高数帮

课时5 传输层

课时5 传输层 考点分析

考点	重要程度	占分	题型
提供的服务	★★		选择题
UDP协议	★★★		选择题
TCP协议	★★★★		选择题、问答题

传输层提供的服务

传输层的功能

从通信和信息处理的角度看，传输层向它上面的应用层提供通信服务，它属于面向通信部分的最高层，同时也是用户功能中的最低层

传输层位于网络层之上，它为运行在不同主机上的进程之间提供了逻辑通信，而网络层提供主机之间的逻辑通信。显然，即使网络层协议不可靠（网络层协议使分组丢失、混乱或重复），传输层同样能为应用程序提供可靠的服务

5.1 服务

传输层提供应用进程之间的逻辑通信（即端到端的通信）。与网络层的区别是，网络层提供的是主机之间的逻辑通信

复用和分用。复用是指发送方不同的应用进程都可使用同一个传输层协议传送数据；分用是指接收方的传输层在剥去报文的首部后能够把这些数据正确交付到目的应用进程

传输层还要对收到的报文进行差错检测（首部和数据部分）

提供两种不同的传输协议，即面向连接的 TCP和无连接的 UDP

传输层的寻址与端口

端口的作用

端口能够让应用层的各种应用进程将其数据通过端口向下交付给传输层，以及让传输层知道应当将其报文段中的数据向上通过端口交付给应用层相应的进程

视频讲解更清晰

端口号

服务器端使用的端口号。它又分为两类，最重要的一类是熟知端口号，数值为0～1023，把这些端口号指派给了 TCP/IP 最重要的一些应用程序，让所有的用户都知道；另一类称为登记端口号，数值为 1024～49151。它是供没有熟知端口号的应用程序使用的，使用这类端口号必须先登记

一些常用的熟知端口号如下：

应用程序	FTP	TELNET	SMTP	DNS	TFTP	HTTP	SNMP
熟知端口号	21	23	25	53	69	80	161

客户端使用的端口号，数值为 49152～65535。由于这类端口号仅在客户进程运行时才动态地选择，因此又称短暂端口号（也称临时端口）

5.1 服务

传输层的寻址与端口

套接字

端口号拼接到IP 地址即构成套接字 Socket

无连接服务与面向连接服务

面向连接服务就是在通信双方进行通信之前，必须先建立连接，在通信过程中，整个连接的情况一直被实时地监控和管理。通信结束后，应该释放这个连接

5.1 服务

无连接服务是指两个实体之间的通信不需要先建立好连接，需要通信时，直接将信息发送到 "网络"中，让该信息的传递在网上尽力而为地往目的地传送

TCP/IP 协议族在IP 层之上使用了两个传输协议：一个是面向连接的传输控制协议（TCP），采用TCP时，传输层向上提供的是一条全双工的可靠逻辑信道；另一个是无连接的用户数据报协协议（UDP），采用 UDP 时，传输层向上提供的是一条不可靠的逻辑信道。

