第六章 传输层协议

课前思考

- 传输层的任务是什么？
- 传输层协议主要涉及哪些内容？
- 为什么说传输层协议是真正的端到端协议？
- 传输地址与网络地址有什么不同？
- 出错或丢失的IP分组会重发吗？

本章内容

6.1 传输层概述

6.2 传输服务质量

6.3 传输层协议机制

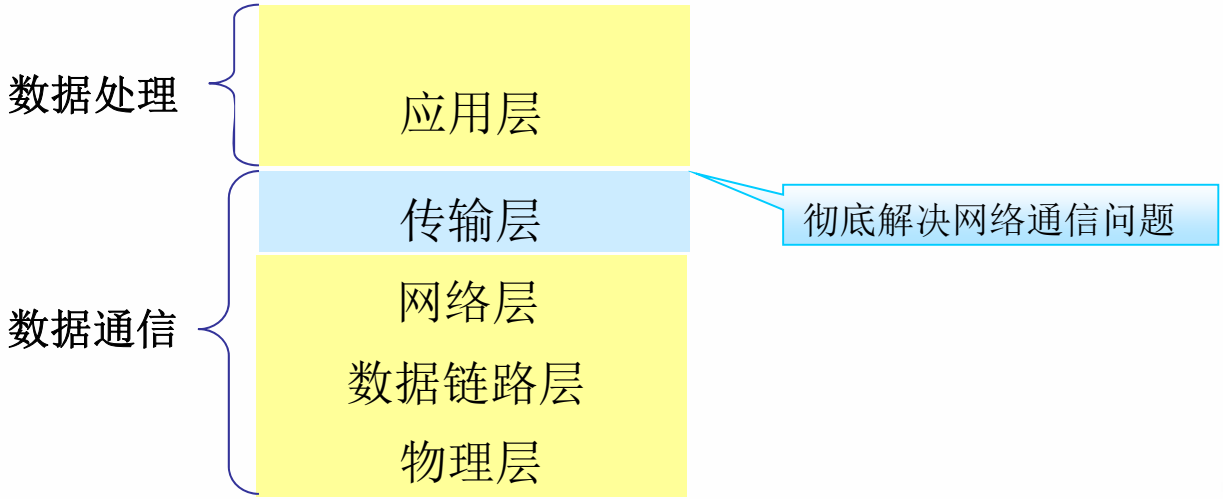
6.4 TCP协议

6.5 UDP协议

6.1 传输层概述

- 传输层地位

  位于网络体系结构的中间，作为数据通信和数据处理的分水岭，具有承上启下的作用。



- 传输层功能

  完成主机进程－主机进程之间的报文传输。

- 传输层是真正的端对端的通信。

  - 实现了进程之间的通信
    - IP协议虽然能把分组交给目的主机，但是这个分组还停留在目的主机的网络层而没有交付给主机中的应用进程。
    - 严格来讲，两个主机进行通信实际上就是两个主机中的应用进程互相通信，应用进程之间的通信又称为端到端的通信。

从网络传输质量的角度

- TCP/IP的网络层是一个典型的提供无连接“尽力而为”的不可靠服务，IP分组在传输过程中可能会出现丢包、乱序或重复等问题，需要在网络层之上增加一个层次来弥补网络层所提供的服务质量的不足，以便为高层提供可靠的端到端通信。
- 网络层及以下部分是由通信子网来完成的，由于历史及经济原因，通信子网往往是公用数据网，是资源子网中的端用户所不能直接控制的，用户不可能通过更换性能更好的路由器或增强数据链路层的纠错能力来提高网络层的服务质量，因此端用户只能依靠在自己主机上所增加的传输层来检测分组的丢失或数据的残缺并采取相应的补救措施。

传输层引入的新概念与新机制

- 一系列实现端到端进程之间的可靠数据传输所必需的机制，包括：
  - ➔ 面向连接服务的建立机制，即能够为高层数据的传输建立、维护与拆除传输连接，以实现透明的、可靠的端到端的传输；
  - ➔ 端到端的错误恢复与流量控制，以能对网络层出现的丢包、乱序或重复等问题做出反应。

传输地址

- 传输层与网络层最大的区别是传输层提供进程通信能力。为了标识相互通信的网络进程，IP网络通信的最终地址不仅要包括主机的IP地址，还要包括可描述网络进程的某种标识。因此，无论是TCP还UDP，都必须首先解决进程的标识问题。
- 网络进程标识。在单机上，为了区别不同的进程，采用进程标识或进程号（Process ID）来唯一地标识进程。即在网络环境中，完整的进程标识需要这样的一种形式：源主机地址＋源进程标识，目标主机地址＋目标进程标识。

传输地址

- 端口号在TCP/IP传输层的作用类似IP地址在网络层的作用或MAC地址在数据链路层的作用
  - IP地址是主机的逻辑标识
  - MAC地址是主机的物理标识
  - 而端口号是网络应用进程的一种逻辑标识
- 由于同一时刻一台主机上可以有大量的网络应用进程在运行，因此需要有多个不同的端口号来对进程进行标识

传输地址

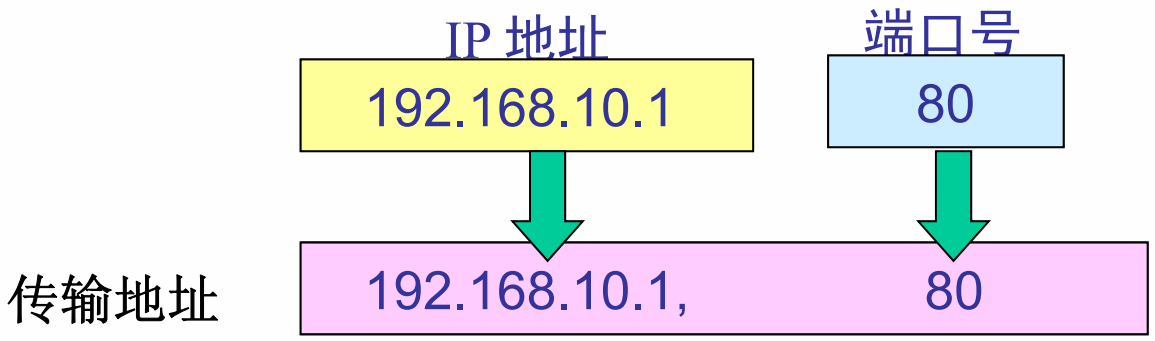
- 端口号有两种基本分配方式，即全局分配和本地分配方式
  - 全局分配是指由一个公认权威的机构根据用户需要进行统一分配，并将结果公布于众，因此这是一种集中分配方式
  - 本地分配是指当进程需要访问传输层服务时，向本地系统提出申请，系统返回本地唯一的端口号，进程再通过合适的系统调用，将自己和该端口绑定起来，因此是一种动态连接方式

传输地址

- 实际的TCP/IP端口号分配综合了以上两种方式，由Internet赋号管理局（IANA）将端口号分为著名端口（well—known ports）、注册端口和临时端口3个部分：
  - （1）著名端口号：取值为0～1023，由IANA统一分配和控制，被规定作为公共应用服务的端口，如WWW、FTP、DNS、NFS和电子邮件服务等。
  - （2）注册端口号：取值为1024～49151，这部分端口被保留用作商业性的应用开发，如一些网络设备厂商专用协议的通信端口等。厂商或用户可根据需要向IANA进行注册，以防止重复。
  - （3）临时端口号：取值为49152～65535，这部分端口未做限定，由本地主机自行进行分配，因此又被称为自由端口

