**TCP 提供面向有连接的通信传输，面向有连接是指在传送数据之前必须先建立连接，数据传送完成后要释放连接。**

[https://mp.weixin.qq.com/s?\_\_biz=MzUyNDE4NzM3MQ==&mid=2247483710&idx=1&sn=079b00bc5e013a37e77dfc9004da000c&scene=21#wechat\_redirect](https://mp.weixin.qq.com/s?__biz=MzUyNDE4NzM3MQ==&mid=2247483710&idx=1&sn=079b00bc5e013a37e77dfc9004da000c&scene=21" \l "wechat_redirect)

TCP首部：

**TCP端口号** TCP的连接是需要四个要素确定唯一一个连接：

**（源IP，源端口号）+ （目的IP，目的端口号）**

**TCP的序号和确认号：**

**32位序号 seq：Sequence numbe**r 缩写seq ，TCP通信过程中某一个传输方向上的字节流的每个字节的序号，通过这个来确认发送的数据有序，比如现在序列号为1000，发送了1000，下一个序列号就是2000。

**32位确认号 ack：Acknowledge number 缩写ack，TCP对上一次seq序号做出的确认号**，用来响应TCP报文段，给收到的TCP报文段的序号seq加1

**TCP的标志位 SYN**，**ACK** 和 **FIN**

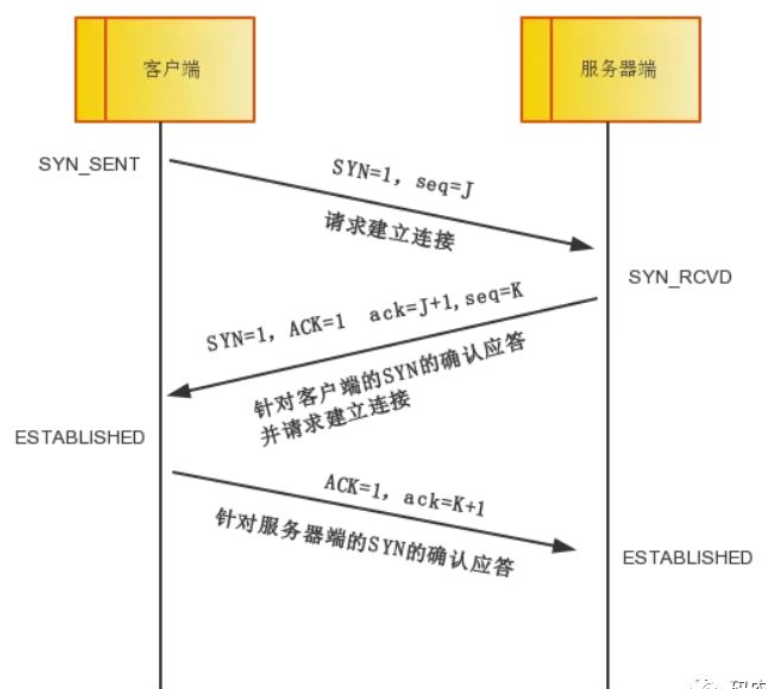
