TCP

**1.TCP (Transmission Control Protocol ,传输控制协议)**

(1)特点：

TCP时面向连接的，先建立TCP连接才能传输数据，传输完成需要释放,以及建立的TCP连接

TCP连接两个端口(套接字)，点对点

TCP提供可靠交付

TCP提供全双工通信

TCP是面向字节流，TCP根据对方给出的窗口值和网络拥塞状况来决定一个数据报的长度，如果缓存数据太长则划分短一些再传送，如果数据太短，则等待积累更多后一起传送。

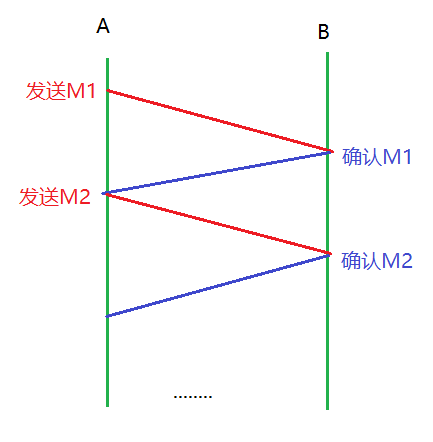
(2).套接字socket = (IP地址：端口号) ，

(3).TCP连接= {socket1,socket2} = {(IP1:port1),(IP2:port2)}（指两个端口的套接字）

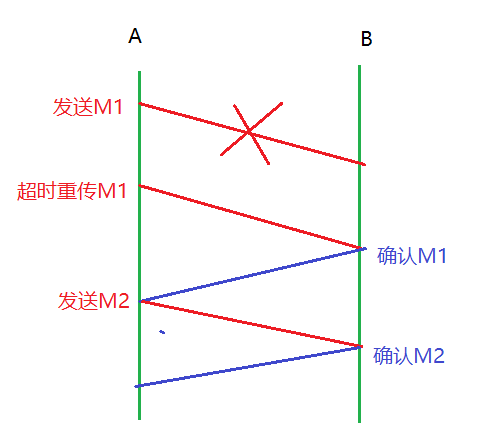
1. **可靠的传输原理**

(1) 停止等待协议(stop-and-wait)：

无差错情况：A发送一个分组M1，B收到向A确认，A继续发送分组M2，依次下去….直到全部发送完。



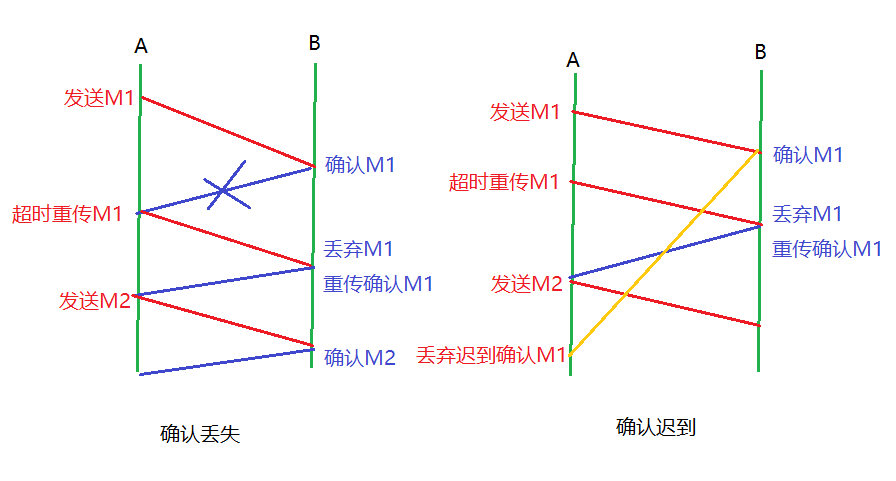
出现差错：B接受分组出现差错则直接丢弃，A没有收到确认消息，就一直等待，直到超时重传。超时计时器必须比数据在分组传输的平均往返时间更长一些。