5.2 UDP协议

UDP 数据报

UDP 概述

UDP仅在 IP 的数据报服务之上增加了两个最基本的服务：复用和分用以及差错检测。如果应用开发者选择 UDP而非TCP，那么应用程序几乎直接与IP打交道

视频讲解更清晰

5.2 UDP协议

UDP具有如下优点：

UDP无须建立连接。因此 UDP不会引入建立连接的时延

无连接状态。TCP 需要在端系统中维护连接状态。此连接状态包括接收和发送缓存、拥塞控制参数和序号与确认号的参数

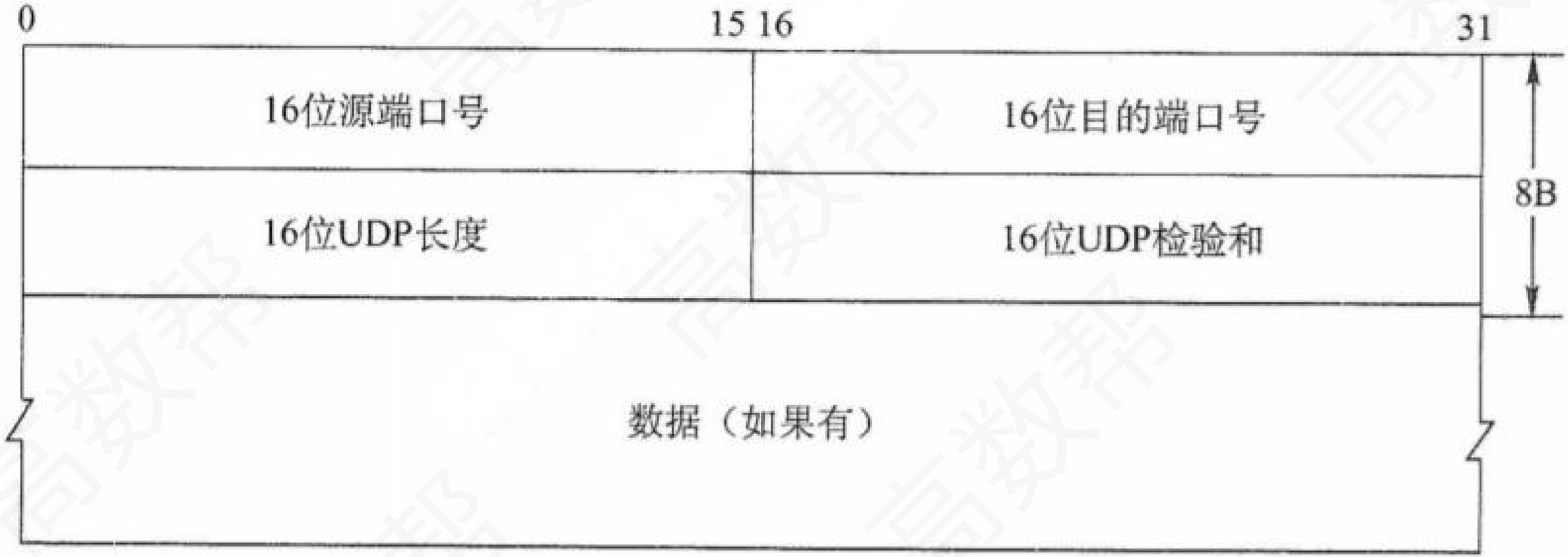
分组首部开销小。TCP有 20B 的首部开销，而 UDP仅有8B 的开销

应用层能更好地控制要发送的数据和发送时间。UDP 没有拥塞控制，因此网络中的拥塞不会影响主机的发送效率

UDP 支持一对一、一对多、多对一和多对多的交互通信

UDP 数据报

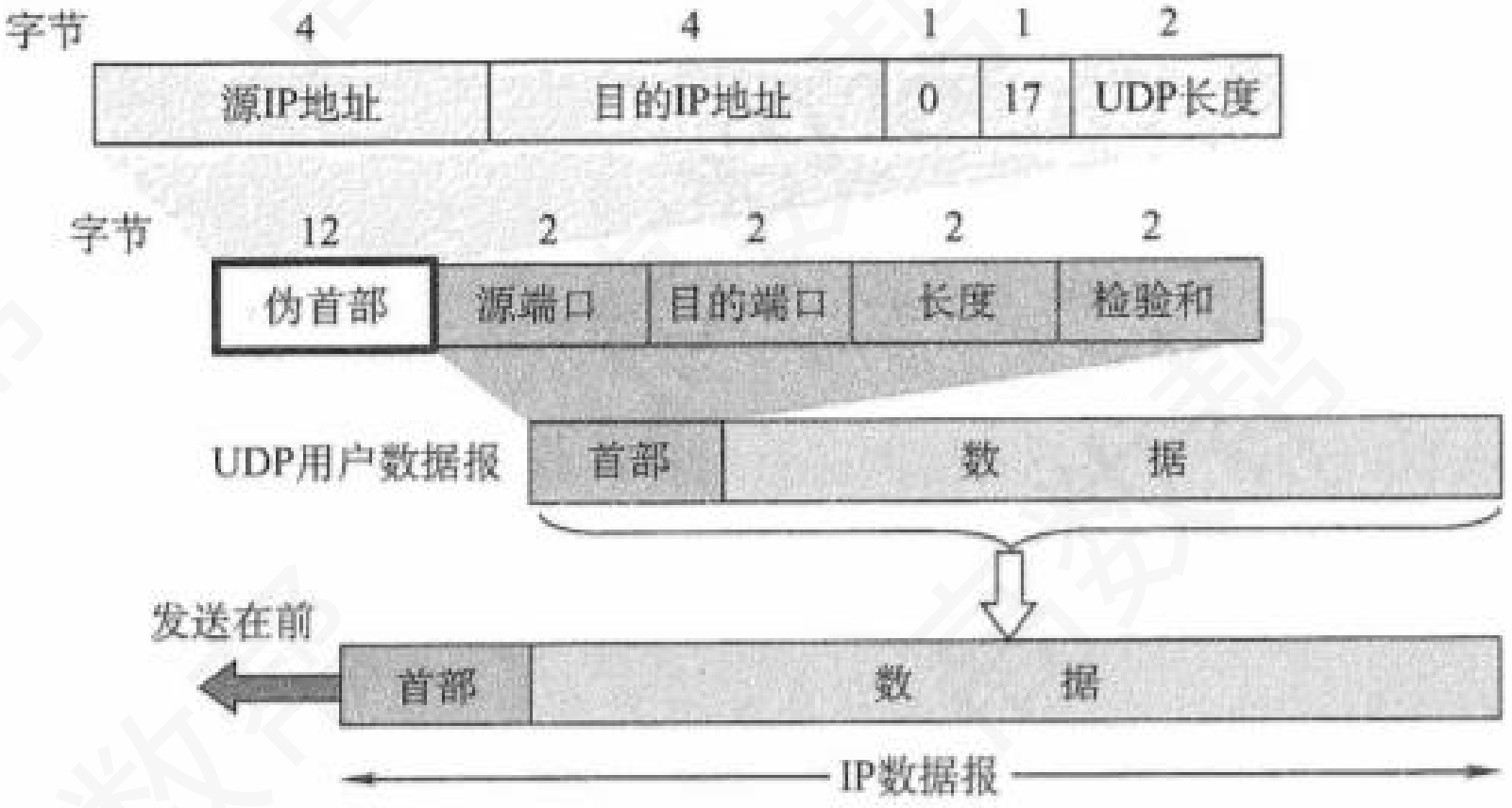
UDP 首部格式



5.2 UDP协议

UDP校验

在计算校验和时，要在UDP数据报之前增加 12B的伪首部，伪首部并不是UDP的真正首部。只是在计算校验和时，临时添加在 UDP数据报的前面，得到一个临时的UDP数据报。校验和就是按照这个临时的 UDP数据报来计算的。伪首部既不向下传送又不向上递交，而只是为了计算校验和



5.3 TCP协议

TCP协议的特点

TCP 是在不可靠的 IP 层之上实现的可靠的数据传输协议，它主要解决传输的可靠、有序、无丢失和不重复问题。TCP 是TCP/IP 体系中非常复杂的一个协议，主要特点如下：

