TCP/IP 详解卷 1：协议 - 11

阅读目的：了解 TCP 协议及 TCP 连接管理。
阅读时间：4 小时
阅读概况：第 12、13 章

第 12 章　TCP：传输控制协议(初步)

1. 概述

虽然 TCP 和 UDP 使用相同的网络层（IPv4 或 IPv6），但是 TCP 给应用程序提供了一种 与 UDP 完全不同的服务。 TCP 提供了一种面向连接的（connection-Oriented）、可靠的字节流服务。

> “面向连接的”是指使用 TCP 的两个应用程序必须在它们可交换数据之前，通过相互联系来建立一个 TCP 连接。

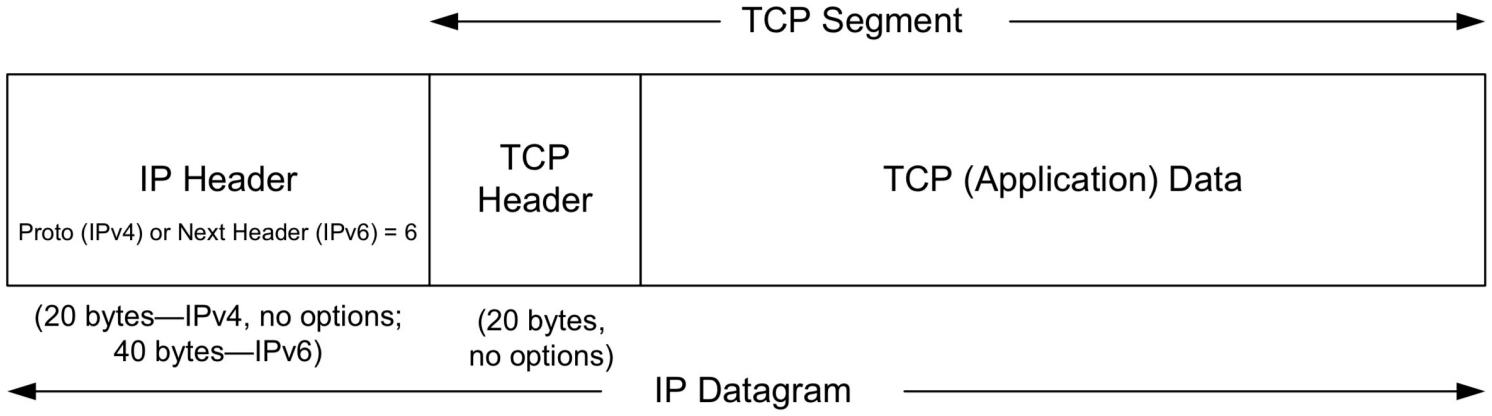
TCP 提供一种字节流抽象概念给应用程序使用。这种设计方案的结果是，没有由 TCP 自动插入的记录标志或消息边界。一个记录标志对应着一个应用程序的写范围指示。一端给 TCP 输入字节流，同样的字节流会出现在另一端。每个端点独立选择自己的读和写大小。

TCP 根本不会解读字节流里的字节内容。它不知道正在交换的数据字节是不是二进制数据、 ASCII 字符、EBCDIC 字符或其他东西。对这个字节流的解读取决于连接中的每个端点的应用程序。

TCP 给应用程序提供一种双工服务。这就是说数据可向两个方向流动，两个方向互相独立。因此，连接的每个端点必须对每个方向维持数据流的一个序列号。一个完整的 TCP 连接是双向和对称的，数据可以在两个方向上平等地流动。

2. TCP 头部和封装

TCP 在 IP 数据报中的封装。



每个 TCP 头部包含了源和目的端口号。这两个值与 IP 头部中的源和目的 IP 地址一起， 唯一地标识了每个连接。一个 IP 地址和一个端口的组合有时被称为一个**端点（endpoint）**或 **套接字（socket）**。