确认丢失和确认迟到：

B的确认M1消息丢失了，A超时重传M1，B收到后丢弃重复的M1，重传确认M1；

B的确认M1消息迟到了，A重传后也收到确认了，这个时候迟到的确认M1来了，A收到后丢弃。



1. **自动重传请求（Automatic Repeat-reQuest，ARQ）**

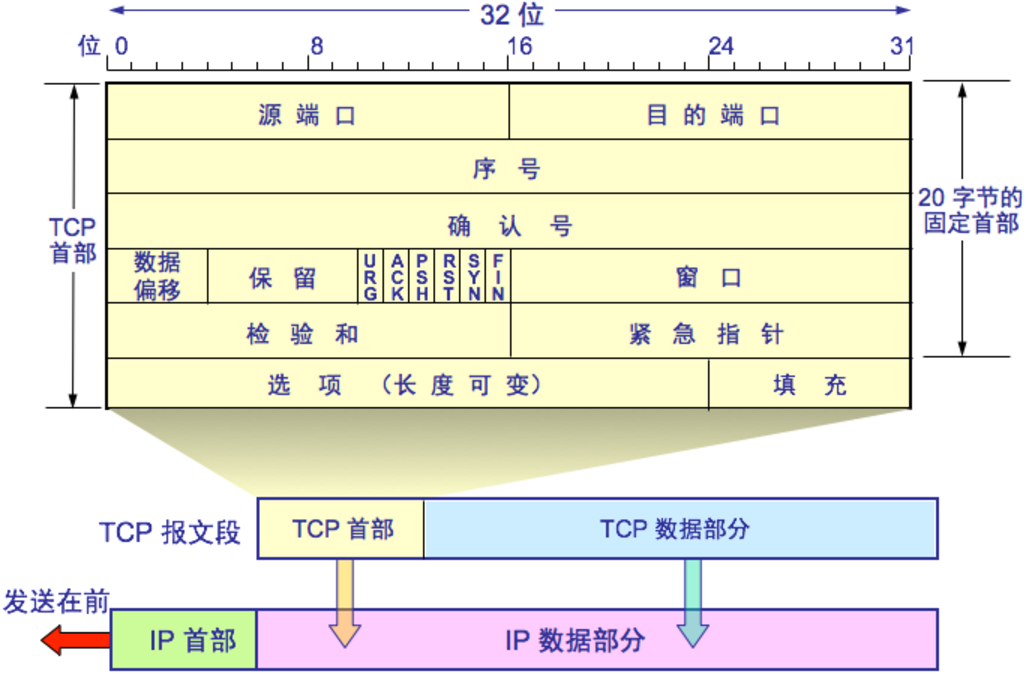
自动重传请求分成为三种，即停等式(stop-and-wait）ARQ，回退n帧（go-back-n）ARQ，以及选择性重传（selective repeat）ARQ。通过接受方请求发送方重传出错的数据报文来恢复出错的报文，ARQ表明重传的请求是自动进行的，接受方不需要请求发送方重传出错的分组,后两者是[滑动窗口](http://baike.baidu.com/view/1199185.htm" \t "http://baike.baidu.com/_blank)技术与请求重发技术的结合，由于窗口尺寸开到足够大时，帧在线路上可以连续地流动，因此又称其为[连续ARQ协议](http://baike.baidu.com/view/1638938.htm" \t "http://baike.baidu.com/_blank)