TCP是面向连接的传输层协议

每条 TCP连接只能有两个端点，每条 TCP 连接只能是点对点的（一对一）

TCP提供可靠的交付服务，保证传送的数据无差错、不丢失、不重复且有序

5.3 TCP协议

TCP协议的特点

TCP提供全双工通信，允许通信双方的应用进程在任何时候都能发送数据，为此 TCP连接的两端都设有发送缓存和接收缓存，用来临时存放双向通信的数据

TCP 是面向字节流的，虽然应用程序和 TCP的交互是一次一个数据块（大小不等），但TCP把应用程序交下来的数据仅视为一连串的无结构的字节流

视频讲解更清晰

TCP 报文段

TCP 传送的数据单元称为报文段。TCP报文段既可以用来运载数据，又可以用来建立连接、释放连接和应答

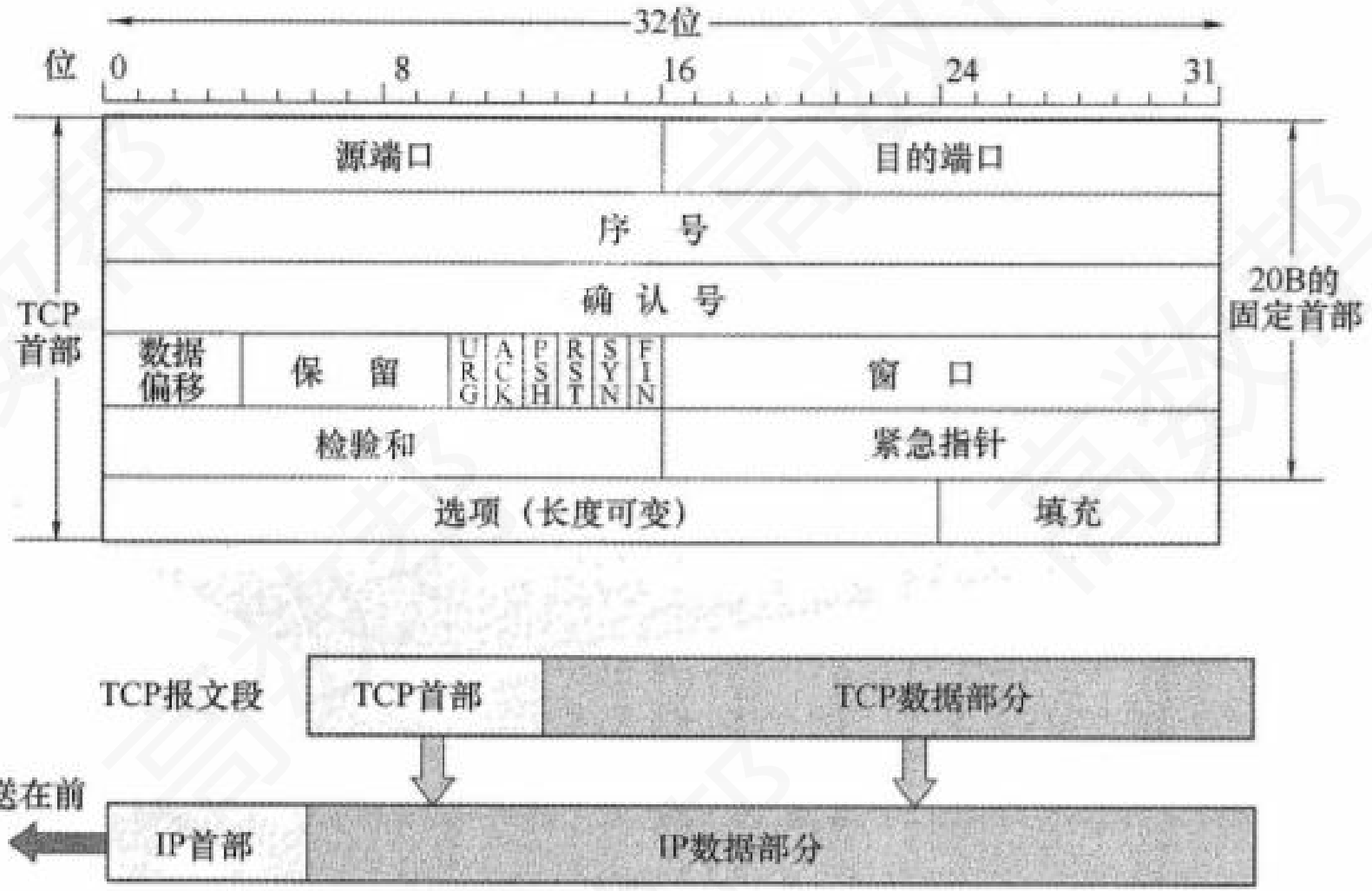
一个 TCP报文段分为首部和数据两部分，整个 TCP报文段作为 IP数据报的数据部分封装在 IP数据报中，其首部的前 20B 是固定的。TCP报文段的首部最短为 20B，后面有 4N字节是根据需要而增加的选项，通常长度为 4B 的整数倍

TCP 报文段

确认号。占4B，是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， 则表明到序号 N-1为止的所有数据都已正确收到。

数据偏移（即首部长度）。占4位，它指出 TCP报文段的数据起始处距离TCP报文段的起始处有多远。"数据偏移"的单位是32位（以4B为计算单位）。因此当此字段的值为 15 时，达到 TCP 首部的最大长度 60B

