001 01011101 00111111 00111111，即十进制的 3 915 202 367；确认号是 11011101 00110101 11011110 01101101，也就是十进制的 3 711 295 085，正好比客户端发送的序号多 1；12 位控制位为 0000 00010010，其中 ACK 和 SYN 控制位都被设置为 1，其余均为 0。

3. 第 3 次握手

客户端收到了服务器端发送的应答包，并且发现 ACK 和确认号是正确的，表示从客户端到服务器端的通信链路是正常的，并且服务器端同意连接请求，这时客户端给服务器端回复一个应答包，在应答包里把收到的服务器端序号加 1 作为确认号，同时将 ACK 控制位设

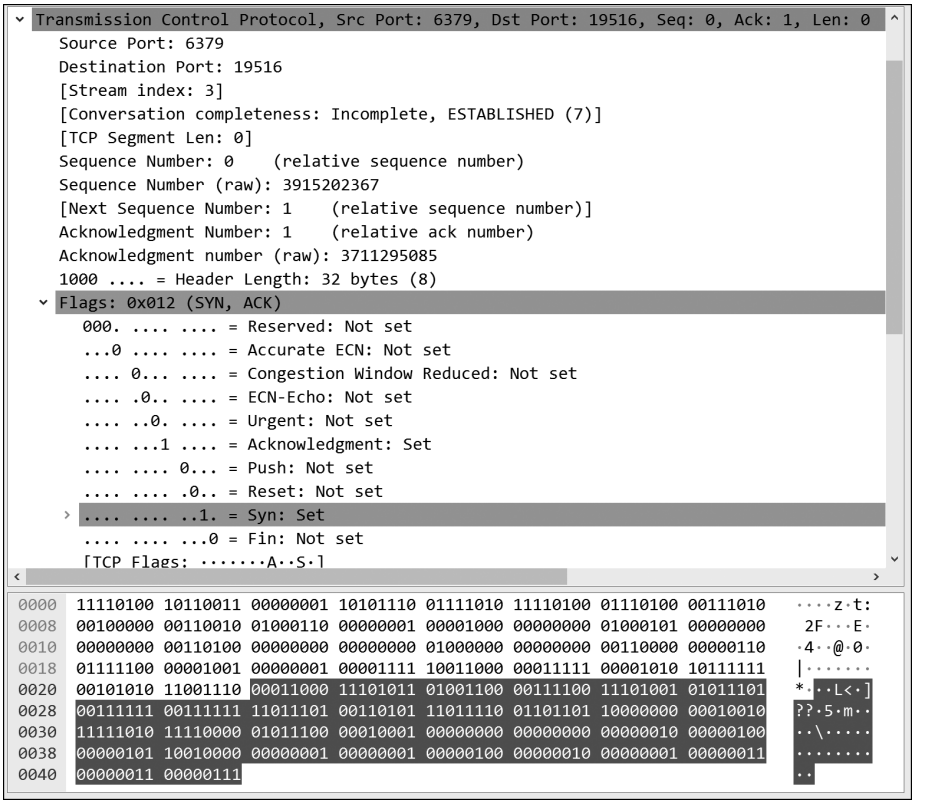

图 3-7 第 2 次握手

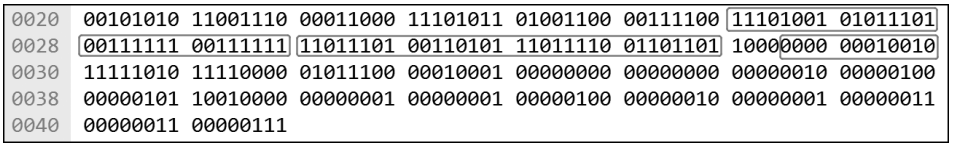

图 3-8 第 2 次握手的关键字段

置为 1,服务器端收到应答包后发现 ACK 和确认号也是正确的,就表明客户端收到了服务器端的报文,这样客户端和服务器端就建立起了 TCP 连接。第 3 次握手的报文如图 3-9 所示。

根据 TCP 的格式,标识出序号、确认号和控制位等关键字段所占用的字节,如图 3-10 所示。

这里序号为 11011101 00110101 11011110 01101101,即十进制的 3 711 295 085,虽然 SYN 报文不包含数据,但是,TCP 协议规定了 SYN 报文也占用 1 个序号,所以客户端第 2 次向服务器端发送报文时,序号就比初始序号多 1;确认号是 11101001 01011101 00111111 01000000,也就是十进制的 3 915 202 368,比服务器端发送的序号多 1;12 位控制位为 0000 00010000,其中 ACK 控制位被设置为 1,其余均为 0。

