1 运输层

1.1 可靠传输的工作原理

1.1.1 停止等待协议

每发送完一个分组就停止发送，等待对方确认。在收到确认后再发送下一个分组。发送方只要超过了一段时间仍然没有收到确认，就认为刚才发送的分组丢失了，因而重传前面发送的分组。（超时重传）

注意三点：

第一：发送方在发送完一个分组后，必须暂存保留已发送的分组的副本。

第二：分组和确认分组都必须进行编号。

第三：超时重传时间应当比数组在分组传送的平均往返时间更长一些。

停止等待协议中可能发生确认丢失和确认迟到的情况，对于确认丢失，发送方再次重传，对于确认迟到，发送方也重传，但对于收到的重复确认丢弃，这种可靠传输协议称为自动重传请求**ARQ**

1.1.2 连续**ARQ**协议

为了提高效率，发送方不使用低效率的停止等待协议，而是采用流水线传输。发送方可连续发送多个分组，接收方采用累计确认的方式，在收到几个分组后，对按序到达的最后一个分组发送确认，发送方每收到一个确认，就将发送窗口向前滑动一个分组的位置。

1.2 TCP可靠传输的实现

1.2.1 以字节为单位的滑动窗口

滑动窗口协议是用来改善吞吐量的一种技术，即容许发送方在接收任何应答之前传送附加的包。接收方告诉发送方在某一时刻能送多少包（称窗口尺寸）。

TCP中采用滑动窗口来进行传输控制，滑动窗口的大小意味着接收方还有多大的缓冲区可以用于接收数据。发送方可以通过滑动窗口的大小来确定应该发送多少字节的数据。当滑动窗口为0时，发送方一般不能再发送数据报，但有两种情况除外，一种情况是可以发送紧急数据，例如，允许用户终止在远端机上的运行进程。另一种情况是发送方可以发送一个1字节的数据报来通知接收方重新声明它希望接收的下一字节及发送方的滑动窗口大小。

强调三点

第一：A的发送窗口是根据B的接受窗口设置的，但在同一时刻，A的发送窗口并不总是和B的接收窗口一样大。

第二：TCP通常对不按序到达的数据是先临时存放在接受窗口中，等待字节流中所缺少的字节收到后，再按序交付上层的应用进程。

第三：TCP要求接收方必须有累计确认的功能，这样减少传输开销。接收方不应过分推迟发送确认，否则会导致发送方不必要的重传。

1.2.2 超时重传时间的选择

TCP采用一种自适应算法，报文段的往返时间RTT，加权平均往返时间RTTs

1.2.3 选择确认SACK

只传送缺少的数据而不重传已经正确到达接收方的数据。

1.3 TCP流量控制

1.3.1 利用滑动窗口实现流量控制

流量控制：让发送方的发送速率不要太快，要让接收方来得及接收。

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。

TCP为每一个连接设置一个续时计时器。只要TCP连接的一方收到对方的零窗口通知，就启动计时器，若计时器设置的时间到期，就发送一个零窗口探测报文段，而对方就在确认这个探测报文段时给出现在的窗口值，如果窗口仍然是0，那么收到这个报文段的一方，就重新设置计时器。如果不是0，那么思索的僵局就可以打破了。

1.4 TCP的拥塞控制

1.4.1 拥塞控制的一般原理

拥塞控制就是防止过的的数据注入到网络中，这样可以是网络中的路由器或链路不致过载。拥塞控制所要做的是都有一个前提，就是万国城承受现有的网络负荷，拥塞控制是一个全局性的过程。

