### 慢开始和拥塞避免算法

**1.1 慢开始**

拥塞控制的目的是：防止大量数据涌入网络，防止路由器和链路过载。拥塞控制是全局性的过程，涉及到所有的主机、路由器。

慢开始算法，发送方维护一个拥塞窗口（cwnd），拥塞窗口的大小取决于网络拥塞情况，动态变化，发送窗口==拥塞窗口。

发送方控制拥塞窗口的原则是：只要网络没有出现拥塞，拥塞窗口就再增大一些，以便把更多的分组发送出去。但只要网络出现拥塞，拥塞窗口就减小一些，以减少注入到网络中的分组数。

慢开始算法: 连接刚刚建立时发送少量数据，探测网络拥塞情况，再由小到大增加拥塞窗口大小。拥塞窗口大小的单位是MSS，即一条报文的最大长度。最初，窗口大小为1，每收到一个报文ack就会将窗口大小+1，因此每一批次结束后，窗口大小会加倍，1、2、4、8。慢开始的慢指的是最初拥塞窗口很小，但增长速率快，指数级增长。

为了防止拥塞窗口增长过快，设置一个增长阈值，ssthresh：

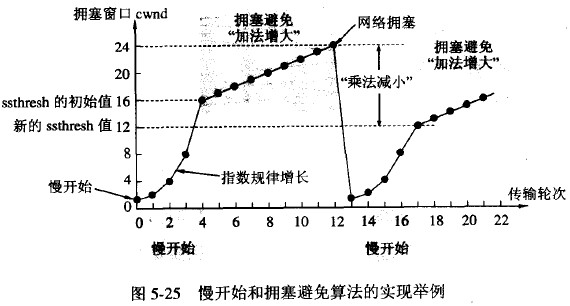
当 cwnd <= ssthresh 时，使用上述的慢开始算法。

当 cwnd > ssthresh 时，停止使用慢开始算法而改用拥塞避免算法。

**1.2 拥塞避免**

让拥塞窗口缓慢增大，每个发送批次后，窗口大小增加1，而不是加倍。线性增长，避免出现网络拥塞。

无论是慢开始或拥塞避免阶段，只要发送方判断网络出现拥塞（其根据就是没有收到确认），就要把慢开始门限ssthresh设置为出现拥塞时的发送 方窗口值的一半（但不能小于2）。然后把拥塞窗口cwnd重新设置为1，执行慢开始算法。称为“乘法减小”。这样做的目的就是要迅速减少主机发送到网络中的分组数，使得发生 拥塞的路由器有足够时间把队列中积压的分组处理完毕。



### 快重传和快恢复

**2.1 快重传**

慢开始和拥塞避免是网络拥塞控制的基础模型。在此基础上做了一些优化。

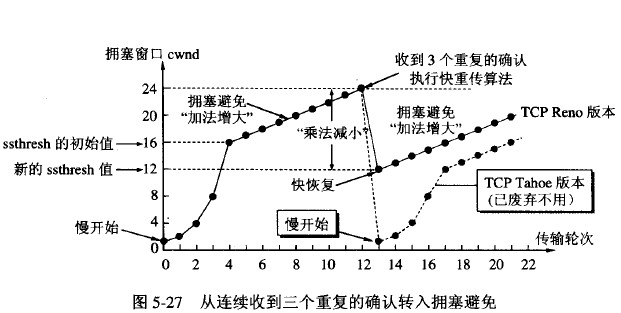
前提：接收端只要收到顺序报文时才返回ack，如果是乱序报文，例如报文1、报文2、报文4，接收端收到4时可以什么都不做。此时报文3可能因为拥塞丢失。发送端发现超时了超时了，没收到报文3的ack，客户端会从报文3开始重发后面的所有报文。

快重传的优化：当接收端收到乱序报文时，返回最后一个顺序报文ack，即收到4时返回“已经收到报文2”。当客户端连续3次收到同一条报文的ack时会重传下一条报文，即报文3。目的是在超时之前将丢失报文补上，不免因为个别报文的丢失，重发后续报文。信道利用率更高，吞吐量也更大。通过提高效率，减低网络拥塞。

**2.2 快恢复**

与开重传配合使用，当收到连续3条重复确认，说明网络中出现拥塞，就要执行“乘法减小”。但快恢复不执行“乘法减小”，而是把cwnd值设置为 慢开始门限ssthresh减半后的数值，然后开始执行拥塞避免算法（“加法增大”），使拥塞窗口缓慢地线性增大。当前TCP采用了快重传+快恢复的拥塞避免策略。

采用快恢复的理由是，既然接收端能收到其他分组（乱序报文），说明网络拥塞程度轻，无需将拥塞窗口大小设置为1，可以避免慢开始，直接跳过慢开始进入拥塞避免阶段。



也有的快重传实现是把开始时的拥塞窗口cwnd值再增大一点，即等于 ssthresh + 3 \* MSS 。这样做的理由是：既然发送方收到三个重复的确认，就表明有三个分组已经离开了网络。这三个分组不再消耗网络 的资源而是停留在接收方的缓存中。可见现在网络中并不是堆积了分组而是减少了三个分组。因此可以适当把拥塞窗口扩大了些。

注意，慢开始算法只在TCP连接刚建立时生效，其他时候快恢复、拥塞避免生效。

### 接收端参与控制

上面4中策略中，拥塞窗口大小只由发送端控制，但实际上接收方发送ack时，会把接收窗口大小（rwnd）发给发送端，发送端再决定发送窗口时使用min(cwnd, rwnd)，cwnd是计算后的发送窗口大小。通过适配接收方的窗口大小，降低拥塞。