stop-and-wait：简单，发送与接受窗口均为1,信道利用率低

Go-back-n:当发送方接受到接收方的状态报告指示出错后，发送方将重传过去n个报文

Selective repeat：当发送方接受到接收方的状态报告指示出错后，发送方将只重传出错的报文

**4 TCP报文的首部格式**



(1).序号:当SYN出现时，序列号实际上是初始序列码（ISN）,而第一个数据字节是ISN+1,指的是TCP数据块的第一字节的号码

(2).确认号:如果设置了ACK控制位，这个值表示一个准备接受的包的序列号

(3).数据偏移:指向数据的开始，单位为4字节，因此选项最多有40个字节

(4).紧急指针:指向后面是优先数据的字节，仅当URG=1时启用

(5).控制位:URG,ACK,PSH,PST,SYN,FIN

(6).URG:紧急数据包，接受端紧急处理，发送方优先处理

(7.)ACK:响应数据报，和确认号有关

(8.)PSH:表示要求对方立刻传送缓冲区内对应的数据报，无需等待

(9).SYN:表示发送方希望建立连接，表示开始连接

(10).FIN:表示传送完毕，通知对方完毕是否同意断线，此时发送端还在等待回复

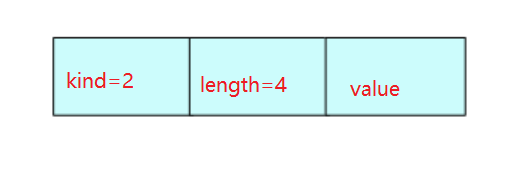
(11).RST:表示立刻结束，无需等待

(12).窗口:告诉对方本机缓存区还可以接受的数据，单位byte

(13)校验和:校验包含TCP首部，TCP数据,TCP伪首部.

(14)TCP选项

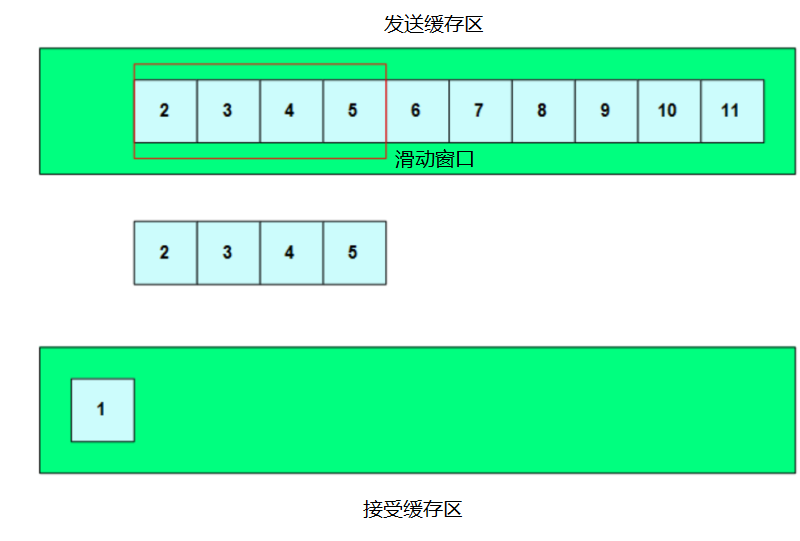
MSS(最大数据段的大小):指定TCP协议所允许的从对方接受的最大数据段



**5 TCP可靠传输的实现**

(1) 以字节为单位的滑动窗口：

维护一个发送窗口和接受窗口，发送窗口根据收到的确认序号不断更新发送窗口，其内部有三个指针，依次指向窗口的起始序号，窗口已发送字节尾部序号，窗口允许发送的尾部序号，这三个指针不断更新；接受窗口则按序接受数据，并发送按序接受的最高序号的确认信息。



1. 选择确认SACK   
    TCP接受方收到的数据字节流不连续，通过选择确认，让发送放只重传缺少的数据，而不是重传所有没有收到确认的数据。
2. **TCP的流量控制**(让发送方的发送速率不要太快，要让接收方来得及接收)

(1) 利用滑动窗口控制流量：发送方的发送窗口不能超过接收方给出的接受窗口的数值   
(2) 传输效率：TCP要控制发送缓存发送的时机来保证传输效率。

缓存数据到达MSS就组成TCP报文发送出去

(3)发送方计时期限到了就把已有的缓存发送出去，但必须小于最长报文段长度MS

(4)解决糊涂窗口综合症：发送方不发送很小的报文段的同时，接收方也不要在缓存只有一点的时候就发送确认信息给发送方

1. **TCP的拥塞控制**

(1) 拥塞控制的概念：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   
(2) 拥塞的几种状态：