传输地址

- 传输地址唯一地标识主机进程
- 传输地址=网络号+主机号+端口号，端口号用来标识应用进程
- 在IP网络，传输地址= IP地址+端口号

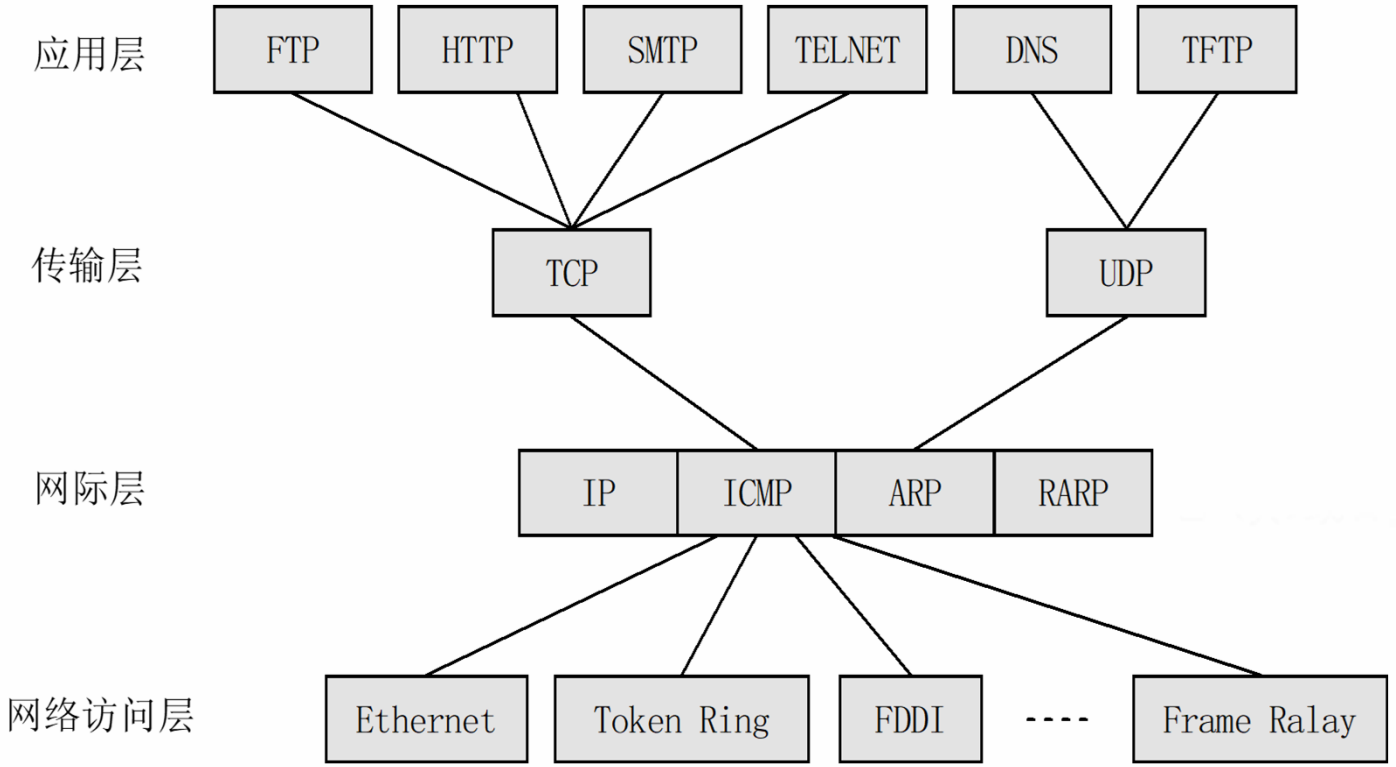


传输层协议涉及的内容

- 传输层协议涉及的内容
  - 寻址：传输地址及其发现。
  - 差错控制：分组顺序、差错检测、重传确认机制等。
  - 流量控制
  - 连接管理：连接建立、连接释放。

传输层的两个主要协议

- 传输层需要有两种不同的传输协议，即用户数据报协议（User Datagram Protocol，UDP）和传输控制协议（Transmission Control Protocol，TCP），它们都是因特网的正式标准。



UDP协议

- UDP在传送数据之前不需要先建立连接。对方的传输层在收到UDP报文后，不需要给出任何确认。虽然UDP不提供可靠交付，但在某些情况下UDP是一种最有效的工作方式。

- 传输层的UDP用户数据报与网际层的IP数据报有很大区别
  - IP数据报要经过互联网中许多路由器的存储转发，但UDP用户数据报是在传输层的端到端抽象的逻辑信道中传送的