流量控制往往指点对点通信量的控制，是一个端到端的问题（接收端控制发送端）。流量控制所要做的就是抑制发送端发送数据的速率，一边接收端来得及接受。

进行拥塞控制需要付出代价。这话搜先需要获得网络内部流量的分布信息。在实施拥塞控制时，还需要在节点之间交换信息和各种命令，以便选择控制的策略和实施控制。

1.4.2 拥塞控制方法

1.4.2.1 慢开始和拥塞控制

慢开始：在主机刚刚开始发送报文段时可先将拥塞窗口cwnd设置为一个最大报文段MSS的数值。在每收到一个对新的报文段的确认后，将拥塞窗口增加至多一个MSS(最大报文段)的数值。用这样的方法逐步增大发送端的拥塞窗口cwnd，可以分组注入到网络的速率更加合理。

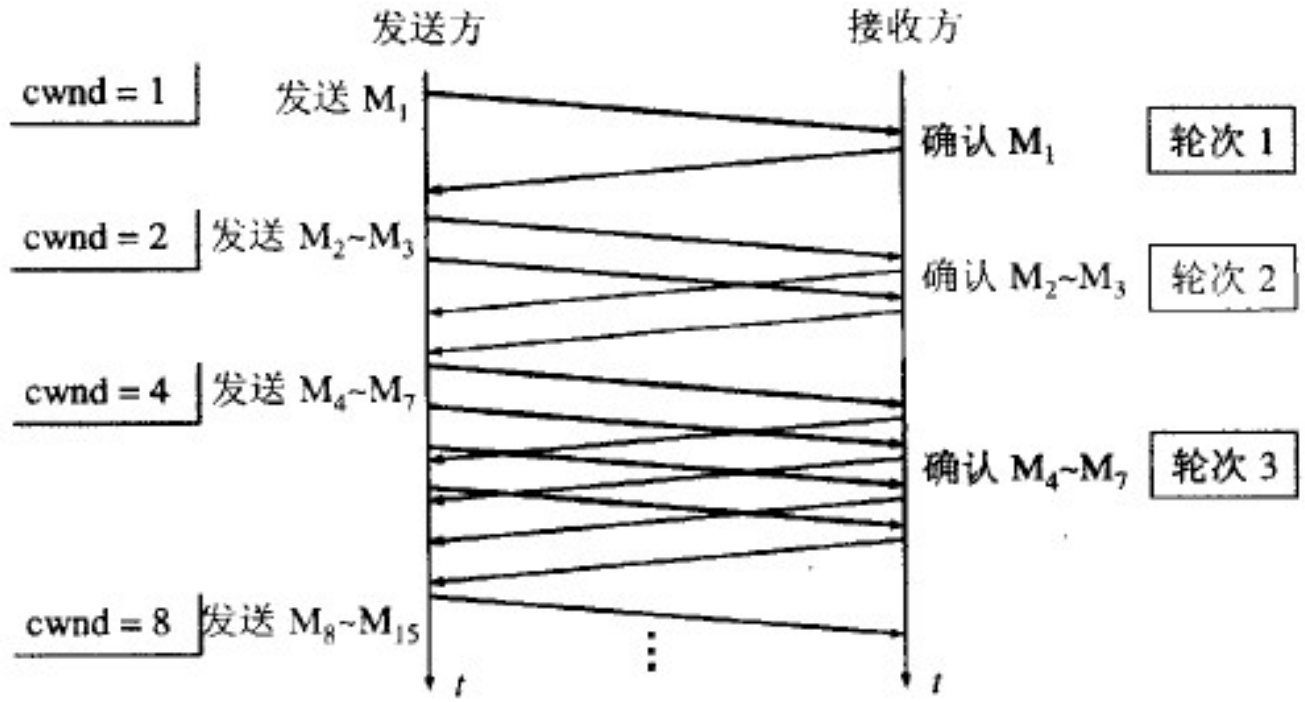


图 5-24　发送方每收到一个确认就把窗口 cwnd 加 1