TCP 报文段

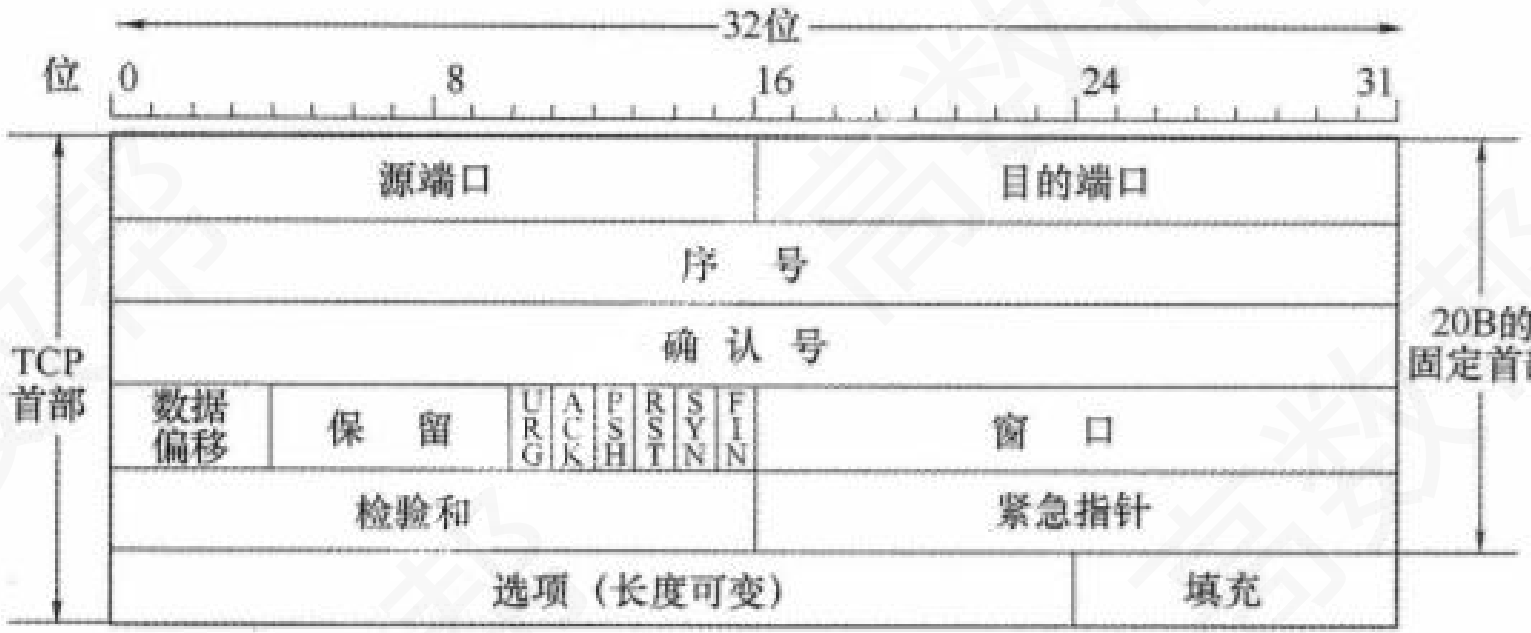


急位 URG。URG=1时，表明紧急指针字段有效。它告诉系统此报文段中有紧急数据，应尽快传送（相当于高优先级的数据）

确认位ACK。仅当ACK=1时确认号字段才有效。当ACK=0时，确认号无效。TCP规定，在连接建立后所有传送的报文段都必须把 ACK 置 1

TCP 报文段

推送位PSH（Push）。接收方TCP收到PSH=1的报文段，就尽快地交付给接收应用进程，而不再等到整个缓存都填满后再向上交付



同步位 SYN。同步 SYN=1表示这是一个连接请求或连接接受报

终止位 FIN（Finish）。用来释放一个连接。当 FIN = 1时，表明此报文段的发送方的数据已发送完毕，并要求释放传输连接。

TCP 连接管理

每个 TCP 连接都有三个阶段：连接建立、数据传送和连接释放。TCP 连接的管理就是使运输连接的建立和释放都能正常进行

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称为客户（Client），而被动等待连接建立的应用进程称为服务器（Server）

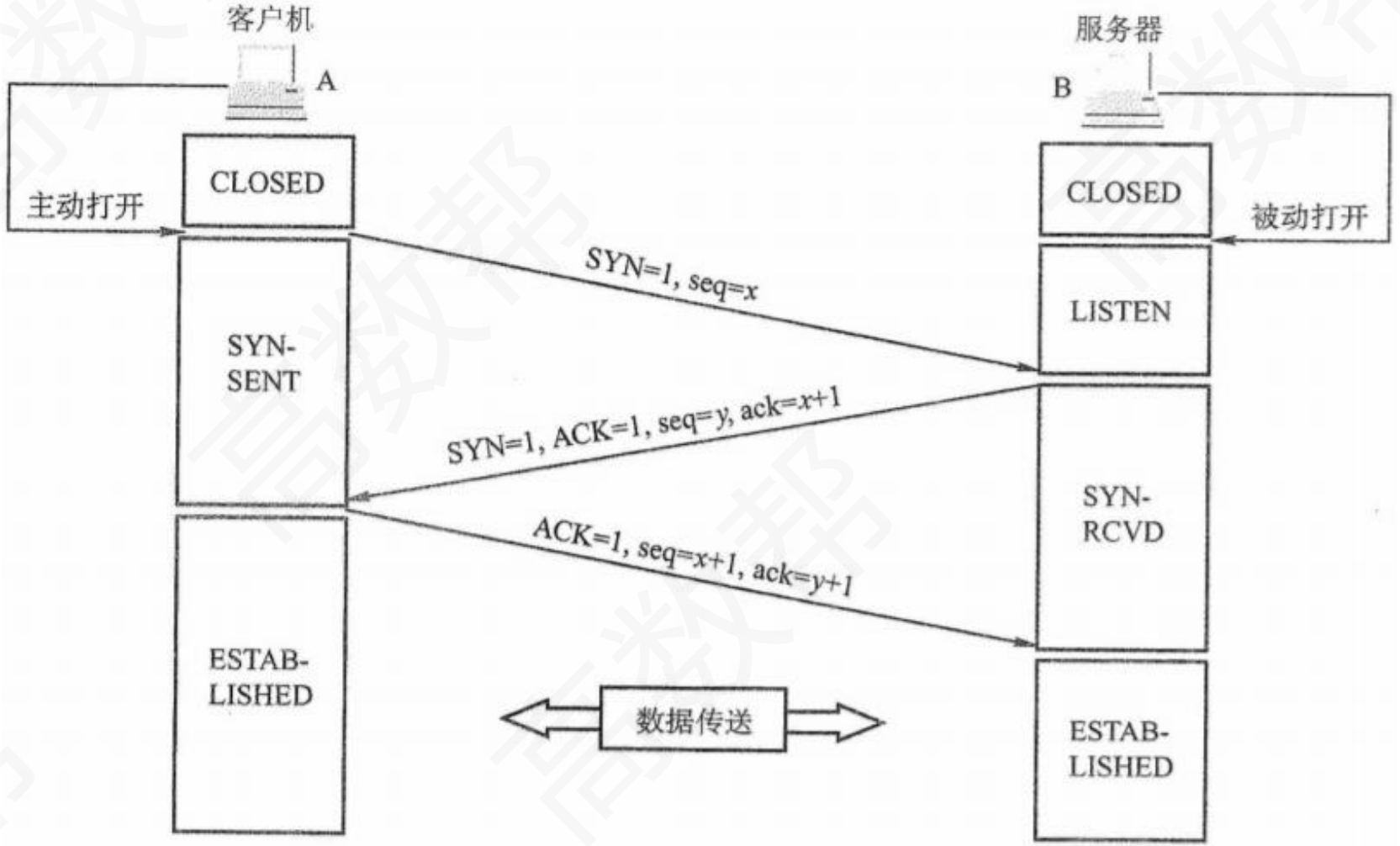
视频讲解更清晰

5.3 TCP协议

TCP 连接的建立

连接的建立经历以下3个步骤，通常称为三次握手

第一步：客户机的TCP首先向服务器的 TCP发送连接请求报文段。这个特殊报文段的首部中 的同步位SYN置1，同时选择一个初始序号seq=x。TCP规定，SYN报文段不能携带数据，但要消耗掉一个序号