該起床囉
我盡力了
譯：好色龍
THIS COMIC MADE POSSIBLE THANKS
MRLOVENSTEIN.COM

要实现可靠的数据流传输服务，必须解决哪几个问题？

1.可靠性：

①防丢失：确认与重传；

②防重复：报文段序号；

2.传输效率、流量控制：滑动窗口机制；

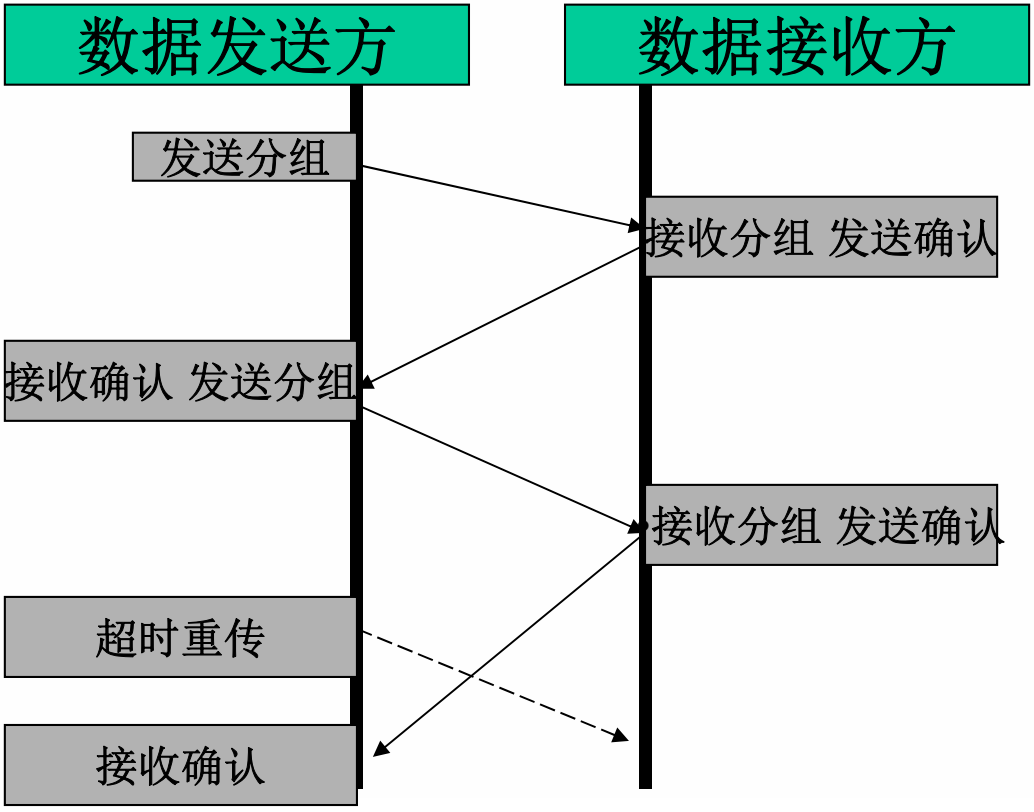
3.拥塞控制：加速递减与慢启动技术；

4.建立连接：三次握手协议；

5.关闭连接：三次握手协议。

提供可靠性

1. 防丢失 – 带重传的肯定确认技术



① 接收方收到数据后向源站发确认（ACK）；

② 设置定时器，源站在限定时间内未收到ACK，则重发。

如何对待重复的数据？

2. 防重复 – 可捎带的累计确认技术

① 为每一分组赋予序号。

② 确认时也指明确认哪个分组。

③ 序号同时保证了分组间的正确顺序。

3. RTT与重传定时器

- 重传定时器主要处理重传时间(RTO)，即报文段的确认等待时间
- 当TCP发送一个报文段时，就创建该报文段的重传定时器
- 若在定时器到期前收到了该报文段的确认，则撤销该定时器
- 若在定时器到期前未收到对该报文段的确认,则重传该报文段，并将该定时器复位

TCP流量管理

- 为什么要进行流量控制

  发送方TCP实体发送数据过快，接收方TCP实体来不及处理，导致接收缓冲区溢出。

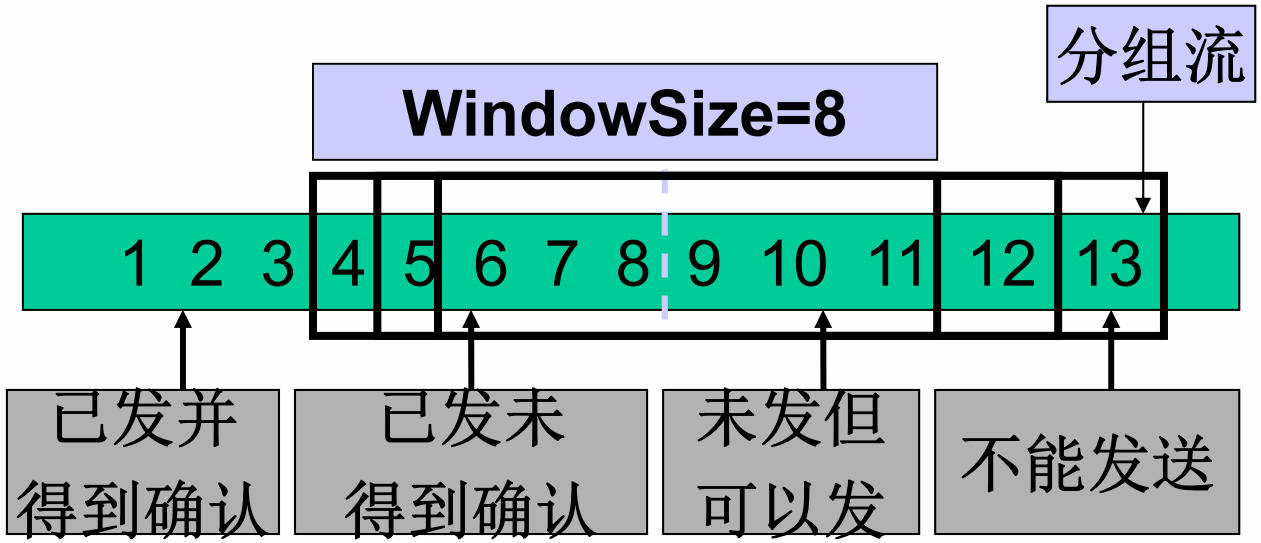
- TCP流量控制采用可变尺寸的滑动窗口协议
  - 由于TCP为应用层提供字节流服务，窗口尺寸以字节为单位。
  - 应用进程根据处理能力读取TCP接收缓冲区中的字节流，导致空闲的接收缓冲区（**接收窗口**）动态变化。
- TCP流量控制过程

