本节内容

TCP协议特点

和TCP报文段

TCP协议的特点

1.TCP是面向连接（虚连接）的传输层协议。打call

2.每一条TCP连接只能有两个端点，每一条TCP连接只能是点对点的。

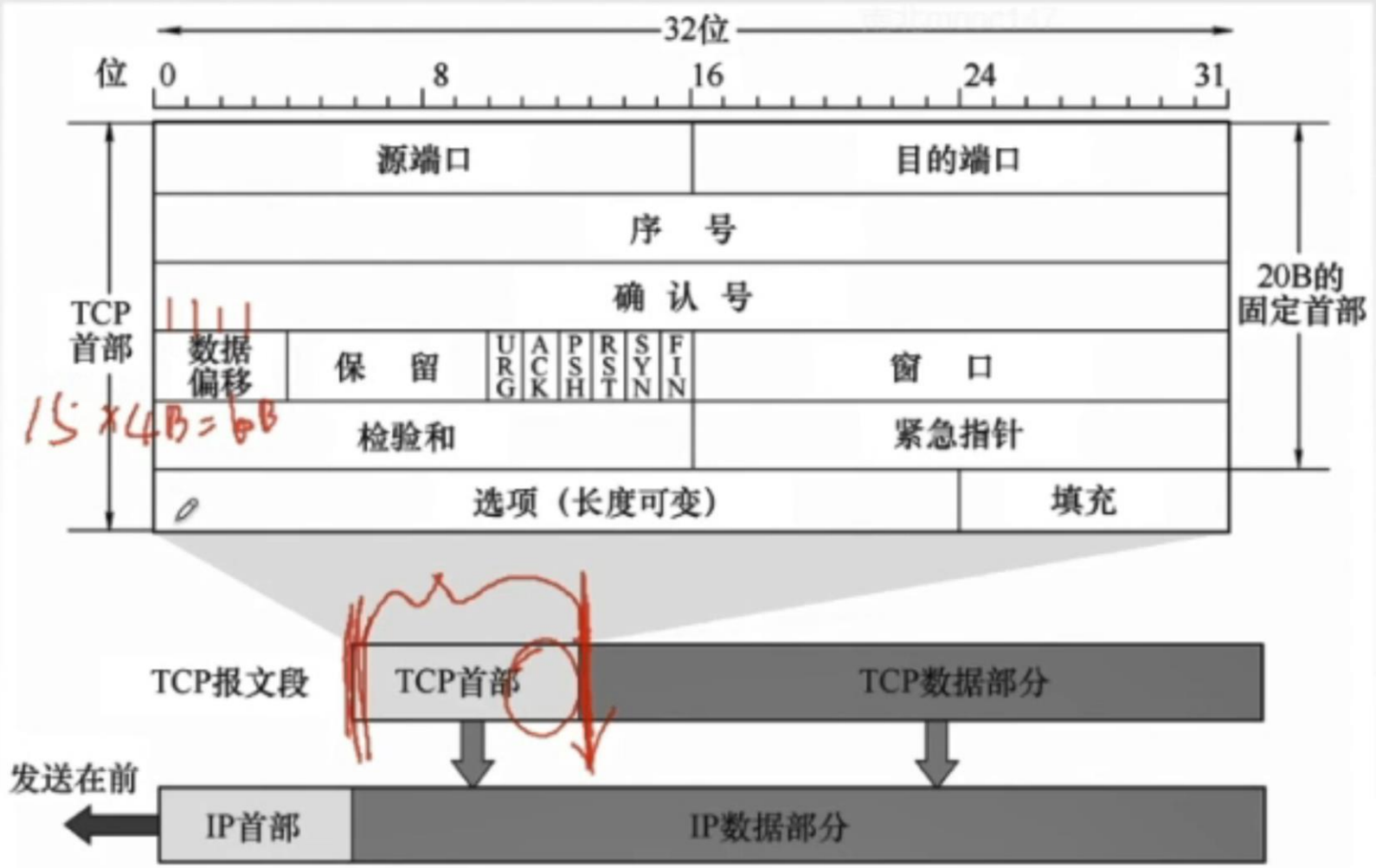
3.TCP提供可靠交付的服务，无差错、不丢失、不重复、按序到达。可靠有序，不丢不重

