网络技术基础

高智刚

M.P. & WeChat：13572460159

E-mail: gaozhigang@nwpu.edu.cn

第六章：传输层

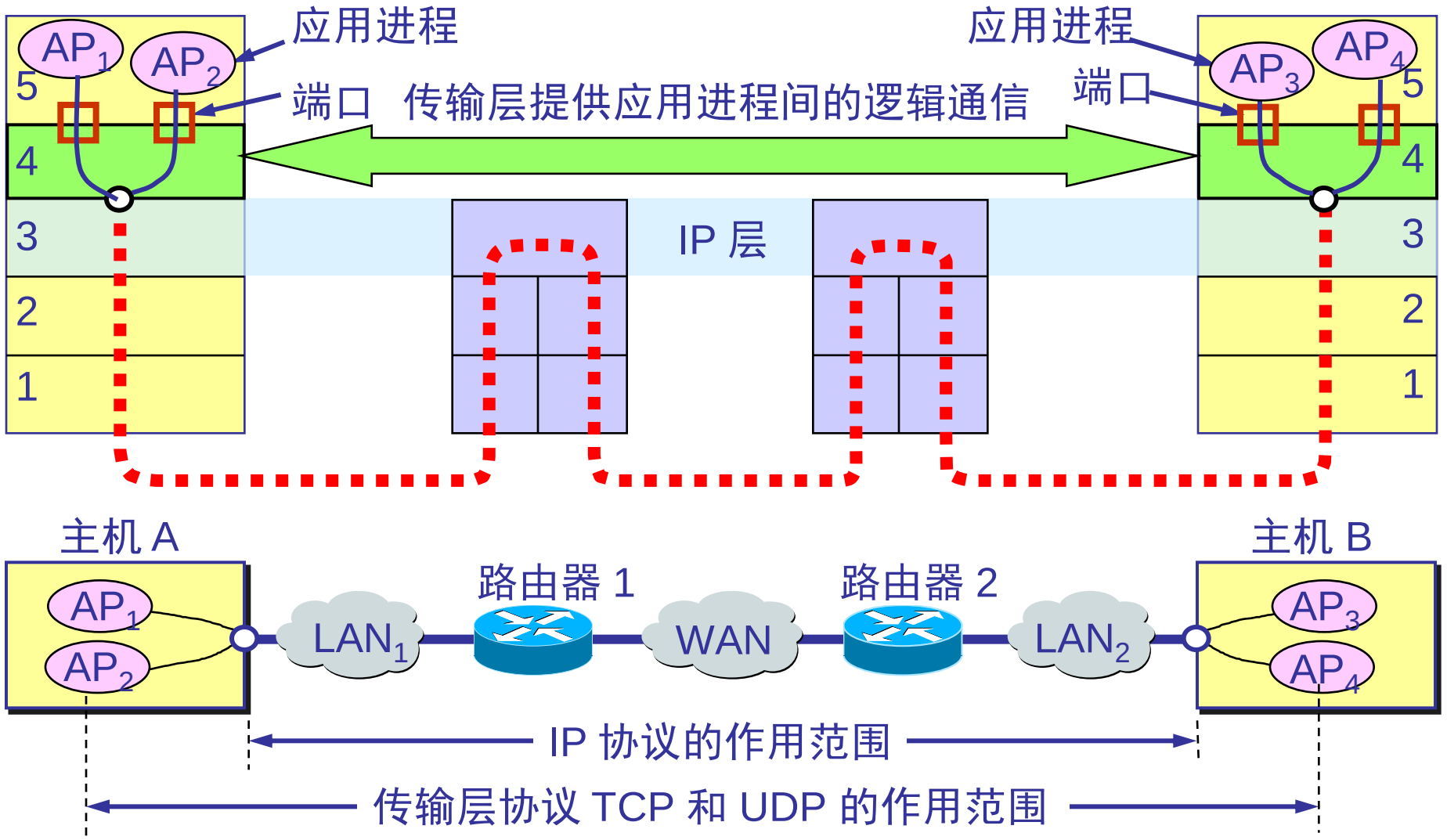
本章主要内容

- 传输层概述
- 传输层端口
- 用户数据报协议UDP
- 传输控制协议TCP

6.1 传输层协议概述

- 进程之间的通信
  - 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
  - 当网络的边缘部分中的两个主机使用网络的核心部分的功能进行端到端的通信时，只有位于网络边缘部分的主机的协议栈才有运输层，而网络核心部分中的路由器在转发分组时都只用到下三层的功能

传输层为相互通信的应用进程提供了逻辑通信

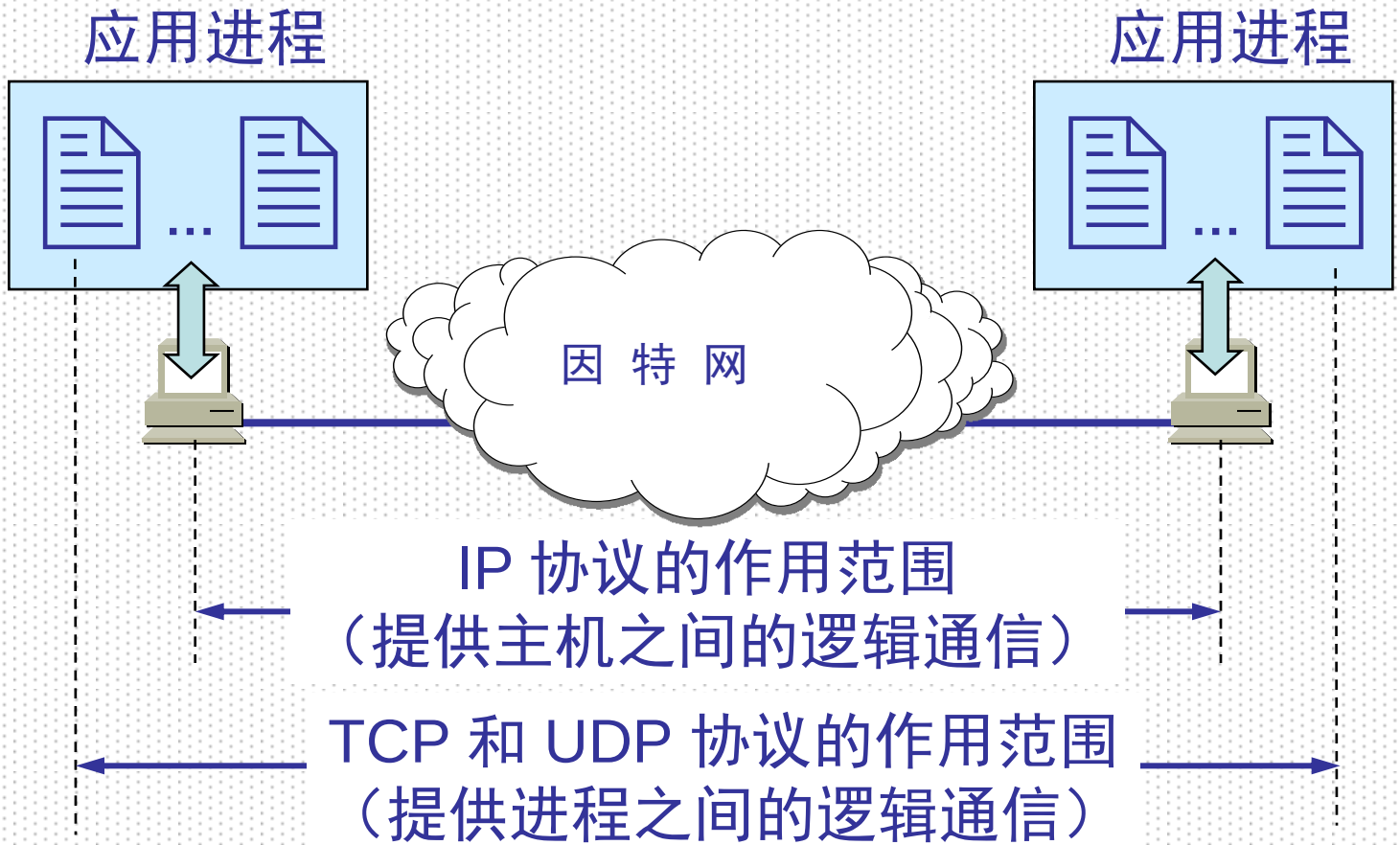


6.1 传输层协议概述

- 应用进程之间的通信
  - 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
  - 传输层的一个很重要的功能就是复用和分用。应用层不同进程的报文通过不同的端口向下交到传输层，再往下就共用网络层提供的服务（复用），交付时由传输层交付给应用层的不同进程（分用）
  - “传输层提供应用进程间的逻辑通信”。“逻辑通信”的意思是：传输层之间的通信好像是沿水平方向传送数据。但事实上这两个运输层之间并没有一条水平方向的物理连接

6.1 传输层协议概述

- 传输层协议和网络层协议的主要区别

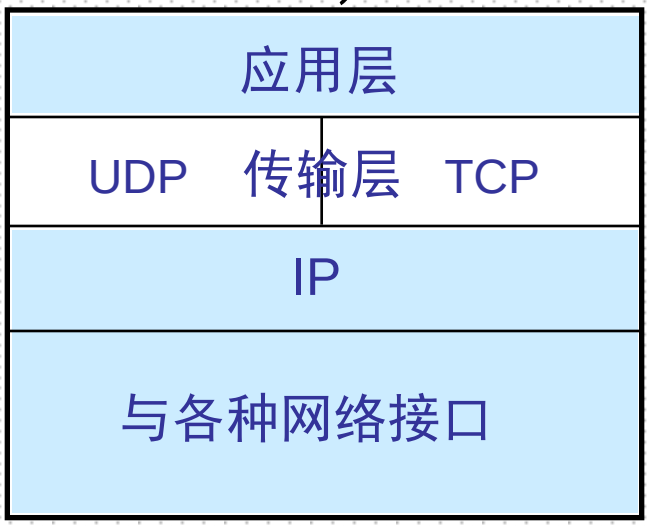


6.1 传输层协议概述

- 传输层的主要功能
  - 传输层为应用进程之间提供端到端的逻辑通信（但网络层是为主机之间提供逻辑通信）
  - 传输层还要对收到的报文进行差错检测
  - 提供两种不同的传输协议，即面向连接的TCP和无连接的UDP