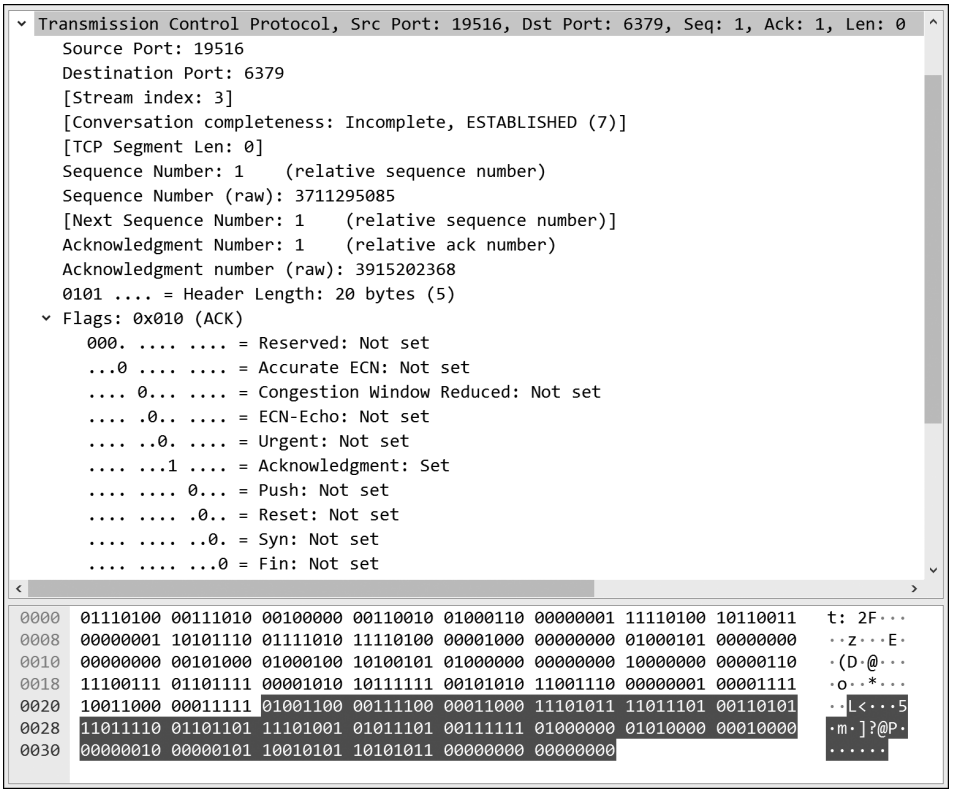

图 3-9 第 3 次握手

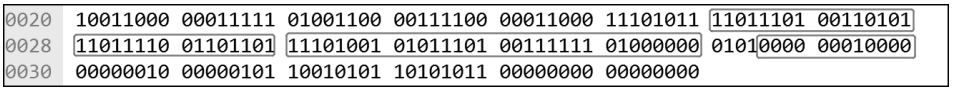

图 3-10 第 3 次握手的关键字段

3.1.3 四次挥手

在 TCP 连接建立以后可以进行数据的传输，当数据传输完毕后，任何一方都可以提出断开连接的请求，这一点和建立连接时不同，建立连接时只能客户端发起连接。TCP 断开连接的过程需要发送 4 次数据包，被称为四次挥手。

下面通过一个实际的示例进行演示，客户端 IP 地址为 10.191.42.206，端口为 21826，服务器端 IP 地址为 1.15.152.31，端口为 6379，为了直观地展示挥手过程，使用 Wireshark 捕获了双方通信的数据包，本次断开连接是由服务器端发起的，如图 3-11 所示。