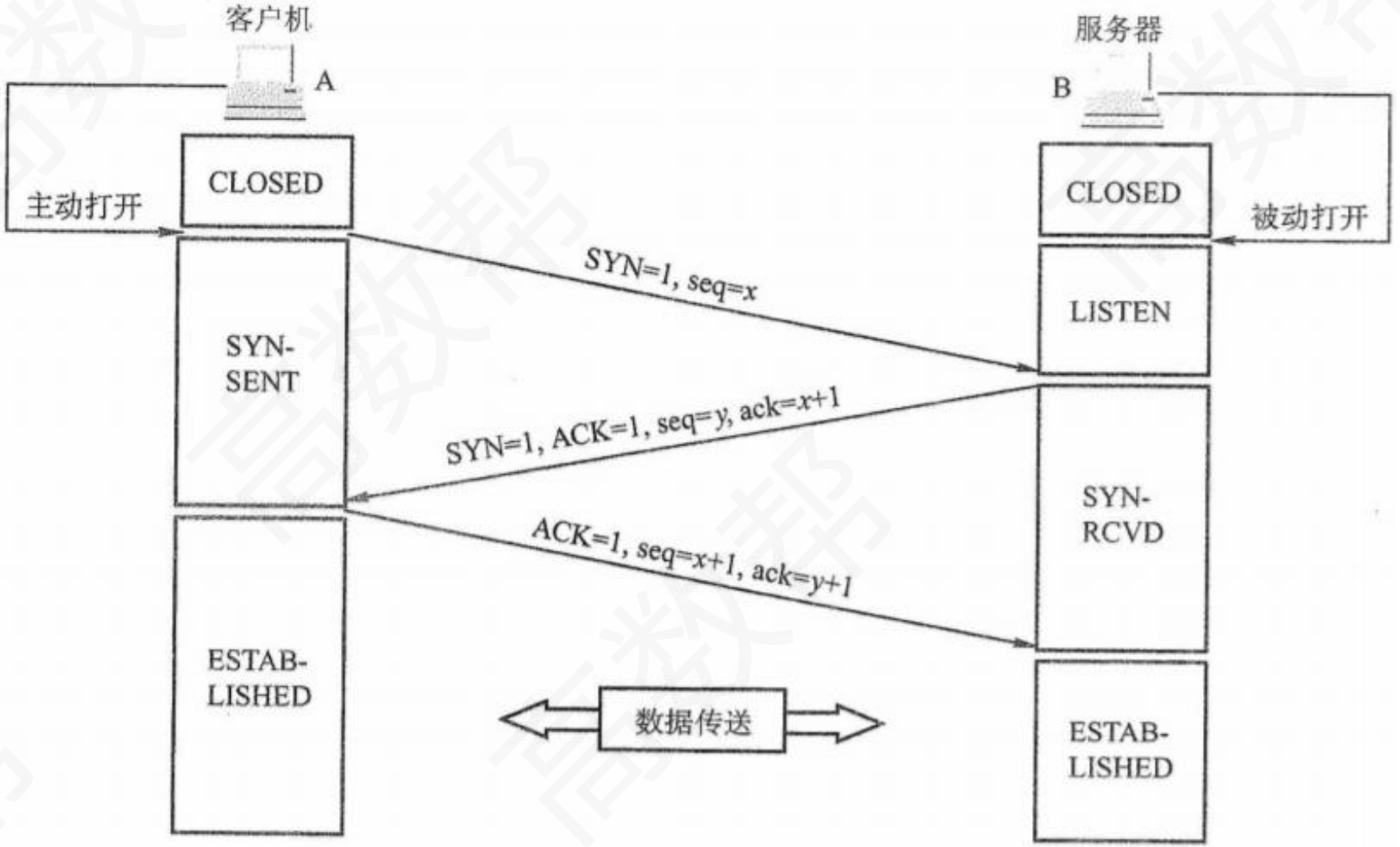


5.3 TCP协议

TCP 连接的建立

连接的建立经历以下3个步骤，通常称为三次握手

第二步：服务器的TCP 收到连接请求报文段后，如同意建立连接，则向客户机发回确认，并为该TCP连接分配缓存和变量。在确认报文段中，把SYN位和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scq=y。注意，确认报文段不能携带数据，但也要消耗掉一 个序号