6.1 传输层协议概述

- 传输层有两个不同协议
  - 用户数据报协议UDP (User Datagram Protocol)
  - 传输控制协议TCP (Transmission Control Protocol)
- 两个对等运输实体在通信时传送的数据单位叫作传输协议数据单元TPDU (Transport Protocol Data Unit)
  - TCP的传输数据协议单元是TCP报文段(segment)
  - UDP的传输数据协议单元是UDP用户数据报



6.1 传输层协议概述

- 两种不同的传输协议
  - 传输层向高层用户屏蔽了下面网络核心的细节（如网络拓扑、所采用的路由选择协议等），使应用进程看见的像是两个传输层实体之间有一条端到端的逻辑通信信道
  - 当传输层采用面向连接的TCP协议时，尽管下面的网络是不可靠的（只提供尽最大努力服务），但这种逻辑通信信道就相当于一条全双工的可靠信道
  - 当传输层采用无连接的UDP协议时，这种逻辑通信信道是一条不可靠信道

6.1 传输层协议概述

- 两种不同的传输协议
  - 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
  - TCP则提供面向连接的服务。TCP不提供广播或多播服务。由于TCP要提供可靠的、面向连接的运输服务，因此不可避免地增加了许多的开销。这不仅使协议数据单元的首部增大很多，还要占用许多的处理机资源

6.1 传输层协议概述

- 两种不同的传输协议
  - 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数据报要经过互连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
  - TCP报文段是在传输层抽象的端到端逻辑信道中传送，这种信道是可靠的全双工信道。但这样的信道却不知道究竟经过了哪些路由器，而这些路由器也根本不知道上面的传输层是否建立了TCP连接

6.1 传输层协议概述

- **传输层的端口**
  - 运行在计算机中的进程是用进程标识符来标志的
  - 运行在应用层的各种应用进程却不应当让计算机操作系统指派它的进程标识符。这是因为在因特网上使用的计算机的操作系统种类很多，而不同的操作系统又使用不同格式的进程标识符
  - 为了使运行不同操作系统的计算机的应用进程能够互相通信，就必须用统一的方法对TCP/IP体系的应用进程进行标志

6.1 传输层协议概述

- 需要解决的问题
  - 由于进程的创建和撤销都是动态的，发送方几乎无法识别其他机器上的进程
  - 有时我们会改换接收报文的进程，但并不需要通知所有发送方
  - 我们往往需要利用目的主机提供的功能来识别终点，而不需要知道实现这个功能的进程

6.1 传输层协议概述

- 协议端口号
  - 解决这个问题的方法就是在运输层使用协议端口号(protocol port number)，或通常简称为端口(port)
  - 虽然通信的终点是应用进程，但我们可以把端口想象是通信的终点，因为我们只要把要传送的报文交到目的主机的某一个合适的目的端口，剩下的工作（即最后交付目的进程）就由TCP来完成

6.1 传输层协议概述

- 软件端口与硬件端口
  - 在协议栈层间的抽象的协议端口是软件端口
  - 路由器或交换机上的端口是硬件端口
  - 硬件端口是不同硬件设备进行交互的接口，而软件端口是应用层的各种协议进程与运输实体进行层间交互的一种地址

6.1 传输层协议概述

- 传输层的端口
  - 端口用一个16位的端口号（Port Number）进行标志，故可提供65536个端口
  - 端口号只具有本地意义，即端口号只是为了标志本计算机应用层中的各进程。在因特网中不同计算机的相同端口号是没有联系的

6.1 传输层协议概述

- 三类端口
  - 保留端口(周知端口)，以全局方式统一分配并公之于众，每一种标准服务器都分配有一个，数值一般为0~1023
  - 登记端口号，数值为1024~49151，为没有保留端口号的应用程序使用的。使用这个范围的端口号必须在IANA登记，以防止重复
  - 客户端口号或短暂端口号，数值为49152~65535，留给客户进程选择暂时使用。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

6.2 用户数据报协议UDP

- UDP概述
  - UDP只在IP的数据报服务之上增加了很少一点的功能，即端口的功能和差错检测的功能
  - 虽然UDP用户数据报只能提供不可靠的交付，但UDP在某些方面有其特殊的优点

6.2 用户数据报协议UDP

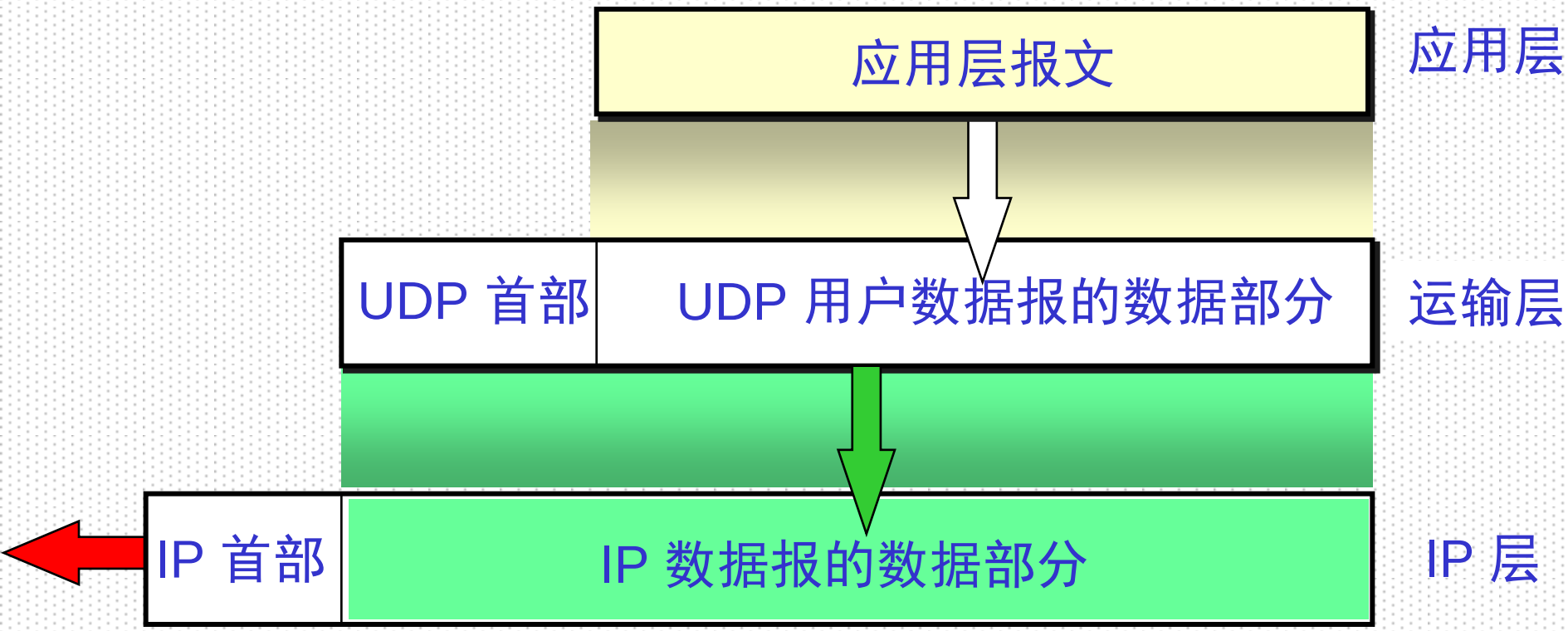
西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- UDP的主要特点
  - UDP是无连接的，即发送数据之前不需要建立连接
  - UDP使用尽最大努力交付，即不保证可靠交付，同时也不使用拥塞控制
  - UDP是面向报文的。UDP没有拥塞控制，很适合多媒体通信的要求
  - UDP支持一对一、一对多、多对一和多对多的交互通信
  - UDP的首部开销小，只有8个字节

- 面向报文的UDP
  - 发送方UDP对应用程序交下来的报文，在添加首部后就向下交付IP层。UDP对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界
  - 应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文
  - 接收方UDP对IP层交上来的UDP用户数据报，在去除首部后就原封不动地交付上层的应用进程，一次交付一个完整的报文
  - 应用程序必须选择合适大小的报文