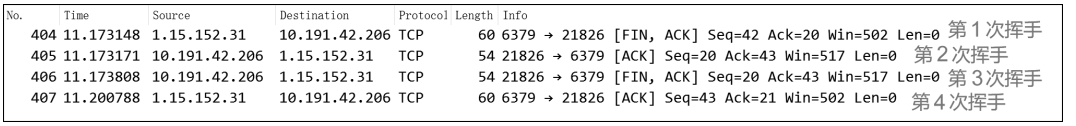

图 3-11 四次挥手

1. 第 1 次挥手

服务器端发起断开连接请求，将控制位 FIN 设置为 1，如图 3-12 所示。

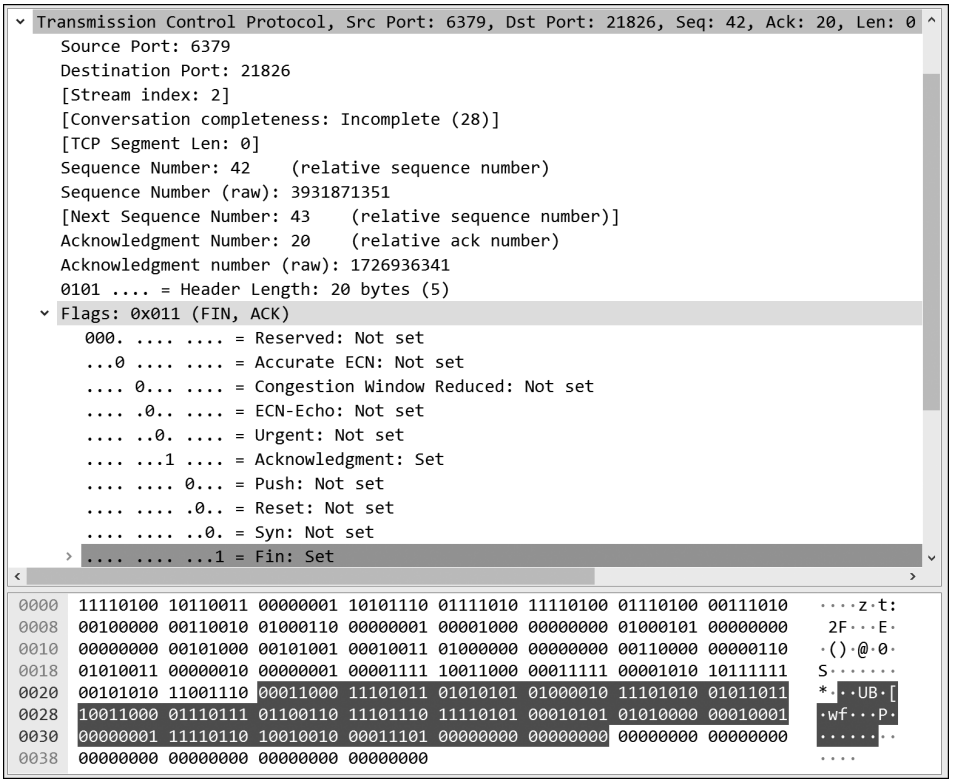

图 3-12　第 1 次挥手

根据 TCP 的格式，标识出序号、确认号和控制位等关键字段所占用的字节，如图 3-13 所示。

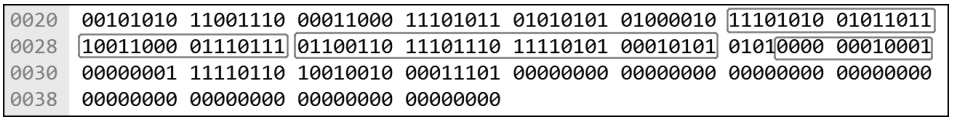

图 3-13　第 1 次挥手的关键字段

可以看到，序号为 11101010 01011011 10011000 01110111，十进制形式为 3 931 871 351；控制位为 0000 00010001，其中 ACK 和 FIN 控制位都是 1，其余均为 0。因为断开连接前双方已经发送了多次数据包，所以本次 ACK 为 1 是正常的，这和发起连接时不同，那时为第 1 次发送数据包，ACK 为 0；此时服务器端进入 FIN_WAIT_1 状态。

2. 第 2 次挥手

客户端收到断开连接的请求后，发送应答包，表示收到了该请求，如图 3-14 所示。

根据 TCP 的格式，标识出序号、确认号和控制位等关键字段所占用的字节，如图 3-15

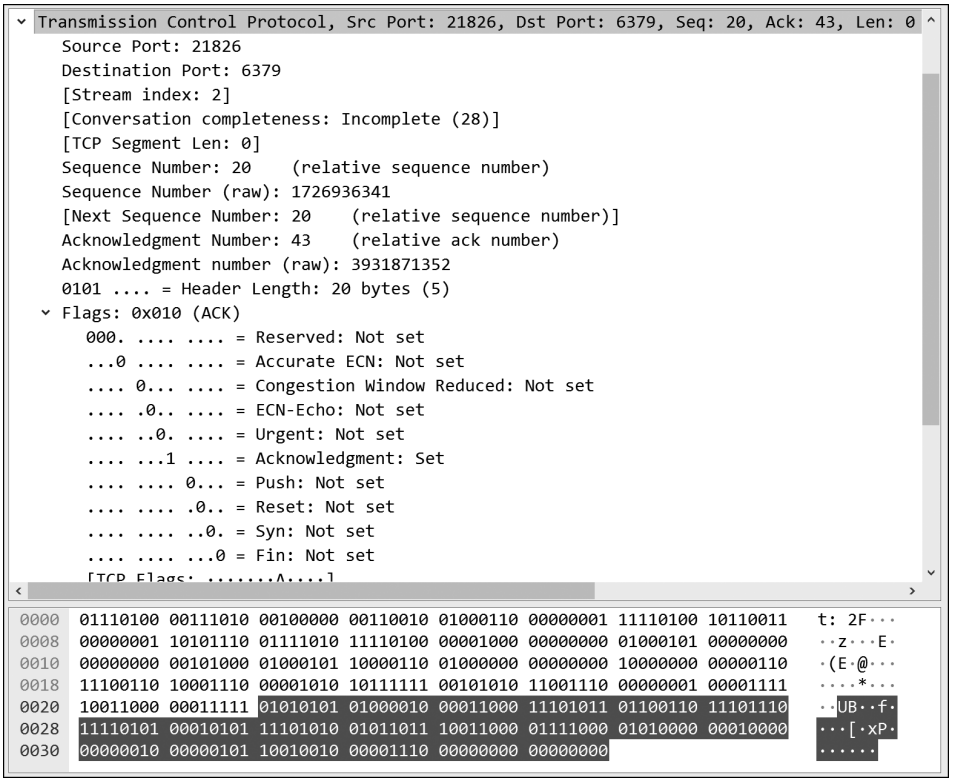

图 3-14 第 2 次挥手

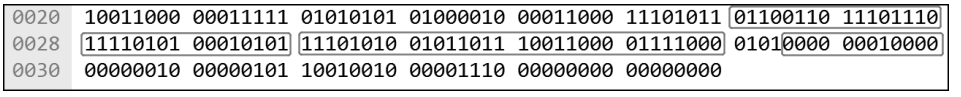

图 3-15 第 2 次挥手的关键字段

所示。

可以看到,序号为 01100110 11101110 11110101 00010101,十进制形式为 1 726 936 341;确认号为 11101010 01011011 10011000 01111000,十进制形式为 3 931 871 352;控制位为 0000 00010000,ACK 控制位为 1,其余均为 0;此时客户端进入 CLOSE_WAIT 状态。