4.TCP提供全双工通信。 → 发送缓存 准备发送的数据&已发送但尚未收到确认的数据
接收缓存 按序到达但尚未被接受应用程序读取的数据&不按序到达的数据

5.TCP面向字节流 → TCP把应用程序交下来的数据看成仅仅是一连串的无结构的字节流。

流：流入到进程或从进程流出的字节序列。

TCP报文段首部格式

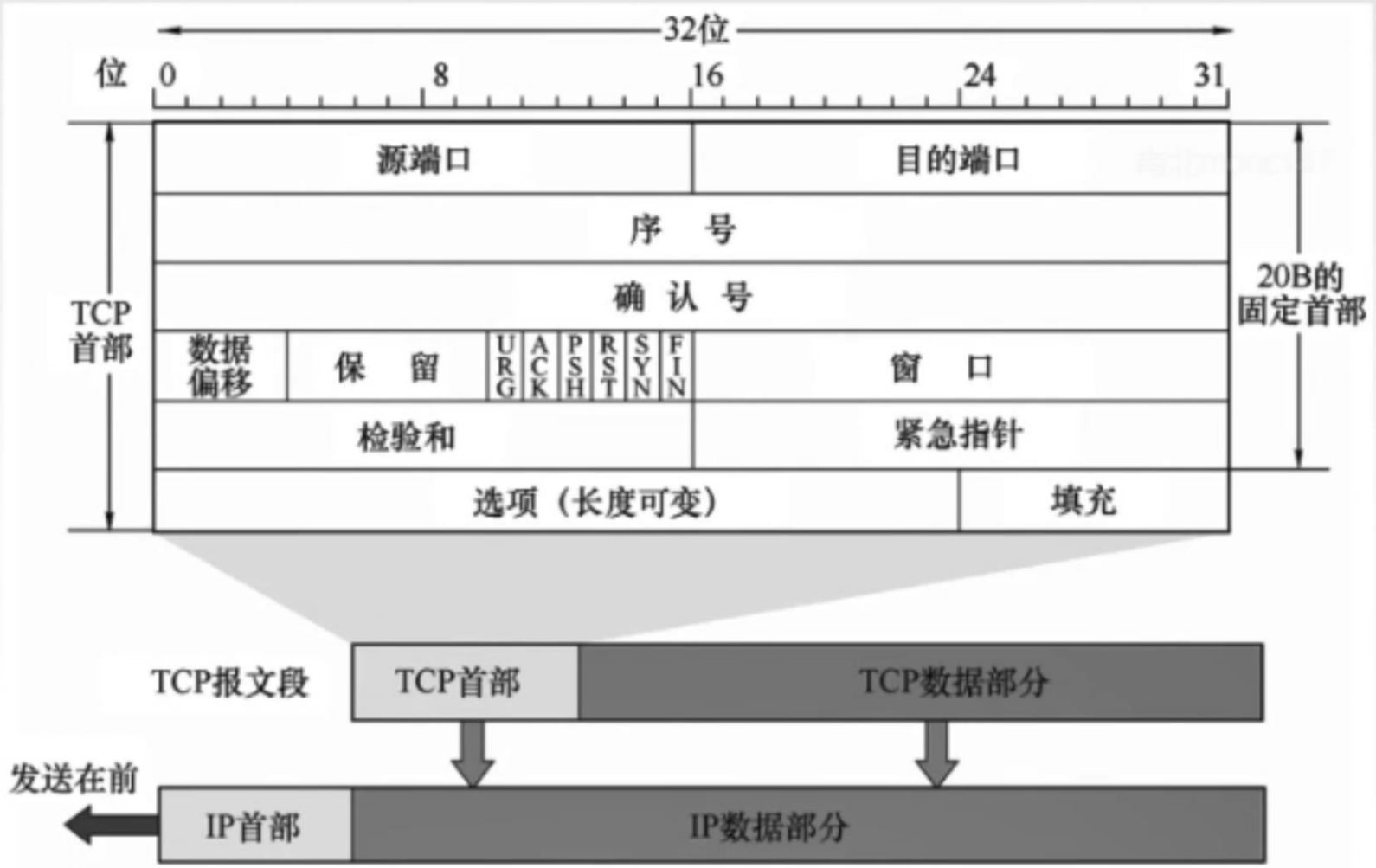


序号：在一个TCP连接中传送的字节流中的每一个字节都按顺序编号，本字段表示本报文段所发送数据的**第一个字节的序号**。

确认号：**期望**收到对方下一个报文段的第一个数据字节的序号。若确认号为N，则证明到序号N-1为止的所有数据都已正确收到。

数据偏移（**首部长度**）：TCP报文段的数据起始处距离TCP报文段的起始处有多远，以4B位单位，即1个数值是4B。

TCP报文段首部格式



6个控制位