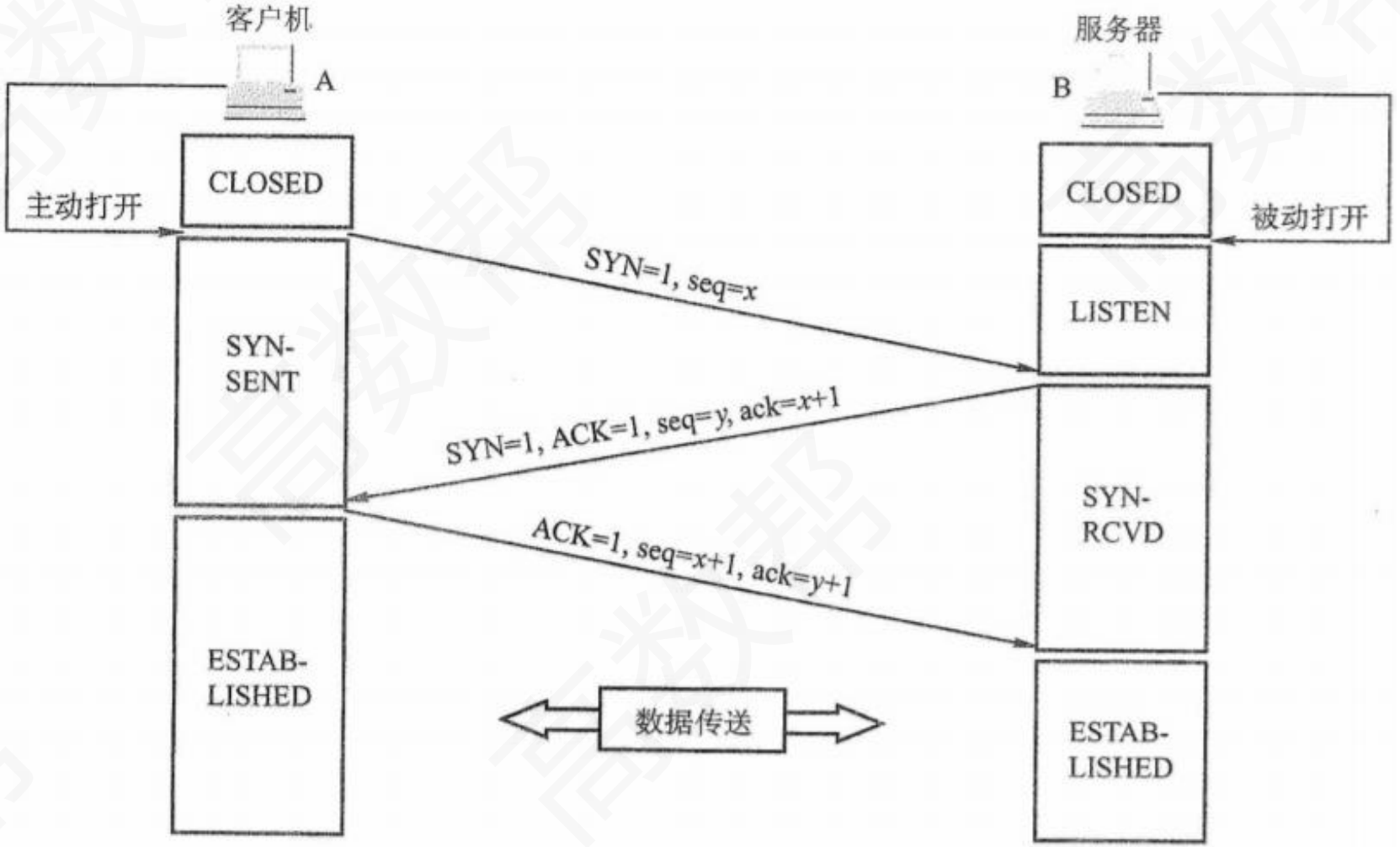


5.3 TCP协议

TCP 连接的建立

连接的建立经历以下3个步骤，通常称为三次握手

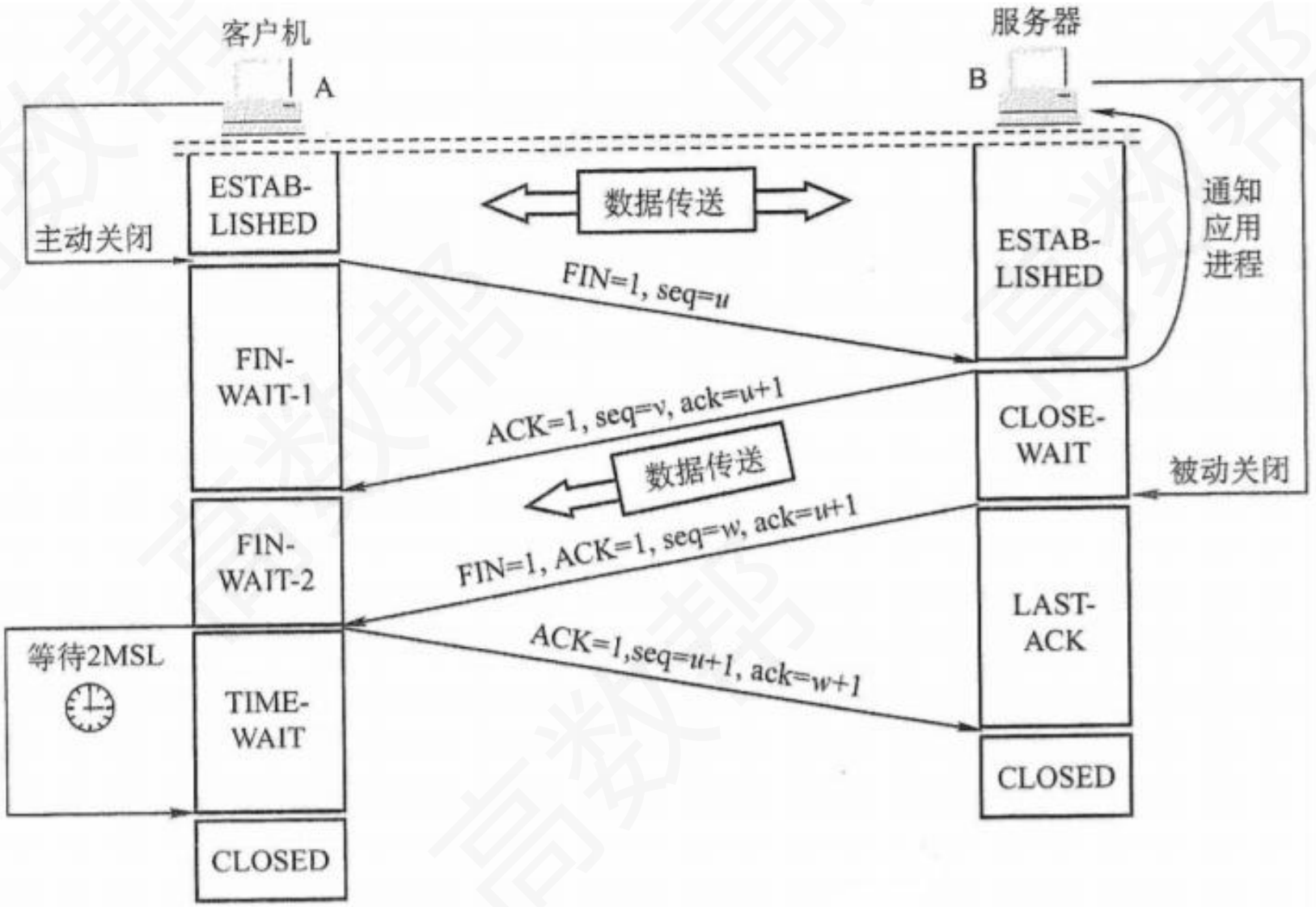
第三步：当客户机收到确认报文段后，还要向服务器给出确认，并为该TCP连接分配缓存和变量。确认报文段的ACK位置1，确认号ack=y+1，序号seq=x+1。该报文段可以携带数据，若不携带数据则不消耗序号



TCP 连接管理

TCP 连接的释放 通常称为四次挥手

第一步：客户机打算关闭连接时，向其TCP发送连接释放报文段，并停止发送数据，主动关闭TCP连接，该报文段的终止位FIN置1，序号seq=u，它等于前面已传送过的数据的最后一个字节的序号加1，FIN报文段即使不携带数据，也消耗掉一个序号