6.2 用户数据报协议UDP

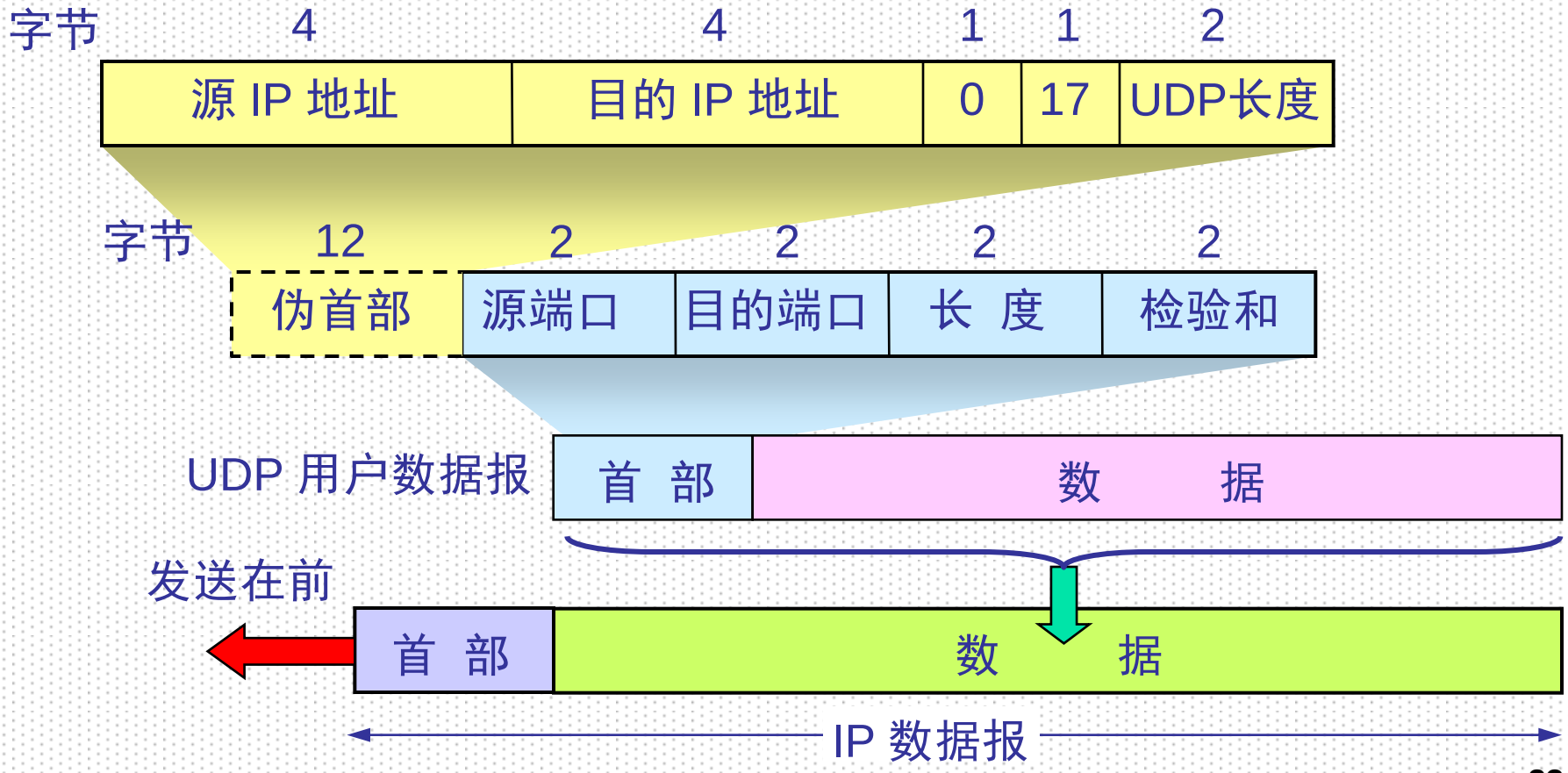
- UDP是面向报文的



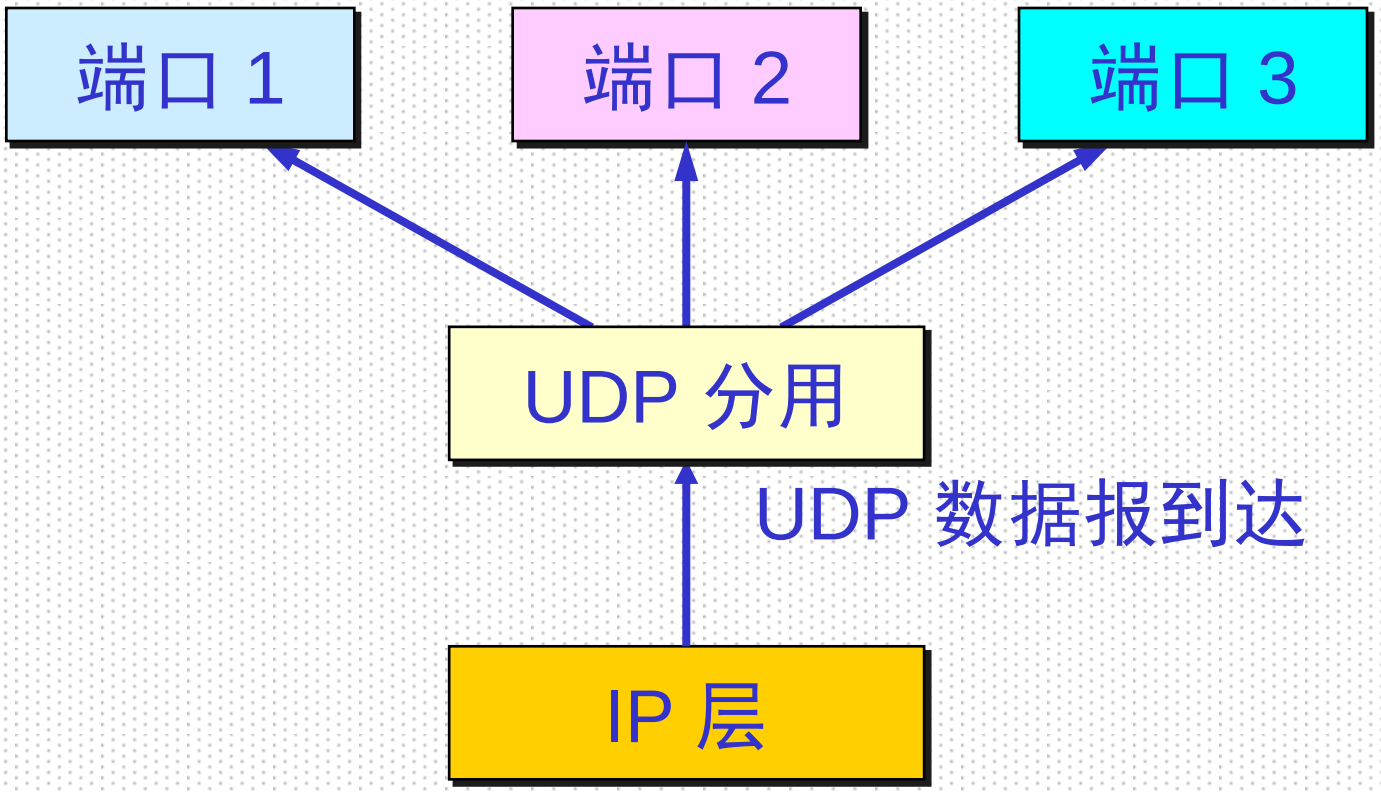
6.2 用户数据报协议UDP

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

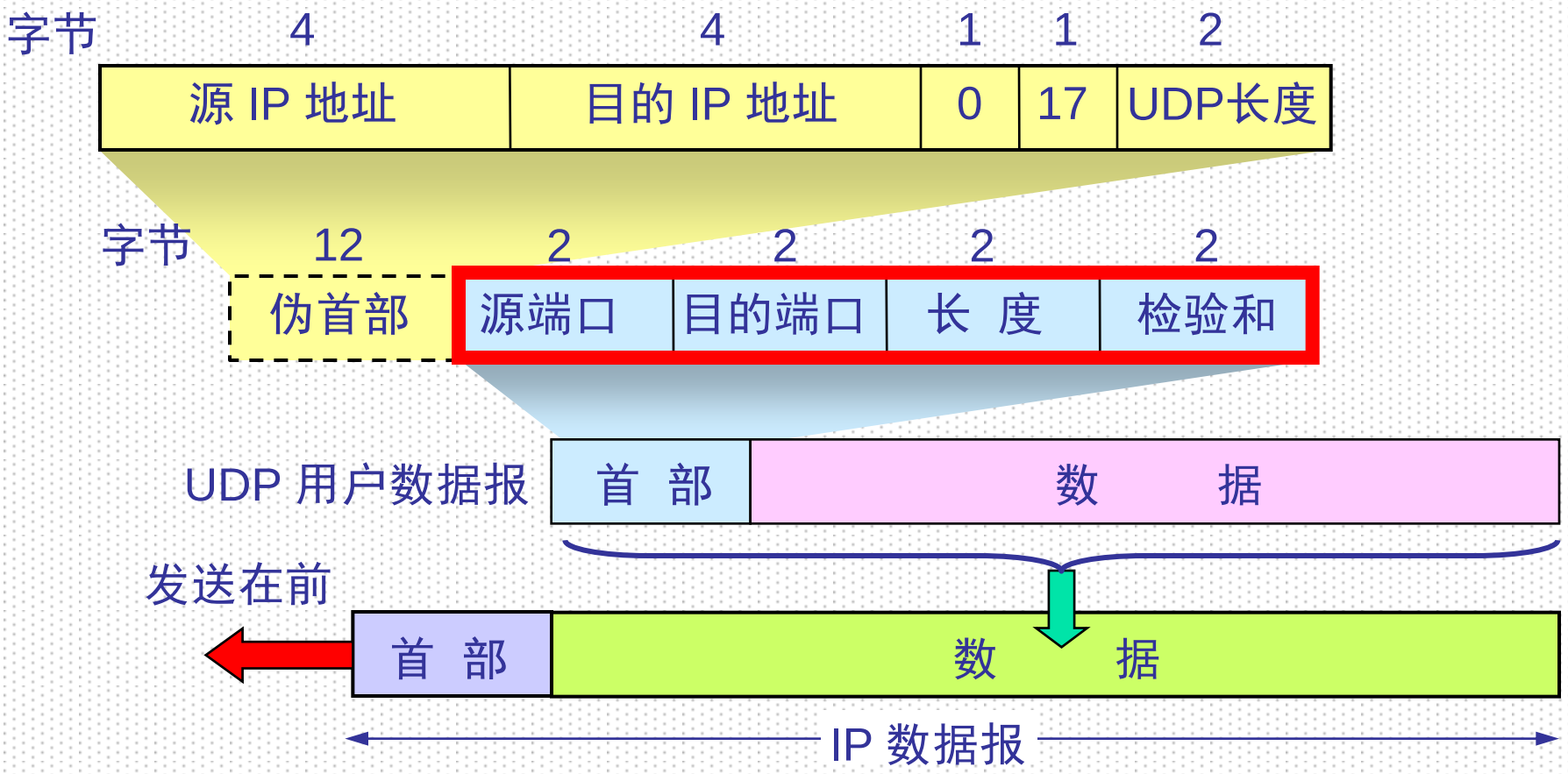
- UDP的首部格式



- UDP基于端口的分用

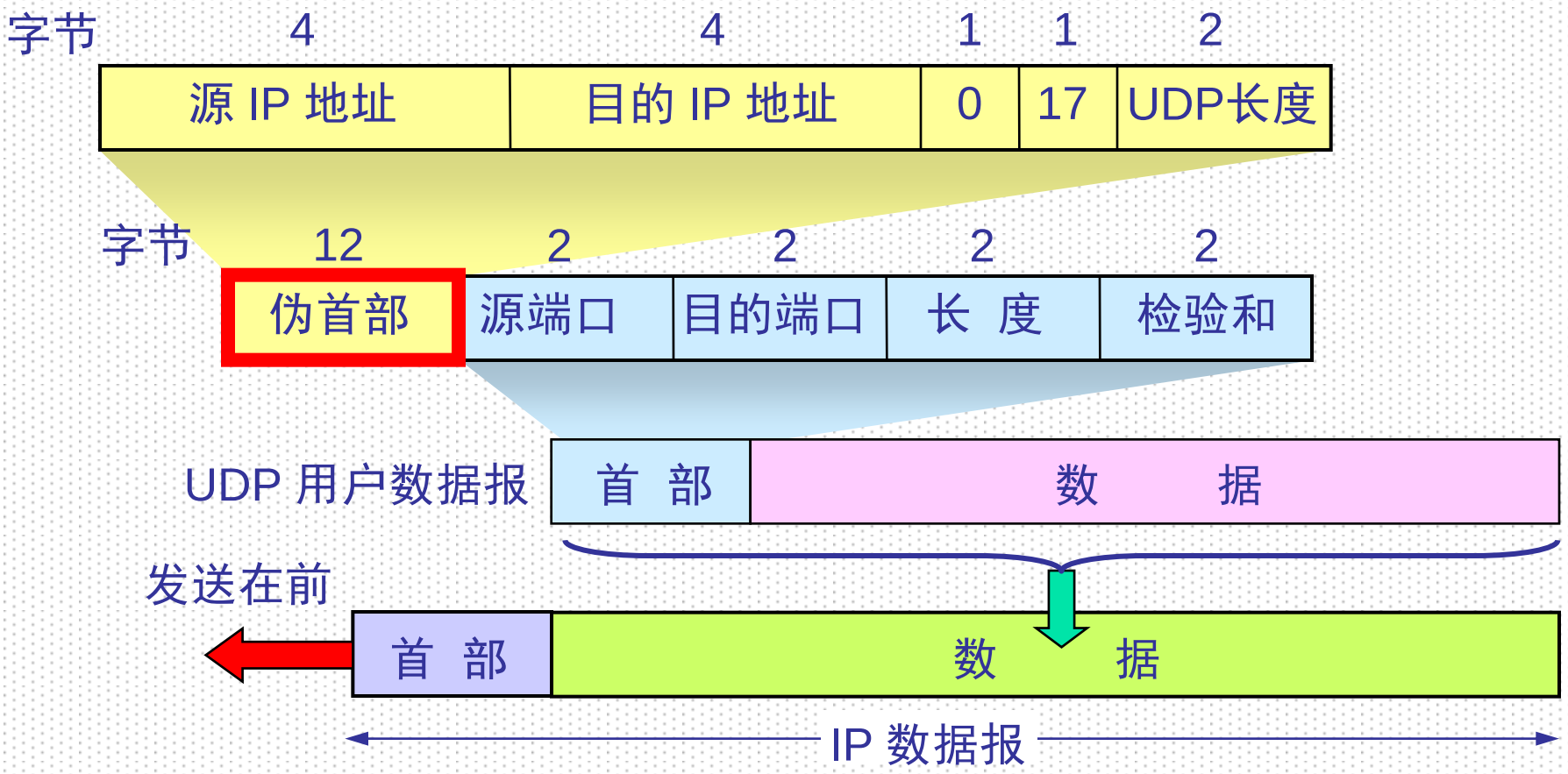


6.2 用户数据报协议UDP



用户数据报UDP有两个字段：数据字段和首部字段。首部字段有8个字节，由4 字段组成，每个字段都是两个字节。

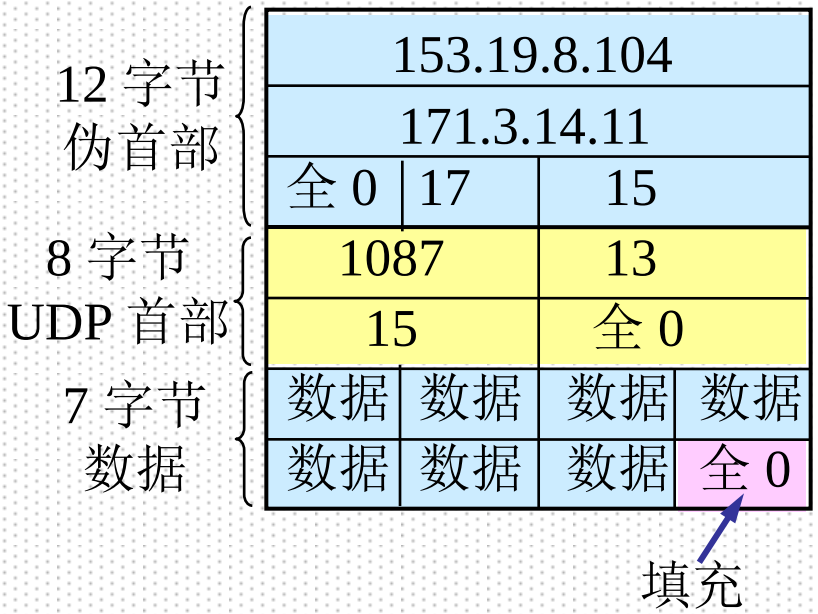
6.2 用户数据报协议UDP



在计算检验和时，临时把“伪首部”和UDP用户数据报连接在一起。伪首部仅仅是为了计算检验和。

6.2 用户数据报协议UDP

- 计算UDP检验和的例子



10011001 00010011 → 153.19
00001000 01101000 → 8.104
10101011 00000011 → 171.3