紧急位URG：URG=1时，标明此报文段中有紧急数据，是高优先级的数据，应尽快传送，不用在缓存里排队，配合紧急指针字段使用。

确认位ACK：ACK=1时确认号有效，在连接建立后所有传送的报文段都必须把ACK置为1。

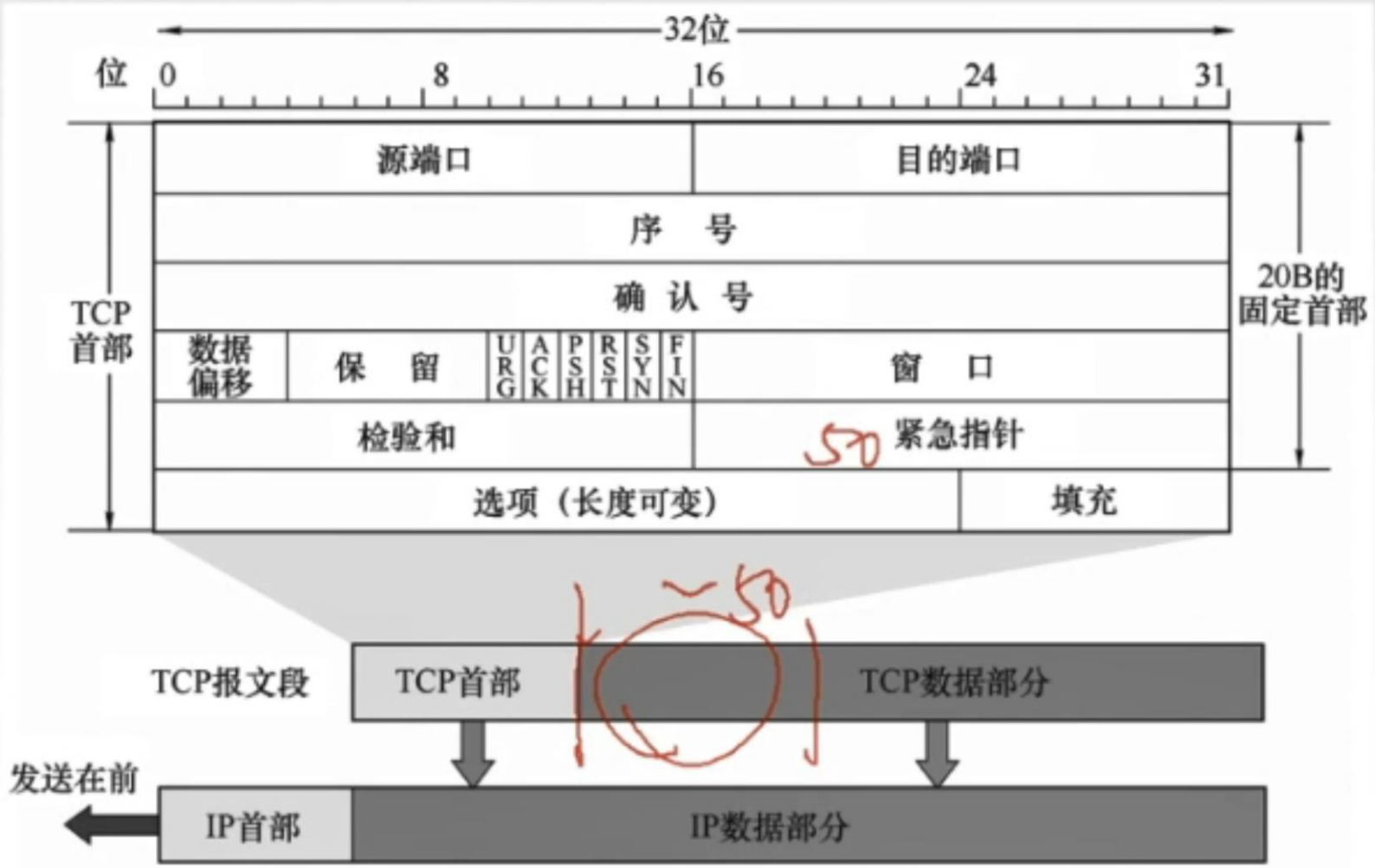
推送位PSH：PSH=1时，接收方尽快交付接收应用进程，不再等到缓存填满再向上交付。

复位RST：RST=1时，表明TCP连接中出现严重差错，必须释放连接，然后再重新建立传输链接。

同步位SYN：SYN=1时，表明是一个连接请求/连接接受报文。

终止位FIN：FIN=1时，表明此报文段发送方数据已发完，要求释放连接。

TCP报文段首部格式



窗口：指的是发送本报文段的一方的接收窗口，即现在允许对方发送的数据量。

检验和：检验首部+数据，检验时要加上12B伪首部，第四个字段为6。

紧急指针：URG=1时才有意义，指出本报文段中紧急数据的字节数。

选项：最大报文段长度MSS、窗口扩大、时间戳、选择确认...