TCP 连接的释放 通常称为四次挥手

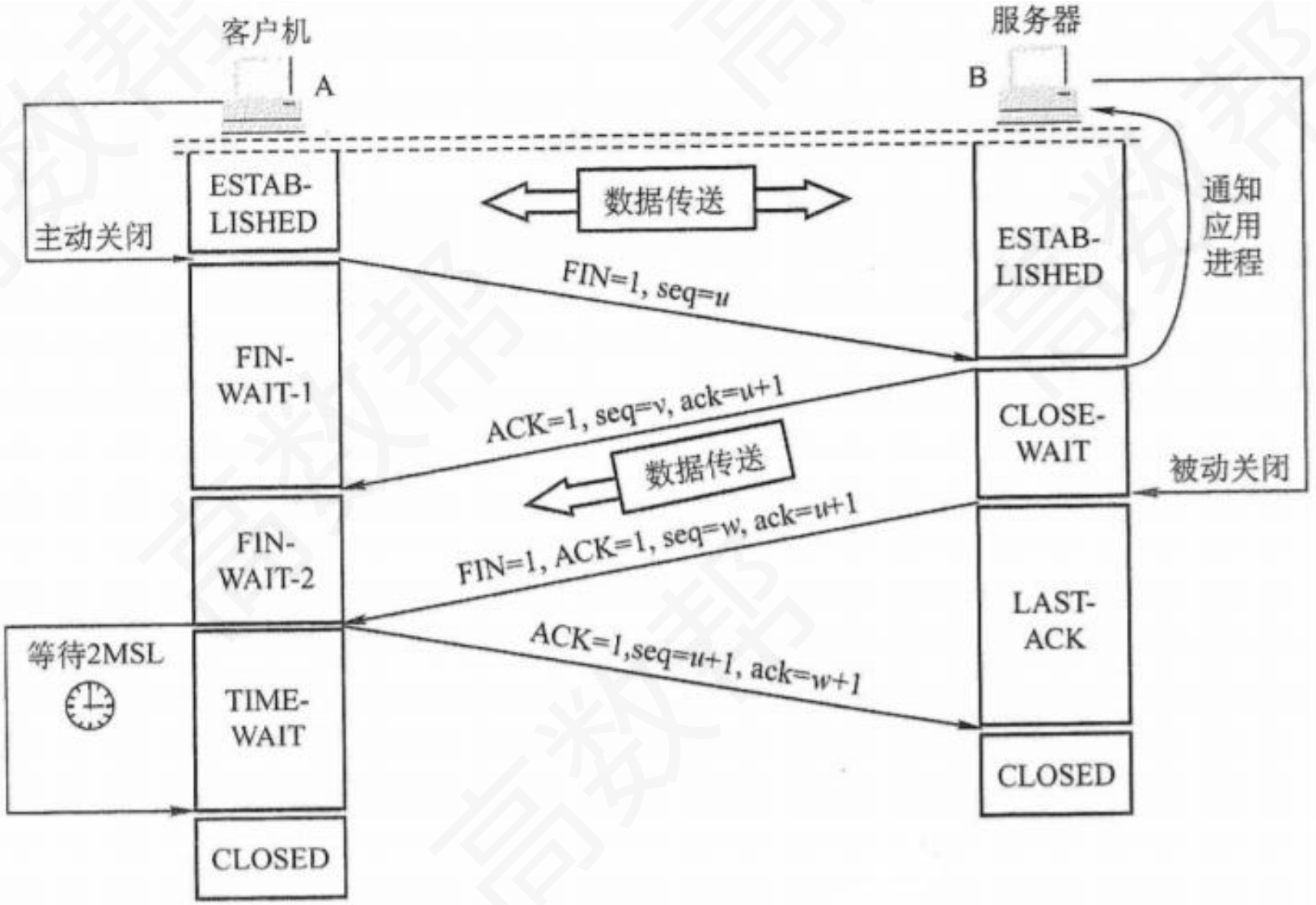
TCP 连接管理

第二步：服务器收到连接释放报文段后即发出确认，确认号ack=u+1，序号seq=w，等于它前面已传送过的数据的最后一个字节的序号加1

此时，从客户机到服务器这个方向的连接就释放了

第三步

第四步



TCP 连接管理

对上述 TCP连接建立和释放的总结如下:

1）连接建立。分为3步:

① SYN=1,seq=x。

②SYN=1,ACK=1,seq=y,ack=x+1。

③ACK=1,seq=x+1,ack=y+1。

2）释放连接。分为 4步:

①FIN=1,seq =u。

②ACK=1,seq=v, ack=u+1。

③FIN=1,ACK=1,seq=w,ack=u+1。

④ACK=1,seq=u+1,ack=w+1。

5.3 TCP协议

TCP可靠传输

序号

TCP首部的序号字段用来保证数据能有序提交给应用层，TCP 把数据视为一个无结构但有序的字节流，序号建立在传送的字节流之上，而不建立在报文段之上

确认

TCP首部的确认号是期望收到对方的下一个报文段的数据的第一个字节的序号

TCP默认使用累计确认，即 TCP 只确认数据流中至第一个丢失字节为止的字节

重传

两种事件会导致 TCP 对报文段进行重传：超时和冗余 ACK

超时：TCP每发送一个报文段，就对这个报文段设置一次计时器。计时器设置的重传时间到期但还 未收到确认时，就要重传这一报文段

冗余ACK（冗余确认）：冗余 ACK 就是再次确认某个报文段的 ACK，而发送方先前已经收到过该报文段的确认

5.3 TCP协议

TCP流量控制

输层和数据链路层的流量控制的区别是：传输层定义端到端用户之间的流量控制，数据链路层定义两个中间的相邻结点的流量控制。另外，数据链路层的滑动窗口协议的窗口大小不能动态变化，传输层的则可以动态变化

TCP 拥塞控制

拥塞控制是指防止过多的数据注入网络，保证网络中的路由器或链路不致过载

拥塞控制与流量控制的区别：拥塞控制是让网络能够承受现有的网络负荷，是一个全局性的过程，涉及所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素。相反，流量控制往往是指点对点的通信量的控制，是个端到端的问题（接收端控制发送端），它所要做的是抑制发送端发送数据的速率，以便使接收端来得及接收

5.3 TCP协议

接收窗口rwnd，接收方根据目前接收缓存大小所许诺的最新窗口值，反映接收方的容量。由接收方根据其放在 TCP 报文的首部的窗口字段通知发送方

拥塞窗口cwnd，发送方根据自己估算的网络拥塞程度而设置的窗口值，反映网络的当前容量。只要网络未出现拥塞，拥塞窗口就再增大一些，以便把更多的分组发送出去。但只要网络出现拥塞，拥塞窗口就减小一些，以减少注入网络的分组数。 发送窗口的上限值 = min[rwnd，cwnd]

TCP 拥塞控制

1.慢开始和拥塞避免

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。用这样的方法逐步增大发送方的 cwnd，可使分组注入网络的速率更加合理

视频讲解更清晰

例如，A向B发送数据，发送方先置拥塞窗口cwnd=1，A发送第一个报文段，A收到B对第一个报文段的确认后，把cwnd从1增大到2；于是A接着发送两个报文段，A收到B对这两个报文段的确认后，把 cwnd从2 增大到4，下次就可一次发送4个报文段。