3. 第 3 次挥手

第 2 次挥手后,如果客户端还有需要发送给服务器端的数据,就继续发送,否则就将 FIN 数据包发送给服务器端,表示客户端也做好了断开连接的准备,报文如图 3-16 所示。

根据 TCP 的格式,标识出序号、确认号和控制位等关键字段所占用的字节,如图 3-17 所示。

可以看到,序号为 01100110 11101110 11110101 00010101,十进制形式为 1 726 936 341,和第 2 次挥手时的序号一样,这是因为四次挥手的报文都不包含数据,序号不改变,只有发送 FIN 数据包时序号才会加 1,第 2 次挥手发送的是确认包,不改变发送方的序号。确认号

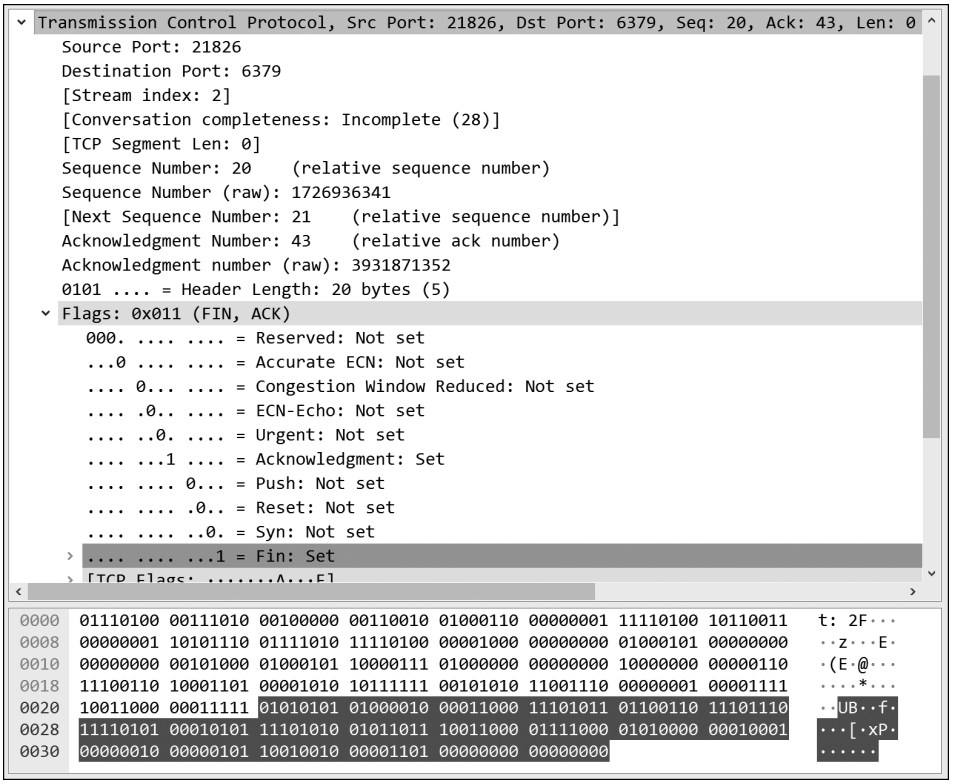

图 3-16 第 3 次挥手

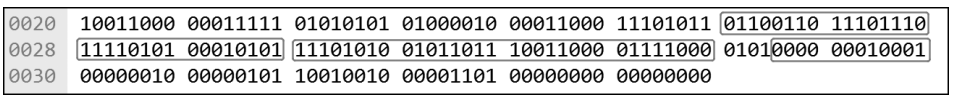

图 3-17 第 3 次挥手的关键字段

为 11101010 01011011 10011000 01111000，十进制形式为 3 931 871 352，因为没有收到新的服务器端数据包，所以也和第 2 次挥手时一样；控制位为 0000 00010001，ACK 和 FIN 控制位都为 1，其余均为 0；此时客户端进入 LAST_ACK 状态。

4. 第 4 次挥手

服务器端收到客户端发送的 FIN 数据包后，进入 TIME_WAIT 状态，然后将一个应答包回复给客户端，客户端收到该应答包后就进入 CLOSED 状态；服务器端在 TIME_WAIT 状态等待 2 × MSL（Maximum Segment Lifetime，报文最大生存时间）时间后也进入 CLOSED 状态；这样双方就断开了 TCP 连接。第 4 次挥手的报文如图 3-18 所示。

根据 TCP 的格式，标识出序号、确认号和控制位等关键字段所占用的字节，如图 3-19 所示。

可以看到，序号为 11101010 01011011 10011000 01111000，十进制形式为 3 931 871 352；确认号为 01100110 11101110 11110101 00010110，十进制形式为 1 726 936 342；控制位为

