三报文“握手”

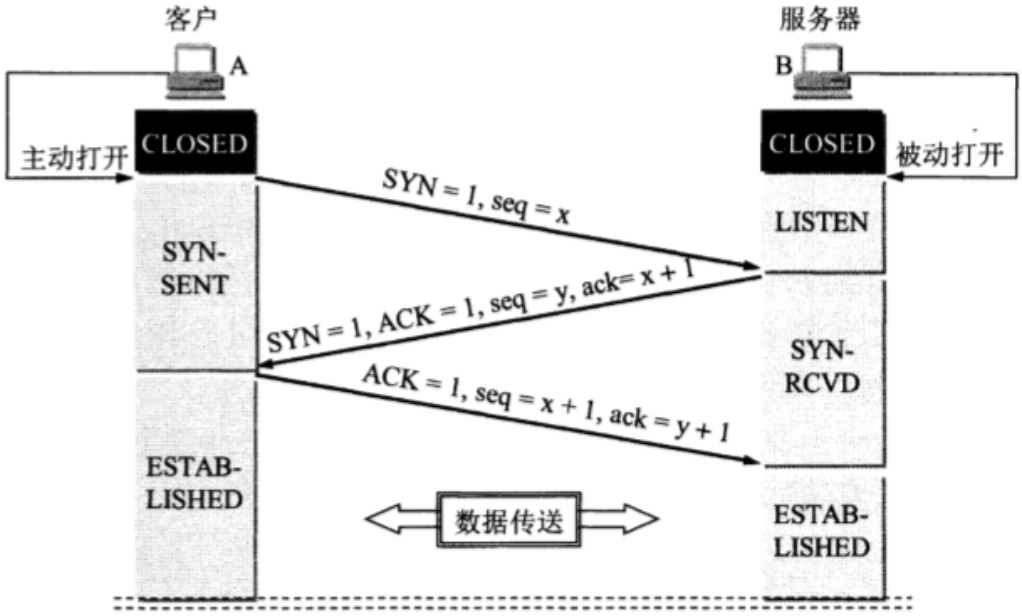

图 5-28 用三报文握手建立 TCP 连接

TCP规定：`SYN=1`的报文段不能携带数据。普通的TCP报文段可以携带数据，但如果不携带数据，则不消耗序号。

1. 最初，客户端 和 服务器 都是处于关闭状态。TCP服务器进程首先创建传输控制块，用来存储TCP链接中的一些重要信息，之后就准备接受TCP客户进程的连接请求。此时，TCP服务器进程就会进入监听状态，等待客户的请求。TCP服务器进程是被动等待来自TCP客户进程的连接请求，而不是主动发起，因此称为被动打开连接。
2. TCP客户进程也是首先建立连接控制块，然后向TCP服务器发送*TCP*连接请求报文段`[SYN=1, seq=x]`，并进入同步已发送状态，称为主动打开连接。`SYN=1`表明这是一个TCP链接请求报文段，序号字段`seq=x`，作为TCP客户进程所选择的初始序号。
3. TCP服务器进程收到客户端发来的连接请求报文段后，如果同意建立连接，则向TCP客户进程发送*TCP*连接确认报文段`[SYN=1, ACK=1, seq=y, ack=x+1]`，并进入同步已接收状态。`SYN=1, ACK=1`表明这是一个TCP连接请求确认报文段，`seq=y`作为TCP服务器进程所选择的初始序号，`ack=x+1`这是对TCP客户进程所选择的初始序号的确认。
4. TCP客户进程收到TCP连接请求确认报文段后，还要向TCP服务器进程发送一个普通的*TCP*确认报文段`[ACK=1, seq=x+1, ack=y+1]`，并进入连接已建立状态。`ACK=1`表明这是一个普通的TCP确认报文段，序号字段`seq=x+1`是因为TCP客户进程发送的第一个TCP报文段的序号为x，并且不携带数据。确认段`ack=y+1`是对TCP服务器进程所选择的初始序号的确认。TCP服务器进程收到该确认报文段后也进入连接已建立状态.

几个问题?

可不可以只握手两次？即不发送最后一次的普通TCP确认报文段

不可以。假设TCP客户进程发出一个TCP连接请求报文段，但该报文段在某些网络结点长时间滞留了，这必然会造成对该报文段的超时重传。假设重传的报文段被TCP服务器进程正常接收，TCP服务器进程向TCP客户进程发送一个TCP连接请求确认报文段，并进入连接已建立状态。由于这是已经改为**两报文握手**，因此TCP服务器进程发送完TCP连接请求确认报文段后，进入的是连接已建立状态，而不像三报文握手那样进入**同步已接收状态**。TCP客户进程收到连接请求确认报文段后也进入连接建立状态。**现在**

TCP双方都处于连接已建立状态，它们可以相互传输数据，之后再通过四报文挥手来释放连接，TCP双方都进入关闭状态。一段时间后，之前滞留在网络中失效的TCP连接请求报文段到达了TCP服务器进程，TCP服务器进程会误认为这是TCP客户进程又发起了一个新的TCP连接请求，于是给TCP客户进程发送TCP连接请求确认报文段，并进入**连接已建立状态**。该报文段到达TCP客户进程，**由于TCP客户进程实际上没有发送新的TCP连接请求**，并且处于关闭状态，因此**不会理会该报文段**。但是TCP服务器进程已经进入了连接已建立状态，他认为新的TCP链接已经建立好了，并一直等待TCP客户进程发来数据，这将白白浪费TCP服务器进程所在主机的很多资源！

如果采用的是三次握手，就算那一次失效的报文传送过来了，服务器收到了失效的报文并回复了确认报文，但是客户端不会再次发出确认。由于服务器收不到确认，就知道客户端并没有请求连接。

四报文“挥手”

