第五章 传输层 TCP拥塞控制

出现拥塞的条件：对资源需求的总和>可用资源。
网络中有许多资源同时呈现供应不足->**网络性能变坏**->网络吞吐量将随输入负荷增大而下降

拥塞控制的目的：**防止过多的数据注入到网络中**。（是一个**全局性过程**）

拥塞控制和流量控制也有相似的地方，即它们都**通过控制发送方发送数据的速率**来**达到控制效果**。

拥塞控制与流量控制的区别：
（1）拥塞控制是让网络能够承受现有的网络的负荷，**是一个全局性的过程**；
（2）流量控制往往是指**点对点的通信量的控制**，即**接受端控制发送端，**它所要做的是抑制发送端发送数据的速率，以便使接受端来得及接收。

拥塞控制的四种算法：

- 慢开始

- 拥塞避免
- 快重传
- 快回复

假定：
1、**数据单方向传送**，而**另一个方向只传送确认**。
2、**接收方总是有足够大的缓存空**间，因而**发送窗口大小取决于拥塞程度**。
发送窗口=Min{接受窗口rwnd，拥塞窗口cwnd}

> 接受窗口：**接收方**根据接受缓存设置的值，并告知给发送方，**反映接受方容量**。
>
> 拥塞窗口：**发送方**根据自己估算的网络拥塞程度而设置的窗口值，**反应网络当前容量**。

慢开始和拥塞避免
（1）慢开始算法
在TCP**刚刚连接好并开始发送TCP报文段时**，**先令拥塞窗口cwnd=1**，即一个最大报文段长度MSS。每收到一个对新报文段的确认后，将cwnd加1，即增大一个MSS。

例如：A向B发送数据，发送时A的拥塞窗口为2，那么A一次可以发送两个TCP报文段，经过一个RTT后（也称一个**传输轮次**），A收到B对刚才两个报文的确认，于是把拥塞窗口调整为4，下一次发送时就可一次发送4个报文段。

使用慢开始算法后，每经过一个传输轮次（即往返时延RTT），拥塞窗口cwnd就会加倍，即cwnd的大小指数式增长。这样，慢开始一直把拥塞窗口cwnd增大到一个规定的慢开始门限ssthresh（阈值），然后改用拥塞避免算法。

（2）拥塞避免算法

拥塞避免算法的做法如下：发送端的拥塞窗口cwnd每经过一个往返时延RTT就**增加一个MSS的大小**，而不是加倍，使cwnd按线性规律缓慢增长（即加法增长）。

而当出现一次超时（网络拥塞）时，令慢开始门限ssthresh等于当前cwnd的一半（即乘法减小）。

根据cwnd 的大小执行不同的算法，可归纳如下：

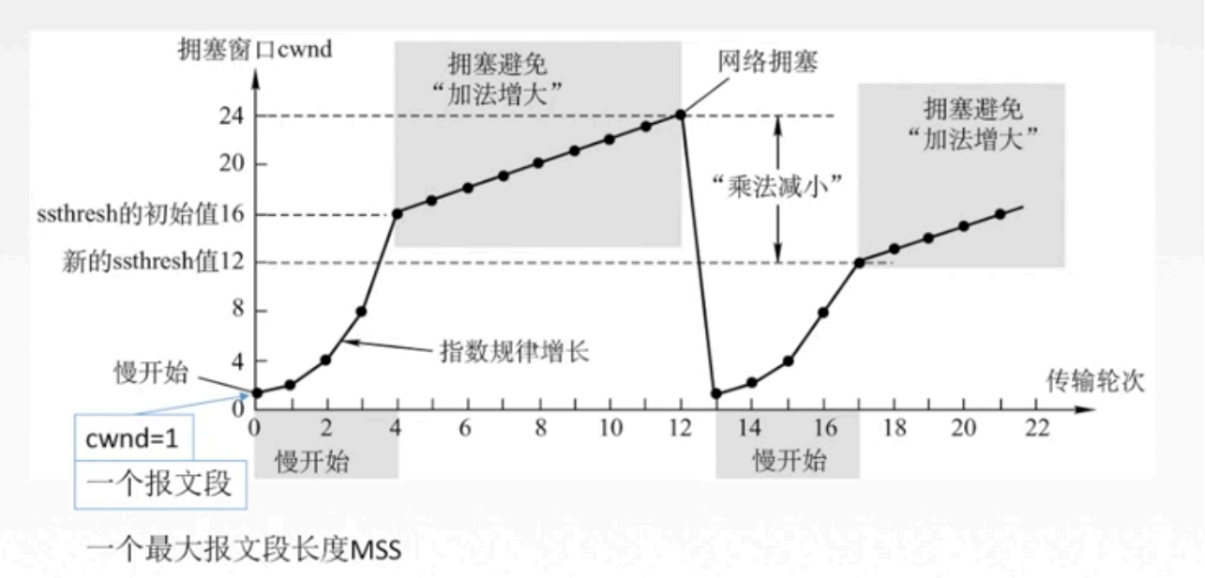
- 当cwnd<ssthresh时，使用慢开始算法。
- 当cwnd>ssthresh时，停止使用慢开始算法而改用拥塞避免算法
- 当cwnd=ssthresh时，即可使用慢开始算法，又可使用拥塞避免算法（通常做法）

