计算机网络05

概述

主机通过异构网络互联，实现主机到主机的通信——》物理层，数据链路层，网络层

但在计算机网络中进行通信的真正实体是位于通信两端主机中的进程

为运行在不同主机上的应用进程提供直接的通信服务——》运输层，运输层协议又称端到端协议

根据应用需求不同，运输层为应用层提供了两种不同的运输协议

面向连接的TCP

无连接的UDP

运输层端口号，复用和分用

计算机上的进程用**进程标识符PID**来标志区分

为了让不同操作系统的电脑应用进程能相互通信，TCP/IP体系使用统一的**端口号**对应用进程标识

端口号只有本地意义，不同电脑中的相同端口号没有联系

端口号用16比特表示，有以下几类

熟知端口号：指派给TCP/IP体系中最重要的一些应用协议，如：HTTP使用80

登记端口号：给没有熟知端口号的应用程序使用，必须在IANA登记，防止重复

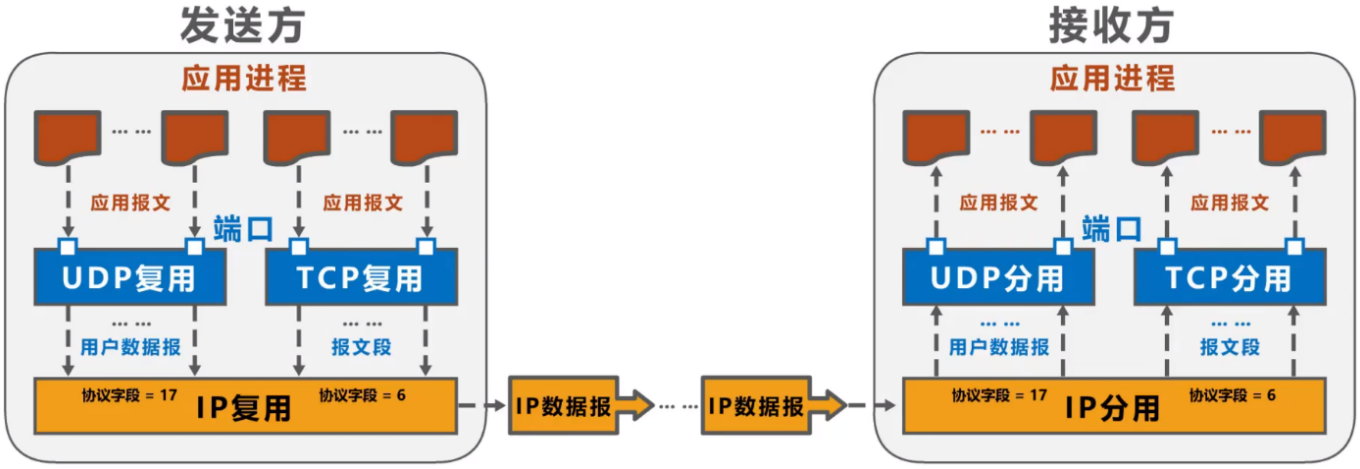
短暂端口号：给客户进程选择短暂使用。当服务器进程收到客户进程的报文时，就知道了客户进程所使用的动态端口号。通信结束后，这个端口号可供其他客户进程以后使用