**SYN：简写为S，同步标志位，用于建立会话连接，同步序列号；**

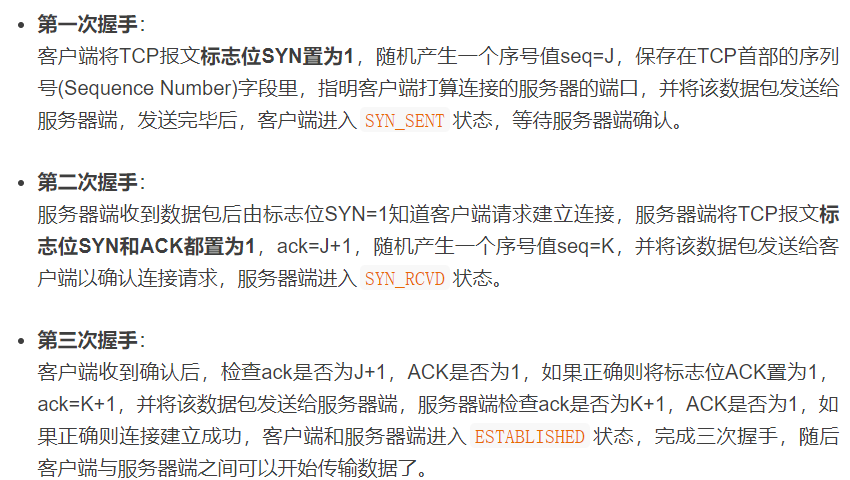
**ACK：简写为.，确认标志位，对已接收的数据包进行确认；**

**FIN：简写为F，完成标志位，表示我已经没有数据要发送了，即将关闭连接；**

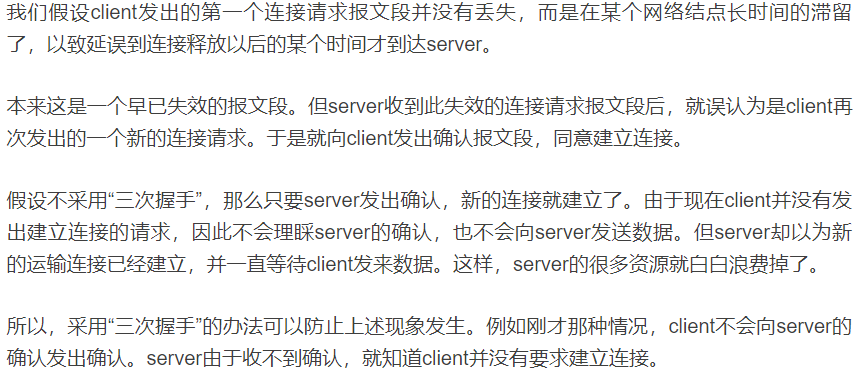
**所谓三次握手(Three-way Handshake)，是指**建立一个 TCP 连接时，需要客户端和服务器总共发送3个报文。****

**三次握手的目的是连接服务器指定端口，建立 TCP 连接，并同步连接双方的序列号和确认号，交换 TCP 窗口大小信息。在 socket 编程中，客户端执行 connect() 时。将触发三次握手**



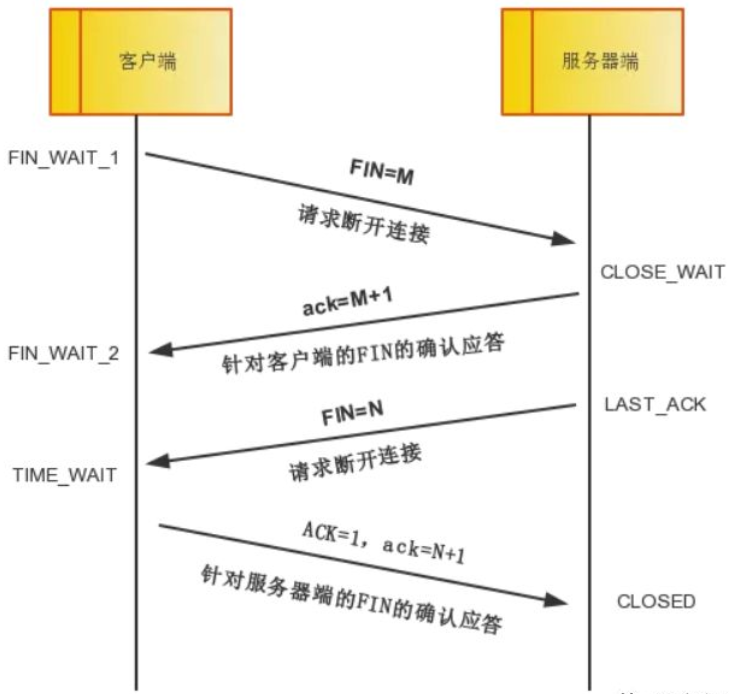


**为什么需要三次握手？**



**TCP 四次挥手关闭连接**

**四次挥手即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开**。在socket编程中，这一过程由客户端或服务端任一方执行close来触发。**由于TCP连接是全双工的，因此，每个方向都必须要单独进行关闭**，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭。