  类似于数据链路层的滑动窗口协议，所不同的是：通过接收窗口当前尺寸调整发送窗口的上限值。

传输效率和流量控制-滑动窗口机制

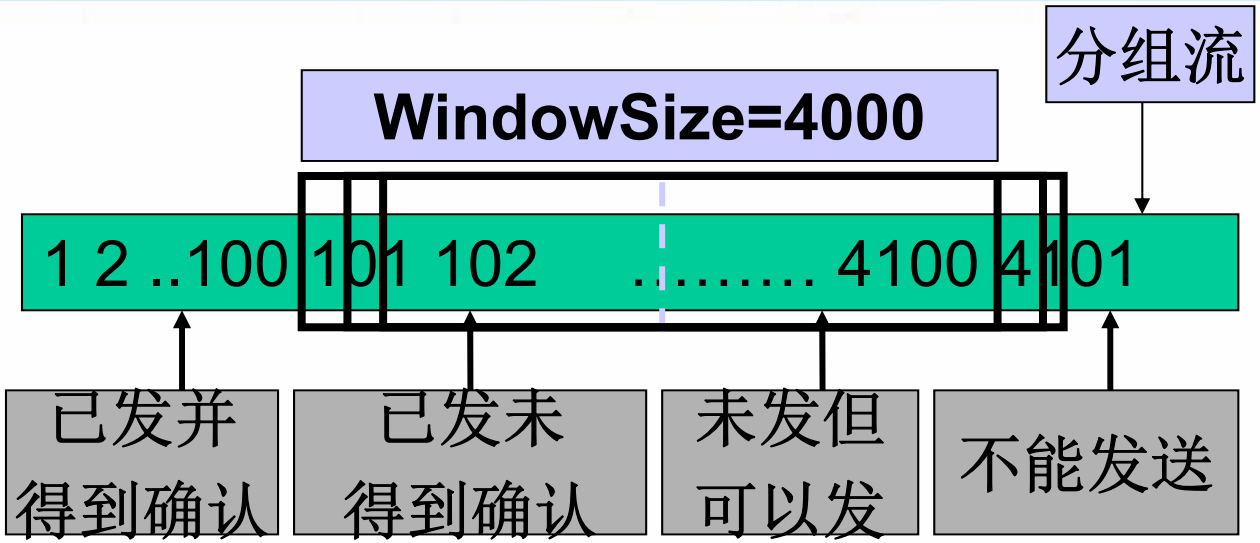
1.一般的滑动窗口机制

思想：允许发送方不必等确认到来就可继续发送下面的分组，但规定一个上限。若多个分组的确认未到时，则暂停发送。



2. TCP的滑动窗口技术

(5) TCP连接两端各有两个窗口（发送窗口和接收窗口）



(1)数据流的各字节被编上序号。

(2)TCP的滑动窗口按字节操作而不是按报文段或分组操作。

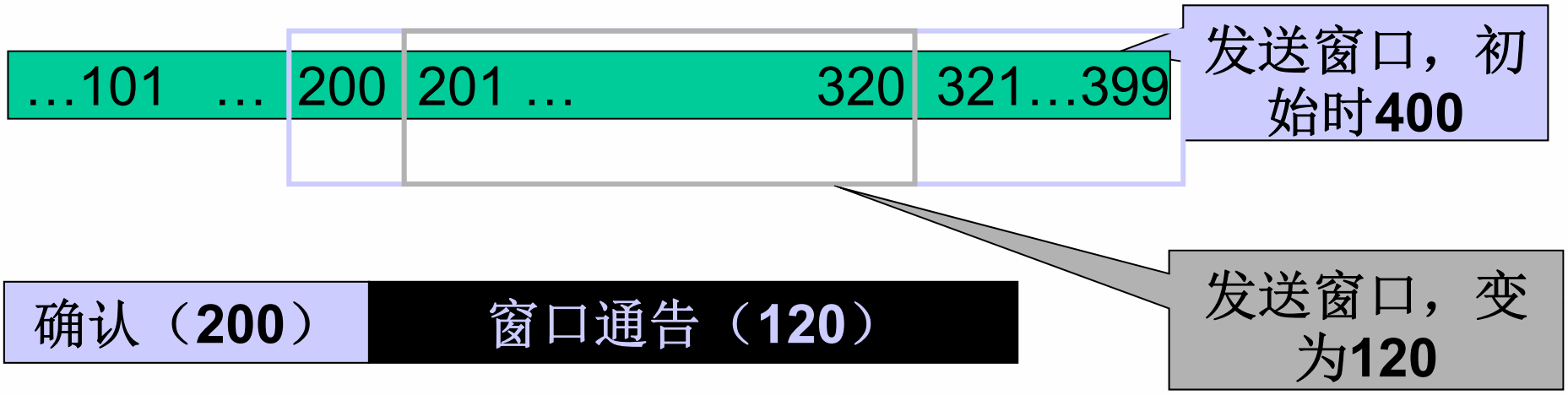
(3)TCP窗口大小为字节数。最大为65535字节。

(4)通信双方都设有发送和接收缓冲区（相当于发送窗口和接收窗口）。默认大小各系统有差异，如4096、8192、16384等。发送缓冲区大小为默认窗口大小。

3. TCP端到端流量控制 – 窗口大小可变技术

时机：目的主机缓冲区变小而不能接收源主机更多的数据时，就要进行流量控制。

TCP技术：可随时改变窗口大小。目的主机在确认时，还向源主机告知目的主机接收缓冲区的大小。



说明：接收方使用0窗口通告来停止所有的传输。此时，除了紧急数据和窗口试探报文外，不发其它数据 。

死锁问题

- 发送方收到确认报文后停止发送，等待接收方发送一个窗口≠0的确认报文后再启动发送。一段时间后，接收方发送了一个窗口≠0的确认报文，
- 但若该确认报文丢失了（发送方未收到）。
  - ➔结果是发送方一直等待不能发送，接收方接收不到新的报文，导致死锁发生。

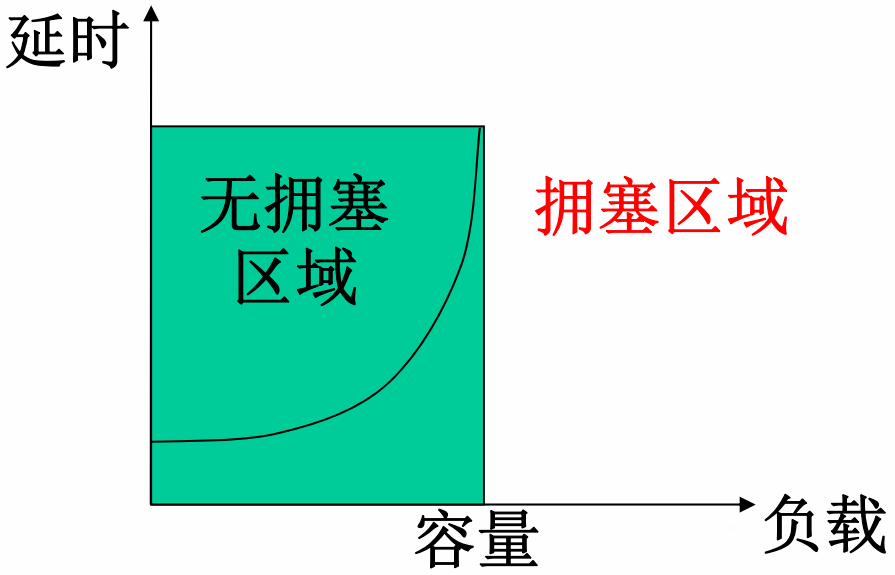
死锁的解决方法

- TCP为每一个连接设计一个持续计时器。只要TCP连接的一方收到对方的窗口=0的确认报文，就启动持续计时器。
- 若持续计时器设置的时间到期仍未收到对方发送的窗口≠0的确认报文时，就发送一个窗口=0的探测报文（仅携带1字节的数据），对方则在发送的该探测报文的确认报文中给出现在的窗口值。
- 若窗口仍然是0，则收到这个报文的一方再重新设置持续计时器继续等待；若窗口≠0，则开始发送数据。