拥塞避免：当拥塞窗口值大于慢开始门限时，停止使用慢开始算法而改用拥塞避免算法。拥塞避免算法使发送的拥塞窗口每经过一个往返时延RTT就增加一个MSS的大小。

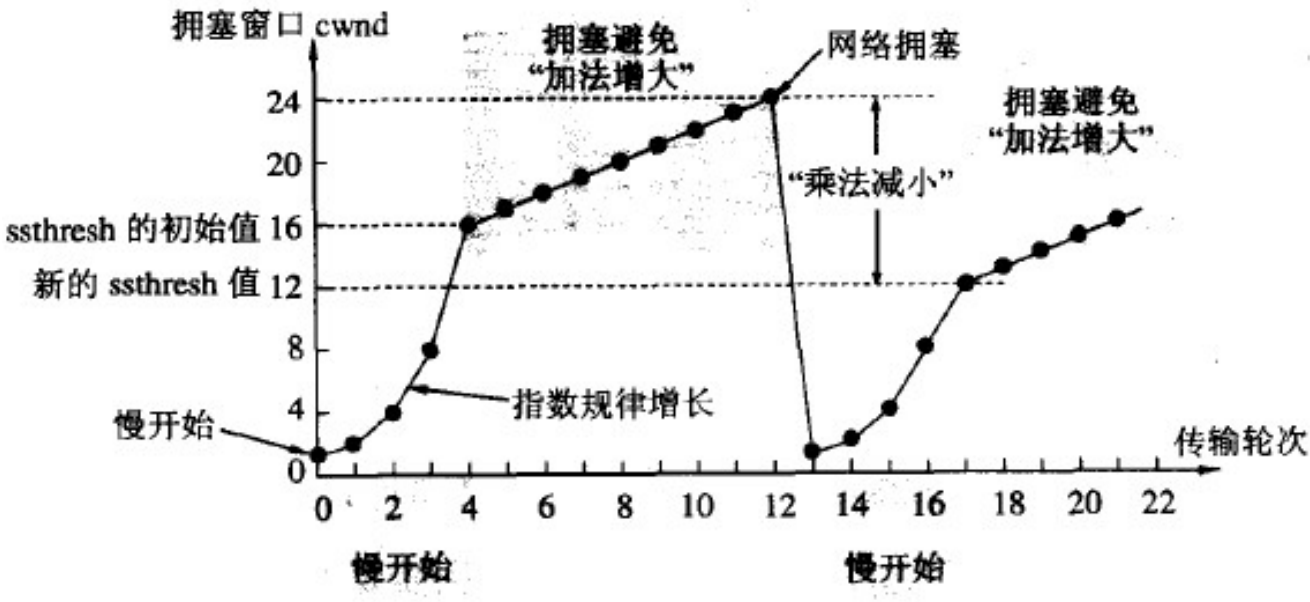


图 5-25　慢开始和拥塞避免算法的实现举例

乘法减小：是指不论在慢开始阶段还是拥塞避免阶段，只要出现一次超时（即出现一次网络拥塞），就把慢开始门限值 ssthresh 设置为当前的拥塞窗口值乘以 0.5。当网络频繁出现拥塞时，ssthresh 值就下降得很快，以大大减少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，在收到对所有报文段的确认后（即经过一个往返时间），就把拥塞窗口 cwnd增加一个 MSS 大小，使拥塞窗口缓慢增大，以防止网络过早出现拥塞。