**为什么连接的时候是三次握手，关闭的时候却是四次握手？**

建立连接时因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。所以建立连接只需要三次握手。

由于TCP协议是一种面向连接的、可靠的、基于字节流的运输层通信协议，**TCP是全双工模式**。这就意味着，关闭连接时，当Client端发出FIN报文段时，只是表示Client端告诉Server端数据已经发送完毕了。**当Server端收到FIN报文并返回ACK报文段，表示它已经知道Client端没有数据发送了，但是Server端还是可以发送数据到Client端的，所以Server很可能并不会立即关闭SOCKET，直到Server端把数据也发送完毕。**

当Server端也发送了FIN报文段时，这个时候就表示Server端也没有数据要发送了，就会告诉Client端，我也没有数据要发送了，之后彼此就会愉快的中断这次TCP连接。

**为什么要等待2MSL？**

MSL：报文段最大生存时间，它是任何报文段被丢弃前在网络内的最长时间。

**TCP协议可靠性是如何保证之滑动窗口，超时重发，序列号确认应答信号**

TCP 是一种提供可靠性交付的协议。

也就是说，通过 TCP 连接传输的数据，无差错、不丢失、不重复、并且按序到达。

但是在网络中相连两端之间的介质，是复杂的，并不确保数据的可靠性交付，那么 TCP 是怎么样解决问题的？

TCP 是通过下面几个特性保证数据传输的可靠性：**序列号和确认应答信号 超时重发控制**

**连接管理 滑动窗口控制 流量控制 拥塞控制**

**通过序列号和确认应答信号提高可靠性**

**在 TCP 中，当发送端的数据到达接收主机时，接收端主机会返回一个已收到消息的通知，这个消息叫做确认应答（ACK）**。当发送端将数据发出之后会等待对端的确认应答。如果有确认应答，说明数据已经成功到达对端。反之，则数据丢失的可能性很大

但是，**如果在一定时间内发送端都没有得到确认应答ACK，发送端就会认为数据丢失，并进行数据重发**。所以，即使产生了丢包，TCP仍然能保证数据能够到达对端，实现可靠的传输