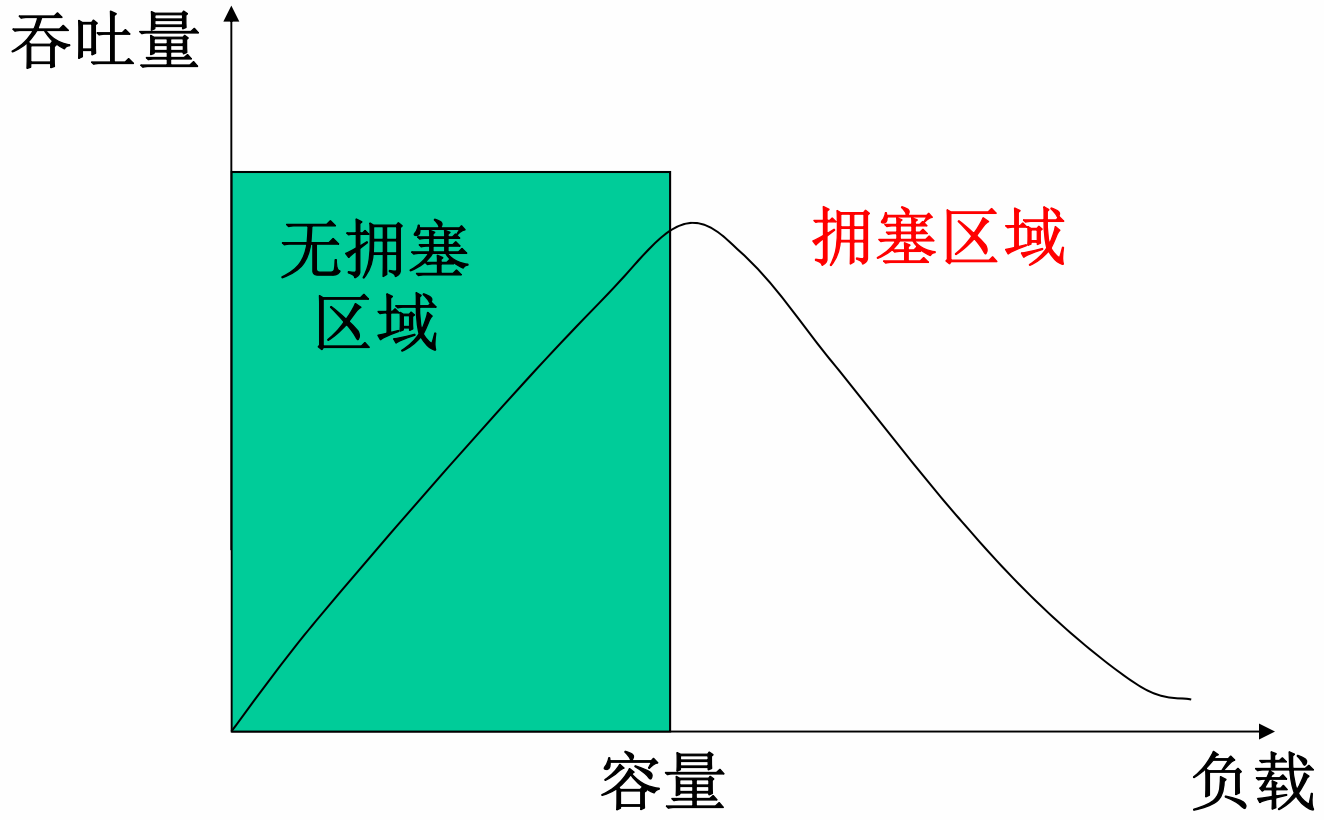
拥塞

- 拥塞：交换节点（如路由器）数据报负载过重的现象
- 网络或互联网会产生拥塞，是因为路由器和交换机有队列(队列是在处理分组前或后存放分组的缓存)，通常有输入队列和输出队列。
  - ➔ 如果分组的到达速率大于分组的处理速率，输入队列就会变得越来越长
  - ➔ 反之，输出队列则越来越长
- 控制拥塞首先是检测，然后是处理

- 拥塞控制涉及网络性能的两个因素:
  - ➔延时和吞吐量
- 延时和负载
  - ➔ 负载<网络容量时，延时为最小值
  - ➔ 负载≈网络容量时，延时就急剧增大
  - ➔ 负载>网络容量时，延时就变为无穷大



吞吐量和网络负载关系图



TCP拥塞控制

- TCP的拥塞控制与流量控制的区别
  - 拥塞控制是为了防止网络过载，导致路由器缓冲区不足而造成IP分组的丢失。
  - 流量控制是为了防止发送端发送速度过快，导致接收端缓冲区不足而造成TCP段的丢失。

TCP拥塞控制

- TCP拥塞控制策略
  - 发送端设置拥塞窗口来反映网络容量，通过拥塞窗口来限制发送方向网络注入数据的速度，即发送端允许发送的数据既不能超过接收窗口的大小，也不能超过拥塞窗口的大小；前者是为了流量控制，后者是为了拥塞控制。
  - 通过慢启动和拥塞避免策略来控制拥塞窗口的大小。

TCP拥塞控制技术

回顾：IP层的拥塞控制技术：

（ICMP源站抑制报文），是一种被动机制。

TCP拥塞控制的必要性：在TCP层，拥塞造成时延增加，这又会造成超时重传，控制不当会进一步加重拥塞。

TCP采用了一种主动控制机制。

拥塞控制技术：

① 拥塞窗口cwnd

② 加速递减技术

③ 慢启动技术

① 拥塞窗口

- 每个发送方均保持两个窗口
  - ➔ 接收方承认的窗口(允许窗口awnd)
  - ➔ 拥塞窗口(cwnd)
    - ➢ 该窗口大小以字节为单位，但是增加和减少以MSS(Maximum Segment Size)为单位；
    - ➢ 初始大小：1个MSS；
    - ➢ 临界值：64KB

① 拥塞窗口

- 每个窗口(awnd和cwnd)都反映出发送方可以传输的字节数
- 取两个窗口中的最小值作为可以发送的字节数
- 有效窗口为发送和接收方分别认为合适的窗口中的最小的
- 拥塞窗口保持指数规律增大,直到数据传输超时或者达到接收方设定的窗口大小

② 慢启动技术

- 慢启动算法

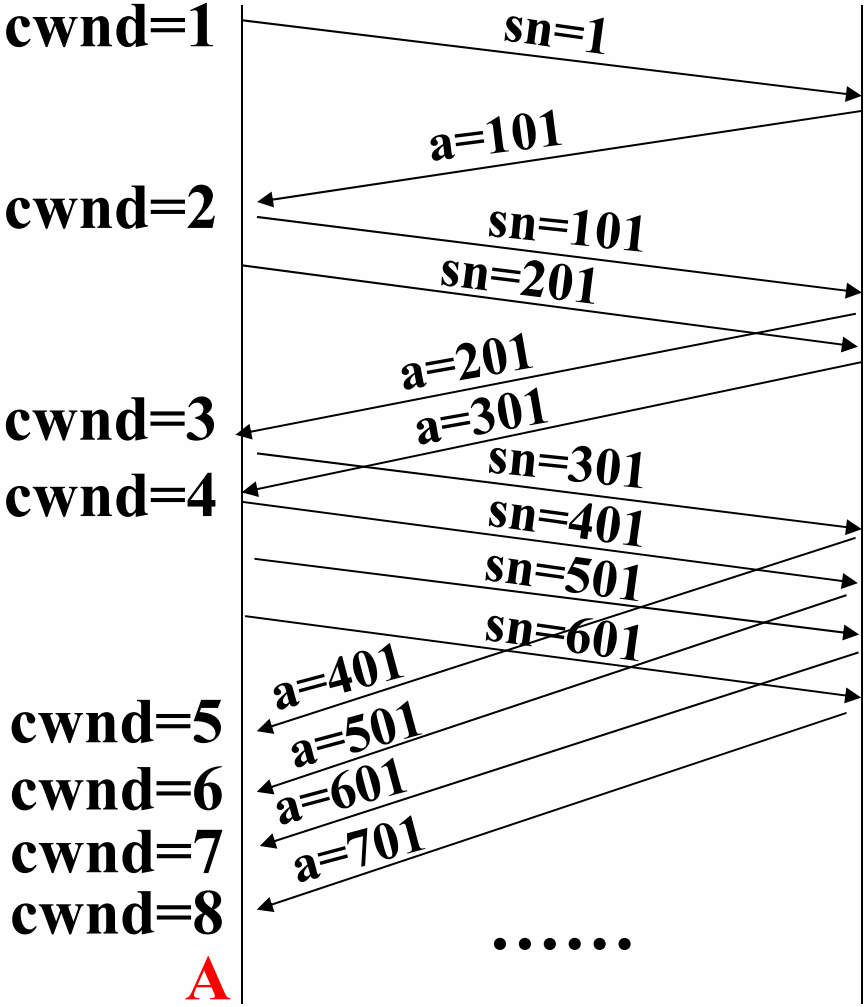
  - ➔ 慢启动为发送方TCP增加一个拥塞窗口cwnd
  - ➔ 当与另一个网络的主机建立TCP连接时，拥塞窗口被初始化为1个报文段(即另一端通告的报文段大小)