发送方复用与接收方分用

同一个协议的报文包装在一起，但以端口区分，称为UDP/TCP复用

发送后，接收方分析报文，区分不同应用的进程，称为UDP/TCP分用

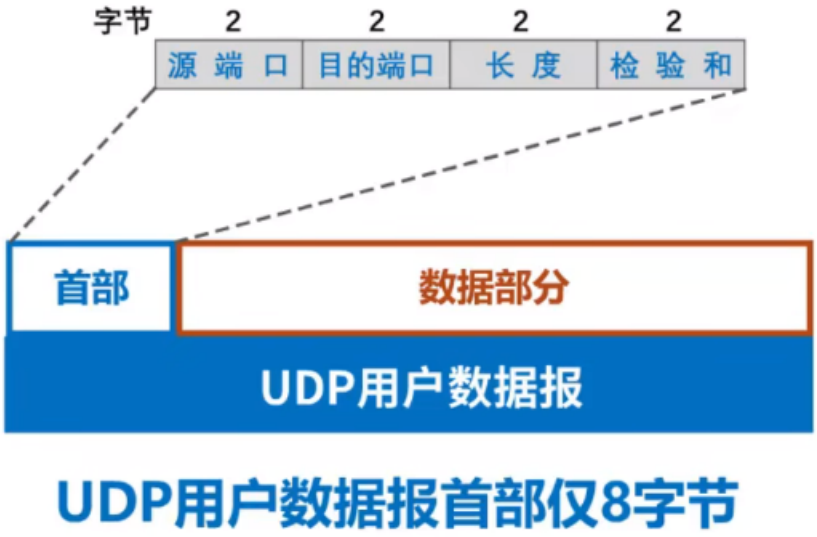
两者是相同工序，相反顺序的过程



UDP与TCP的对比

UDP (User Datagram Protocol)

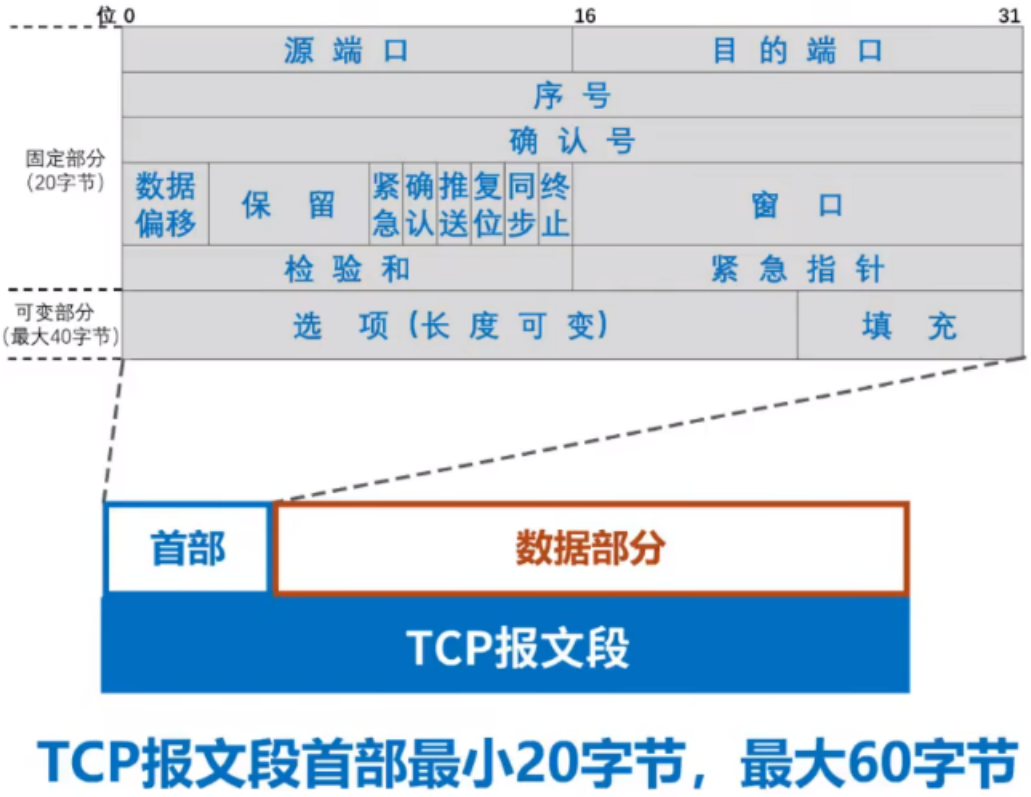
- 随机进行数据传输，不用建立连接
- 支持单播，多播，广播
- 将应用层的报文不经处理，加上UDP头直接传输——》UDP是面向应用报文的
- 提供不可靠传输服务（适用于IP电话，视频会议等实时应用）
- UDP用户数据报首部较简单



UDP用户数据报首部仅8字节

TCP (Tranimission Control Protocol)

- 开始时“三报文握手”建立连接，结束后“四报文挥手”释放连接
- 基于TCP连接的可靠信道——》仅支持单播
- 将应用层报文看作无意义的数据流，打散编号分组乱序传输。这要求接收方应用进程能处理乱序的数据——》TCP是面向字节流的
- 虽然发送方和接收方的字节数不一样，但内容是一样的——》实现可靠传输。
- 向上层提供面向连接的可靠传输服务（适用于文件传输）
- TCP报文段首部较复杂