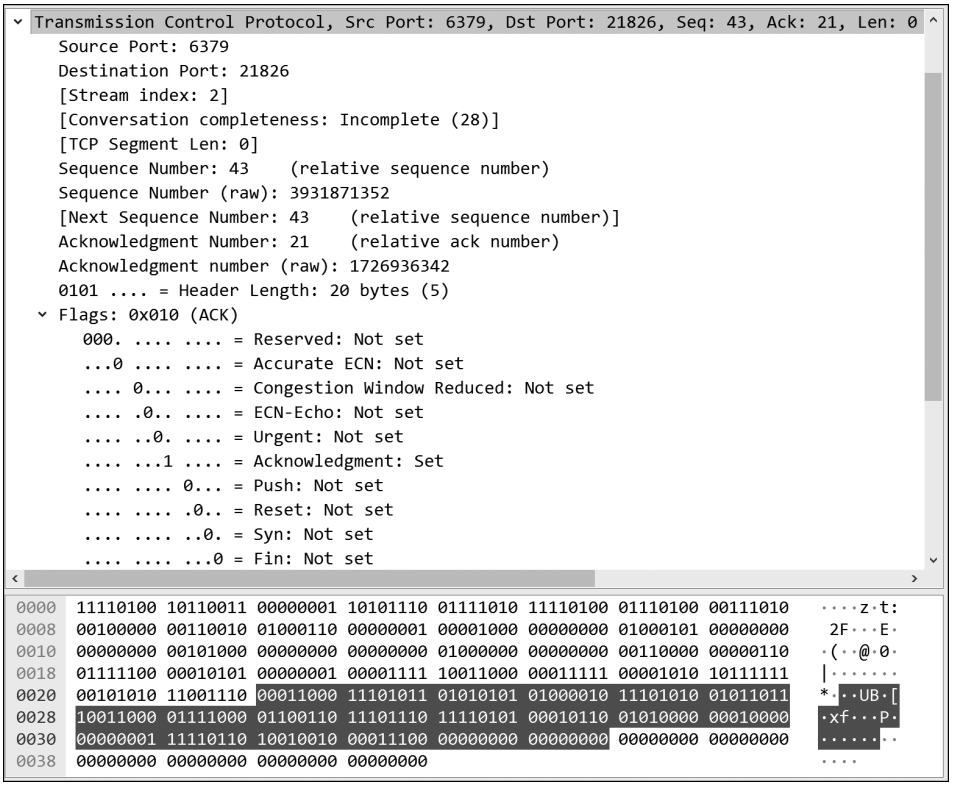

图 3-18 第 4 次挥手

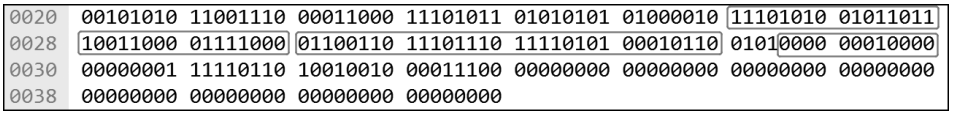

图 3-19 第 4 次挥手的关键字段

0000 00010000，ACK 控制位为 1，其余均为 0。

3.1.4 滑动窗口机制

在不可靠链路上实现可靠传输的基本原理是发送确认和超时重传，也就是说，当发送端发送完一个分组后，就处于等待状态，接收端将确认应答发送给发送端，发送端收到确认应答后开始下一次分组传输；如果发送端在规定的超时时间内没有收到确认应答，就重新传输该分组。这种确保分组可靠传输的机制又称为停等协议(stop-and-wait)，是一种最简单、最基础的数据传输协议，但是，这种协议有一个非常致命的缺点，就是信道利用率非常低，每发送一个分组都需要等待确认，大部分时间被用于等待应答包。

解决停等协议低效问题的方法是发送端可以连续发送多个分组，不必每个分组都等待确认，接收端告诉发送端它的接收能力，只要在这个接收能力之内，发送端可以一直发送。

接收端回复确认应答时，可以累计确认，例如，接收端收到了 3 个连续的分组，只需向发送端发送最后一个分组的确认信息，表示最后一个及前面的分组全部收到了；这种机制被 TCP 采纳，称为滑动窗口机制。

TCP 是双工协议，会话的双方都各自维护一个发送端窗口和接收端窗口，接收端窗口的大小是由自己的系统、硬件及处理能力等因素决定的，发送端窗口的大小取决于对方的通知，在 TCP 的报文格式中，有一个窗口大小字段，是接收端对自己接收能力的声明。对于发送端的数据，结合滑动窗口的数据确认状态，可以分为 4 种数据类型，如图 3-20 所示。

图 3-20　滑动窗口的 4 种数据类型

1. 已发送已确认

表示已经发送给接收端，并且接收端确认收到了数据，该数据不属于发送端窗口的范围，在图 3-20 中编号 104 及以前的字节都处于已发送已确认状态。

2. 已发送未确认

表示已经从发送端发送出去了，但是还没有收到接收端确认报文的数据，该数据属于发送端窗口的范围，如果被接收端确认，则确认的数据将变为已发送已确认状态，同时窗口位置会右移。在图 3-20 中编号 105～109 的字节都处于已发送未确认状态。