慢启动算法

- 每收到一个ACK，拥塞窗口就增加一个报文段(cwnd以字节为单位,慢启动以报文段大小为单位进行增加)
- 发送方取拥塞窗口与通告窗口中的最小值为发送上限
- 拥塞窗口是发送方使用的流量控制，而通告窗口是接收方使用的流量控制

慢启动的示意和影响

每收到一个ACK，cwnd增1



每个报文段100字节

大约4个往返时间，A就会用一个连续的段流将管道填满

② 慢启动技术

指数递增：每次成功发送1个MSS长度的报文段，则发送方拥塞窗口加倍；

线性递增：增长到临界值后，每次增加1个MSS

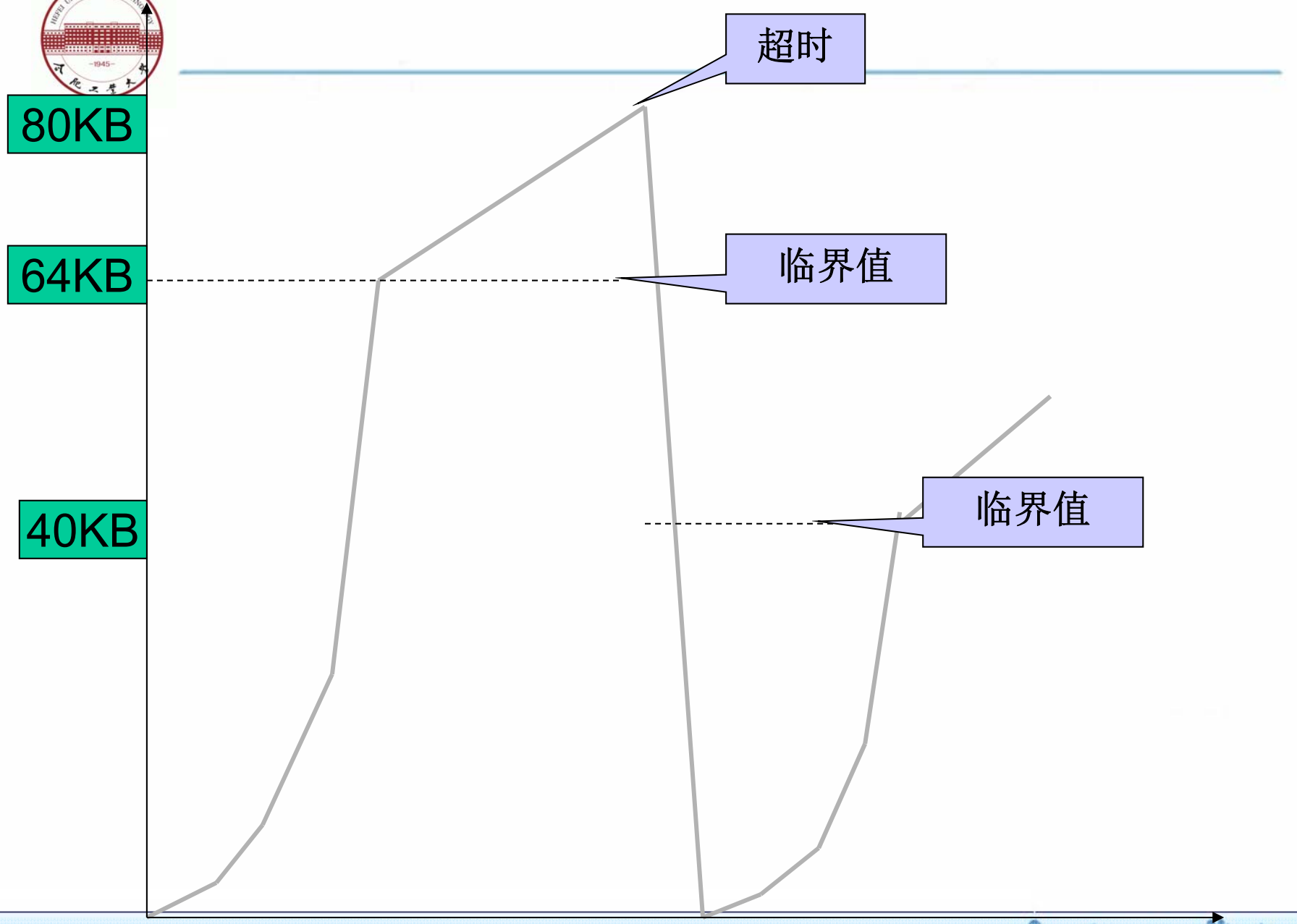
发送窗口 = min (接收方窗口通告，cwnd)

- 假定由于分组受到损坏引起的丢失是非常少的(远小于1%)，因此分组丢失就意味着在源主机和目标主机之间的某处网络上发生了拥塞。
- 有两种分组丢失的指示
  - 发生超时和接收到重复的确认

③ 加速递减技术

指数级递减：出现超时重传时，将临界值设为当前拥塞窗口的1/2，拥塞窗口恢复为1个MSS大小；

指数退避：对保留在发送窗口中的报文段，将重传时限加倍。


超时
80KB
64KB
临界值
临界值
40KB

TCP确认机制的特性

确认号：希望接收的对方下一报文段序号(已成功接收到的数据字节序+1)

说明：

(1)序号和确认序号在一起使得确认可捎带进行。

(2)TCP采用累计确认策略。

累计确认

- 仅对连续接收的数据段进行确认（累计确认）
  - 返回的确认数据段中的“确认号”字段值仅代表对端已正确接收的连续数据段（最高字节序号+1）
  - 不一定是已正确接收数据段中的“最高序号+1”，因为中间可能还有数据因为网络延迟暂时未收到，或出现了传输错误而丢失了
  - 假设每个数据段的长度大小均为100字节，接收端收到了序号为1、101、201、401四个数据段。其中序号301的数据段暂时未收到，此时接收端返回的确认数据段中的“确认号”是301，而不是501，也就是对前三个数据段进行确认，不会对后面的401数据段进行确认。当后面收到了301数据段后，可能会返回一个“确认号”为501的数据段，表示301和401数据段均已正确接收。