1.4.2.2 快重传和快恢复

快重传算法规定：发送端只要一连收到三个重复的ACK即可断定有分组丢失了，就应该立即重传丢手的报文段而不必继续等待为该报文段设置的重传计时器的超时。

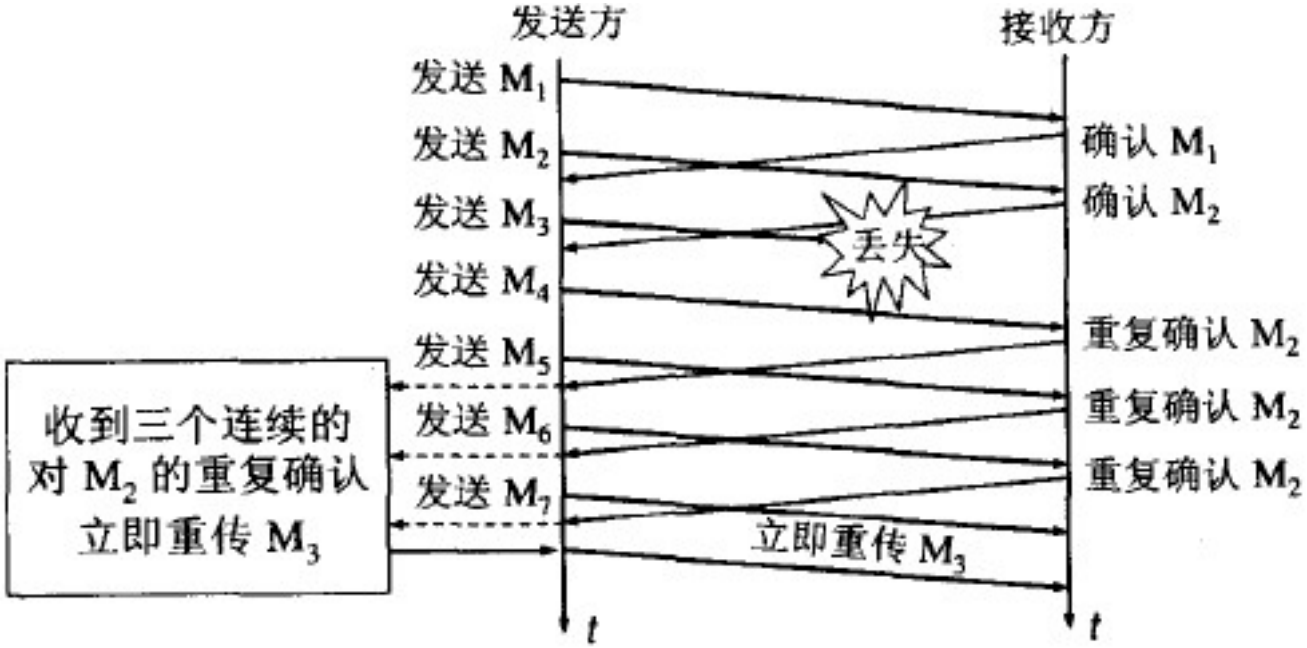


图 5-26　快重传的示意图

快恢复算法：当发送端收到连续三个重复的ACK时，就执行“乘法减小”算法，把慢开始门限ssthresh减半。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，使拥塞窗口缓慢的线性增加。

