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,位于传输层,TCP为应用层提供全双工服务。这意味数据能在两个方向上独立地进行传输。因此，连接的每一端必须保持每个方向上的传输数据序号。确认号，和序列号类似，**不过它是用来确认已经收到的序号并下次想收到的序号**

### Acknowledgement of delay

通常TCP在收到数据的时候不会立刻发送一个ACK确认，它会延迟发送，可以和对方需要的数据一起发送（数据捎带ACK）或者是等待第二个数据来了直接回复第二个ACK，通常的实现采用的延迟是200ms(就是说它会等待200ms有没有数据一起发送)

### Nagle

在数据传输过程中，通常会遇到一些小分组的传输,为了提传输效率，所以提出了Nagle算法。Nagle算法的基本定义是任意时刻，最多只能有一个未被确认的小段。 所谓“小段”，指的是小于MSS尺寸的数据块，所谓“未被确认”，是指一个数据块发送出去后，没有收到对方发送的ACK确认该数据已收到。****只有收到前一数据(大小小于MSS)的ACK消息时，Nagle算法才发送下一数据,****采用了Nagle算法会让用户感觉到时延

### Retransmission

重传超时时间(RTO)根据往返时间(RTT)计算得到

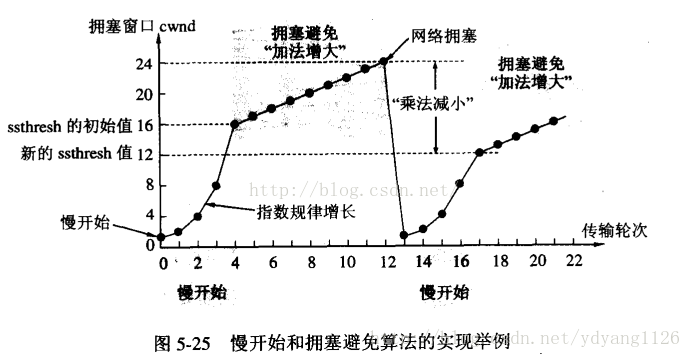
### 滑动窗口

确认可靠保序,并且控制流量,因为tcp是全双工，所以两边都有滑动窗口,但是其中会遇到一个问题，就是接收方发送的的的窗口更新数据丢失，这样会让发送方进入到无限等待状态，因为他要等待窗口更新为非0。为了解决这个问题TCP采用了坚持定时器（persist timer）去探测窗口更新。