轻度拥塞：当网络的吞吐量明显小于理想的吞吐量时进入轻度拥塞

拥塞状态：当提供的负载达到某一数值时，网络的吞吐量反而随提供的负载的增大而下降时进入拥塞状态

死锁：当提供的负载继续增大到某一数值时，网络的吞吐量就下降到零，网络已无法工作，则进入死锁

(3) 开环和闭环控制

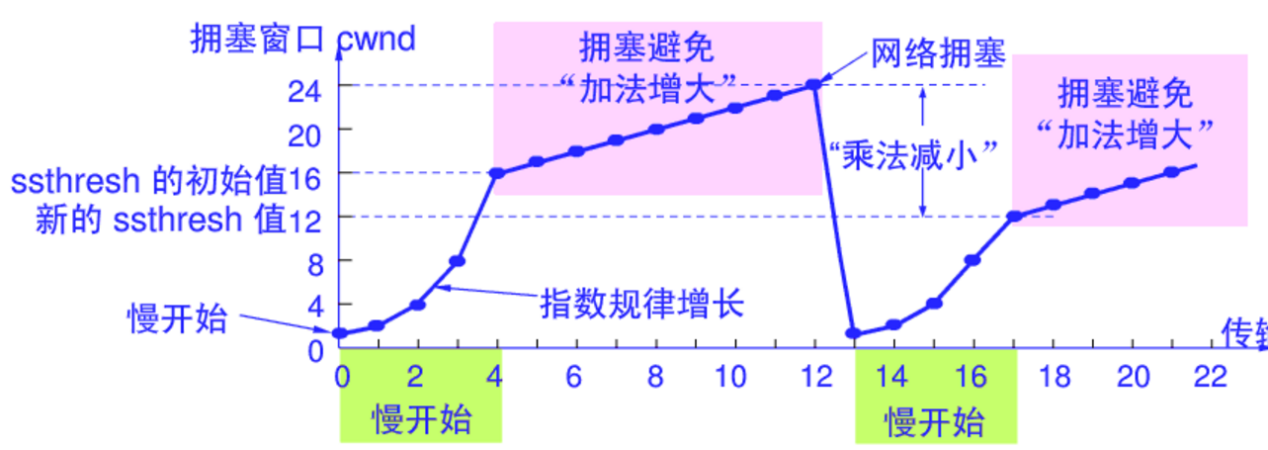
开环控制：在设计网络时事先将有关发生拥塞的因素考虑周到，力求网络在工作时不产生拥塞

闭环控制：监测网络系统以便监测拥塞在何时何处发生；把拥塞发生的信息传送到可采取行动的地方；调整网络系统的运行以解决出现的问题

(4) 控制拥塞的方法：

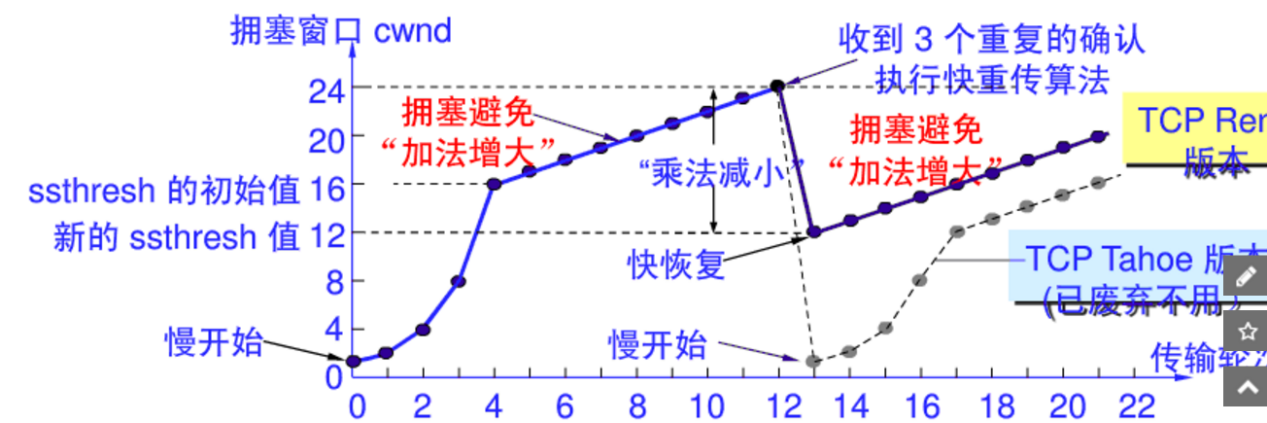
慢开始和拥塞避免   
慢开始：由小到大增加发送窗口和拥塞窗口(cwnd小于等于发送窗口)