不连续序号的数据将先被缓存

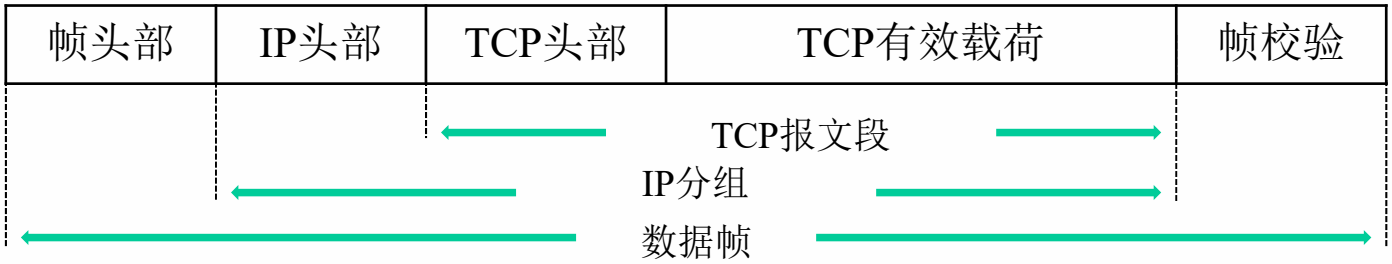
- 当主机接收到的数据段序号不连续时，不连续部分不会向应用层进行提交，而是先缓存在“接收窗口”中，等待接收到终端的序号的数据段后再一起提交。
- 对于没有按先后次序正确接收的数据，在向应用层提交时会重新按数据段序号进行组合，然后再提交给应用层。
- 如果主机先后接收到序号分别为1、101、201、301、601、501、801的数据段（假设数据段大小均为100字节）
  - 先向应用层提交1、101、201、301数据段
  - 收到401号数据段后，再按401、501、601顺序重组，并向应用层提交

问题：采用累计确认，为什么？

缺点：发送方无法收到所有成功传输的报文段的确认信息，对往返时间样本的精确测量带来影响。

6.4 TCP协议

- TCP最初在Unix环境下实现，现在也在Windows环境下实现，通过Socket提供服务，见P.261。
- 报文段（即TPDU)封装在IP分组中， IP分组封装在数据帧中。



- TCP提供全双工数据传输服务，如果一台主机进程A和另一台主机进程B，建立了一条传输链路，则意味着A，B都可以同时发送和接收TCP报文段。
- TCP不支持组播和广播。
- TCP连接提供可靠的字节流服务

TCP报文段格式

TCP报文段是TCP协议的数据交换单元，其格式如下：

- **源/目的端口号：**各占16位，表示发送方和接收方的端口号。
- **顺序号：**32位，表示TCP段中的数据部分第1个字节的编号。
- **确认号：**32位，表示期望接收数据的第1个字节的编号，同时表示对确认号之前数据的确认。
- **数据偏移：**4位，表示TCP段头长度，包括固定和可变部分，单位为字（32位）
- URG：紧急数据标志。当有紧急数据时，该标志为“1”。
- ACK：该标志若为“1”，则表示**确认号**有效；若为“0”，则**确认号**无效。
- PSH：表示要求马上发送数据，不必等到缓冲区满时才发送。
  例如，TELNET协议中，每输入一个字符就必须立即发送。
- RST：该标志用于对本次TCP连接进行复位。通常在TCP连接发生故障时设置本位，以便双方重新同步，并初始化某些连接变量。如接收方收到它不希望接收的报文段，则将RST置“1”。

- **SYN**：用于建立TCP连接。
  - SYN置为“1”且ACK置为“0”，表示请求建立TCP连接
  - SYN置为“1”且ACK置为“1”，表示确认TCP连接。
- **FIN**：用于释放连接。若FIN置为“1”，则表示没有数据要发送了，但仍可以接收数据。
- **窗口大小**：16位，用于TCP流量控制。表示从确认的字节号开始还可以接收多少字节。窗口大小也允许为“0”，表示确认号以前的字节已收到，但暂停接收数据。
- **校验和**：用于对TCP报文段进行校验（类似于IP校验和），但校验范围包括头部和数据部分。计算校验和需包括一个TCP伪头部，这样有助于检测分组是否被错误递交。

TCP伪头

- **紧急指针：** 当URG为“1”时，紧急指针给出TCP段中紧急数据的长度，单位为字节；数据字段的起始位置（**顺序号**）作为紧急数据的开始位置。

- **选项：** 目前TCP只规定一种选项，即最大的TCP段长（MSS），缺省值是556B。TCP实体双方可通过协商确定一个特定的最大TCP段长。

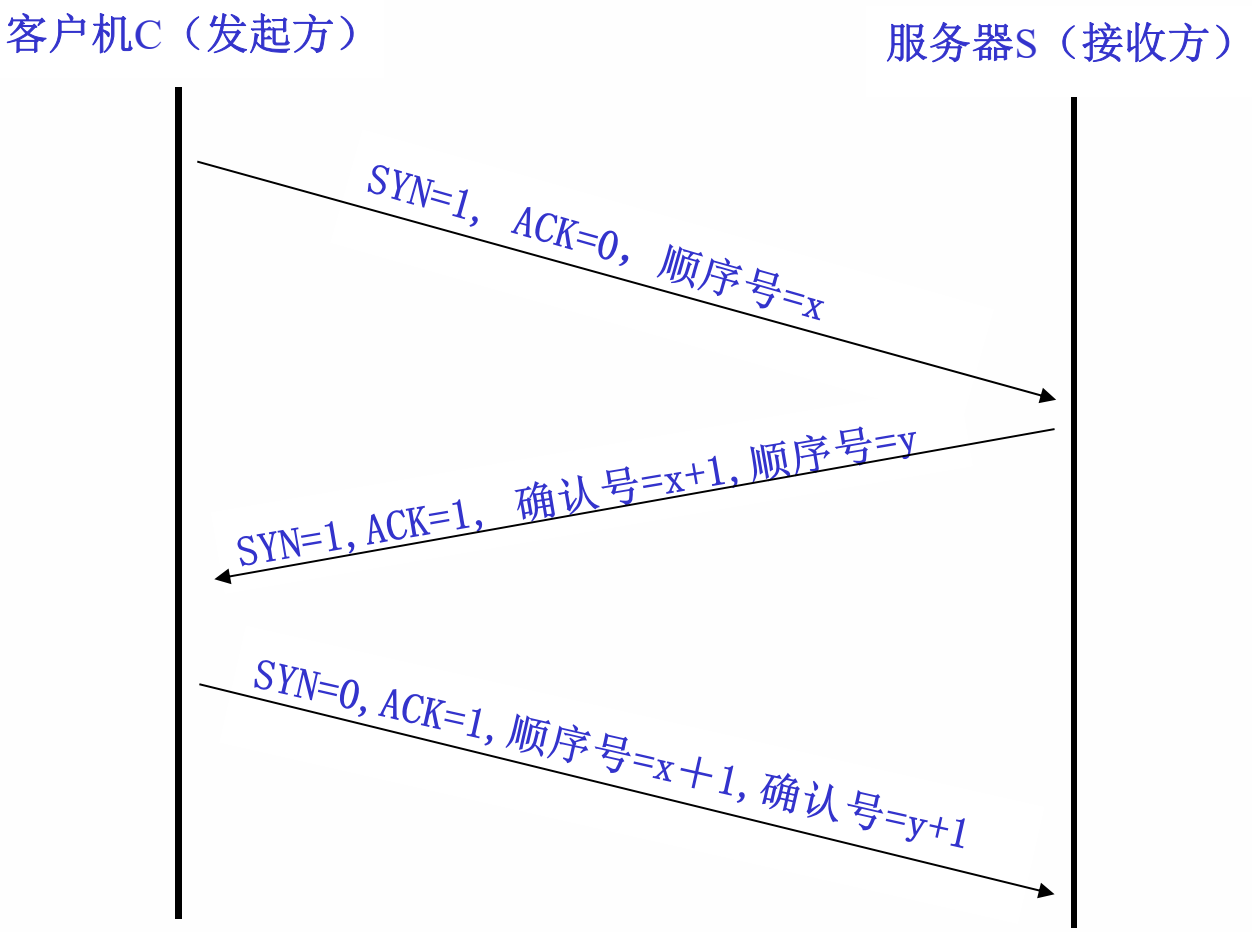
  若MSS选的太小，则由于帧头和报头开销而网络利用率会降低；若MSS选的太大，则由于IP层分段而增加额外开销。一般认为，MSS应尽可能选大些，只要IP层不需要分段就行。