3. 未发送（接收端允许）

表示还没有从发送端发送出去，但接收端同意接收数据，该数据属于发送端窗口范围，已经被加载到缓存中，需要尽快发送出去。在图 3-20 中编号 110～114 的字节都处于未发送（接收端允许）状态。

4. 未发送（接收端未允许）

表示还没有从发送端发送出去，而且接收端也没有同意接收数据，这部分数据超出了当前接收端所能接收的能力。在图 3-20 中编号 115 及之后的字节都处于未发送（接收端未允许）状态。

假如上述发送端数据有了如下变化：一是发送端收到了接收端的确认报文，确认编号 105 和 106 的数据已接收；二是原先未发送的数据 110 已经发送了出去，那么此时滑动窗口的状态如图 3-21 所示。

图 3-21　窗口滑动

可以看到，窗口整体向右滑动了两字节，并且窗口范围内已发送未确认和未发送（接收端允许）的数据范围也发生了相应的改变。

3.2 UDP

UDP 也是传输层协议的一种，相对 TCP 来讲，它不需要建立连接，是不可靠、无序的，但是报文更简单，在特定场景下有更高的数据传输效率，在现代的网络通信中同样有广泛的应用。

UDP 报文格式如图 3-22 所示，包括固定 8 字节的首部和可变的数据部分。

图 3-22 UDP 报文格式

UDP 报文首部各个字段的说明如下。

1. 源端口

该字段占用 16 位，表示应用程序使用哪个端口发送数据，如果不需要对方回复，则可以使用 0。

2. 目的端口

该字段占用 16 位，表示接收端的端口。

3. 长度

该字段占用 16 位，表示包括首部字段在内的 UDP 数据包的长度。

4. 校验和

该字段占用 16 位，用来对数据包进行校验，检查在传输过程中是否存在差错，如果出错就丢弃。

一个典型的 UDP 报文如图 3-23 所示，该报文使用 Wireshark 捕获，图中封包字节区中

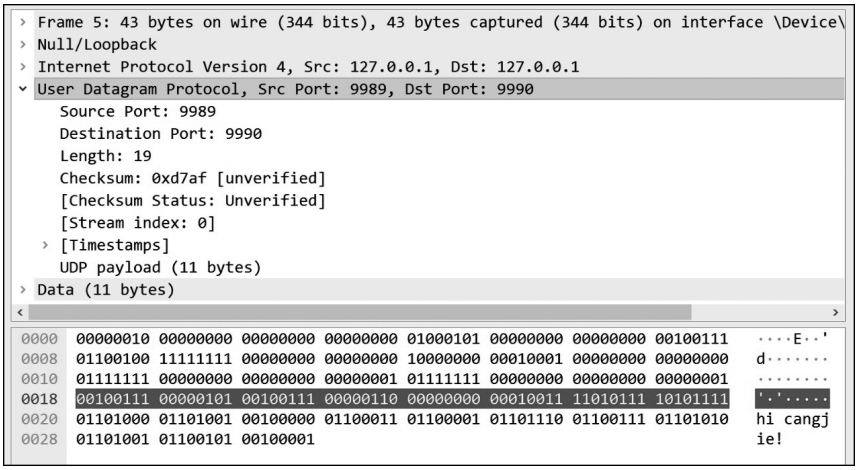

图 3-23 UDP 报文

的高亮部分即为 UDP 的首部。

3.3 IP

目前 IP 的两个版本(IPv4 和 IPv6)都处于正常使用状态,随着时间的推移,IPv6 会逐渐取代 IPv4 的统治地位,但这是一个漫长的过程,对于网络编程开发者来讲,两个协议都需要了解,特别是两个协议的报文格式有较大的区别,下面分别进行说明。

3.3.1 IPv4

IPv4 的报文格式如图 3-24 所示,如果不包含选项和数据,则单纯的报文首部占用固定的 20 字节,如果包含选项,则选项的字节数必须为 4 的倍数,不足部分使用 0 填充。

图 3-24 IPv4 报文格式

IPv4 报文首部各个字段的说明如下。

1. 版本

该字段占用 4 位,对于 IP 来讲,可能是 0100 或者 0110,分别代表 IPv4 和 IPv6,该字段处于报文头位置,非常重要,决定了后续数据按照哪一个 IP 版本进行解析。

2. 首部长度

表示首部占用多少个 32 位,也就是多少个 4 字节。因为 4 位二进制数最多表示 15,所以首部最多 60 字节,去掉首部固定部分的 20 字节,首部选项部分最多占用 40 字节。

3. 服务类型

占用 8 位,指定特殊的报文处理方式,用于为不同的 IP 数据包定义不同的服务质量。

4. 总长度

占用 16 位,表示 IP 报文的总大小,包括报文首部和携带的数据,按照字节计数,所以 IPv4 报文的最大长度为 65 535。

5. 标识

用来实现 IP 分片的重组,标识分片属于哪个进程,不同进程通过不同 ID 区分,与标志位和分片偏移字段一起使用。

6. 标志位

占用 3 位,用来确认是否还有 IP 分片或是否能执行分片操作。

7. 分片偏移

占用 13 位,用于标识 IP 分片的位置,实现 IP 分片的重组。

8. 存活时间

IPv4 报文可以经过最多三层设备(如路由器)数,存活时间(Time To Live,TTL)的初始值由源主机设置,该值在经过一个处理它的三层设备时减 1,当 TTL 值减为 0 时被丢弃,此时会发送 ICMP 报文通知源主机,其所发送的报文并未到达目标地址。该字段主要是为了防止路由出现环路问题而导致 IP 报文在网络中不停地被转发。