00001110 00001011 → 14.11
00000000 00010001 → 0 和 17
00000000 00001111 → 15
00000100 00111111 → 1087
00000000 00001101 → 13
00000000 00001111 → 15
00000000 00000000 → 0（检验和）
01010100 01000101 → 数据
01010011 01010100 → 数据
01001001 01001110 → 数据
01000111 00000000 → 数据和 0（填充）

按二进制反码运算求和 10010110 11101101 → 求和得出的结果

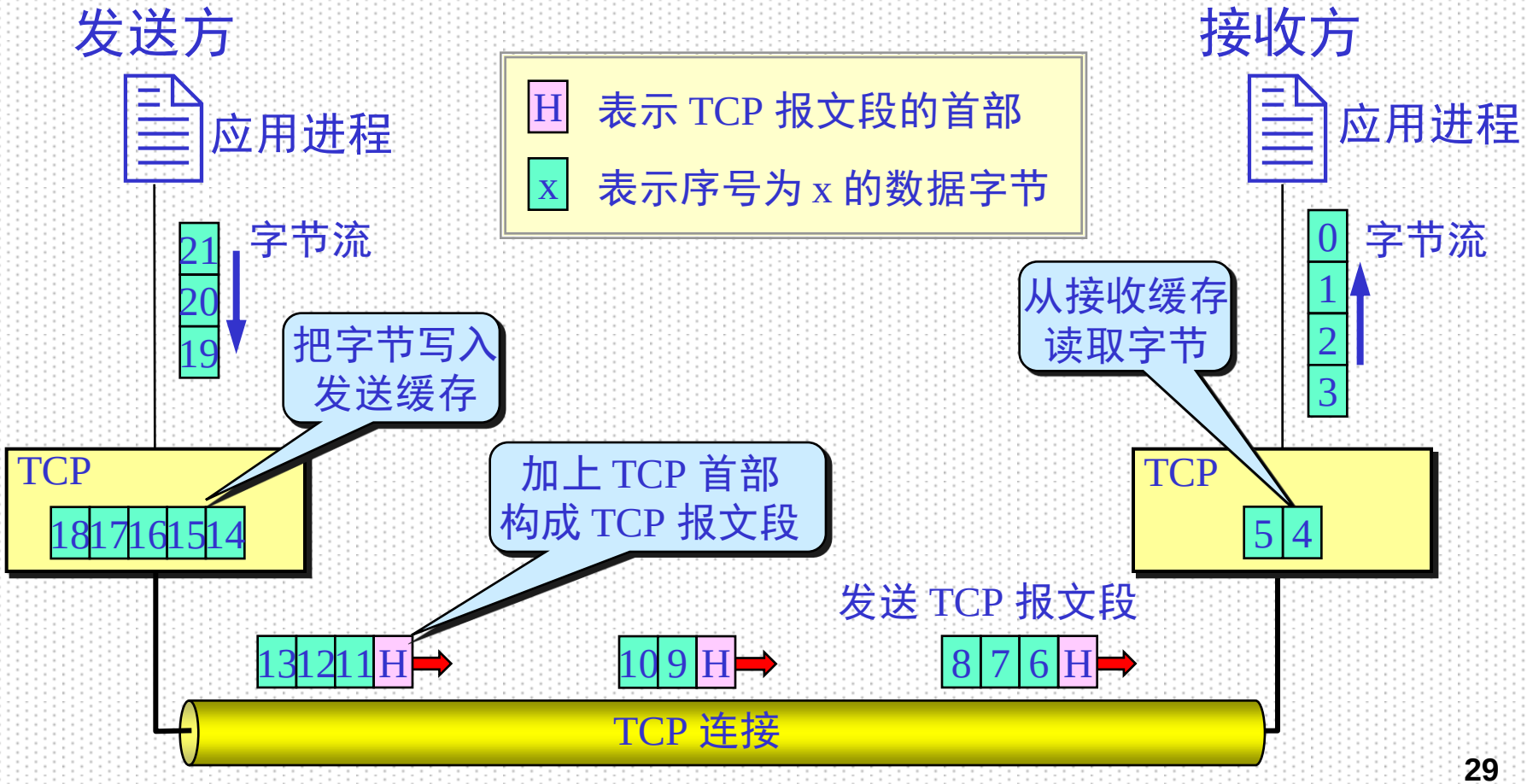
将得出的结果求反码 01101001 00010010 → 检验和

6.3 传输控制协议TCP概述

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- TCP最主要的特点
  - TCP是面向连接的传输层协议
  - 每一条TCP连接只能有两个端点(endpoint)，每一条TCP连接只能是点对点的（一对一）
  - TCP提供可靠交付的服务
  - TCP提供全双工通信
  - 面向字节流