本节内容

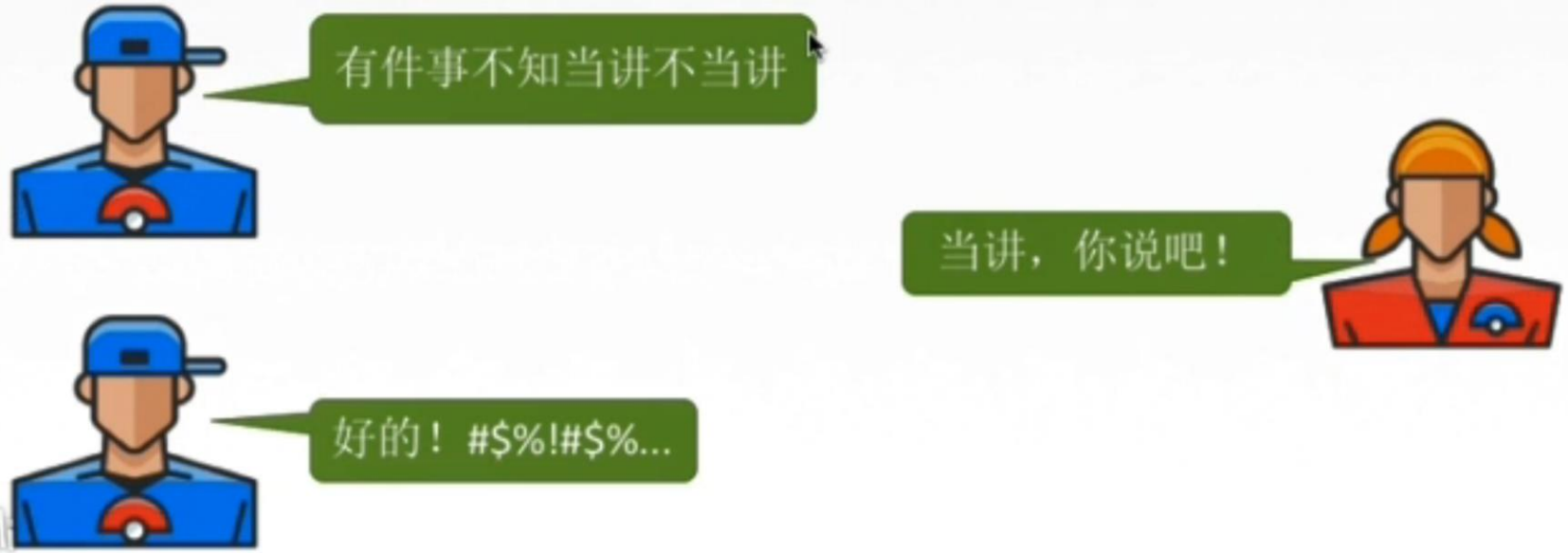
TCP连接管理

TCP连接管理

TCP连接传输三个阶段：

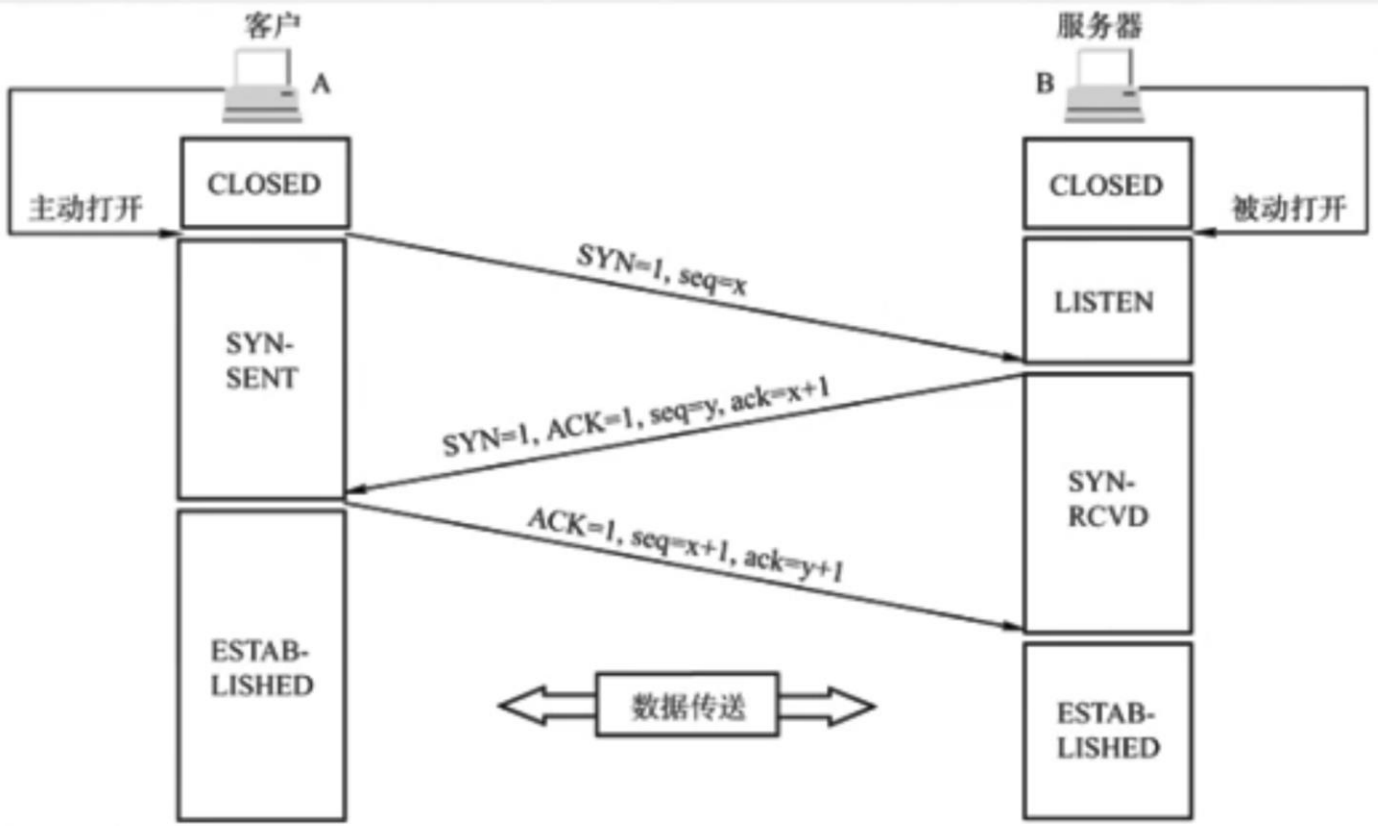
连接建立 → 数据传送 → 连接释放

TCP连接的建立采用**客户服务器方式**，主动发起连接建立的应用进程叫做客户，而被动等待连接建立的应用进程叫服务器。



TCP的连接建立

假设运行在一台主机（客户）上的一个进程想与另一台主机（服务器）上的一个进程建立一条连接，客户应用进程首先通知客户TCP，他想建立一个与服务器上某个进程之间的连接，客户中的TCP会用以下步骤与服务器中的TCP建立一条TCP连接：