- **填充：** 以0为填充位，确保TCP头部以32位边界结束。

- **数据：** TCP载荷数据，由于IP分组长度限制，TCP最大载荷长度=65535B−20B-20B=65495B。

建立连接-三次握手

- 采用“三次握手”



- 客户端发出连接请求TCP段（SYN＝1，ACK＝0，顺序号=x，其中x为随机数），指明想要连接的服务器端口号（目的端口），设置TCP段的最大段长及其它参数。

- 服务器的TCP实体收到该请求后，检查是否有进程在监听目的端口，
  - 如果没有，则返回拒绝TCP段（RST＝1）的作为应答，拒绝请求。
  - 如果有，则该进程可以接受或拒绝连接请求，

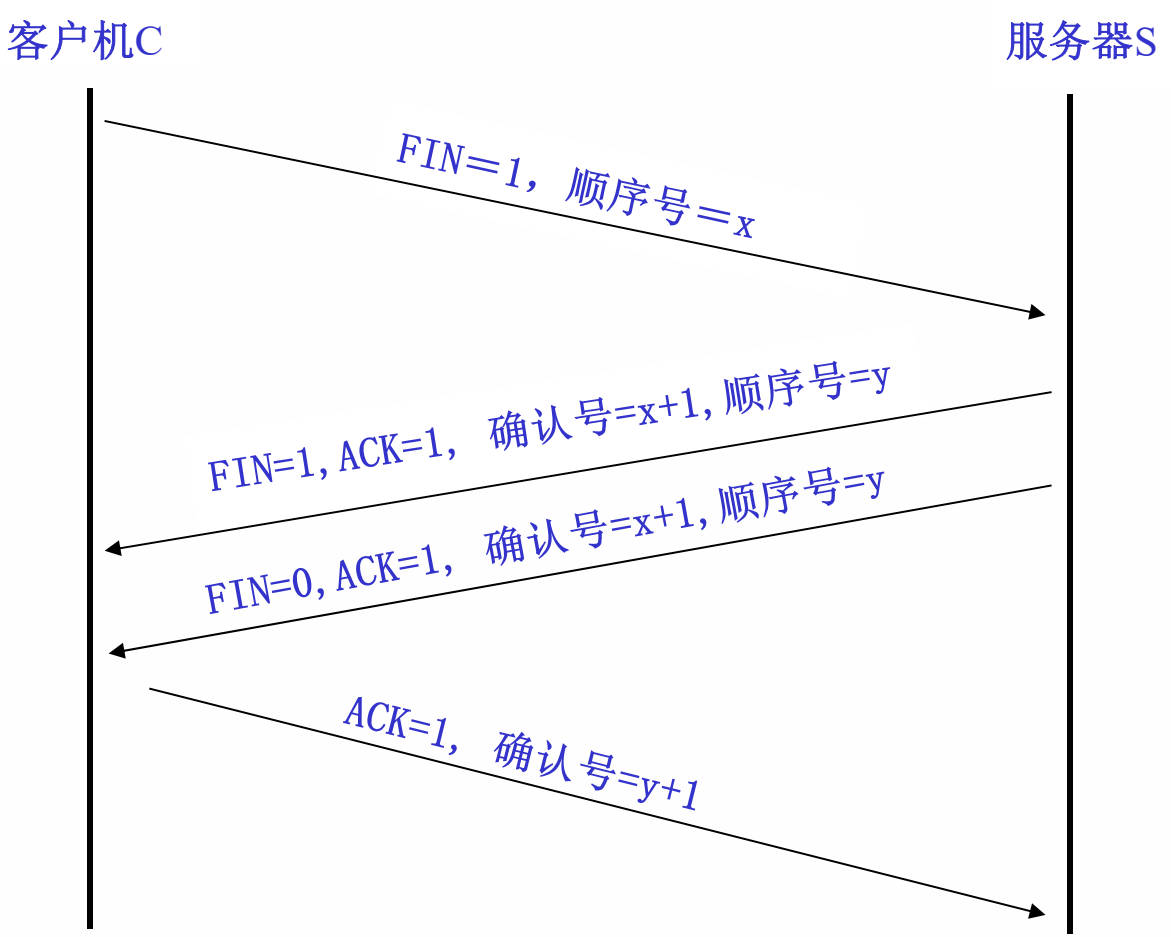
    如果接受，则返回确认TCP段（SYN＝1，ACK＝1，顺序号=y，确认号=x+1,其中y为随机数）。

    如果拒绝则返回拒绝TCP段（RST＝1）。

- 客户端收到确认TCP段后，也发送一个确认TCP段（SYN＝0，ACK＝1），并允许最后一个确认TCP段直接开始发送数据。此时连接建立完毕。

释放连接-三次握手

- TCP连接释放：仍采用“三次握手”



- 主机A发送断连请求TCP段（FIN＝1），向主机B表明“我已无数据要发送，但如果你要发送数据，我仍能接收”。
- 主机B收到断连请求后，

  如果无数据传输，则返回应答TCP段（ FIN＝1，ACK＝1）；如果有数据传输，则返回应答TCP段（ FIN＝0，ACK＝1 ）；
- 主机A返回确认TCP段（ACK＝1）。

6.5 UDP协议

- UDP是非面向连接，不可靠的传输协议。
  - 不需要建立连接。
  - 不支持流量控制和拥塞控制。
  - UDP段在传输过程中可能会丢失、失序和延迟。
  - 支持广播和组播，这对多媒体传输是非常有用的。

- UDP实质上在IP基础上，增加了端口机制，实现了主机进程间的数据传输。
- UDP报文段

- UDP计算“校验和”方法同TCP，也需要包括一个UDP伪头；一旦发现出错，则简单地丢弃，“重传”由应用层完成。

UDP伪头

- UDP不保证可靠传输，既没有重传机制，也没有流量控制和拥塞控制，因此主机不需要维护具有许多参数、复杂的连接状态表，因此减少了开销和发送数据之前的延时，适合很多实时应用，如IP电话、视频会议等。

TCP与UDP的区别

协议	是否连接	传输可靠性	应用场合	速度
TCP	面向连接	可靠	传输大量的数据	慢
UDP	非连接	不可靠	少量数据	快

本章小结

- 内容

  主要介绍传输层协议及其相关技术，包括传输服务质量（QoS）、传输地址、传输连接的建立和拆除、流量控制、拥塞控制 、Internet的主要传输层协议实例TCP、UDP等。

- 重点

  TCP协议原理， TCP报文段格式及每个字段的含义，传输连接的建立和拆除过程。