- TCP面向流的概念


发送方
应用进程
字节流
21
20
19
把字节写入
发送缓存
TCP
18 17 16 15 14
H 表示 TCP 报文段的首部
x 表示序号为 x 的数据字节
加上 TCP 首部
构成 TCP 报文段
13 12 11 H
10 9 H
发送 TCP 报文段
8 7 6 H
TCP 连接
接收方
应用进程
字节流
0
1
2
3
从接收缓存
读取字节
TCP
5 4

6.3 传输控制协议TCP概述

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 应当注意
  - TCP连接是一条虚连接而不是一条真正的物理连接
  - TCP对应用进程一次把多长的报文发送到TCP的缓存中是不关心的
  - TCP根据对方给出的窗口值和当前网络拥塞的程度来决定一个报文段应包含多少个字节（UDP发送的报文长度是应用进程给出的）
  - TCP可把太长的数据块划分短一些再传送，也可以等待积累有足够多的字节后再构成报文段发送出去

6.3 传输控制协议TCP概述

- TCP的连接
  - TCP把连接作为最基本的抽象，每一条 TCP连接有两个端点
  - TCP连接的端点不是主机，不是主机的IP地址，不是应用进程，也不是传输层的协议端口，TCP连接的端点叫做套接字（socket）或插口
  - 端口号拼接到（contatenated with）IP地址即构成了套接字

6.3 传输控制协议TCP概述

- **套接字**(socket)

套接字 socket = (IP地址: 端口号)

每一条**TCP**连接唯一地被通信两端的两个端点（即两个套接字）所确定。即：

TCP 连接 ::= {socket1, socket2}
= {(IP1: port1), (IP2: port2)}

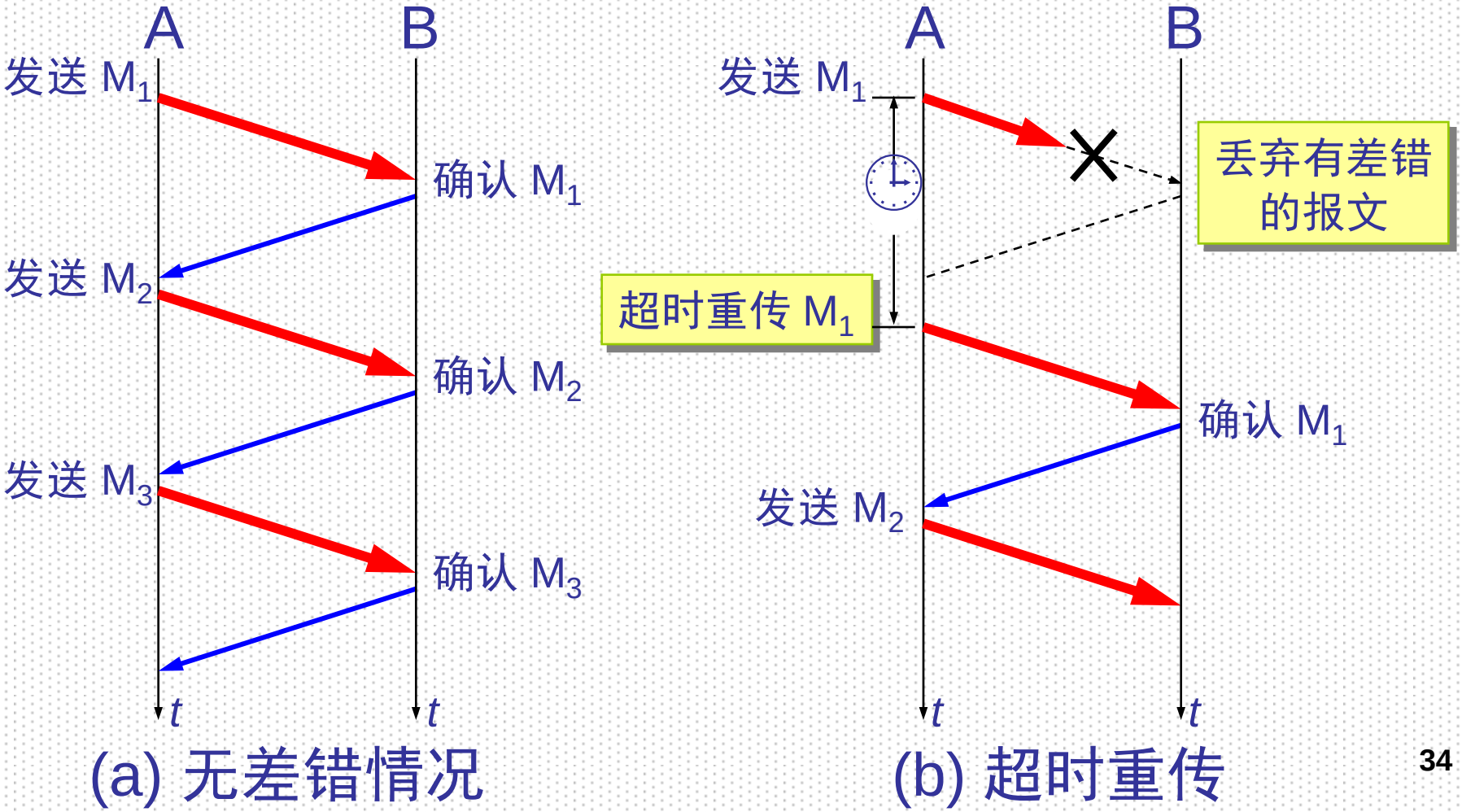
6.3 传输控制协议TCP概述

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 同一个名词socket有多种不同的意思
  - 应用编程接口API称为socket API，简称为socket
  - socket API 中使用的一个函数名也叫作socket
  - 调用socket函数的端点称为socket
  - 调用socket函数时其返回值称为socket描述符，可简称为socket
  - 在操作系统内核中连网协议的Berkeley实现，称为socket实现