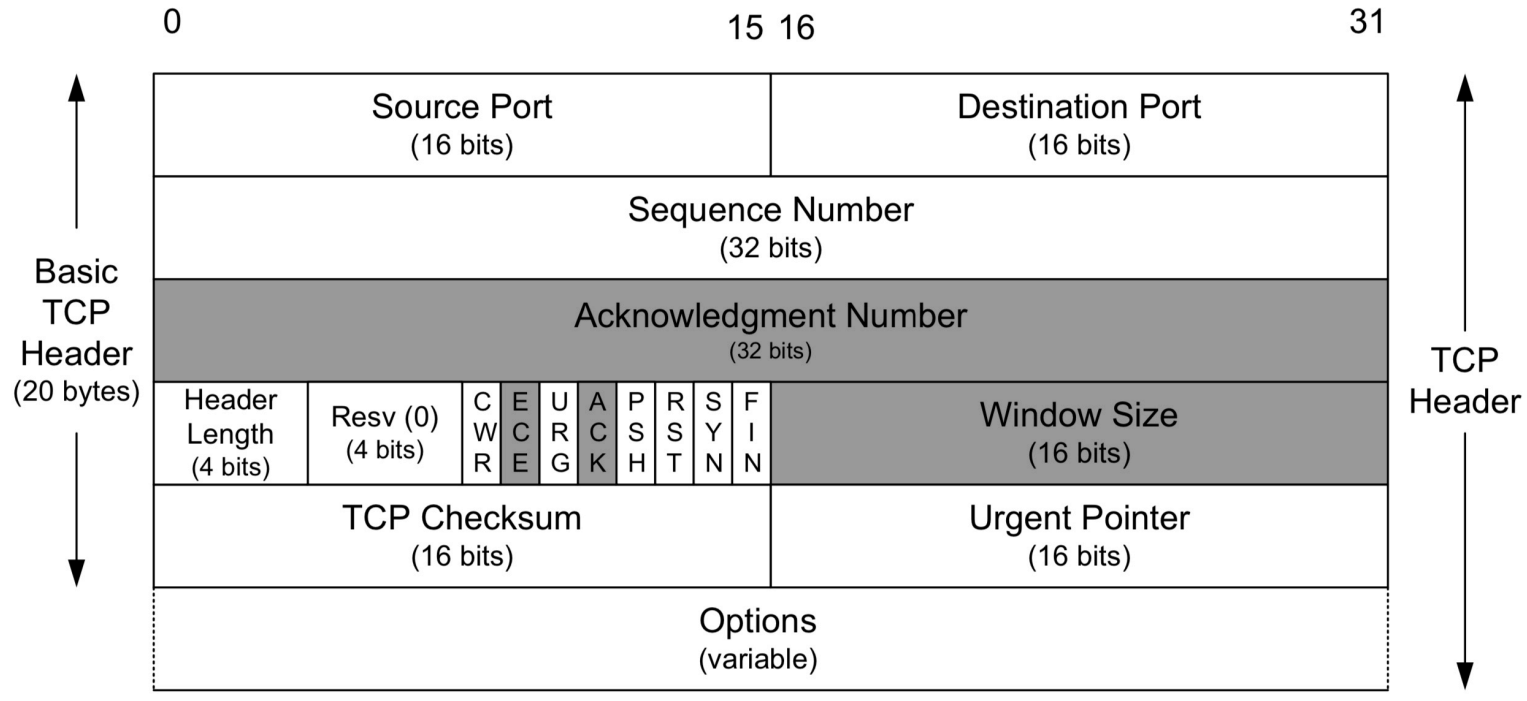


Figure 12-3 The TCP header. Its normal size is 20 bytes, unless options are present. The *Header Length* field gives the size of the header in 32-bit words (minimum value is 5). The shaded fields (*Acknowledgment Number*, *Window Size*, plus *ECE* and *ACK* bits) refer to the data flowing in the opposite direction relative to the sender of this segment.

TCP 提供一种可靠、面向连接、字节流、传输层的服务(通过使用许多这些技术而构建)。TCP 头部里的所有字段大多数都与可靠传递的抽象概念有着直接关系。TCP 把应用程序数据组包成报文段，发送数据时设置超时，确认被其他端点接收到的数据，给次序杂乱的数据进行重新排序，丢弃重复的数据，提供端到端的校验和。

第 13 章　TCP 连接管理（上）

1. 概述

TCP 是一种面向连接的单播协议。在发送数据之前，通信双方必须在彼此间建立一条连接。TCP 服务模型是一个字节流。 TCP 必须检测并修补所有在 IP 层（或下面的层）产生的数据传输问题，比如丢包、重复以及错误。