9. 协议

标识上层所使用的协议,常用协议号如下。

- ICM:1
- IGMP:2
- TCP:6
- EGP:8
- UDP:17

10. 首部校验和

用于校验 IP 数据包是否完整或被修改,不包含数据部分,若校验失败,则丢弃数据包。

11. 源 IP 地址和目的 IP 地址

标识发送端和接收端的 IP 地址,各占用 32 位。

一个典型的 IPv4 报文如图 3-25 所示,该报文使用 Wireshark 捕获,图中封包字节区中的高亮部分即为 IPv4 协议的首部。

3.3.2 IPv6

IPv6 的报文格式如图 3-26 所示,首部中去除了 IPv4 中选项的概念,使用扩展首部代替首部中的选项功能;IPv6 首部固定占用 40 字节,这样可以大幅地提高路由器的处理效率,从而有助于提高网络的整体吞吐量。

IPv6 报文首部各个字段的说明如下。

1. 版本

该字段占用 4 位,和 IPv4 的作用一样,在 IPv6 中固定为 0110。

2. 流量区分

类似于 IPv4 中的服务类型,为报文赋予不同的类型,占用 8 位。

3. 流标识

用来标识对传输有特殊要求的流,占用 20 位。

4. 有效载荷长度

表示除固定首部以外的报文字节数,占用 16 位。

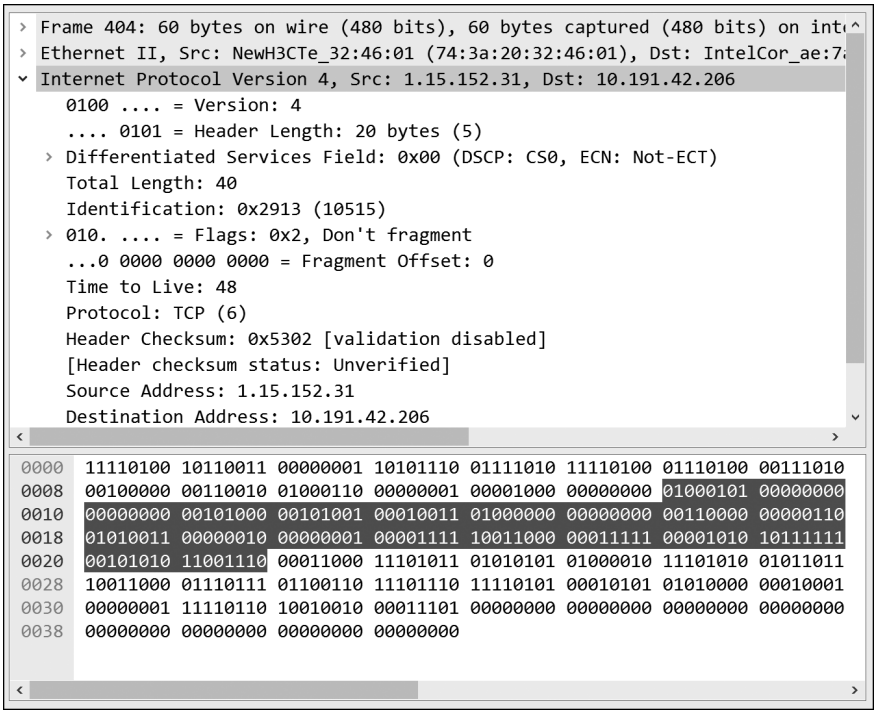

图 3-25　IPv4 报文

位 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

版本	流量区分	流标识	
有效载荷长度		下一个首部	跳数限制
源IP地址（16字节）			
目的IP地址（16字节）			
扩展首部			
有效载荷			

图 3-26　IPv6 报文格式

5. 下一个首部

相当于 IPv4 的协议字段，占用 8 位。

6. 跳数限制

相当于 IPv4 中的存活时间，占用 8 位。

7. 源 IP 地址和目的 IP 地址

标识发送端和接收端的 IP 地址，各占用 128 位。

一个典型的 IPv6 报文如图 3-27 所示，该报文使用 Wireshark 捕获，图中封包字节区中的高亮部分即为 IPv6 的首部。

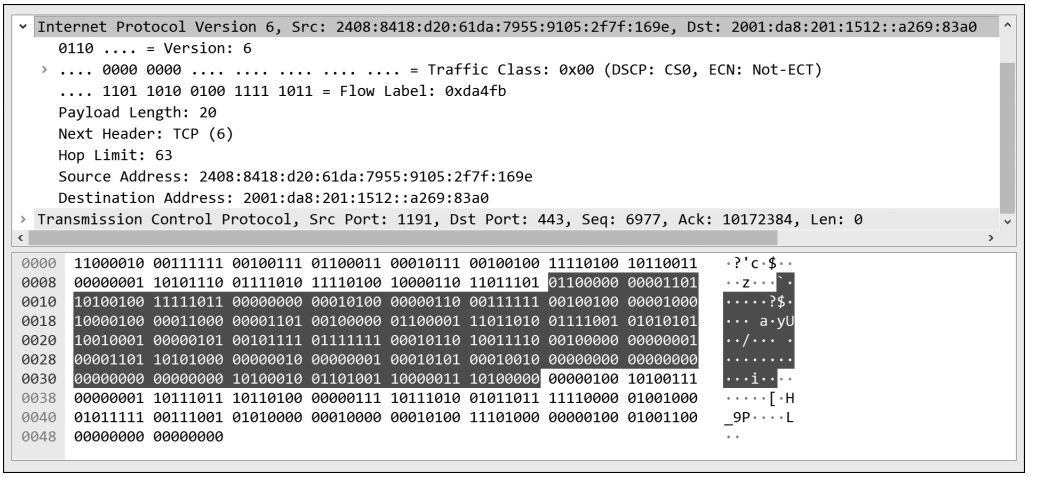

图 3-27 IPv6 报文

3.4 TCP/IP 高级选项

在 TCP 的发送端和接收端，对于报文的发送和确认可以配置不同的策略，这就是 TCP 套接字的 NODELAY 与 QUICKACK；对于连接存活的检测，可以通过 KEEPALIVE 配置，这些都是 TCP Socket 通信的常用高级选项，下面分别进行介绍。

1. NODELAY

以太网的最大传输单元(Maximum Transmission Unit，MTU)为 1500 字节，TCP 在发送报文时，为了避免被发送方分片会主动把数据分割成小段后再交给网络层，这个最大的分段大小称为 MSS(Max Segment Size)，MSS 的计算公式如下：

MSS＝MTU－IP 首部大小－TCP 首部大小