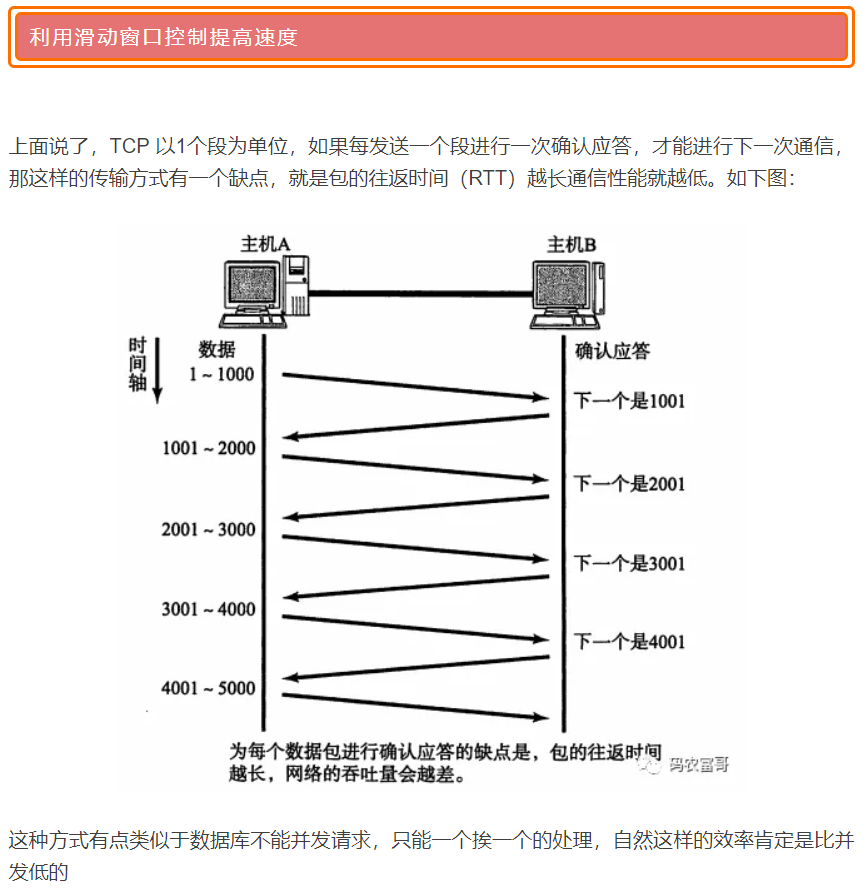
**发送端没有得到确认应答ACK的原因，主要分两种情况：**

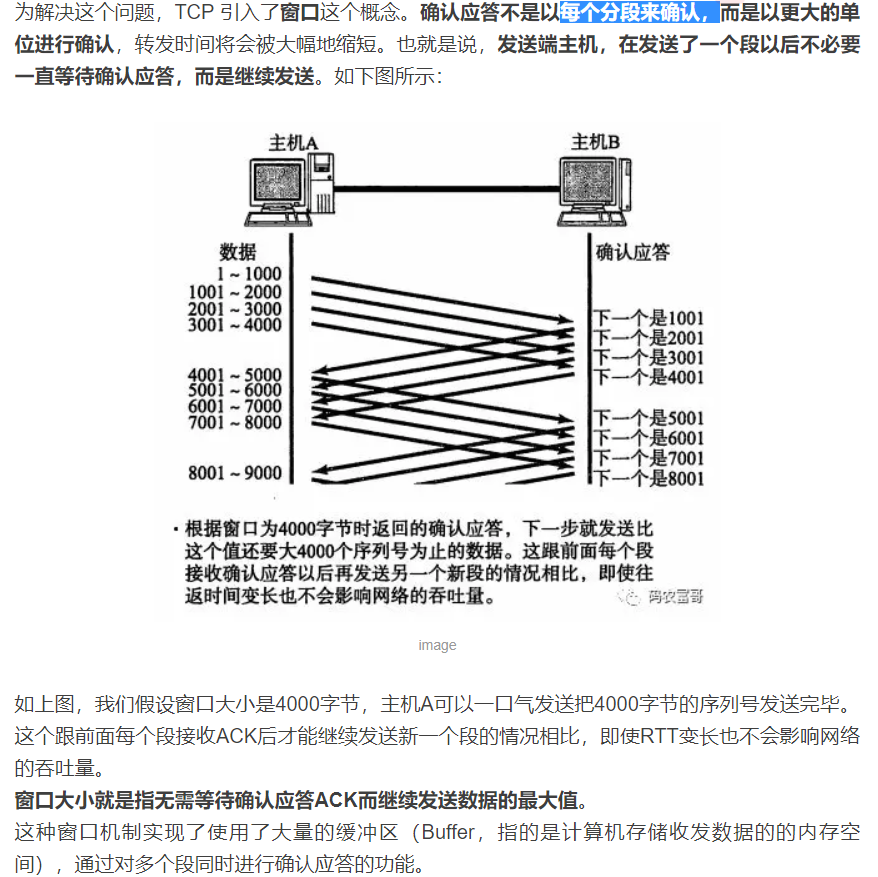
1. **发送端发送的数据丢包 2. 接收端发送的确认应答ACK丢包或延迟**

