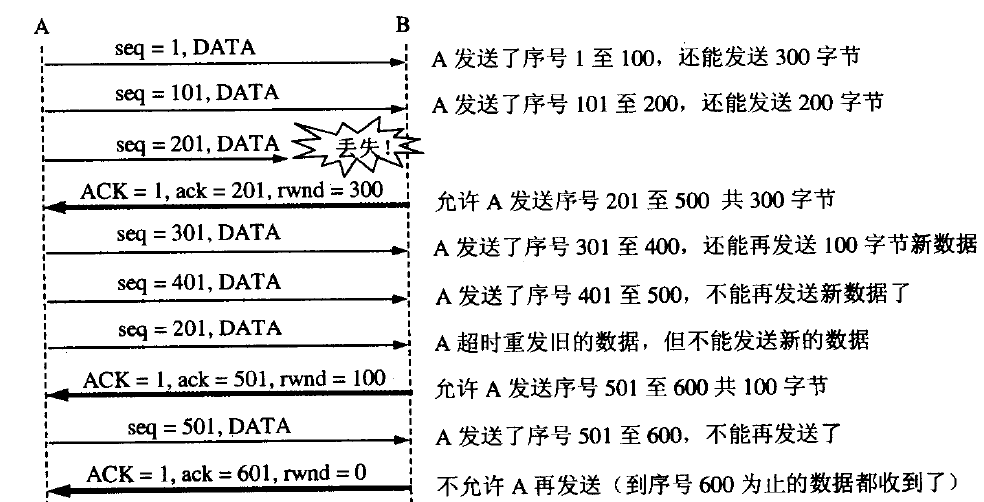
## 1.TCP的流量控制

所谓的**流量控制就是让发送方的发送速率不要太快，让接收方来得及接受**。利用滑动窗口机制可以很方便的在TCP连接上实现对发送方的流量控制。**TCP的窗口单位是字节，不是报文段，发送方的发送窗口不能超过接收方给出的接收窗口的数值。**



如图所示，说明了利用可变窗口大小进行流量控制。设主机A向主机B发送数据。双方确定的窗口值是400.再设每一个报文段为100字节长，序号的初始值为seq=1,图中的箭头上面大写ACK，表示首部中的却认为为ACK，小写ack表示确认字段的值。

     接收方的主机B进行了三次流量控制。第一次把窗口设置为rwind=300，第二次减小到rwind=100最后减到rwind=0，即不允许发送方再发送过数据了。这种使发送方暂停发送的状态将持续到主机B重新发出一个新的窗口值为止。

     假如，B向A发送了零窗口的报文段后不久，B的接收缓存又有了一些存储空间。于是B向A发送了rwind=400的报文段，然而这个报文段在传送中丢失 了。A一直等待收到B发送的非零窗口的通知，而B也一直等待A发送的数据。这样就死锁了。为了解决这种死锁状态，TCP为每个连接设有一个持续计时器。只 要TCP连接的一方收到对方的零窗口通知，就启动**持续计时器**，若持续计时器设置的时间到期，就发送一个零窗口**探测报文段**（仅携带1字节的数据），而对方就在确认这个探测报文段时给出了现在的窗口值。

### TCP报文段发送时机的选择

     TCP报文段发送时机主要有以下几种选择途径。

     1）TCP维持一个变量，它等于**最大报文段长度MSS**，只要缓存中存放的数据达到MSS字节就组装成一个TCP报文段发送出去。

     2）由发送方的应用程序指明要求发送报文段，即TCP支持的**推送**操作

     3）是发送方的一个计时器期限到了，这时就把当前已有的缓存数据装入报文段发送出去。

## 2. TCP的拥塞控制（两个图）

1.拥塞控制的原理

∑对资源的需求>可用资源

     所谓拥塞控制就是防止过多的数据注入到网络中，这样可以使网络中的路由器或链路不致过载。

流量控制往往指的是点对点通信量的控制，是个端到端的问题。

拥塞控制是很难设计的，因为它是一个动态的问题，许多情况下，甚至正是拥塞控制机制本身成为引起网络性能恶化甚至死锁的原因

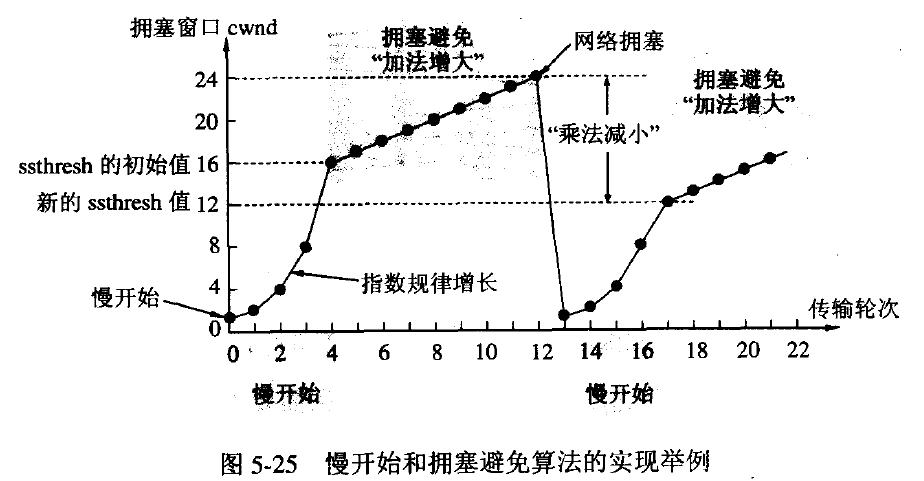
因特网建议标准RFC2581定义了进行拥塞控制的四种算法，即**慢开始（Slow-start)、拥塞避免（Congestion Avoidance)、快重传（Fast Restrangsmit)和快回复（Fast Recovery）**