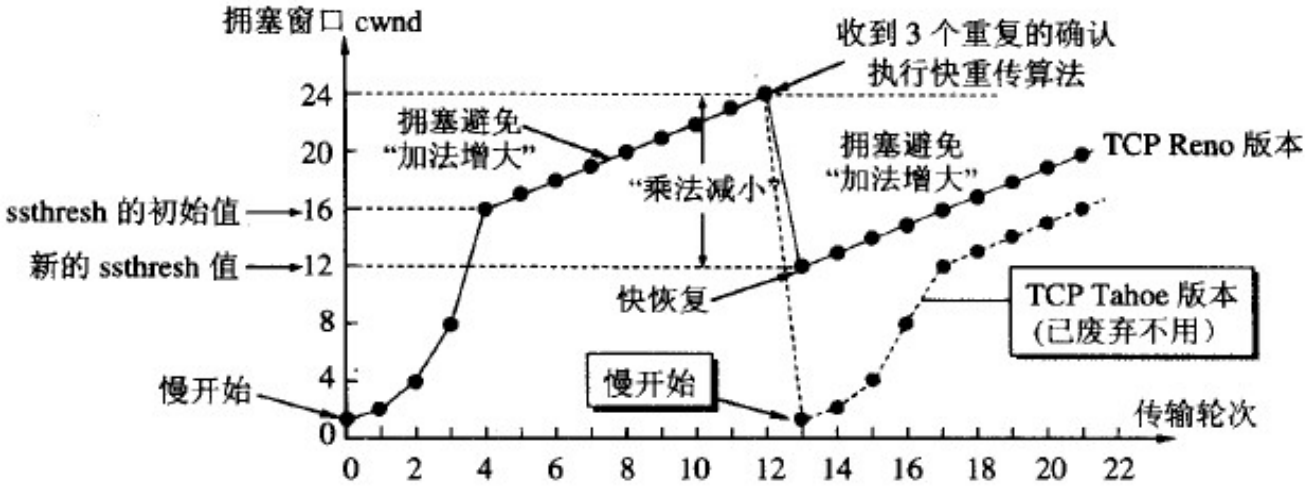


图 5-27　从连续收到三个重复的确认转入拥塞避免

注意：发送窗口的上限值应当取为接收窗口rwnd和拥塞窗口cwnd中较小的一个。

1.4.3 随机早期检测RED

网络层的策略对TCP拥塞控制影响最大的就是路由器的分组丢弃策略。路由器的队列通常是按照FIFO的规则处理到来的分组。

路由器的队列维持两个参数，队列长度最小门限THmin和最大门限THmax。**RED**算法如下：

（1）若平均队列长度小于最小门限，则把新到达的分组放入队列进行排队。

（2）若平均队列长度大于最大门限，则把新到达的分组丢弃。

（3）若平均队列长度在最小门限与最大门限之间，则按照某一概率将新到达的分组丢弃。

1.5 TCP的运输连接管理

1.5.1 TCP建立连接

TCP(Transmission Control Protocol)　传输控制协议

TCP是主机对主机层的传输控制协议，提供可靠的连接服务，采用三次握手确认建立一个连接:

位码即tcp标志位,有6种标示:

SYN(synchronous建立联机)

ACK(acknowledgement 确认)

PSH(push传送)

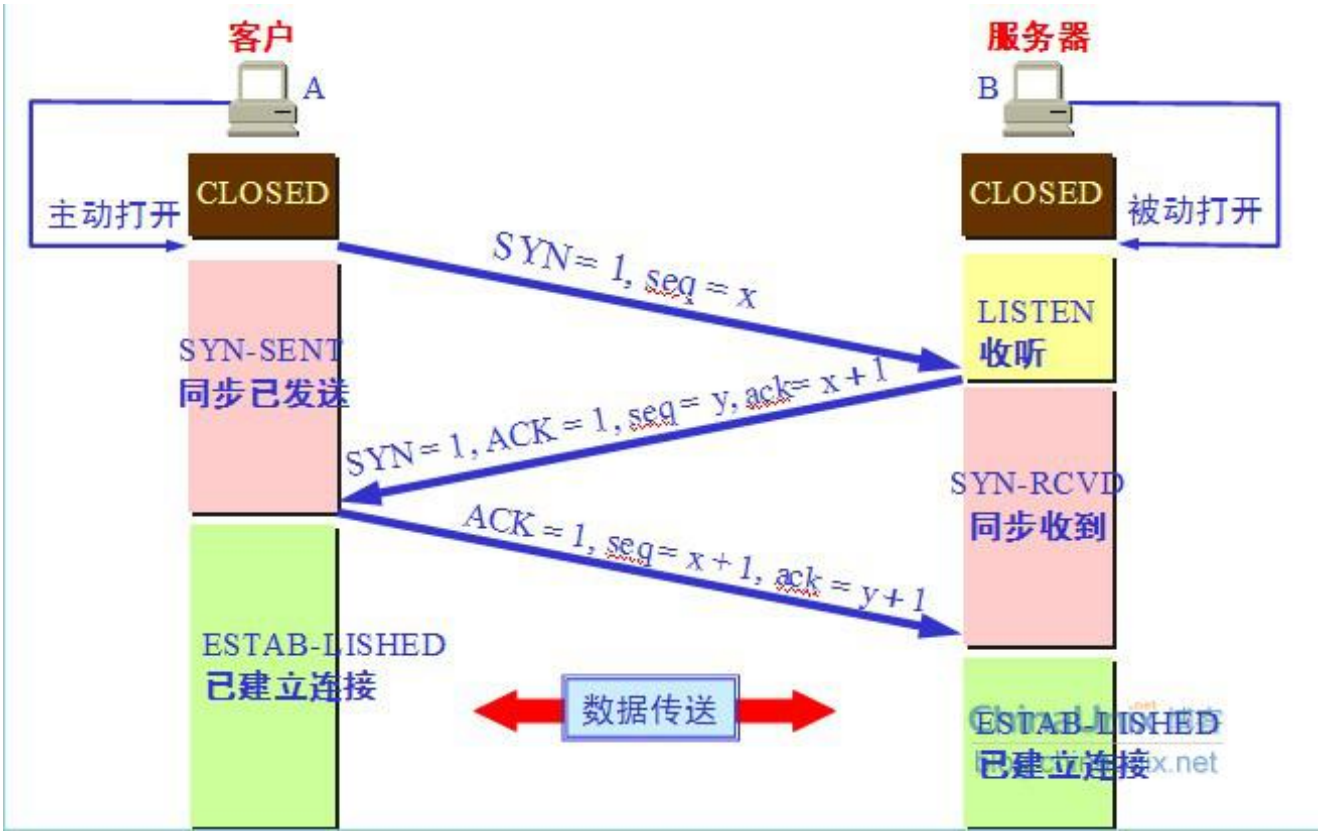
FIN(finish结束)