6.4 可靠传输的工作原理

- 停止等待协议

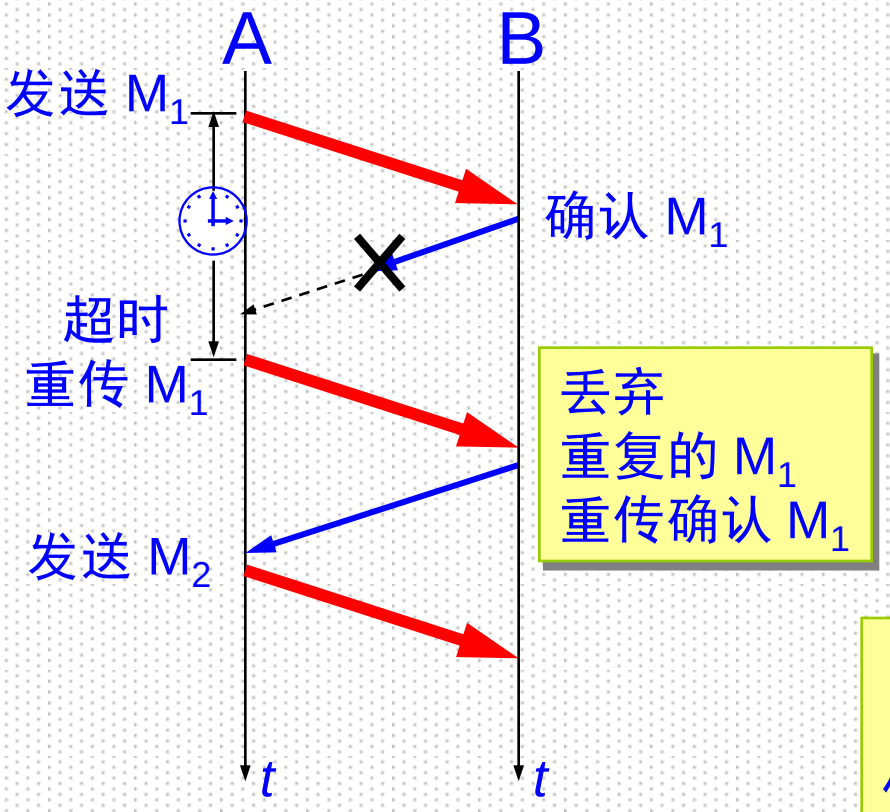


6.4 可靠传输的工作原理

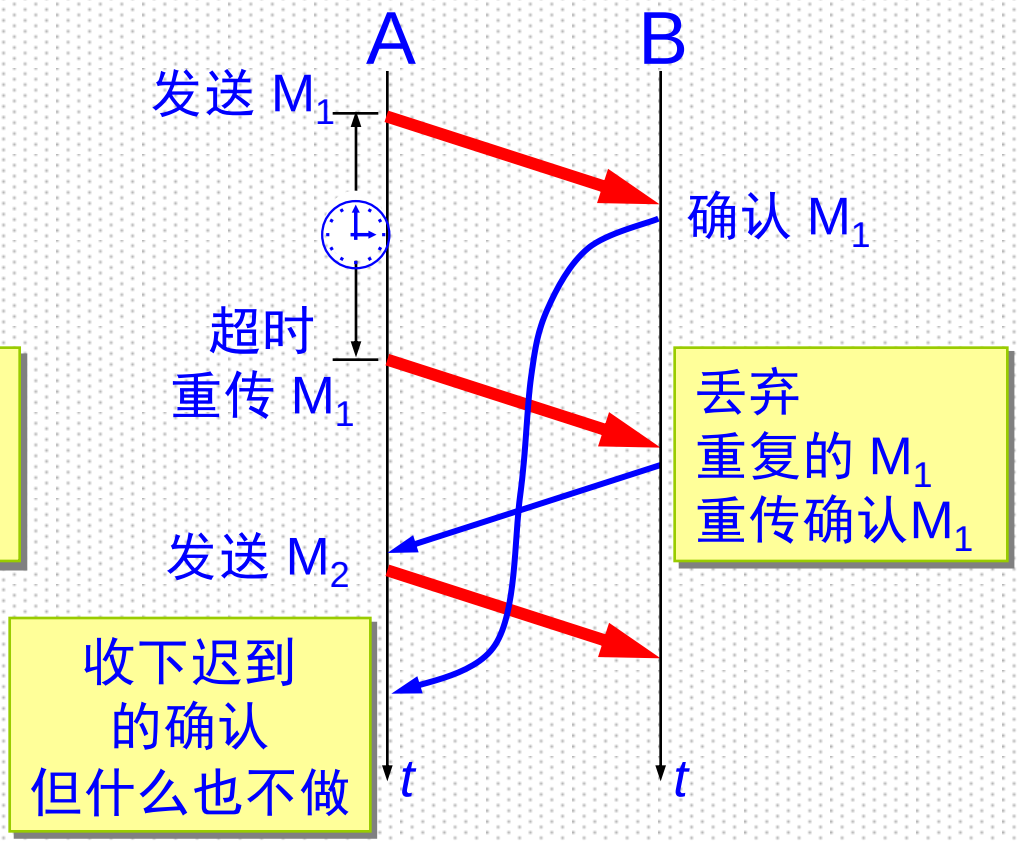
- 注意事项
  - 在发送完一个分组后，必须暂时保留已发送的分组的副本
  - 分组和确认分组都必须进行编号
  - 超时计时器的重传时间应当比数据在分组传输的平均往返时间更长一些

6.4 可靠传输的工作原理

- 确认丢失和确认迟到



(a) 确认丢失



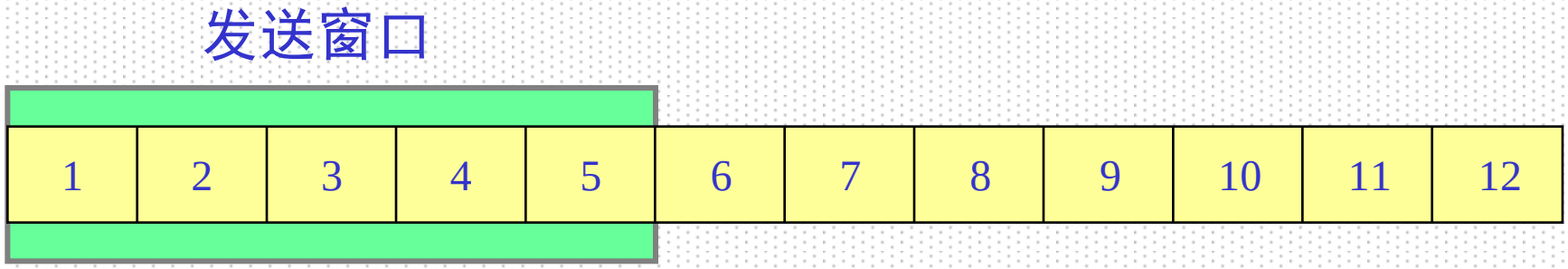
(b) 确认迟到

6.4 可靠传输的工作原理

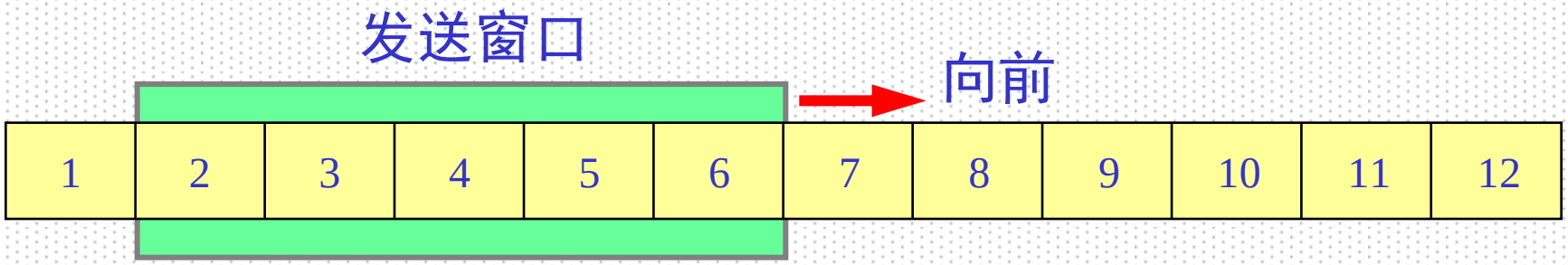
- 可靠通信的实现
  - 使用上述的确认和重传机制，我们就可以在不可靠的传输网络上实现可靠的通信
  - 这种可靠传输协议常称为自动重传请求ARQ (Automatic Repeat reQuest)
  - ARQ表明重传的请求是自动进行的，接收方不需要请求发送方重传某个出错的分组

6.4 可靠传输的工作原理

- 连续ARQ协议



(a) 发送方维持发送窗口（发送窗口是 5）



(b) 收到一个确认后发送窗口向前滑动

6.4 可靠传输的工作原理

- 累积确认
  - 接收方一般采用累积确认的方式。即不必对收到的分组逐个发送确认，而是对按序到达的最后一个分组发送确认，这样就表示：到这个分组为止的所有分组都已正确收到了
  - 优点：容易实现，即使确认丢失也不必重传
  - 缺点：不能向发送方反映出接收方已经正确收到的所有分组的信息

6.4 可靠传输的工作原理

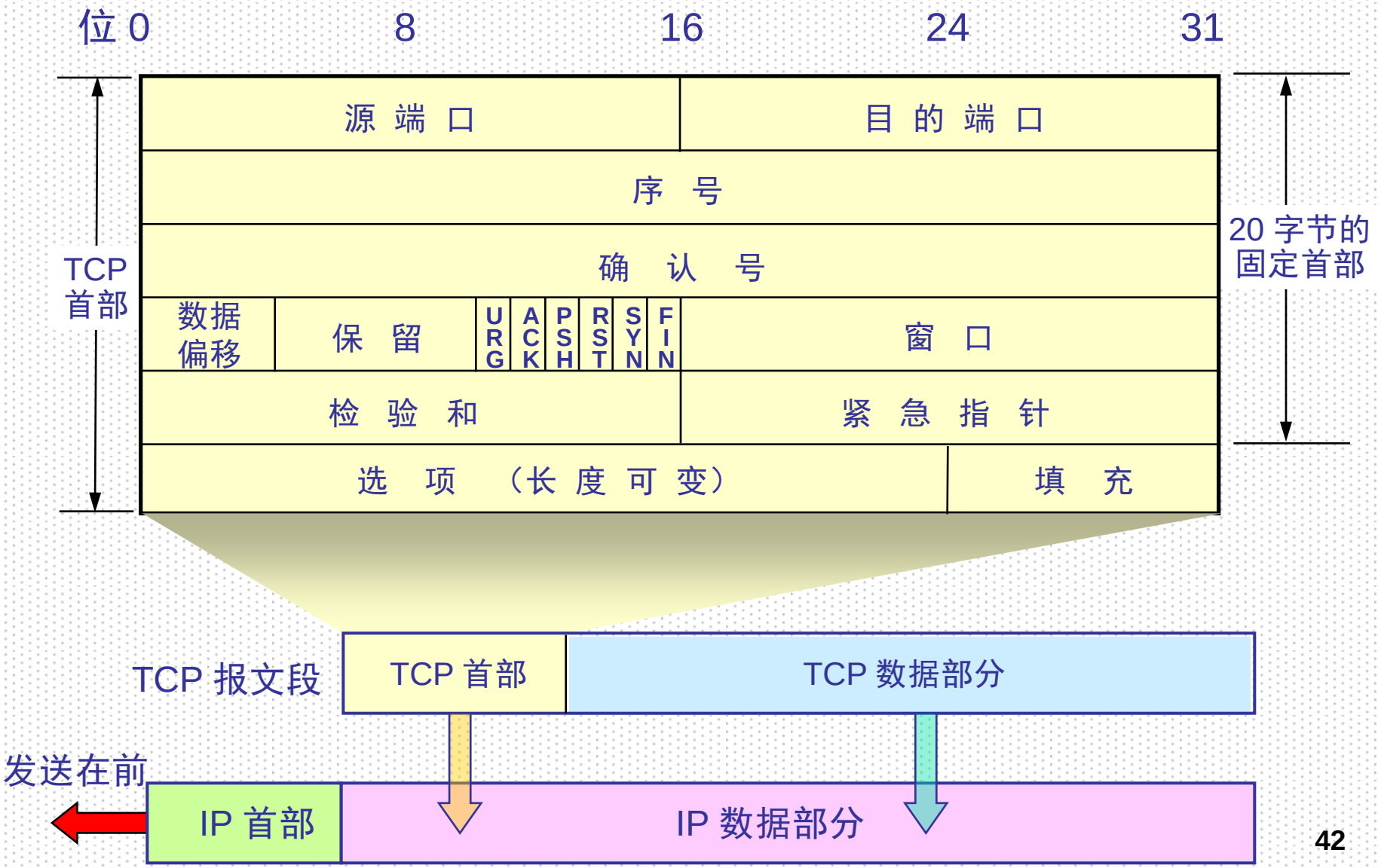
- Go-back-N（回退N）
  - 如果发送方发送了前5个分组，而中间的第3个分组丢失了。这时接收方只能对前两个分组发出确认。发送方无法知道后面三个分组的下落，而只好把后面的三个分组都再重传一次
  - 这就叫做Go-back-N（回退N），表示需要再退回来重传已发送过的N个分组
  - 当通信线路质量不好时，连续ARQ协议会带来负面影响