TCP报文段首部最小20字节，最大60字节

TCP流量控制

如果发送方数据发送过快，接收方来不及接收，可能造成数据的丢失——》流量控制

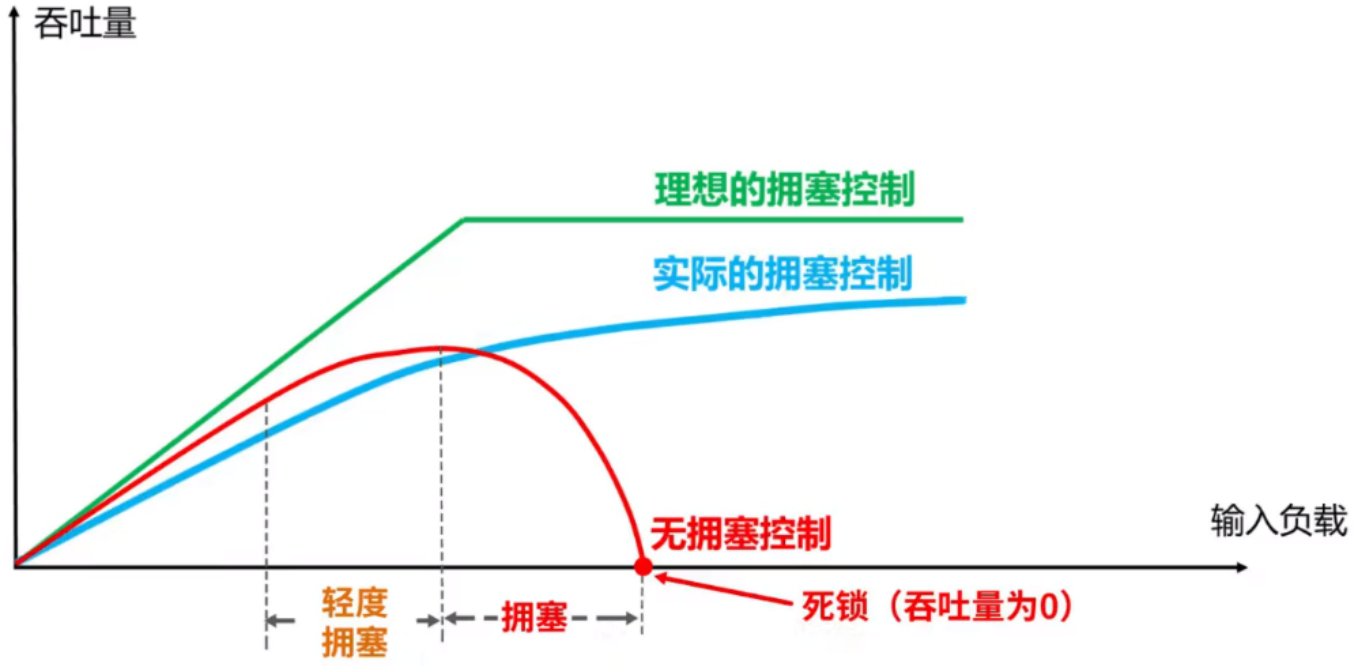
流量控制（flow comtrol）让发送方的发送速率不要太快，让接收方来得及接受。TCP利用滑动窗口机制1实现对发送方的流量控制

滑动窗口

和之前的差不多，就是接收方会发送报文告诉发送方调整自己的发送窗口大小

TCP拥塞控制

网络拥塞：某段时间内，对网络中的某一资源（贷款，交换结点的缓存）的需求超过了该资源所能提供的可用部分，网络性能就会变坏。若不控制，将会使整个网络吞吐量随着输入负载而下降