2. TCP 的连接和终止

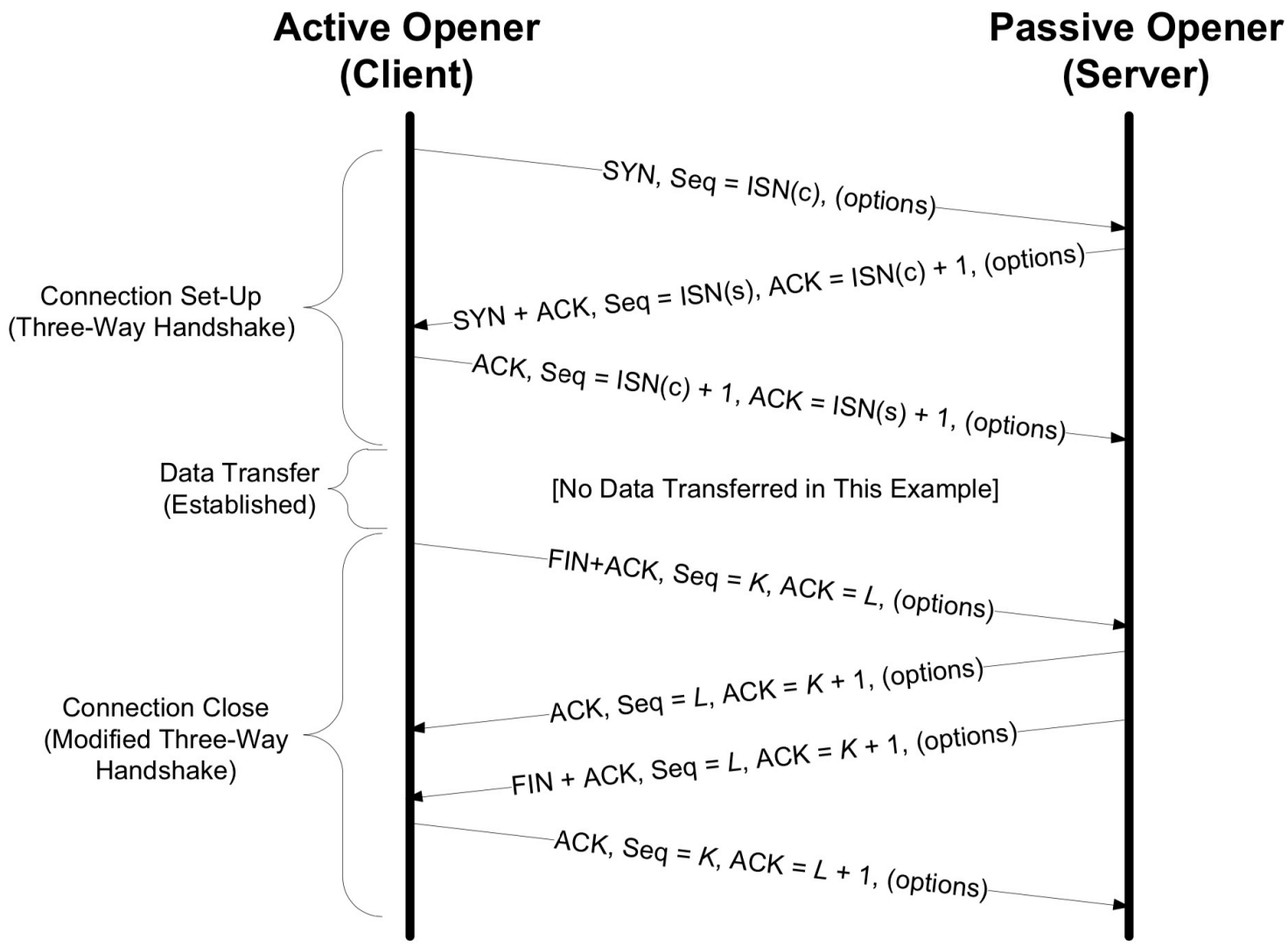
一个 TCP 连接通常分为 3 个阶段：**启动、数据传输（也称作“连接已建立”）和退出（关闭）**。

为了建立一个 TCP 连接， 需要完成以下步骤：

1. 主动开启者（通常称为客户端）发送一个 SYN 报文段（即一个在 TCP 头部的 SYN 位字段置位的 TCP/IP 数据包)，并指明自己想要连接的端口号和它的客户端初始序列号(记作 ISN(c))。通常，客户端还会借此发送一个或多个选项。客户端发送的这个 SYN 报文段称作段 1。

2. 服务器也发送自己的 SYN 报文段作为响应，并包含了它的初始序列号(记作 ISN(s))。该段称作段 2。此外，为了确认客户端的 SYN，服务器将其包含的 ISN(c) 数值加 1 后作为返回的 ACK 数值。因此，每发送一个 SYN，序列号就会自动加 1。这样如果出现丢失的情况，该 SYN 段将会重传。