TCP的流量控制是由连接的每一端通过声明窗口大小来提供。Window size 是一个16bit的字段，所以窗口最大为65535。

### TCP 拥塞控制

拥塞控制算法主要有四种：慢启动，拥塞避免，快速重传和快速恢复。

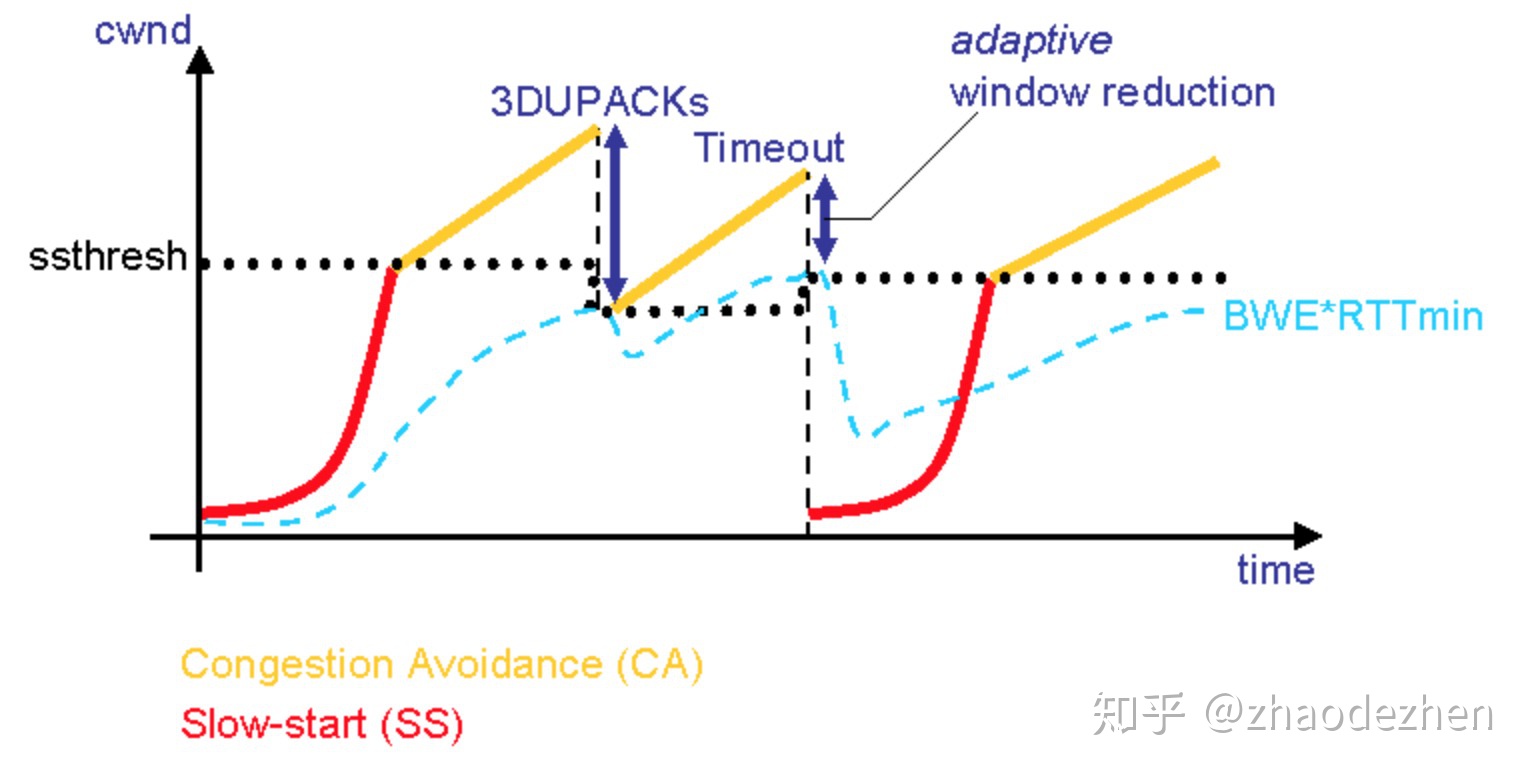


从慢启动状态到拥塞避免

当拥塞发生时，ssthresh被设置为当前窗口大小的一半（cwnd和接收方通告窗口大小的最小值，但最少为2个报文段）。如果是超时重传，cwnd被设置为1个报文段（这就是慢启动，其实慢启动也不慢，它是指数性增长，只是它的起始比较低）当达到ssthresh时，进入拥塞避免算法（拥塞避免是线性增长）。

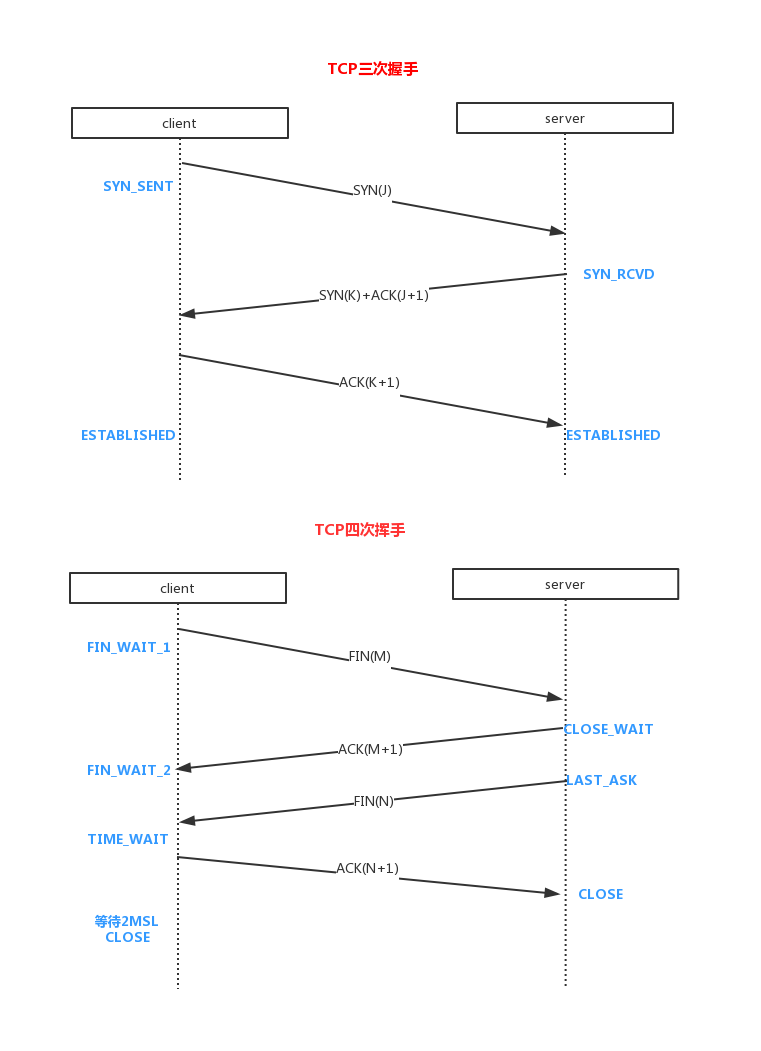
快速重传和快速恢复

当接收端收到一个顺序混乱的数据，它应该立刻回复一个重复的ACK。这个ACK的目的是通知发送端收到了一个顺序紊乱的数据段，以及期望的序列号,在收到三个重复ACK之后（包含第一次收到的一共四个同样的ACK），TCP不等重传定时器超时就重传看起来已经丢失（可能数据绕路并没有丢失）的数据段。因为这个在网络上并没有超时重传那么恶劣，所以不会进入慢启动，**而进入快速恢复**。快速恢复首先会把ssthresh减半(一般还会四舍五入到数据段的倍数)，然后cwnd=ssthresh+收到重复ACK报文段累计的大小。



慢启动窗口增长速度是指数的

### TCP 握手挥手



因为网络环境是复杂多变的，有可能自己的最后一个ACK丢失导致对方重传FIN。所以主动发起方要等待2MSL来预防对方重传

ACK：表示确认字段是重要的。客户端发送的初始SYN数据包之后的所有数据包都应设置此标志。   
PSH：推送功能。要求将缓冲的数据推送到接收应用程序。   
RST：重置连接   
SYN：同步序列号。只有从每一端发送的第一个数据包应该设置此标志。其他一些标志和字段会根据此标志更改含义，有些仅在设置时有效，有些仅在明确时有效。   
FIN：来自发送方的最后一个数据包。