RST(reset重置)

URG(urgent紧急)

Sequence number(顺序号码)

Acknowledge number(确认号码)



1.B的TCP服务器进程先创建传输控制块TCB（Transmission Control Block 存储了每个连接中的一些重要信息，如：TCP连接表，到发送和接收缓存的指针，到重传队列的指针，当前的发送和接受序号，等）,准备接收客户进程的连接请求，然后服务器进程就处于LISTEN状态,等待客户的连接请求.如有,就做出响应.

A的TCP客户进程也是先创建传输控制块,然后向B发出连接请求报文段,这时候首部中的同步位SYN=1,同时选择一个初始序号 seq=x。TCP规定,SYN报文段(SYN=1的报文段) 不能携带数据,但要消耗一个序号.这时,TCP客户进程进入SYN-SENT状态.

2.B收到连接请求报文段后,如同意建立连接,则向A发送确认.在确认报文段中,应把SYN和ACK位都置1,确认号是ack=x+1，同时也为自己选择一个初始序号seq=y。这个报文段也不能携带数据,同样要消耗一个序号.这时TCP服务器进程进入SYN-RCVD(同步收到)状态.

3.TCP客户进程收到B的确认后,还要向B给出确认。确认报文段的ACK置1，确认号ack=y+1，而自己的序号seq=x+1。TCP的标准规定，ACK报文段可以携带数据。但如果不携带数据则不消耗序号，这种情况下，下一个数据报文段的序号仍是seq=x+1。这时，TCP连接已经建立，A进入ESTABLISHED（已建立连接）状态。
当B收到A的确认后，也进入ESTABLISHED状态。

服务器发送完SYN－ACK包，如果未收到客户确认包，服务器进行首次重传，等待一段时间仍未收到客户确认包，进行第二次重传，如果重传次数超过系统规定的最大重传次数，系统将该连接信息从半连接队列中删除。注意，每次重传等待的时间不一定相同。所以D错误。