拥塞控制算法

假定以下条件

数据是单方向传送的，而另一个方向只传送确认

接收方总是有足够大的缓存空间，因而发送发发送窗口的大小由网络拥塞程度决定

以最大报文段MSS的个数为讨论问题的单位，而不是以字节为单位

发送方维护叫拥塞窗口cwnd的状态变量，其值取决于网络的拥塞程度，并且动态变化

cwnd维护原则：没有拥塞，窗口变大；有，减少

判断出现网络拥塞的依据：没有按时收到应当到达的确认报文

发送方将 发送窗口swnd = cwnd

发送发维护一个慢开始门限ssthresh状态变量

当cwnd < ssthresh 时，使用慢开始算法

>, 改用拥塞避免算法

=，两者都可用

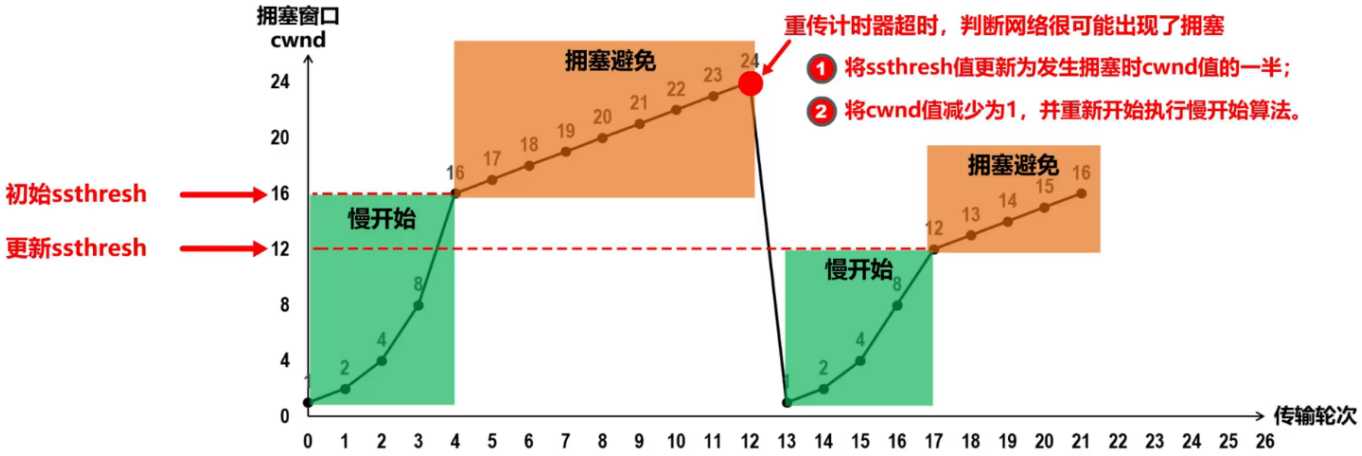
慢开始算法

开始时，cwnd设置为1，每完成一次传输轮次，cwnd指数增长（2, 4, 8, 16...）直至达到规定的ssthresh

拥塞避免算法

到达ssthresh后，不再指数增长，只增长1。

若出现网络拥塞，则将ssthresh更新为当前cwnd的一半（24–>12）然后重新从1开始计算



缺点——》当网络并未拥塞，个别文件丢失时，误启动慢开始

——》又设计了快重传和快恢复

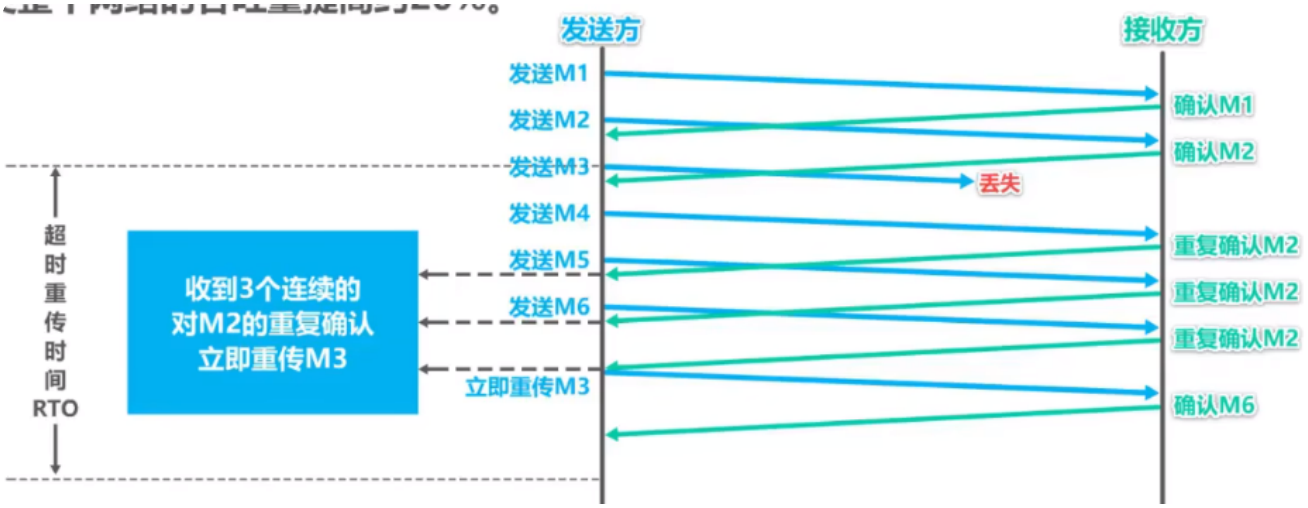
快重传

使发送方尽快进行重传，而不是等超时重传计时器超时再重传

要求接收方立即发送确认

接收方即使接收到失序的报文段也要立即对已收到的报文段的重复确认

发送方收到3个重复确认后，将立即重传，而不是等计时器超时