每次都乘以2；

拥塞避免：当拥塞窗口超过慢开始门限ssthresh后，让拥塞窗口缓慢增大，即每经过一个RTT就加1(即加法增大)。如果网络发生超时，即有可能发生拥塞，就将ssthresh减小为cwnd的一半(乘法减小)。

快重传和快恢复   
快重传：每收到一个失序的报文段就立即发出重复确认，使发送方及早知道有报文段没有到达对方，二不要等待自己发送数据时进行捎带；

快恢复：执行乘法减小时，把ssthresh减小一半后不执行慢开始把cwnd设置为1，而是把cwnd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。



发送方窗口的上限值 = Min[rwnd，cwnd]

1. **TCP的连接管理**

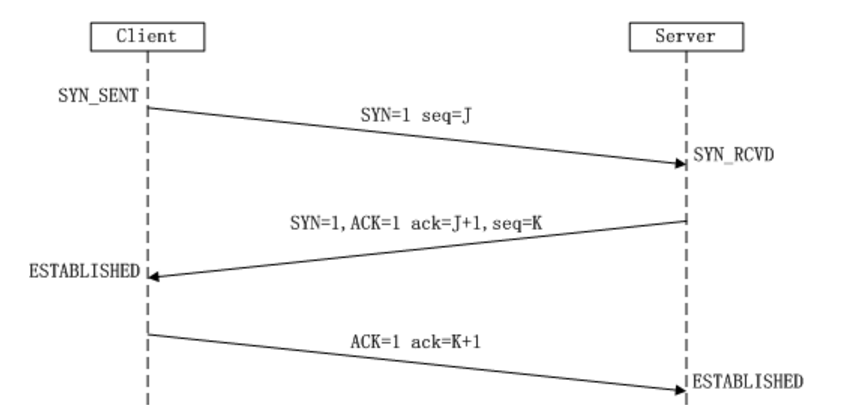
(1) 客户和服务器：主动发起连接的叫客户，被动等待连接的叫服务器   
(2) TCP的连接建立：

Client的TCP向Server发出连接请求报文，其首部的同步位SYN=1，序号seq = J，Client进入SYN-SENT状态

Server收到连接请求报文后，如果同意连接，就发送确认报文。其首部的SYN和ACK都置为1，同时选择一个序号seq = K，确认号为ack = J+1，Server进入SYN-REVD状态，

Client收到Server发出的确认报文后，还要向Server发出确认报文，其首部ACK=1，确认号ack为K+1，序号seq为J+1，然后Client进入ESTABLISHED状态，即建立连接

Server收到Client发出的确认报文后，立即进入ESTABLISHED状态，双方可以开始传输数据



(2) TCP的连接释放：

Client先向其TCP发出连接释放请求，并停止发送数据。其TCP报文的首部FIN=1，序号seq=u(前一个发送数据的序号+1)，这是Client进入FIN\_WAIT1状态