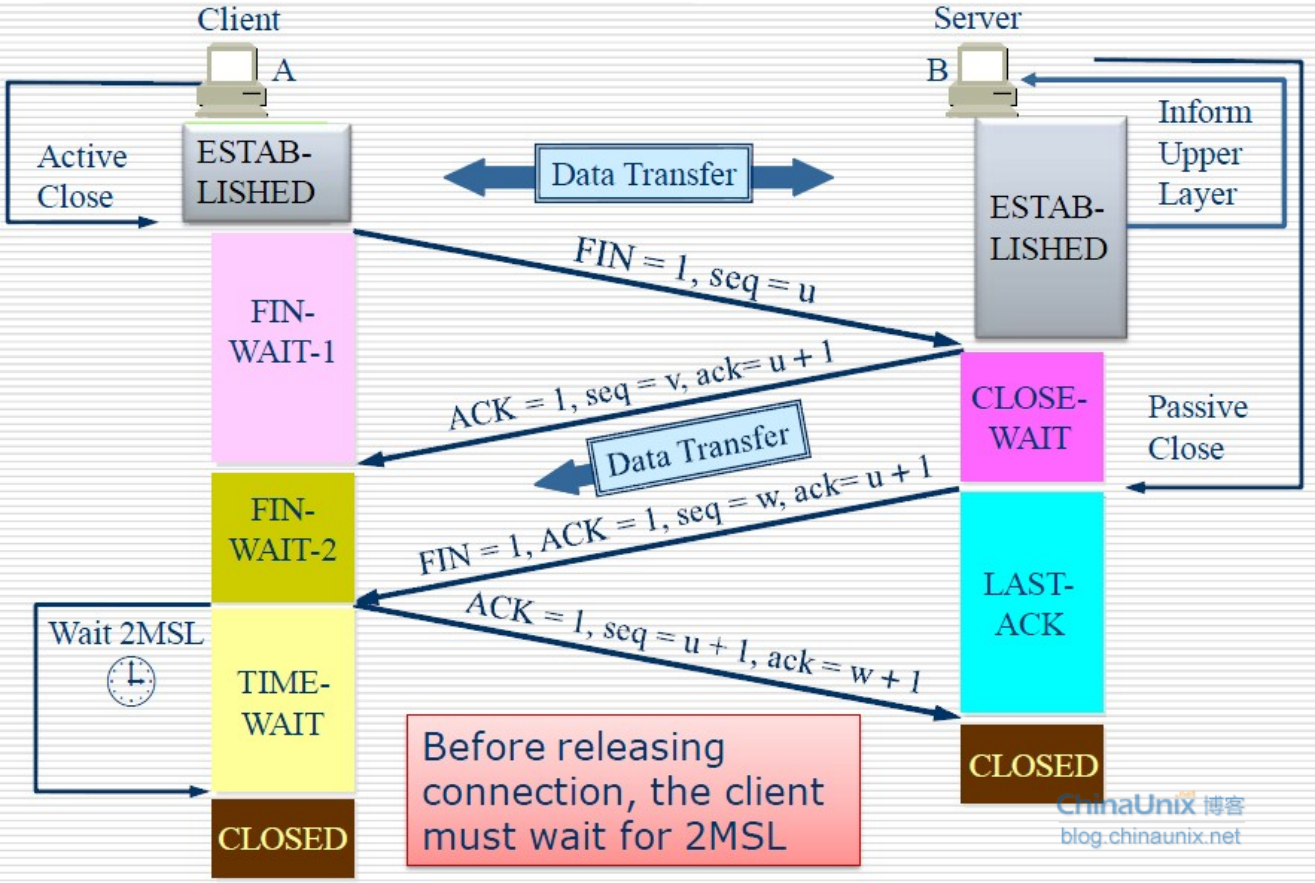
为什么要有第三次确认?

这主要是为了防止已失效的连接请求报文段突然又传送到了B,因而产生错误.