快恢复

发送发收到3个重复确认，不启动慢开始算法，而是执行快恢复算法——》将ssthresh和cwnd调整为当前窗口的一半，开始执行拥塞避免算法

TCP超时重传时间的选择

超时重传时间应略大于往返时间RTT

但不能直接使用某次测量得到的RTT样本来计算

应使用以下公式，计算加权平均往返时间

■ RFC6298建议使用下式计算超时重传时间RTO：

$$RTO = RTT_S + 4 \times RTT_D$$

加权平均往返时间RTTs

$RTT_{S1} = RTT_1$

新的 $RTT_S = (1 - \alpha) \times$ 旧的RTT_S

$+ \alpha \times$ 新的RTT样本

$0 \le \alpha < 1$

已成为建议标准的RFC6298推荐的 α 值为1/8，即0.125。

RTT偏差的加权平均RTTD

$RTT_{D1} = RTT_1 \div 2$

新的 $RTT_D = (1 - \beta) \times$ 旧的RTT_D

$+ \beta \times |RTT_S -$ 新的RTT样本 $|$

$0 \le \beta < 1$

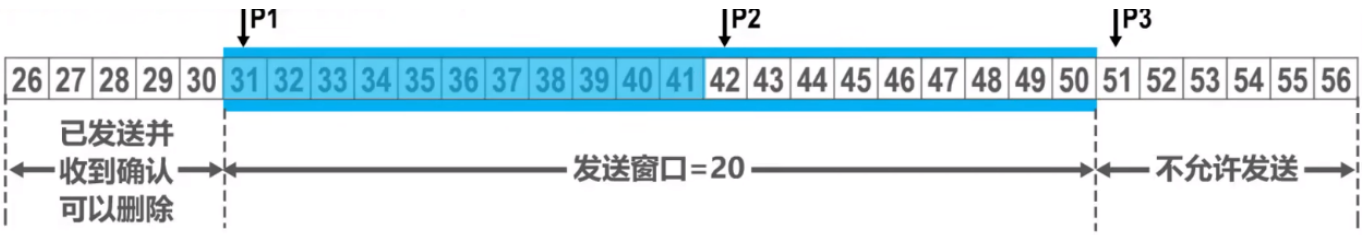
已成为建议标准的RFC6298推荐的 β 值为1/4，即0.25。

出现超时重传——》无法测准往返时间RTT

——》出现超时重传，不计算往返时间，但每重传一次，将RTO增大一些

TCP可靠传输的实现

和前面的窗口传输差不多



如何描述发送窗口的状态?