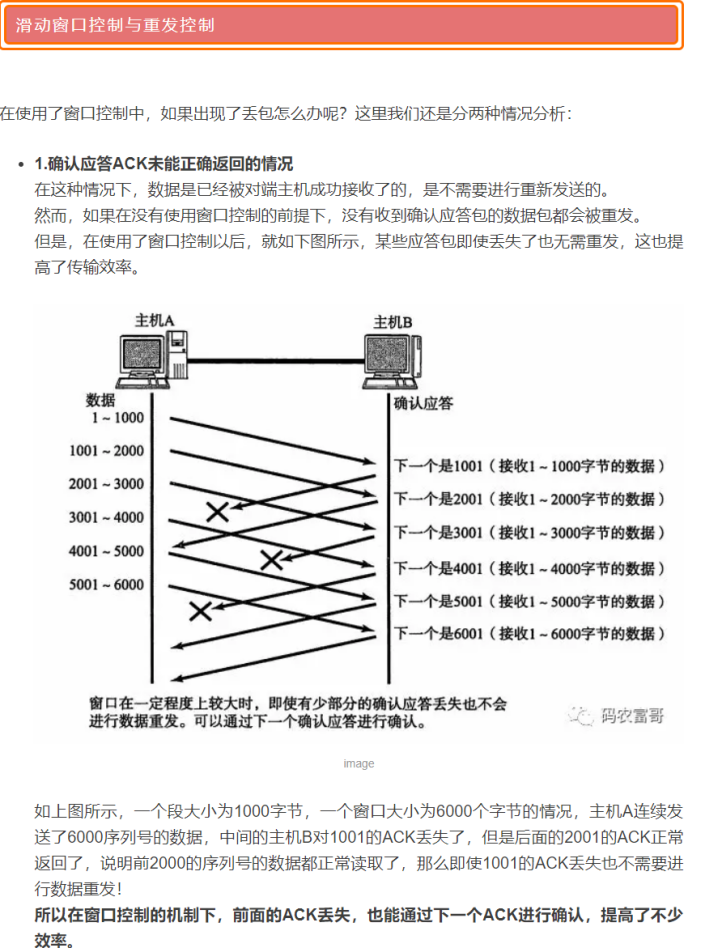
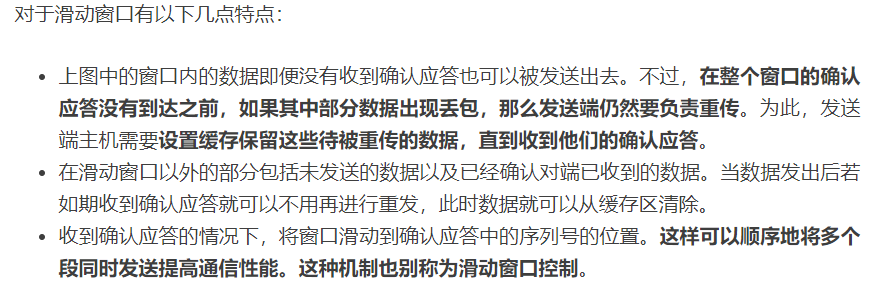
**通过序列号和确认应答号，TCP 能够识别是否已经接收数据，又能够判断是否需要接收，从而实现可靠传输 重发控制，重复控制**