假设没有客户端的第三次确认。A发送连接请求，但连接请求丢失而未收到B的确认，于是A重新发出了一次请求连接，后来B收到了请求，发出确认建立了连接，在完成通信后连接释放，成功完成一次通信。A虽发送了两次连接请求,但第一次丢失,第二次成功,所以此时不存在已失效的连接请求报文。现假设A第一次发送的连接请求报文因为网络问题长时间滞留了，到第二次通信完成释放连接后才到达B,这就是一个已失效的报文,但此时B以为是A的又一次连接请求,于是发出了确认,若没有第三次确认,连接就成功建立了。由于A现在并没有发出建立请求,所以B回复的确认A不理睬,所以不与B进行通信,然而B就一直等着A发数据给他,就这样,B傻乎乎的等了很久,白白浪费了自己的

青春(B的资源)。

1.5.2 TCP的连接释放



1.数据传输结束后，通信双方都可释放连接。现在A,B都处于ESTABLISHED状态。A先发送连接释放报文段，并停止在发送数据，主动关闭TCP连接。A把连接释放报文段首部离得FIN置1，其序号seq=u，u等于前面已发送的数据的最后一个字节的序号加1.这是A进入FIN-WAIT-1（终止等待1）状态，等待B的确认。TCP规定，FIN报文段即使不携带数据，也消耗一个序号。

2.B收到连接释放报文段后即发出确认，确认号是ack=u+1,而这个报文段自己的序号是v，等于B前面已发送的数据的最后一个字节的序号加1.然后B进入CLOSE-WAIT(关闭等待)状态。至此A->B的连接就释放了，这时TCP连接处于半关闭（half-close）状态，即A已经没有数据要发送了，但B若发送数据，A仍然接收。

3.A收到B的确认后，就进入FIN-WAIT-2(终止等待2)状态，等待B发出的连接释放报文段。若B已经没有数据要发给A，其应用进程就通知TCP释放连接。这时B发出的连接释放报文段必须使FIN=1.假定B的序号是w（在半关闭状态B可能又发送了一些数据）。B还必须从夫上次已发送过的确认号ack=u+1。这时B进入LAST-ACK状态，等待A确认。

4.A在收到B的连接释放报文段后，必须对此发出确认。在确认报文段中把ACK置1，确认号ack=w+1，自己的序号为seq=u+1（前面发送的FIN报文段要消耗一个序号）。然后进入TIME-WAIT（时间等待）状态。现在TCP连接还没有释放掉。必须经过时间等待计时器（TIME-WAIT timer）设置的时间2MSL后，A才进入到CLOSED状态。（MSL：Maximum Segment Lifetime 最长报文段寿命）

为什么在TIME-WAIT要等待呢？

1.为了保证A发送的最后一个ACK报文段能够到达B。这个ACK报文段可能丢失，因而使处在LAST-ACK状态的B收不到确认。B会超时重传FIN+ACK报文段，A就能在2MSL时间内收到这个重传的FIN+ACK报文段，接着A重传一次确认，重启计时器。最好，AB都正常进入到CLOSED状态。如果A在TIME-WAIT状态不等待一段时间，而是再犯送完ACK报文后立即释放连接，那么就无法收到B重传的FIN+ACK报文段，因而也不会在发送一次确认报文。这样，B就无法按照正常步骤进入CLOSED状态。

2.防止 已失效的连接请求报文出现在本连接中。

1.私有IP地址范围：

A: 10.0.0.0~10.255.255.255 即10.0.0.0/8

B:172.16.0.0~172.31.255.255即172.16.0.0/12

C:192.168.0.0~192.168.255.255 即192.168.0.0/16

这些地址是不会被Internet分配的，它们在Internet上也不会被路由，虽然它们不能直接和Internet网连接，但通过技术手段仍旧可以和 Internet通讯（NAT技术）。我们可以根据需要来选择适当的地址类，在内部局域网中将这些地址像公用IP地址一样地使用。在Internet上，有些不需要与 Internet通讯的设备，如打印机、可管理集线器等也可以使用这些地址，以节省IP地址资源。

2.OSI模型 共有七层

由一到七层

物理层、数据链路层 、网络层 、传输层 、会话层 、表示层 和应用层

3.Cookie

客户端保存了不同服务器的cookie，每个服务器只能获取对应的cookie，而不能获取全部的

4.VLAN

VLAN（virtual local area network）虚拟局域网，把大的局域网划分为几个单独的互不相通的虚拟局域网，隔离广播风暴。