使用三个指针P1，P2，P3分别指向相应的字节序号

- ☐ 小于P1的是已发送并已收到确认的部分；
- ☐ 大于等于P3的是不允许发送的部分；
- ☐ P3 - P1 = 发送窗口的尺寸；
- ☐ P2 - P1 = 已发送但尚未收到确认的字节数；
- ☐ P3 - P2 = 允许发送但当前尚未发送的字节数（又称为可用窗口或有效窗口）；

TCP连接的建立

解决三问题

TCP双方确知对方的存在

双方能够协商一些参数（如最大窗口值）

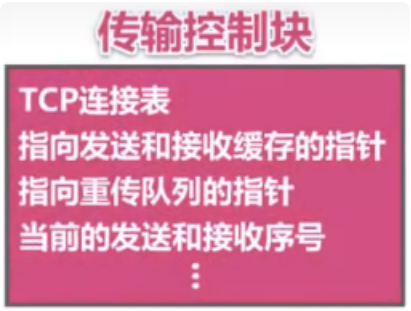
双方能够对运输实体资源（如缓存大小）进行分配

三握手过程

主动发起请求——》TCP客户

被动接受请求——》TCP服务器

双方先建立传输控制快



客户向服务器发送TCP连接请求报文段

同步位SYN = 1，表明这是一个TCP连接请求报文段

序号字段seq = x，作为TCP客户进程所选择的初始序号