ROUND 1:

客户端发送**连接请求报文段**，无应用层数据。

SYN=1，seq=x(随机)

ROUND 2:

服务器端为该TCP连接**分配缓存和变量**，并向客户端返回**确认报文段**，允许连接，无应用层数据。

SYN=1，ACK=1，seq=y(随机)，ack=x+1

ROUND 3:

客户端为该TCP连接**分配缓存和变量**，并向服务器端返回确认的确认，可以携带数据。

SYN=0，ACK=1，seq=x+1，ack=y+1

SYN洪泛攻击

SYN洪泛攻击发生在OSI第四层，这种方式利用TCP协议的特性，就是三次握手。攻击者发送TCP SYN，SYN是TCP三次握手中的**第一个数据包**，而当服务器返回ACK后，该攻击者就不对其进行再确认，那这个TCP连接就处于挂起状态，也就是所谓的半连接状态，服务器收不到再确认的话，还会重复发送ACK给攻击者。这样更加会浪费服务器的资源。攻击者就对服务器发送非常大量的这种TCP连接，由于每一个都没法完成三次握手，所以在服务器上，这些TCP连接会因为挂起状态而消耗CPU和内存，最后服务器可能死机，就无法为正常用户提供服务了。

TCP的连接释放

我说完了。

好的。我想说.....

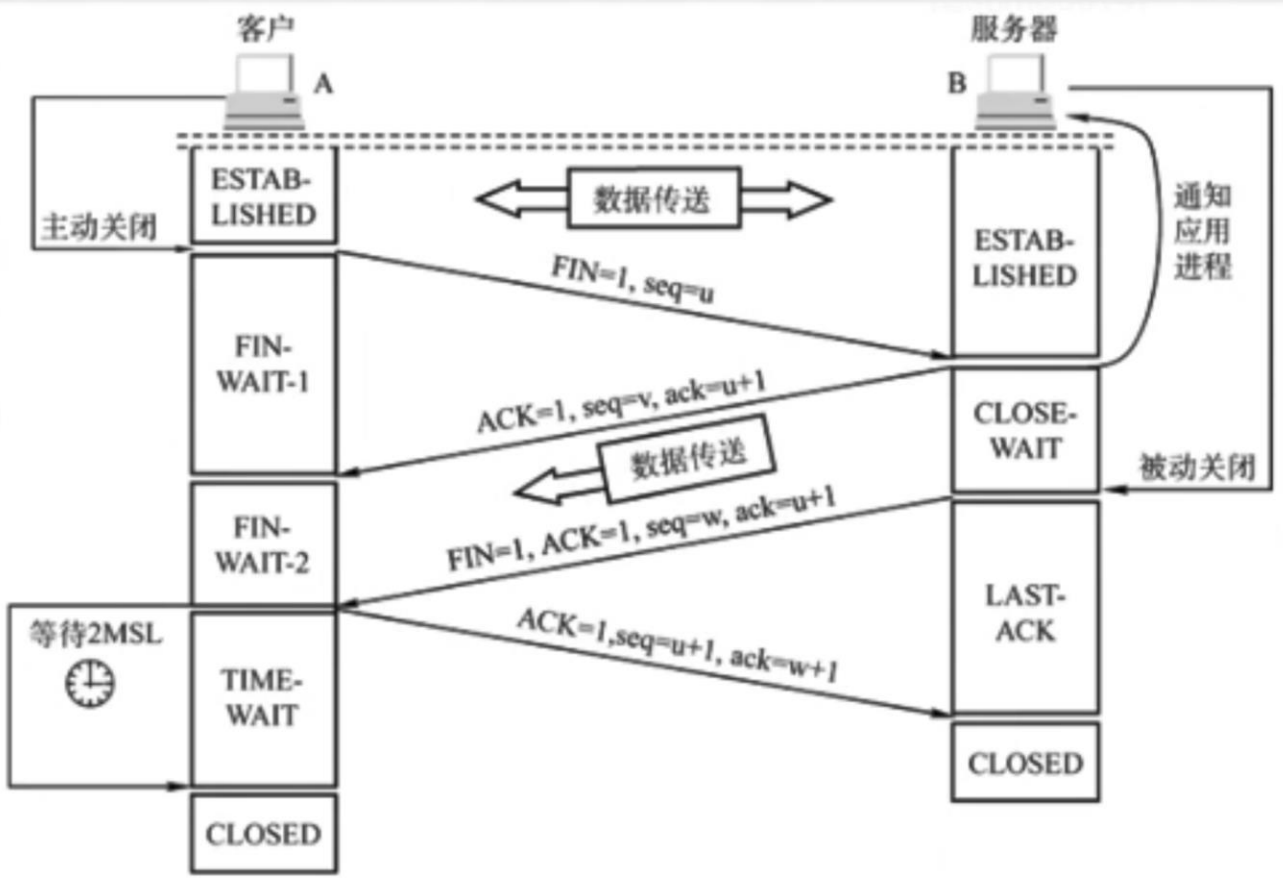
我说完了。

好吧。

四次握手

TCP的连接释放

参与一条TCP连接的两个进程中的任何一个都能终止该连接，连接结束后，主机中的“资源”（缓存和变量）将被释放。



ROUND 1:

客户端发送**连接释放报文段**，停止发送数据，主动关闭TCP连接。

FIN=1，seq=u

ROUND 2:

服务器端回送一个确认报文段，客户到服务器这个方向的连接就释放了——半关闭状态。

ACK=1，seq=v，ack=u+1

ROUND 3:

服务器端发完数据，就发出连接释放报文段，主动关闭TCP连接。

FIN=1，ACK=1，seq=w，ack=u+1

ROUND 4:

客户端回送一个确认报文段，再等到时间等待计时器设置的2MSL（最长报文段寿命）后，连接彻底关闭。 ACK=1，seq=u+1，ack=w+1

本节内容

TCP可靠传输

TCP可靠传输

传输层　使用TCP实现可靠传输

网络层　提供尽最大努力交付，不可靠传输

可靠　保证接收方进程从缓存区读出的字节流与发送方发出的字节流是完全一样的。

TCP实现可靠传输的机制

1.校验　2.序号　3.确认　4.重传

与UDP校验一样，
增加伪首部

重传

?

确认重传不分家，TCP的发送方在**规定的时间**内**没有收到确认**就要重传已发送的报文段。超时重传

重传时间

TCP采用自适应算法，动态改变重传时间RTTs（加权平均往返时间）。

等太久了！！！

冗余ACK（冗余确认）

每当比期望序号大的失序报文段到达时，发送一个**冗余ACK**，指明下一个期待字节的序号。

发送方已发送1，2，3，4，5报文段

接收方收到1，返回给1的确认（确认号为2的第一个字节）

接收方收到3，仍返回给1的确认（确认号为2的第一个字节）

接收方收到4，仍返回给1的确认（确认号为2的第一个字节）

接收方收到5，仍返回给1的确认（确认号为2的第一个字节）

发送方收到**3个对于报文段1的冗余ACK** ➡ 认为2报文段丢失，重传2号报文段　快速重传

本节内容

TCP流量控制

TCP流量控制

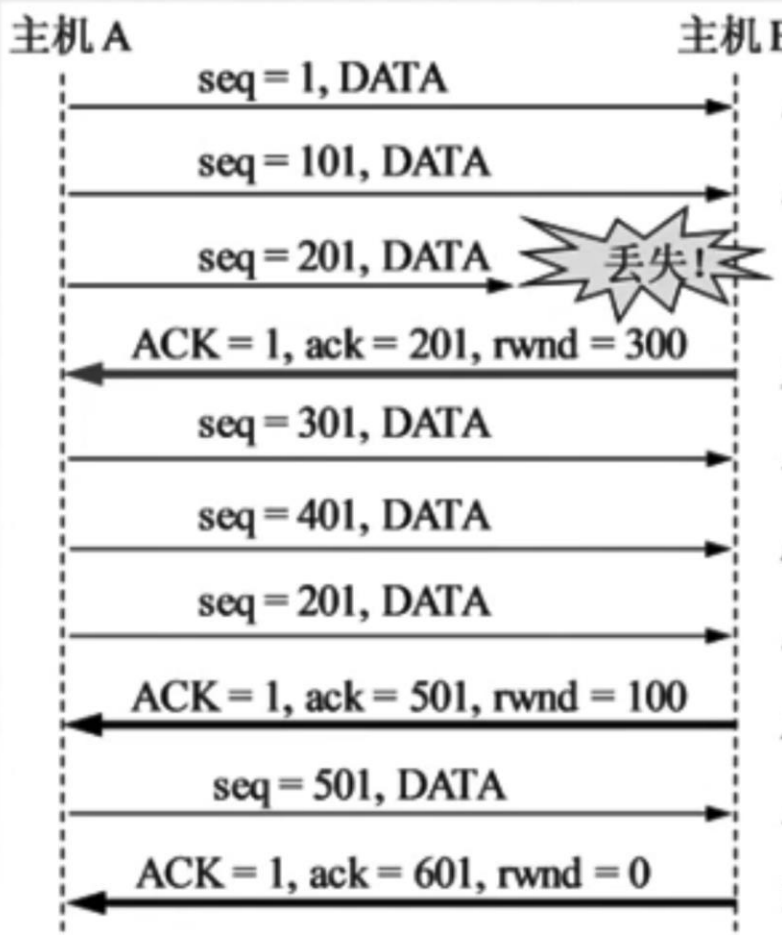
流量控制：让**发送方慢点**，要让接收方来得及接收。

TCP利用**滑动窗口**机制实现流量控制。

在通信过程中，接收方根据自己**接收缓存的大小**，动态地调整发送方的发送窗口大小，即接收窗口rwnd（接收方设置确认报文段的**窗口字段**来将rwnd通知给发送方），发送方的**发送窗口取接收窗口rwnd和拥塞窗口cwnd的最小值**。