**2.2 慢开始（Slow-Start）和拥塞避免（Congestion Avoidance）结合**

**慢开始算法：**是指开始发送数据时，并不清楚网络的负荷情况，会先发送一个1字节的试探报文，当收到确认后，就发送2个字节的报文，继而4个，8个以此指数类推。

（需要注意的是，慢开始的“慢”并不是指拥塞窗口的增长速率慢，而是指在TCP开始发送报文时先设置拥塞窗口=1。）

**拥塞避免算法：**是让拥塞窗口缓慢地增大，即cwnd加1，而不是如慢开始算法一样加倍。



#### ****根据上图的实例进行分析，一开始的慢开始算法的指数增长是很恐怖的，所以为了防止拥塞窗口cwnd增长过快需要设置一个门限ssthresh，这里是16。****

#### ****（1）当 cwnd < ssthresh 时，使用上述的慢开始算法。****

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

#### ****（3）当 cwnd = ssthresh 时，既可使用慢开始算法，也可使用拥塞控制避免算法。****

#### ****无论在慢开始阶段还是在拥塞避免阶段，只要发送方没有收到确认，就认为这时候拥塞了，就要把慢开始门限ssthresh设置为此时发送方窗口值的一半（上例中是把发送方窗口值24修改为12）。然后把拥塞窗口cwnd重新设置为1，执行慢开始算法。****

#### ****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****

#### ****乘法减小和加法增大****

**乘法减小：是指不论在慢开始阶段还是拥塞避免阶段，只要出现超时，就把慢开始门限减半，即设置为当前的拥塞窗口的一半（于此同时，执行慢开始算法）。当网络出现频繁拥塞时，ssthresh值就下降的很快，以大大将小注入到网络中的分组数。**

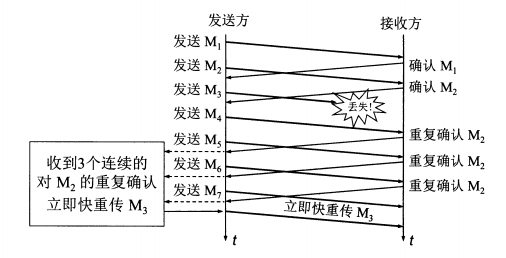
**加法增大：是指执行拥塞避免算法后是拥塞窗口缓慢增大，以防止网络过早出现拥塞。**

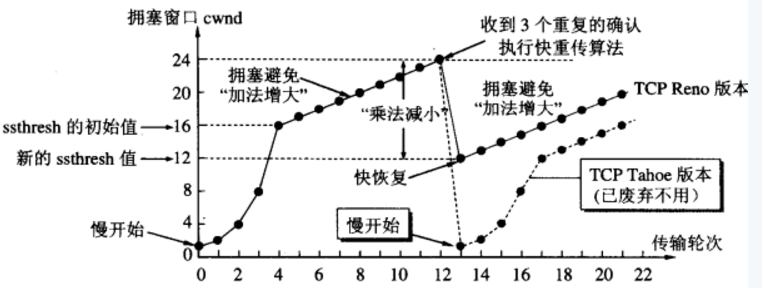
**2.3 快重传（Fast Retransmit）和快恢复（Fast Recovery）结合**

**快重传是指，如果发送端接收到3个以上的重复ACK，不需要等到重传定时器溢出就重新传递，所以叫做快速重传，而快速重传以后，因为走的不是慢启动而是拥塞避免算法，所以这又叫做快速恢复算法。**

**如果没有快速重传和快速恢复，TCP将会使用定时器来要求传输暂停。在暂停这段时间内，没有新的数据包被发送。所以快速重传和快速恢复旨在快速恢复丢失的数据包。**

**---------------------**



与快重传配合使用的还有快恢复算法，结合上图的实例来分析，其过程有以下两个要点：

（1）当发送方在cwnd=24时连续收到三个重复确认，就把慢开始门限ssthresh减半（就是上图中的24修改为12）。

（2）接下来**不执行慢开始算法，而是把cwnd值设置为门限ssthresh减半后的数值（即cwnd不是设置为1而是设置为12），然后开始执行的是拥塞避免算法**，使拥塞窗口缓慢地线性增大。

这里为什么替换掉了慢开始算法呢？

这是因为收到重复的ACK不仅仅告诉我们一个分组丢失了，由于接收方只有在收到另一个报文段时才会产生重复的ACK，所以还告诉我们该报文段已经进入了接收方的缓存。也就是说，在收发两端之间仍然有**流动的数据**，而我们**不想执行慢启动来突然减少数据流**。