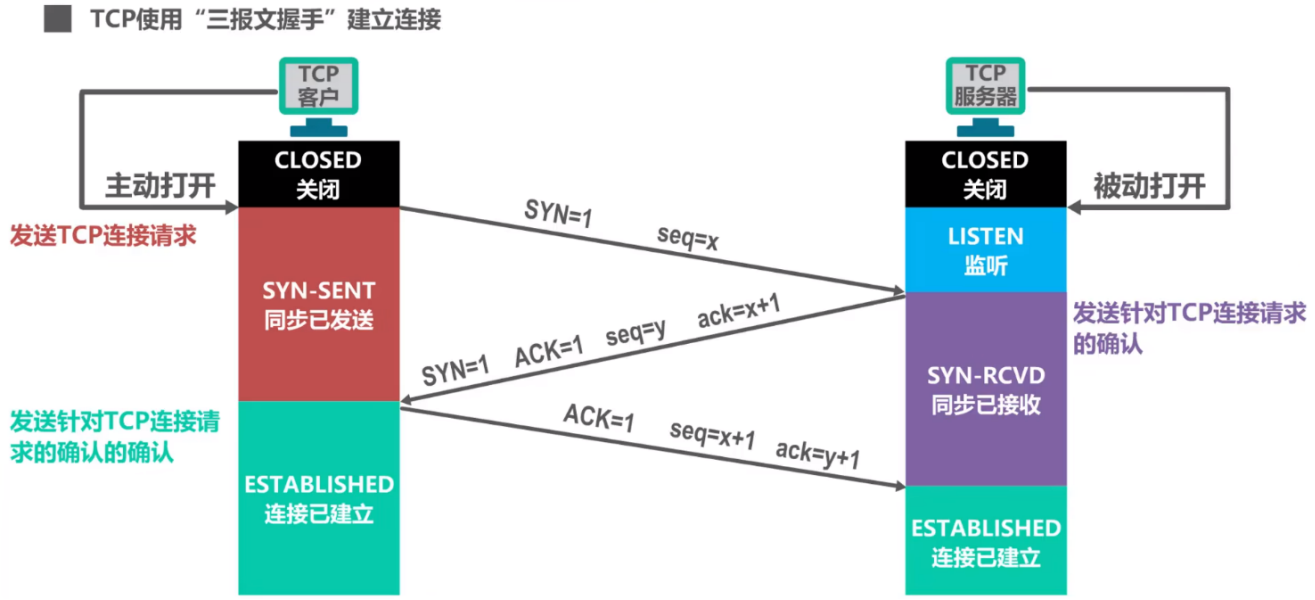
服务器同意连接，发送针对TCP连接请求的确认。

同步位SYN = 1, 确认位ACK = 1，表明这是针对TCP连接请求的确认

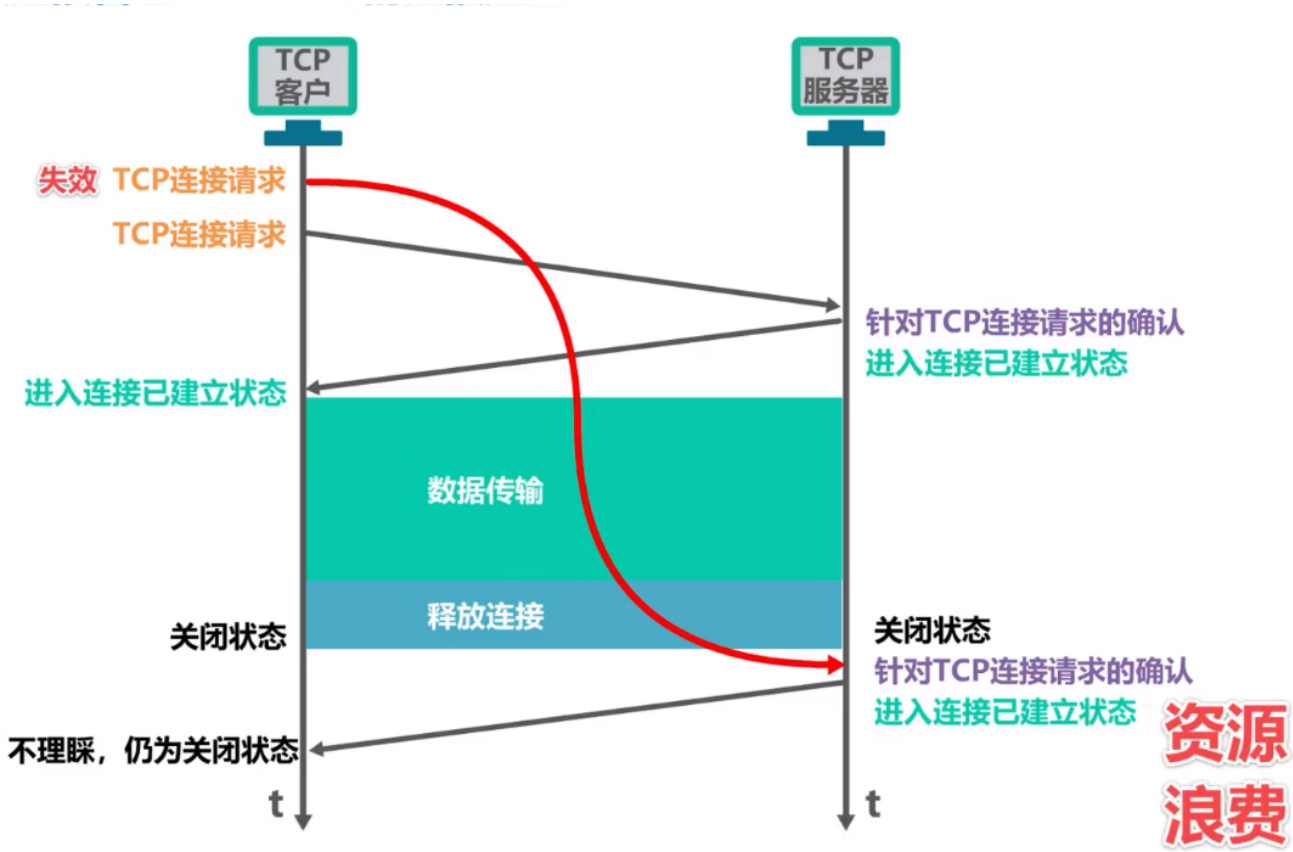
序号字段seq = y，作为TCP服务器进程所选择的初始序号

确认号字段ack = x+1, 表示确认TCP客户进程所选择的初始序号为x

客户发送一个普通的TCP确认报文段——》连接完成



第三次握手是否多余？不，为了防止以下情况



TCP连接的释放

过程

客户发送TCP连接释放

终止位FIN = 1, 确认位ACK = 1，表明这是一个TCP连接释放报文段

序号字段seq = u，等于TCP客户进程之前已经**传送过**的数据的最后一个字节的序号+1

确认号字段ack = v，客户之前**已收到**的数据的最后一个字节的序号+1

服务器发送TCP普通确认，并且TCP服务进程通知高层应用进程：连接释放