（3）网络拥塞的处理

网络出现拥塞时，无论是在慢开始阶段还是在拥塞避免阶段，只要发送方检测到超时事件的发生（未按时收到确认，重传计时器超时），就要把慢开始门限ssthresh设置为出现拥塞时的发送方的cwnd值的一半（但不能小于2）。

然后把拥塞窗口重新设置为1，执行慢开始算法。

利用以上措施要完全避免网路拥塞是不可能的。



上图为慢开始和拥塞避免算法的实现过程：
1、初识时，**拥塞窗口置为1，即cwnd=1，慢开始门限置为16**，即**ssthresh=16**.慢开始阶段，cwnd的初值为1，以后发送方每收到一个确认ACK，cwnd值为1，也即经过每个传输轮次（RTT），cwnd呈指数规律增长。

2、拥塞窗口cwnd增长到慢开始门限ssthresh时，就改用拥塞避免算法，cwnd按线性规律加法增长。

3、假定cwnd=24时，网络发生拥塞，更新ssthresh的值为12，即变为超时时cwnd的值24的一半，cwnd重制为1，并执行慢开始算法，cwnd=12时，改为拥塞避免算法。

注意：在慢开始（指数增长）阶段，**若2cwnd>ssthresh。则下一个RTT的cwnd等于ssthresh，而不等于2cwnd**，即cwnd不能跃过ssthresh值。

快重传和快恢复

快重传和快恢复算法是对慢开始和拥塞避免算法的改进

（1）快重传

快重传技术使用了冗余ACK来检测丢包的发生。

同样，冗余ACK也用于网络拥塞额的检测（丢了包当然意味着网络可能出现了拥塞）。快重传并非取消重传计时器，而是在某些情况下可更早地重传丢失的报文段。

当发送方连续**收到三个重复的ACK报文时**，直**接重传对方尚未收到的报文段**，而**不必等待那个报文段设置的重传计时器超时**。

（2）快恢复

快恢复算法的原理如下：发送端收到连续三个冗余ACK（重复确认）时，执行“乘法减小”算法，把慢开始门限ssthresh设置为出现拥塞时发送方cwnd的一半，它把cwnd的值设置为慢开始门限ssthresh改变后的数值，然后开始执行拥塞避免算法，**使拥塞窗口缓慢地陷性增大**。

由于跳过了cwnd从1起始的慢开始过程，所以称为快恢复，快恢复算法的实现如下：虚线为慢开始的处理过程

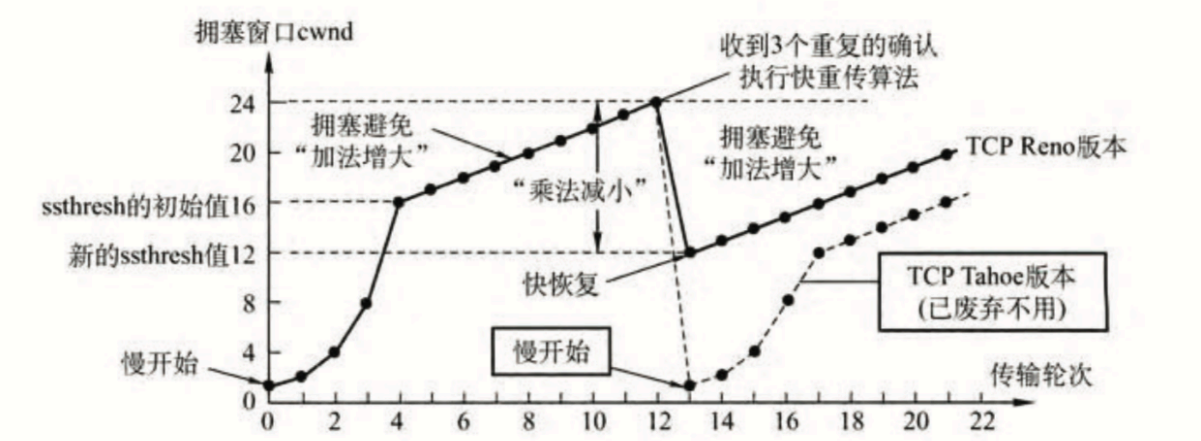


图 5.12　快恢复算法的实现过程