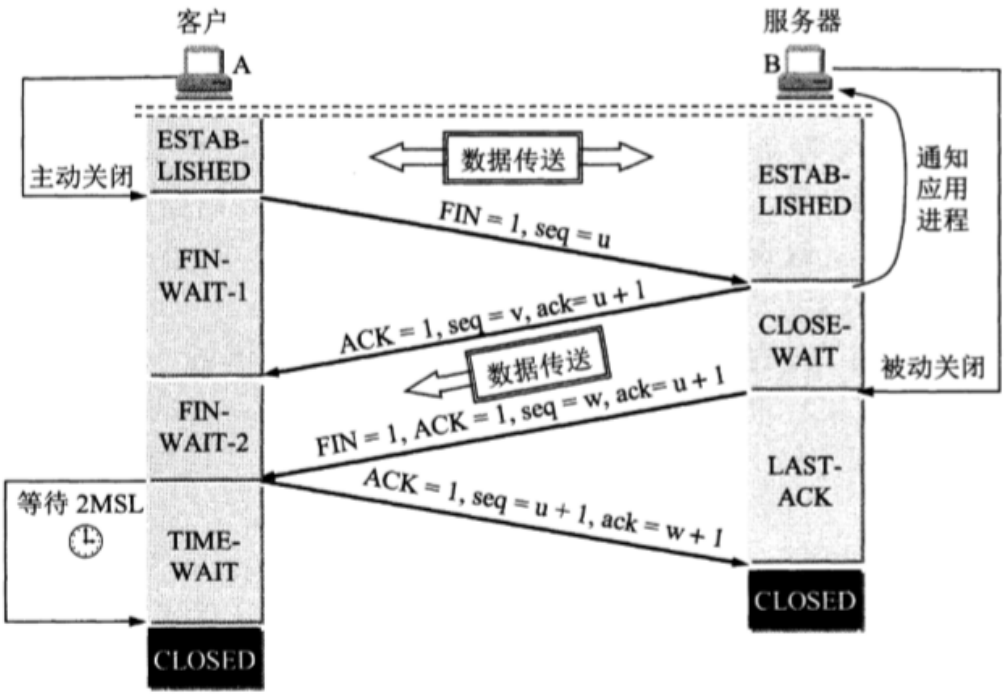

图 5-29　TCP 连接释放的过程

TCP规定：终止位FIN=1的报文段即使不携带数据，也要消耗一个序号。

1. TCP客户进程主动关闭TCP连接，TCP客户进程会发送TCP连接释放报文段`[FIN=1, seq=u]`，并进入**终止等待-1**状态。`seq=u`的 u = 结束报文段序号 + 1，`FIN=1`是终止报文段。
2. 服务器接收到客户端发来的FIN包之后，会发送一条普通的TCP确认报文段`[ACK=1, seq=v, ack=u+1]`，并进入关闭等待状态。`ACK=1`表明这是一个普通的TCP确认报文段，序号`seq=v`中的v等于TCP服务器进程之前已传送过的数据的最后一个字节的序号加1，确认段`ack=u+1`是对TCP连接释放报文段的确认。TCP服务器进程会告诉高层应用进程，TCP客户进程要断开与自己的TCP连接。此时，从TCP客户进程到TCP服务器进程这个方向的连接就释放了。这时TCP连接属于半关闭状态，即TCP客户进程没有数据要发送了，但TCP服务器进程如果有数据要发送，TCP客户进程仍要接收。**从TCP服务器进程到TCP客户进程这个方向的连接并未关闭。**
3. TCP客户进程收到TCP服务器进程的确认段后，进入**终止等待-2**状态。TCP服务器进程发送TCP连接释放报文段`[FIN=1, ACK=1, seq=w, ack=u+1]`，并进入最后确认状态。`FIN=1, ACK=1`表明这是TCP连接释放报文段，同时也对之前收到的报文段进行确认。`seq=w`是对之前TCP客户进程发送数据的字节序号。`ack=u+1`这是对之前收到的TCP连接释放报文段额重复确认。
4. TCP客户进程收到上面报文段后，又对TCP服务器进程发送一个普通的TCP确认报文段`[ACK=1, seq=u+1, ack=w+1]`，之后进入时间等待状态。`ACK=1`表明这是普通的TCP确认报文段，序号`seq=u+1`是因为TCP客户进程之前发送的TCP连接释放报文段虽然不携带数据，但要消耗一个序号。确认段`ack=w+1`这是对所收到的TCP连接释放报文段的确认。**TCP服务器进程收到该报文段后**

就进入关闭状态，而TCP客户进程还要经过2MSL后才能进入关闭状态。MSL是最长报文段寿命，建议为2分钟。

几个问题?

为什么客户进程最后要等待2MSL时间?

防止客户进程最后发去的 ACK 没传送到服务器，如果服务器没收到客户端的 ACK，肯定会选择重发一次 FIN 包，那么此时如果客户端已经关闭了，客户端就不能再发 ACK 确认收到了。这会造成TCP服务器进程反复重传TCP连接释放字段，无法进入关闭状态。2MSL等待时长可以确保收到最后一个服务器发送的TCP确认报文段而进入关闭状态。另外TCP客户进程再经历2MSL时间就可以使本次连接所产生的所有报文段都从网络中消失。

为什么不能三次挥手呢?

- 首先如果去掉最后一次挥手，那么服务器端就不知道自己要关闭的确认报文有没有传输成功，可能半路上就失败了，但是此时客户端不知道，导致客户端一直在等待服务器关闭，但是此时服务器端直接就关闭了；

如果已经建立了连接，但是客户端突然出现故障了怎么办?

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。

- 服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时。
- 若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

写的好的文章：https://blog.csdn.net/qzcsu/article/details/72861891