3. 为了确认服务器的 SYN，客户端将 ISN(s) 的数值加 1 后作为返回的 ACK 数值。这称作段 3。



通过发送上述 3 个报文段就能够完成一个 TCP 连接的建立。它们也常称作**三次握手**。 三次握手的目的不仅在于让通信双方了解一个连接正在建立，还在于利用数据包的选项来承载特殊的信息，**交换初始序列号（Initial Sequence Number， ISN）。**

TCP 协议规定通过发送一个 FIN 段（即 FIN 位字段置位的 TCP 报文段）来发起关闭操作。只有当连接双方都完成关闭操作后，才构成一个完整关闭：

1. 连接的主动关闭者发送一个 FIN 段指明接收者希望看到的自已当前的序列号（K）。FIN 段还包含了一个 ACK 段用于确认对方最近一次发来的数据（L）。
2. 连接的被动关闭者将 K 的数值加 1 作为响应的 ACK 值，以表明它已经成功接收到主动关闭者发送的 FIN。此时，上层的应用程序会被告知连接的另一端已经提出了关闭的请求。通常，这将导致应用程序发起自已的关闭操作。接着，被动关闭者将身份转变为主动关闭者，并发送自已的 FIN。该报文段的序列号为 L。
3. 为了完成连接的关闭，最后发送的报文段还包含一个 ACK 用于确认上一个 FIN。值得注意的是，如果出现 FIN 丢失的情况，那么发送方将重新传输直到接收到一个 ACK 确认为止。

3. 初始序列号

在发送用于建立连接的 SYN 之前，通信双方会选择一个初始序列号。初始序列号会随时间而改变，因此每一个连接都拥有不同的初始序列号。

初始序列号可被视为一个 32 位的计数器。该计数器的数值每 4 微秒加 1。此举的目的在于为一个连接的报文段安排序列号，以防止出现与其他连接的序列号重叠的情况。尤其对于**同一连接的两个不同实例**而言，新的序列号也不能出现重叠的情况。

> 由于一个 TCP 连接是被一对端点所唯一标识的，其中包括由 2 个 IP 地址与 2 个端口号构成的 4 元组，因此即便是同一个连接也会出现不同的实例。如果连接由于某个报文段的长时间延迟而被关闭，然后又以相同的 4 元组被重新打开，那么可以相信延迟的报文段又会被视为有效数据重新进人新连接的数据流中。

4. TCP 选项

TCP 头部包含了多个选项。选项**列表结束（End of option List， EOL）、无操作（No Operation，NOP）以及最大段大小（Maximum Segment Size， MSS）**是定义于原始 TCP 规范中的选项。

Table 13-1 The TCP option values. Up to 40 bytes are available to hold options.

Kind	Length	Name	Reference	Description and Purpose
0	1	EOL	[RFC0793]	End of Option List
1	1	NOP	[RFC0793]	No Operation (used for padding)
2	4	MSS	[RFC0793]	Maximum Segment Size
3	3	WSOPT	[RFC1323]	Window Scaling Factor (left-shift amount on window)
4	2	SACK-Permitted	[RFC2018]	Sender supports SACK options
5	Var.	SACK	[RFC2018]	SACK block (out-of-order data received)
8	10	TSOPT	[RFC1323]	Timestamps option
28	4	UTO	[RFC5482]	User Timeout (abort after idle time)
29	Var.	TCP-AO	[RFC5925]	Authentication option (using various algorithms)
253	Var.	Experimental	[RFC4727]	Reserved for experimental use
254	Var.	Experimental	[RFC4727]	Reserved for experimental use

最大段大小选项

最大段大小是指 TCP 协议所允许的从对方接收到的最大报文段，因此这也是通信对方在发送数据时能够使用的最大报文段。

最大段大小只记录 TCP 数据的字节数而不包括其他相关的 TCP 与 IP 头部。当建立一条 TCP连接时，通信的每一方都要在 SYN 报文段的 MSS 选项中说明自已允许的最大段大小。这 16 位的选项能够说明最大段大小的数值。在没有事先指明的情况下，最大段大小的默认数值为 536 字节。

> 任何主机都应该能够处理至少 576 字节的 IPv4 数据报。如果接照最小的 IPv4 与 TCP 头 部计算， TCP 协议要求在每次发送时的最大段大小为 536 字节，这样就正好能够组成一个 576（20+20+536=576）字节的 IPv4 数据报。