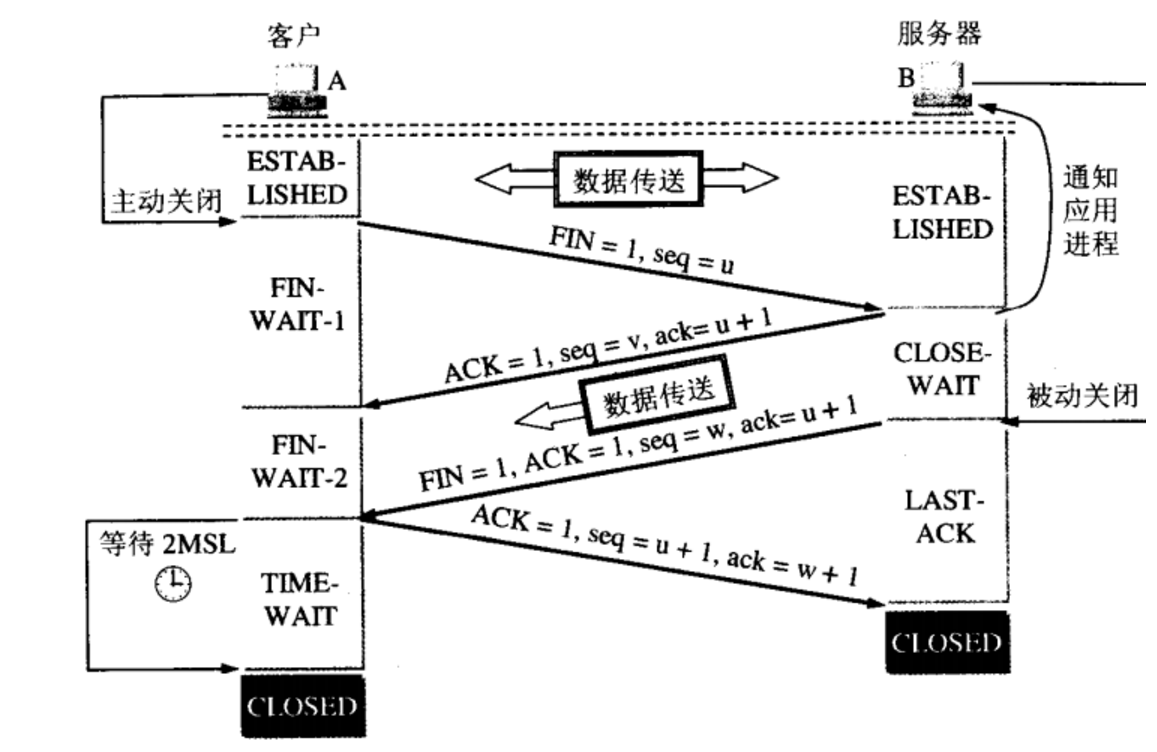
Server收到Client发来的连接释放请求后，立即发出确认，确认号ack = u+1，序号seq=v，并把ACK置1，Server进入CLOSE-WAIT状态

Client收到确认报文后，进入FIN\_WAIT2状态，等待Server发出连接释放请求

Server等待其数据发送完后，其应用程序就通知TCP释放连接，其首部FIN=1，ACK=1，seq=w，ack=u+1，进入LAST-ACK最后确认状态

Client收到Server的连接释放请求后，必须对此确认，其报文首部ACK=1，ack=w+1，seq=u+1，然后进入TIME-WAIT状态，

Server收到Client的确认报文后，进入CLOSED状态，Client等到2MSL后进入CLOSED状态。



**9.TCP有限状态机**

CLOSED:初始状态

LISTEN:服务器正在等待请求

SYN RCVD:一个请求已经到达，等待ACK

SYN SENT:应用程序已经开始打开连接

ESTABLISHED:正常数据传输状态

FIN WAIT1:应用程序说它已经结束连接

FIN WAIT2:另一方已经同意释放连接

TIMED WAIT:等待所有的分组渐渐消失

CLOSED WAIT:另一方已经发起释放连接的过程

CLOSING:双方试图同时关闭连接