慢开始一直把 cwnd增大到一个规定的慢开始门限 sstresh（阈值），然后改用拥塞避免算法

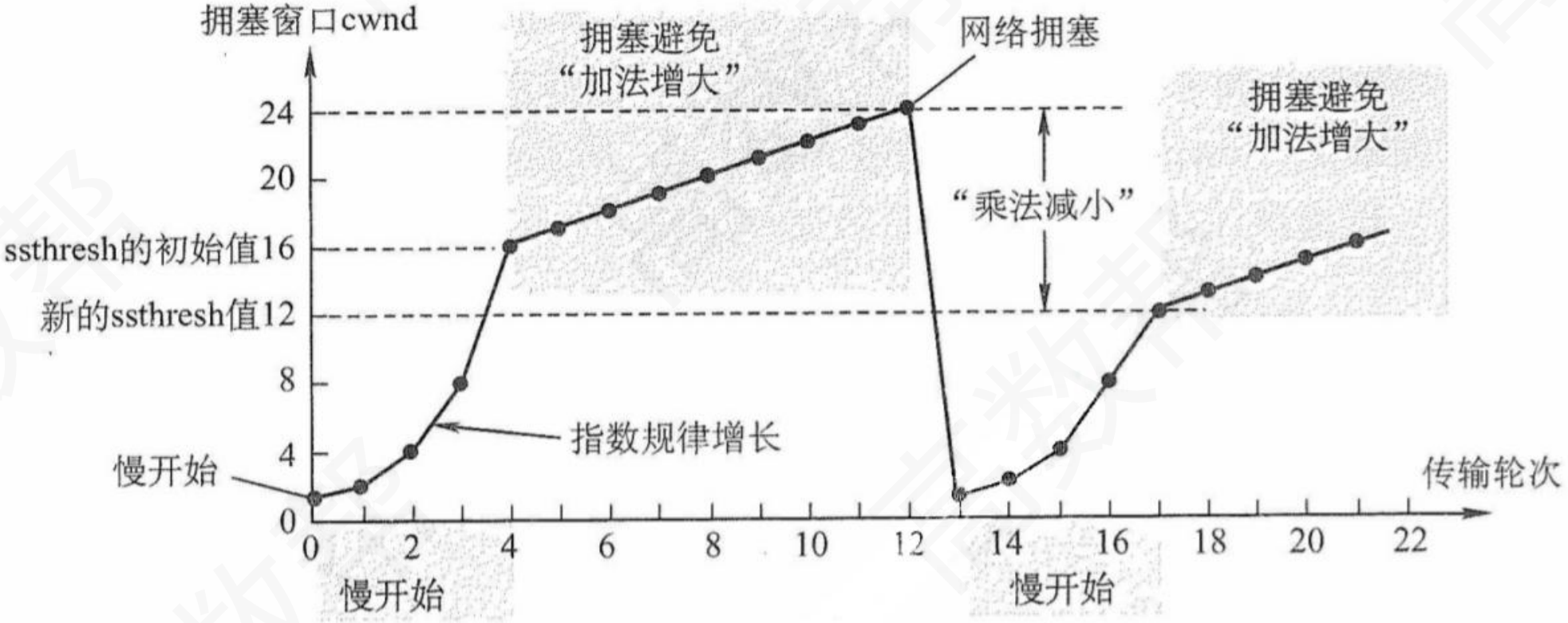
拥塞避免算法

每经过一个往返时延 RTT 就把发送方的拥塞窗口cwnd加1，而不是加倍，使拥塞窗口cwnd按线性规律缓慢增长（即加法增大），这比慢开始算法的拥塞窗口增长速率要缓慢得多

5.3 TCP协议

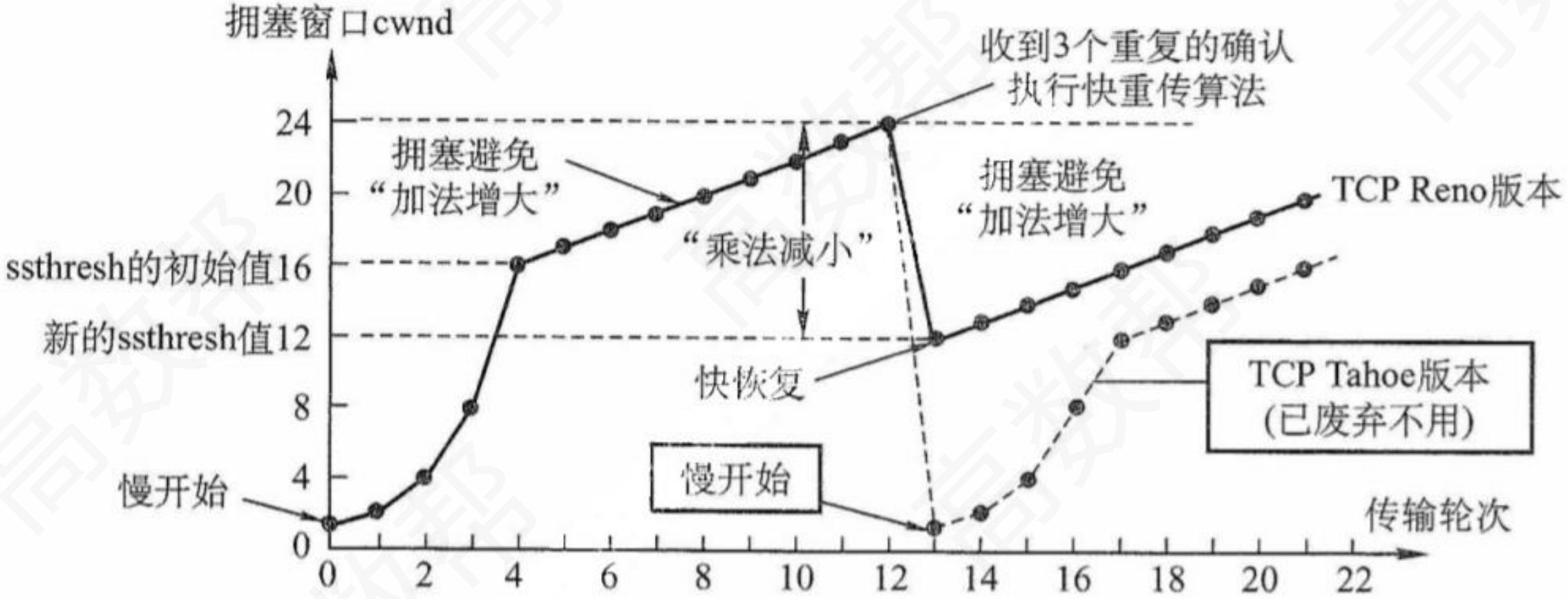
网络拥塞的处理

无论在慢开始阶段还是在拥塞避免阶段，只要发送方判断网络出现拥塞（未按时收到确认），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cwnd 重新设置为1，执行慢开始算法



2.快重传和快恢复

快重传 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



快恢复

当发送方连续收到三个冗余 ACK（即重复确认）时，把慢开始门限 ssthresh 设置为此时发送方cwnd 的一半。与慢开始不同之处是它把cwnd 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（"加法增大"），使拥塞窗口缓慢地线性增大