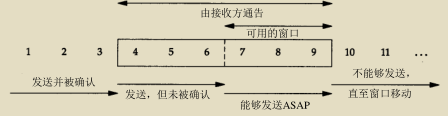


1. 发送端在接收到ACK报文后根据ACK报文中的窗口字段来调整发送窗口的大小，发送窗口的大小调整只能是左边界右移（缩小），或者右边界右移（扩大）。调整完发送端口的大小后，移动整个发送窗口，使窗口的左边界定位在ACK报文返回的ACK号上。
2. 在下图中，我们将字节从 1至11进行标号。接收方通告的窗口称为提出的窗口（offere dwindow），它覆盖了从第4字节到第9字节的区域，表明接收方已经确认了包括第 3字节在内的数据，且通告窗口大小为 6。窗口大小是与确认序号相对应的。发送方计算它的可用窗口，该窗口表明多少数据可以立即被发送。当接收方确认数据后，这个滑动窗口不时地向右移动。窗口两个边沿的相对运动增加或减少了窗口的大小。我们使用三个术语来描述窗口左右边沿的运动：

1) 称窗口左边沿向右边沿靠近为窗口合拢。这种现象发生在数据被发送和确认时。

2) 当窗口右边沿向右移动时将允许发送更多的数据，我们称之为窗口张开。这种现象发生在另一端的接收进程读取已经确认的数据并释放了 T C P的接收缓存时。

3) 当右边沿向左移动时，我们称之为窗口收缩。 Host Requirements RFC强烈建议不要使用这种方式。但T C P必须能够在某一端产生这种情况时进行处理。



1. 慢启动

发送方一开始便向网络发送多个报文段，直至达到接收方通告的窗口大小为止，当发送方和接收方处于同一个局域网时，这种方式是可以的。但是如果在发送方和接收方之间存在多个路由器和速率较慢的链路时，就有可能出现一些问题。一些中间路由器必须缓存分组，并有可能耗尽存储器的空间。 现在，TCP需要支持一种被称为“慢启动 (slow start)”的算法。该算法通过观察到新分组进入网络的速率应该与另一端返回确认的速率相同而进行工作。

慢启动为发送方的TCP增加了另一个窗口：拥塞窗口(congestion window)，记为c w n d。当与另一个网络的主机建立TCP连接时，拥塞窗口被初始化为 1个报文段（即另一端通告的报文段大小）。每收到一个ACK，拥塞窗口就增加一个报文段（cwnd以字节为单位，但是慢启动以报文段大小为单位进行增加）。发送方取拥塞窗口与通告窗口中的最小值作为发送上限。拥塞窗口是发送方使用的流量控制，而通告窗口则是接收方使用的流量控制。

发送方开始时发送一个报文段，然后等待 ACK。当收到该ACK时，拥塞窗口从1增加为2，

即可以发送两个报文段。当收到这两个报文段的 ACK时，拥塞窗口就增加为4。这是一种指数增加的关系。在某些点上可能达到了互联网的容量，于是中间路由器开始丢弃分组。这就通知发送方它的拥塞窗口开得过大。当我们在下一章讨论 TCP的超时和重传机制时，将会看到它们是怎样对拥塞窗口起作用的。现在，我们来观察一个实际中的慢启动。

1. TFTP使用了停止等待协议。数据发送方在发送下一个数据块之前需要等待接收对已发送数据的确认。而TCP所使用的被称为滑动窗口协议的另一种形式的流量控制方法。该协议允许发送方在停止并等待确认前可以连续发送多个分组。由于发送方不必每发一个分组就停下来等待确认，因此该协议可以加速数据的传输。
2. 滑动窗口的操作可能会产生一个严重的问题，这就是发送方产生数据很慢，或者接收方消耗数据很慢，或者两种情况都存在。以上的情况都会使传输的数据段很小，使网络效率很差。这就是糊涂窗口综合症。
3. 发送端产生的糊涂窗口综合症：

如果发送端为产生数据很慢的应用程序，例如，每次产生一个字节。这个应用程序每次将一个字节的数据写入发送端缓存，如果发送端没有特殊的指令，则它产生只包含一个字节数据的报文段，结果就会有很多41字节的报文段在互联网中传送。为了解决发送端的糊涂窗口，应该强迫发送端等待，让它收集数据以便发送大块数据。但如果等待太长时间则会使应用产生时延，如果等待时间太短则会产生较短报文段。Nagle算法是一个较好的方法：

1. 发送端TCP把它从应用程序收到的第一块数据发送出去，即使只有一个字节也发送。
2. 在发送第一个报文段后，发送端TCP就在输出缓存中积累数据并等待，直到接收端TCP返回确认，或者数据已经积累到可以装成最大发送长度的报文段。这时发送端就可以发送这个报文。
3. 对于剩下的数据的传输，不断重复第2步。
4. 接收端产生的糊涂窗口症状：

如果接收端消耗数据很慢，一次中消耗一个字节，那么接收端通知发送端窗口为1，则发送端一次只能发送一个字节，这样接收端就产生了糊涂窗口症状。有两种方法解决接收端的糊涂窗口症状：

1. Clark解决方法：Clark解决方法是只要有数据到就产即发送确认，但在缓存空间小于最大报文段长度以前，或者用缓存空间小于总缓存空间的一半以前，一直都向发送端宣布窗口大不为0。
2. 延迟的确认：当报文段到达时不产即发送确认，而是接收到足够多的数据后才发送确认，也就是说接收多个报文却只回复一个确认。延迟发送确认防止了发送端TCP滑动它的窗口。当发送端TCP发送完数据后就停下来了，这样就防止了糊涂窗口症状，并且减小了通信量。