6.4 可靠传输的工作原理

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- TCP可靠通信的具体实现
  - TCP连接的每一端都必须设有两个窗口：一个发送窗口和一个接收窗口
  - TCP的可靠传输机制用字节的序号进行控制。TCP所有的确认都是基于序号而不是基于报文段
  - TCP两端的四个窗口经常处于动态变化之中
  - TCP连接的往返时间RTT也不是固定不变的。需要使用特定的算法估算较为合理的重传时间

6.5 TCP报文段的首部格式



6.5 TCP报文段的首部格式

源端口和目的端口字段——各占2字节。端口是传输层与应用层的服务接口。传输层的复用和分用功能都要通过端口才能实现。

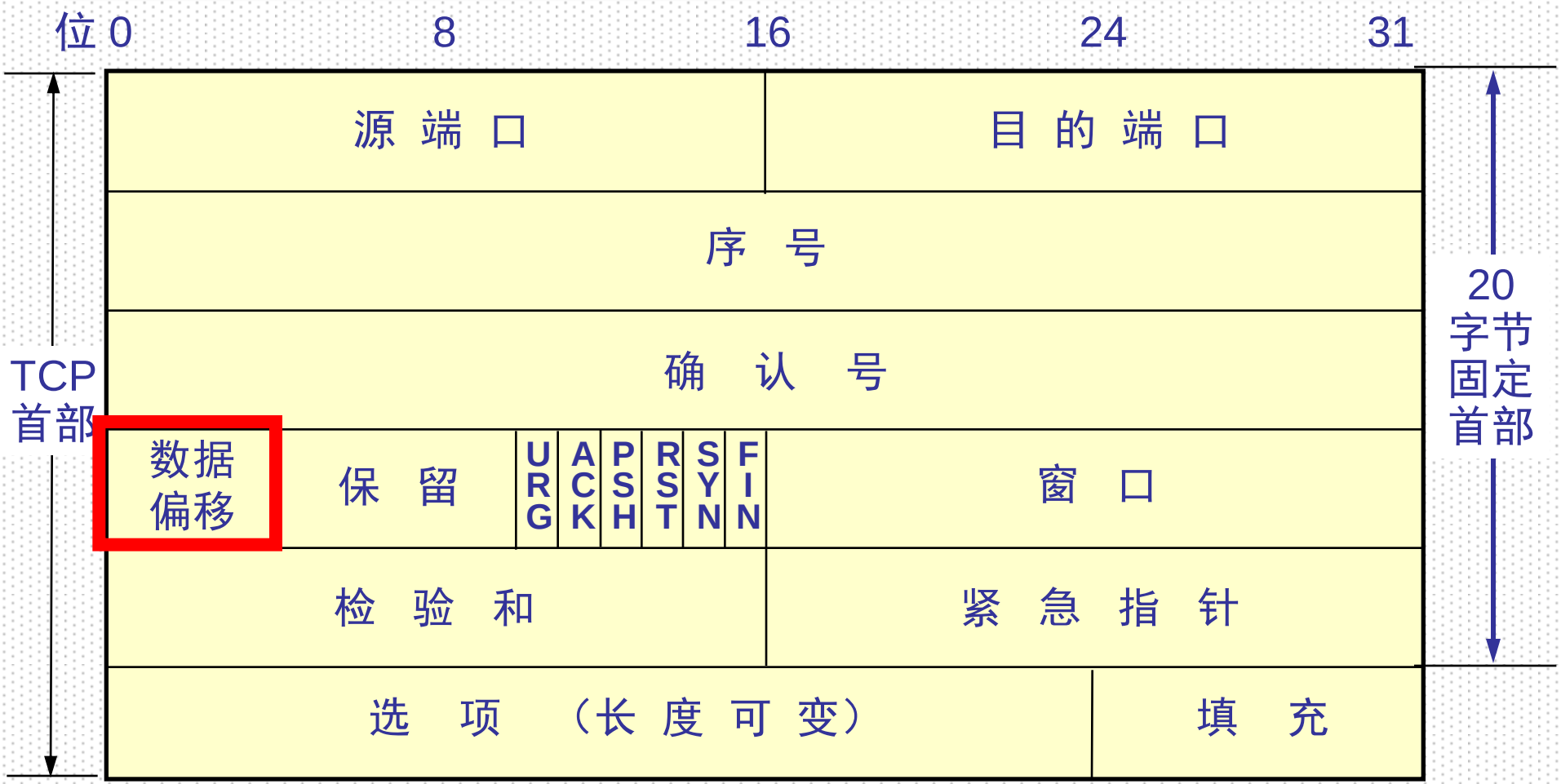
6.5 TCP报文段的首部格式

序号字段——占4字节。TCP连接中传送的数据流中的每一个字节都编上一个序号。序号字段的值则指的是本报文段所发送的数据的第一个字节的序号。

6.5 TCP报文段的首部格式

确认号字段——占4字节，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。

6.5 TCP报文段的首部格式



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位字（以4字节为计算单位）。

6.5 TCP报文段的首部格式

保留字段——占 6 位，保留为今后使用，目前应置为0。

6.5 TCP报文段的首部格式

紧急URG——当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）。

6.5 TCP报文段的首部格式

确认ACK——只有当ACK=1时确认号字段才有效。
当ACK=0时，确认号无效。

6.5 TCP报文段的首部格式

推送PSH(PuSH)——接收TCP收到PSH=1的报文段，就尽快地交付接收应用进程，而不再等到整个缓存都填满了后再向上交付。

6.5 TCP报文段的首部格式

复位RST(ReSeT)——当RST=1时，表明TCP连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。

6.5 TCP报文段的首部格式

同步SYN——同步SYN=1表示这是一个连接请求或连接接受报文。

6.5 TCP报文段的首部格式

终止FIN(FINis)——用来释放一个连接。FIN=1表明此报文段的发送端的数据已发送完毕，并要求释放运输连接。

6.5 TCP报文段的首部格式

窗口字段——占2字节，用来让对方设置发送窗口的依据，单位为字节。

6.5 TCP报文段的首部格式

检验和——占2字节。检验和字段检验的范围包括首部和数据这两部分。在计算检验和时，要在TCP报文段的前面加上12字节的伪首部。

6.5 TCP报文段的首部格式

紧急指针字段——占16位，指出在本报文段中紧急数据共有多少个字节（紧急数据放在本报文段数据的最前面）。

6.5 TCP报文段的首部格式

MSS(Maximum Segment Size)
是TCP报文段中的数据字段的最大长度。
数据字段加上TCP首部
才等于整个的TCP报文段。

选项字段——长度可变。TCP最初只规定了一种选项，即最大报文段长度MSS。MSS告诉对方TCP：“我的缓存所能接收的报文段的数据字段的最大长度是MSS个字节。”

6.5 TCP报文段的首部格式

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 其他选项
  - 窗口扩大选项——占3字节，其中一个字节表示移位值S。新的窗口值等于TCP首部中的窗口位数增大到(16+S)，相当于把窗口值向左移动S位后获得实际的窗口大小
  - 时间戳选项——占10字节，其中最主要的字段时间戳值字段（4字节）和时间戳回送回答字段（4字节）
  - 负确认选项——用于选择重传ARQ

6.5 TCP报文段的首部格式

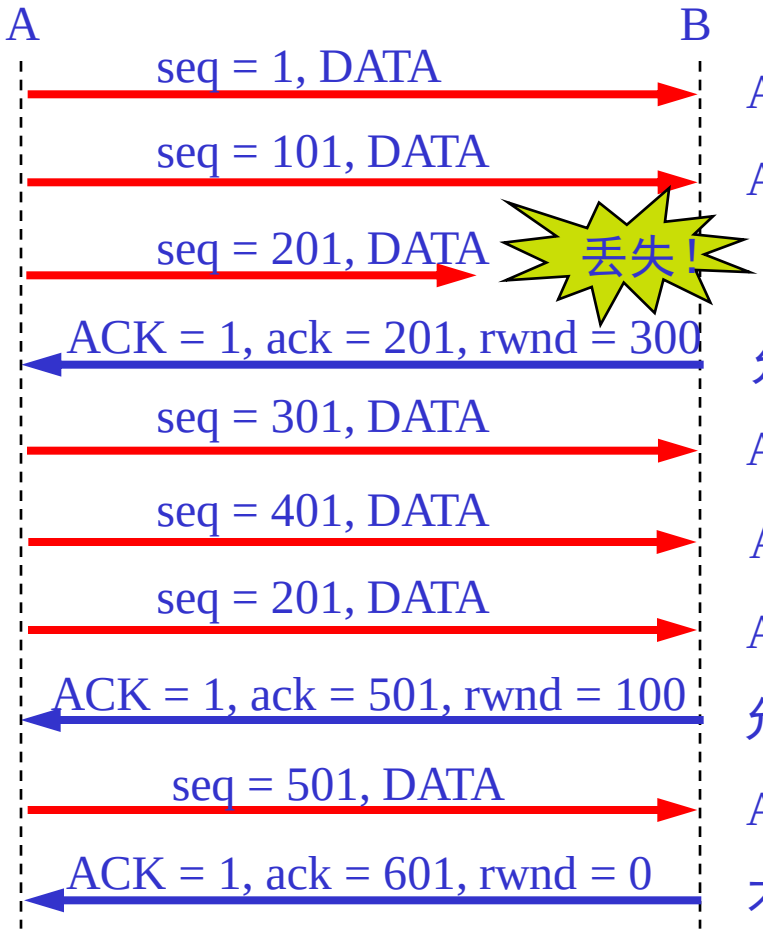
填充字段——使整个首部长度是4字节的整数倍。

6.6 TCP的流量控制

- 利用滑动窗口实现流量控制
  - 一般说来，我们总是希望数据传输得更快一些。但如果发送方把数据发送得过快，接收方就可能来不及接收，这就会造成数据的丢失
  - 流量控制(flow control)就是让发送方的发送速率不要太快，既要让接收方来得及接收，也不要使网络发生拥塞
  - 利用滑动窗口机制可以很方便地在TCP连接上实现流量控制