TCP流量控制

A向B发送数据，连接建立时，B告诉A：“我的rwnd=400（字节）”，设每一个报文段100B，报文段序号初始值为1。



A发送了序号1至100，还能发送300字节

A发送了序号101至200，还能发送200字节

允许A发送序号201至500共300字节

A发送了序号301至400，还能再发送100字节新数据

A发送了序号401至500，不能再发送新数据了

A超时重发旧的数据，但不能发送新的数据

允许A发送序号501至600共100字节

A发送了序号501至600，不能再发送了

不允许A再发送（到序号 600为止的数据都收到了）

TCP为每一个连接设有一个持续计时器，只要TCP连接的一方收到对方的零窗口通知，就启动持续计时器。

若持续计时器设置的时间到期，就发送一个零窗口**探测报文段**。接收方收到探测报文段时给出现在的窗口值。

若窗口仍然是0，那么发送方就重新设置持续计时器。

TCP拥塞控制

出现拥塞的条件：

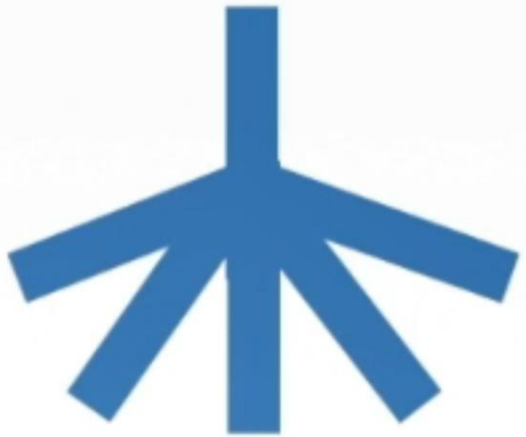
对资源需求的总和 > 可用资源

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：

防止过多的数据注入到网络中。 全局性

拥塞控制 & 流量控制



拥塞控制四种算法

慢开始　　拥塞避免

快重传　　快恢复

假定：

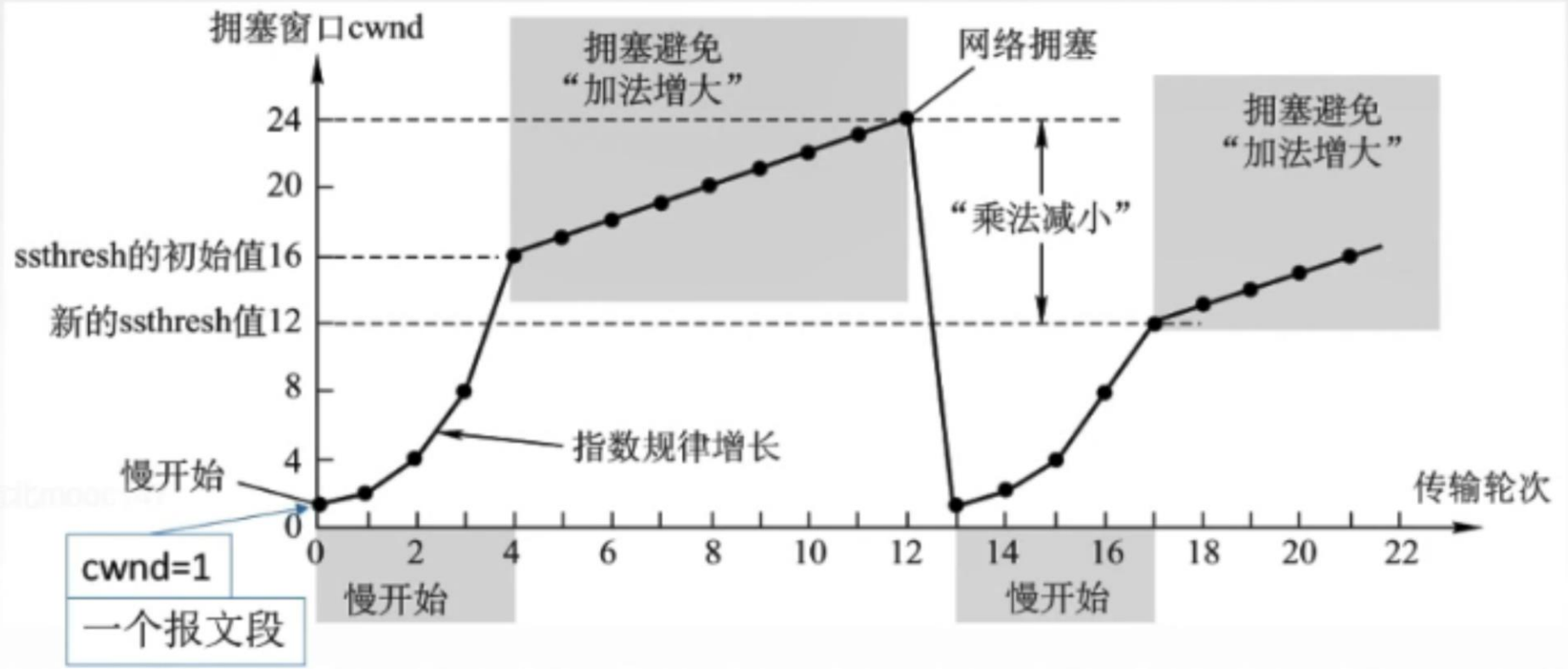
1.数据单方向传送，而另一个方向只传送确认

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{接收窗口rwnd，拥塞窗口cwnd}

接收窗口	**接收方**根据接受缓存设置的值，并告知给发送方，反映接收方容量。
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，反映网络当前容量。