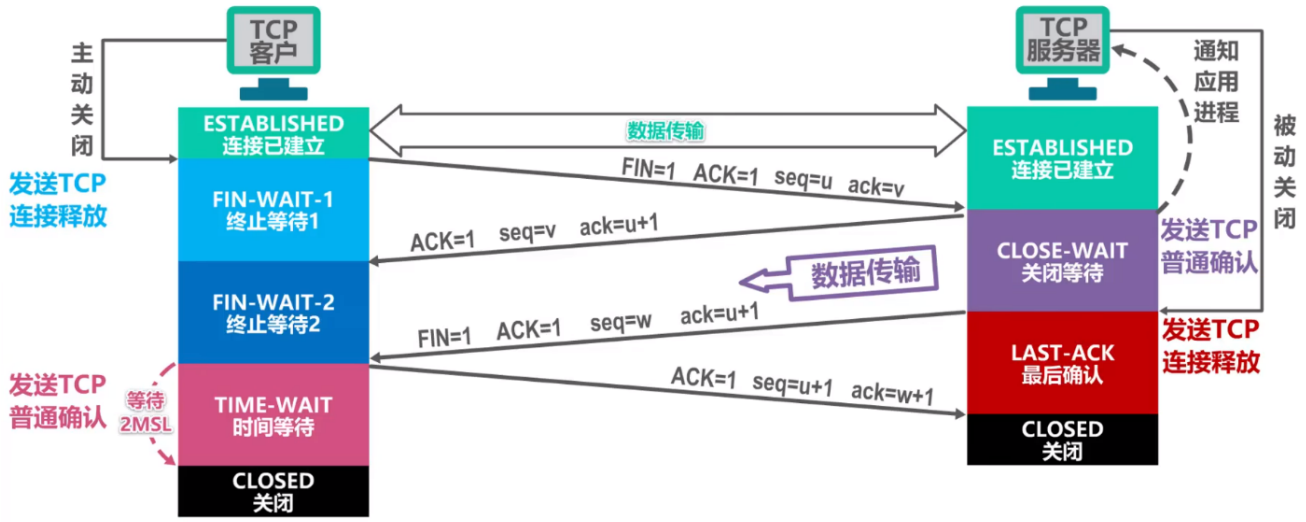
此时，服务器仍可以向客户发送消息，客户进程等待中

若没有信息需要发送，高层应用通知通知TCP服务器进程释放连接

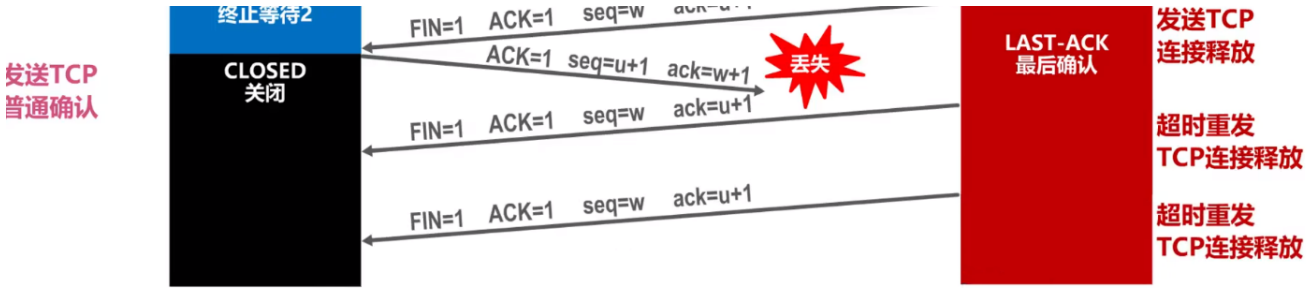
服务器进程发送TCP连接释放报文段

客户发送TCP普通确认

■ TCP通过“四报文挥手”来释放连接



为什么要等待2MSL?



保活计时器

若客户中途暴毙，服务器怎么办呢

■ TCP服务器进程每收到一次TCP客户进程的数据，就重新设置并启动**保活计时器**（2小时定时）。

■ 若保活计时器定时周期内未收到TCP客户进程发来的数据，则**当保活计时器到时后，TCP服务器进程就向TCP客户进程发送一个探测报文段**，以后则每隔75秒钟发送一次。若一连发送10个探测报文段后仍无TCP客户进程的响应，TCP服务器进程就认为TCP客户进程所在主机出了故障，接着就关闭这个连接。