因为 IP 报文首部(IPv4)占用 20 字节，TCP 报文首部也占用 20 字节，所以 MSS 的大小为 1460 字节，如果一个报文的有效数据小于 MSS，则该报文可以被称为小包。

在一些特定情况下，如 Telnet 会有一字节一字节进行报文发送的情景，每次发送一字节的有效数据，就要额外发送 40 字节的首部信息，这在广域网上容易导致拥塞。为了解决类似的小包问题，TCP 引入了 Nagle 算法，该算法的规则如下：

(1) 如果包长度达到 MSS，则允许发送。

(2) 如果该包包含 FIN，则允许发送。

(3) 如果所有发出去的数据包均被确认，则允许发送。

(4) 如果发生了超时，则立即发送。

如果不满足上述条件，则 Nagle 算法会把数据累积起来不发送，使用 TCP 的 NODELAY 选项可以禁用 Nagle 算法，在默认情况下 Nagle 算法是启用的。

2. QUICKACK

接收端在收到数据后，并不是马上回复确认 ACK，而是延迟一段时间再确认，主要基于以下考虑：

（1）为了合并 ACK，如果连续收到两个 TCP 报文，并不一定需要 ACK 两次，只要回复最终的 ACK 就可以确认，从而降低网络流量。

（2）如果接收端恰好有数据要发送，就可以在发送数据的 TCP 报文里带上 ACK 信息。这样避免了大量的 ACK 以一个单独的 TCP 报文发送，也可以减少网络流量。

在默认情况下 TCP 套接字启用了延迟确认，但是，如果发送端启用了 Nagle 算法，而且接收端启用了延迟确认会出现什么情况呢？发送端尽可能延迟发送，接收端尽可能延迟确认，这会使网络延迟问题变得严重，所以尽可能不要同时启用，要么关闭 Nagle 算法，要么关闭延迟确认；通过设置 TCP 套接字的 QUICKACK 即可关闭延迟确认。

3. KEEPALIVE

在 TCP 连接建立后，如果网络一切正常，并且双方都没有主动发起关闭连接的请求，则此 TCP 连接理论上可以永久保持，但是，网络情况比较复杂，在双方长时间未通信时，如何得知对方还活着？如何得知当前连接是否仍然具备通信能力？而且，TCP 建立连接需要 3 次握手，需要双方分配资源，这说明 TCP 连接的建立是昂贵的，如果有必要，则可以持续保持连接，如果确认对方不再存活了，就可以关闭连接，从而释放资源。

TCP 的连接保活机制正是用来解决这个问题的，被称为 KEEPALIVE，当连接的一方等待超过一定时间后自动给对方发送一个空的报文，如果对方回复确认了，则证明连接还存活着，如果对方没有确认且进行多次尝试都是一样的结果，就认为连接已经丢失，没必要继续保持。该机制默认为关闭的，连接的任何一方都可开启此功能，通过 3 个主要参数来配置该功能。

1）tcp_keepalive_time

空闲时长，即每次正常发送心跳的周期，默认值为 7200s(2h)。

2）tcp_keepalive_intvl

探测报文的发送间隔，默认值为 75s。

3）tcp_keepalive_probes

计数器，如果没有接收到对方的确认报文，则继续发送保活探测报文次数，默认值为 9。