慢开始和拥塞避免



一个最大报文段长度MSS

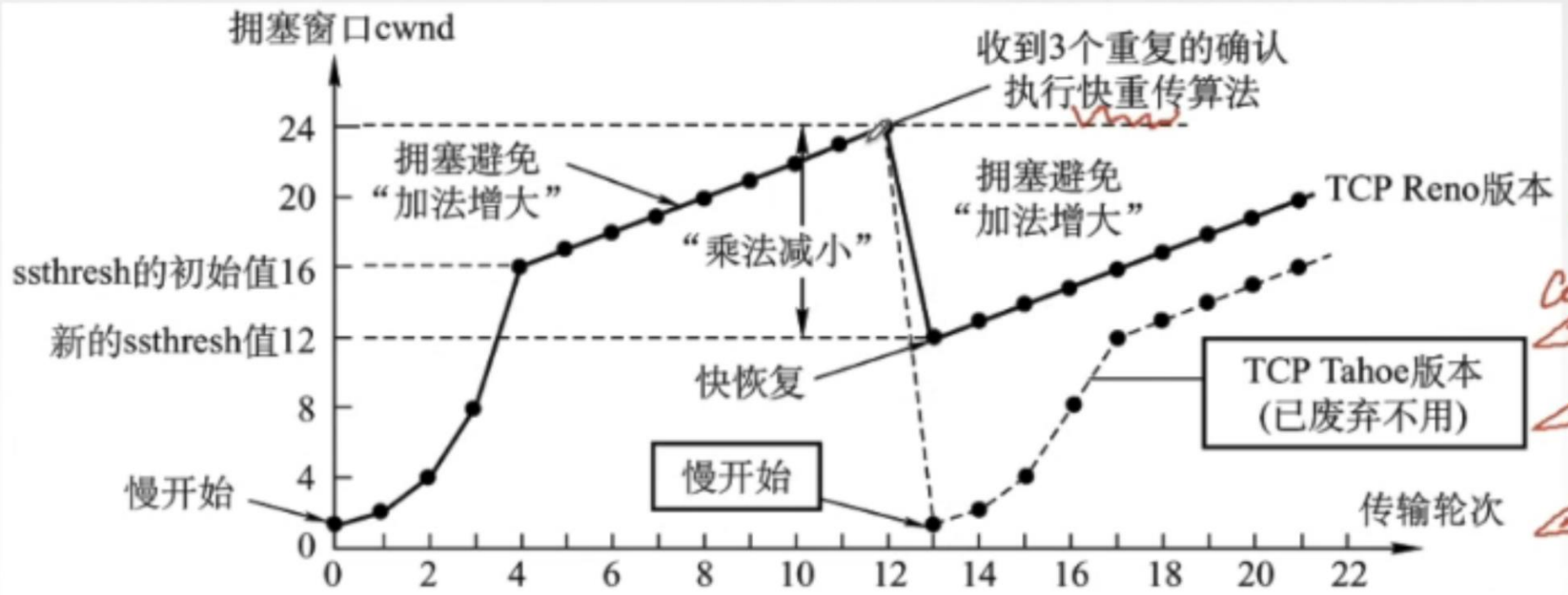
一个**传输轮次**：

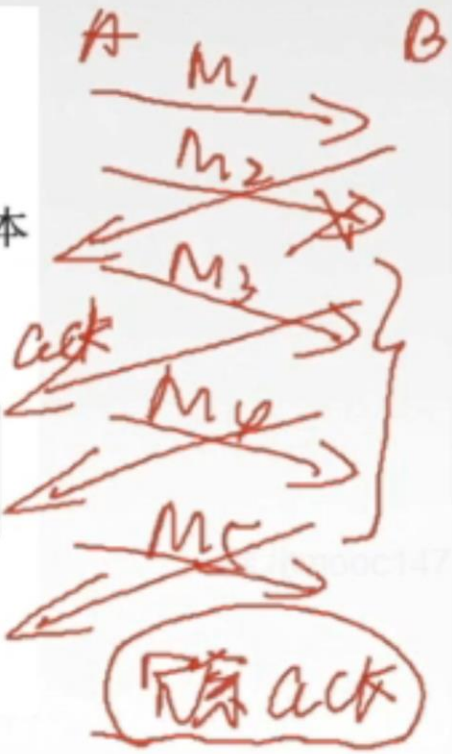
发送了一批报文段并收到它们的确认的时间。

一个往返时延**RTT**。

开始发送一批拥塞窗口内的报文段到开始发送下一批拥塞窗口内的报文段的时间。

快重传和快恢复


拥塞窗口cwnd
收到3个重复的确认
执行快重传算法
24
拥塞避免
“加法增大”
20
“乘法减小”
拥塞避免
“加法增大”
TCP Reno版本
ssthresh的初始值16
新的ssthresh值12
8
快恢复
TCP Tahoe版本
(已废弃不用)
4
慢开始
慢开始
传输轮次
0
0 2 4 6 8 10 12 14 16 18 20 22


A
B
M1
M2
M3
ack
M4
M5
丢弃ack