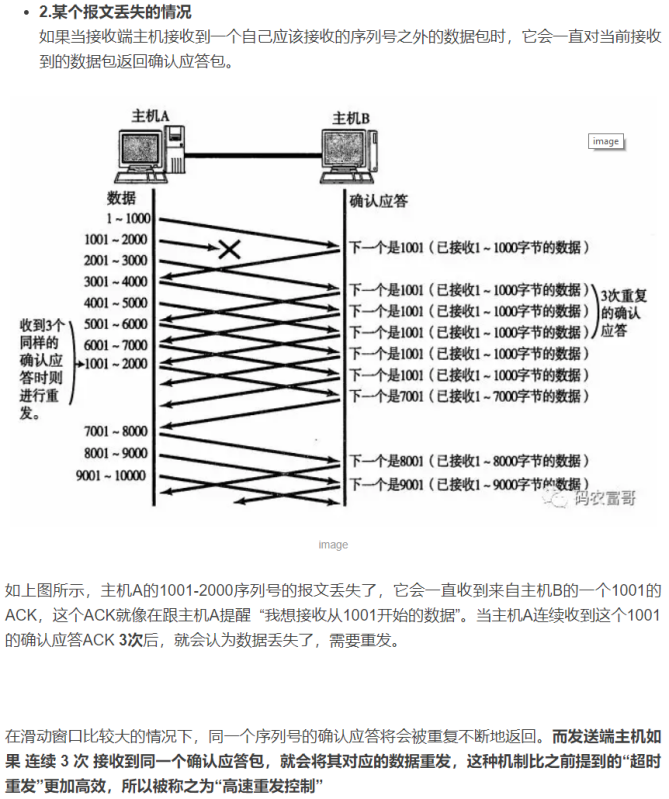
**超时重发如何确定呢？**

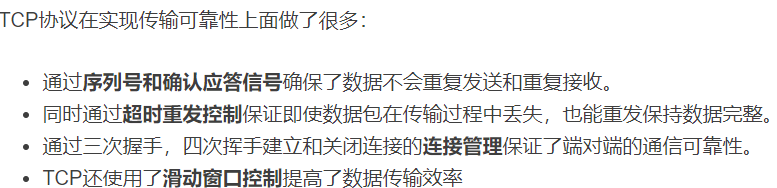
**重发超时是指在重发数据之前，等待确认应答到来的那个特定时间间隔 如果超过这个时间仍未收到确认应答，发送端将进行数据重发。**











**拥塞控制与流量控制的区别** <https://juejin.im/post/5e5757d3e51d4526db750611>

**流量控制** 是作用于**接收者**的，它是控制发送者的发送速度从而使接收者来得及接收，防止丢失数据包的。

**拥塞控制** 拥**塞控制是作用于网络的**，它是防止过多的数据注入到网络中，避免出现网络负载过大的情况

**流量控制** 所谓流量控制，就是让发送端不要发送的过快，让接收端能来得及接收

假设没有流量控制，发送端根据自己的实际情况发送数据，如果发送的速度太快，导致接收端的接收缓冲区很快填满了，此时发送端如果继续发送数据，接收端处理不过来，这时接收端就会把本来应该接收的数据丢弃，这会触发发送端的重发机制，从而导致网络流量的无端浪费。**所以TCP需要提供一种机制：让发送端根据接收端实际的接收能力控制发送的数据量。这就是所谓的流量控制**。TCP 利用滑动窗口实现流量控制的机制，而滑动窗口大小是通过TCP首部的窗口大小字段来通知对方。

**拥塞控制原理**

有了TCP的滑动窗口控制，收发主机之间即使不再以一个“段”为单位，而是以一个“窗口”为单位发送确认应答信号，所以发送主机够连续发送大量数据包。然而，如果在通信刚开始的时候就发送大量的数据包，也有可能会导致网络的瘫痪。

在拥塞控制中，发送方维持一个叫做**拥塞窗口**cwnd（congestion window）的状态变量。拥塞窗口的大小取决于网络的拥塞程度，并且动态地在变化。

发送窗口取拥塞窗口和接收端窗口的最小值，避免发送接收端窗口还大的数据。

拥塞控制使用了两个重要的算法： 慢启动算法， 拥塞避免算法。

**慢启动算法**： 慢启动算法的思路是，不要一开始就发送大量的数据，先试探一下网络的拥塞程度，也就是说由小到大逐渐增加拥塞窗口的大小。**慢算法中，每个传输轮次后将 cwnd 加倍。**

**拥塞避免算法：** 拥塞避免算法也是逐渐的增大 cwnd 的大小，只是采用的是**线性增长**而不是像慢启动算法那样的指数增长**。**

**快重传 快恢复**

<https://blog.csdn.net/qq_36953135/article/details/77506009>