流量控制举例

A向B发送数据。在连接建立时，
B告诉A：“我的接收窗口 rwnd = 400（字节）”。



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 超时重传旧的数据，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）

6.7 TCP的拥塞控制

- **拥塞控制的一般原理**
  - 在某段时间，若对网络中某资源的需求超过了该资源所能提供的可用部分，网络的性能就要变坏——产生拥塞(congestion)
  - 出现资源拥塞的条件：

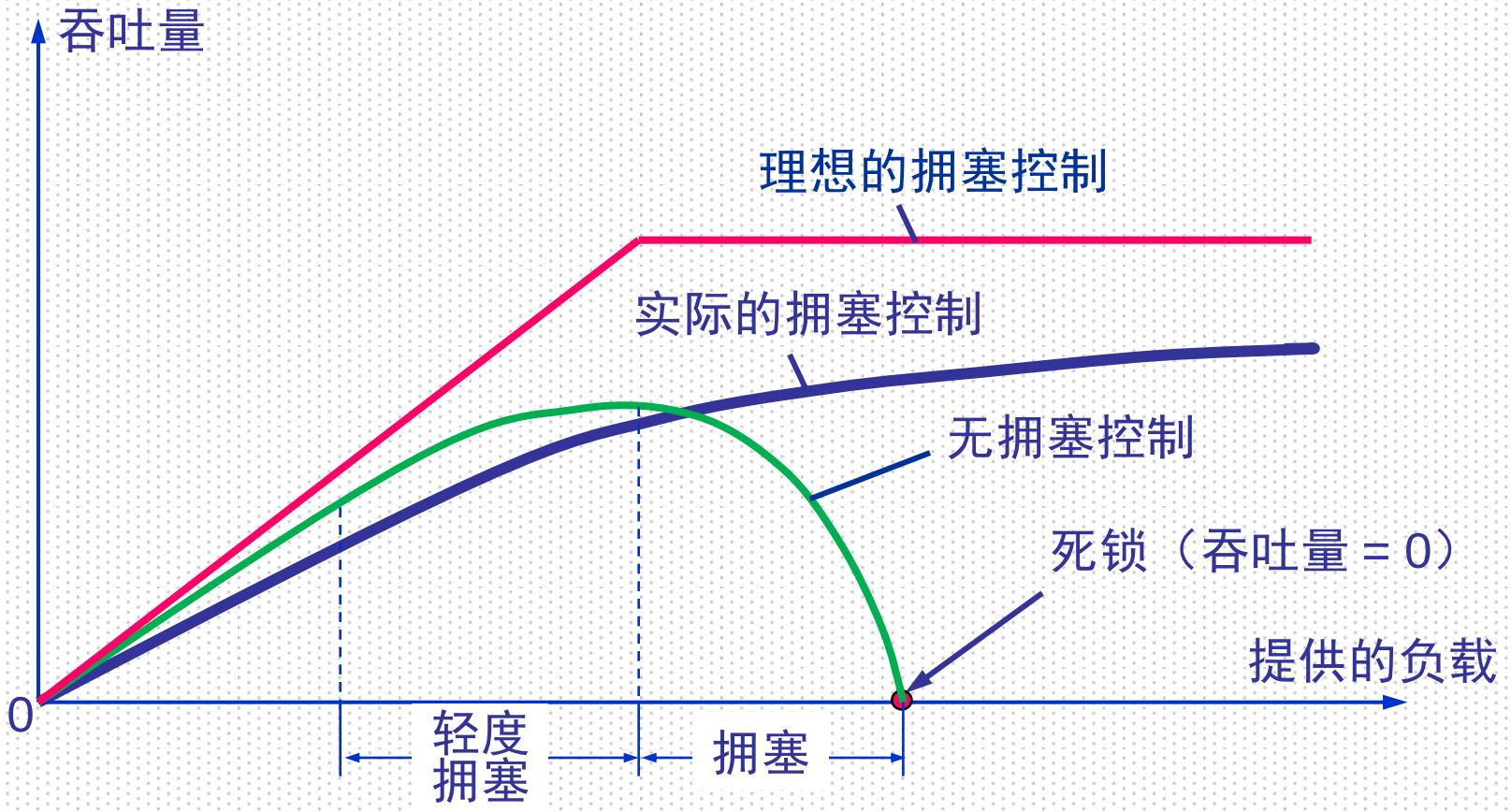
    对资源需求的总和＞可用资源
  - 若网络中有许多资源同时产生拥塞，网络的性能就要明显变坏，整个网络的吞吐量将随输入负荷的增大而下降

6.7 TCP的拥塞控制

- 拥塞控制与流量控制的关系
  - 拥塞控制所要做的都有一个前提，就是网络能够承受现有的网络负荷
  - 拥塞控制是一个全局性的过程，涉及到所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素
  - 流量控制往往指在给定的发送端和接收端之间的点对点通信量的控制
  - 流量控制所要做的就是抑制发送端发送数据的速率，以便使接收端来得及接收

6.7 TCP的拥塞控制

- 拥塞控制所起的作用



6.7 TCP的拥塞控制

- **拥塞控制的一般原理**
  - 拥塞控制是很难设计的，因为它是一个动态的（而不是静态的）问题
  - 当前网络正朝着高速化的方向发展，这很容易出现缓存不够大而造成分组的丢失。但分组的丢失是网络发生拥塞的征兆而不是原因
  - 在许多情况下，甚至正是拥塞控制本身成为引起网络性能恶化甚至发生死锁的原因。这点应特别引起重视

6.7 TCP的拥塞控制

- 开环控制和闭环控制
  - 开环控制方法就是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
  - 闭环控制是基于反馈环路的概念。属于闭环控制的有以下几种措施:
    - 监测网络系统以便检测到拥塞在何时、何处发生
    - 将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

6.8 TCP的传输连接管理

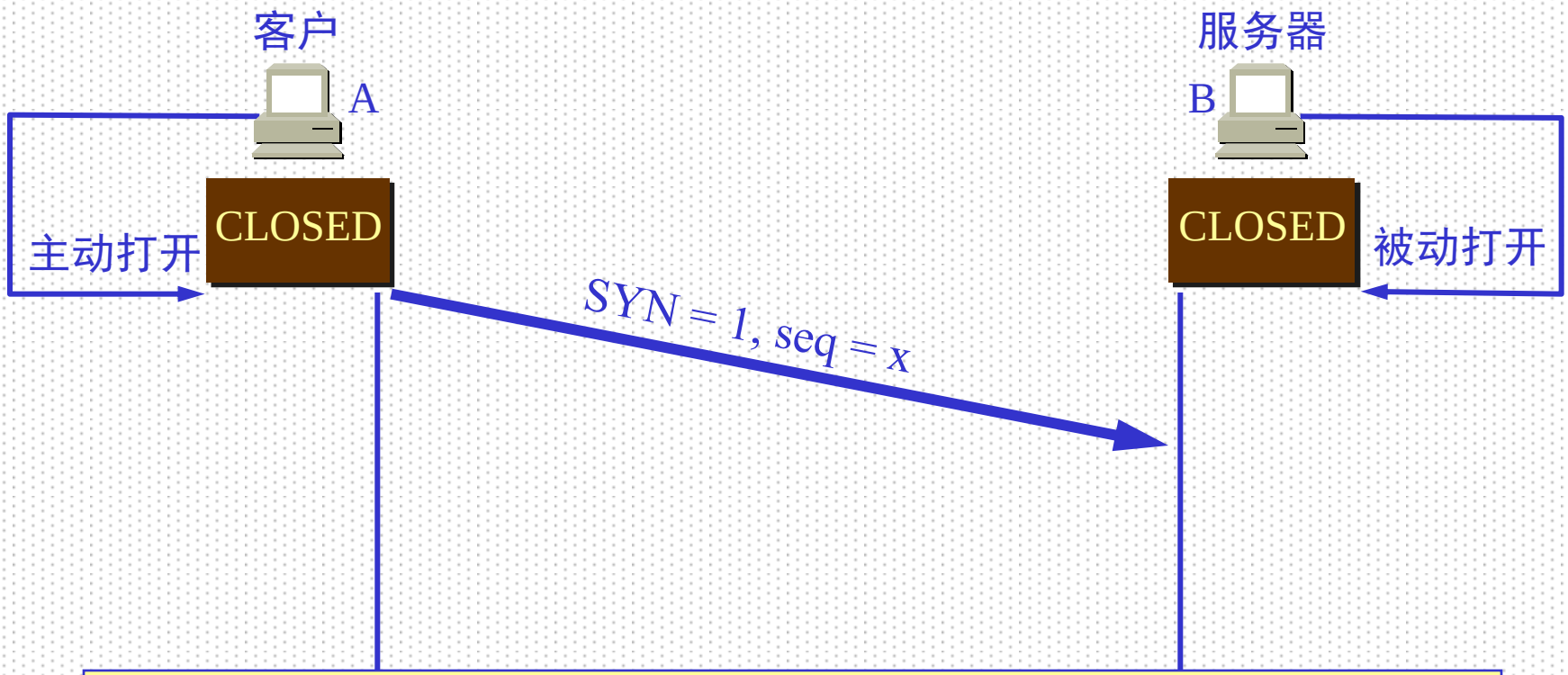
- 传输连接的三个阶段
  - 传输连接有三个阶段，即：连接建立、数据传送和连接释放。传输连接的管理就是使传输连接的建立和释放都能正常地进行
  - 连接建立过程中要解决以下三个问题：
    - 要使每一方能够确知对方的存在
    - 要允许双方协商一些参数（如最大报文段长度，最大窗口大小，服务质量等）
    - 能够对运输实体资源（如缓存大小，连接表中的项目等）进行分配

6.8 TCP的运输连接管理

- 客户服务器方式
  - TCP连接的建立都是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(client)
  - 被动等待连接建立的应用进程叫做服务器(server)

6.8 TCP的运输连接管理

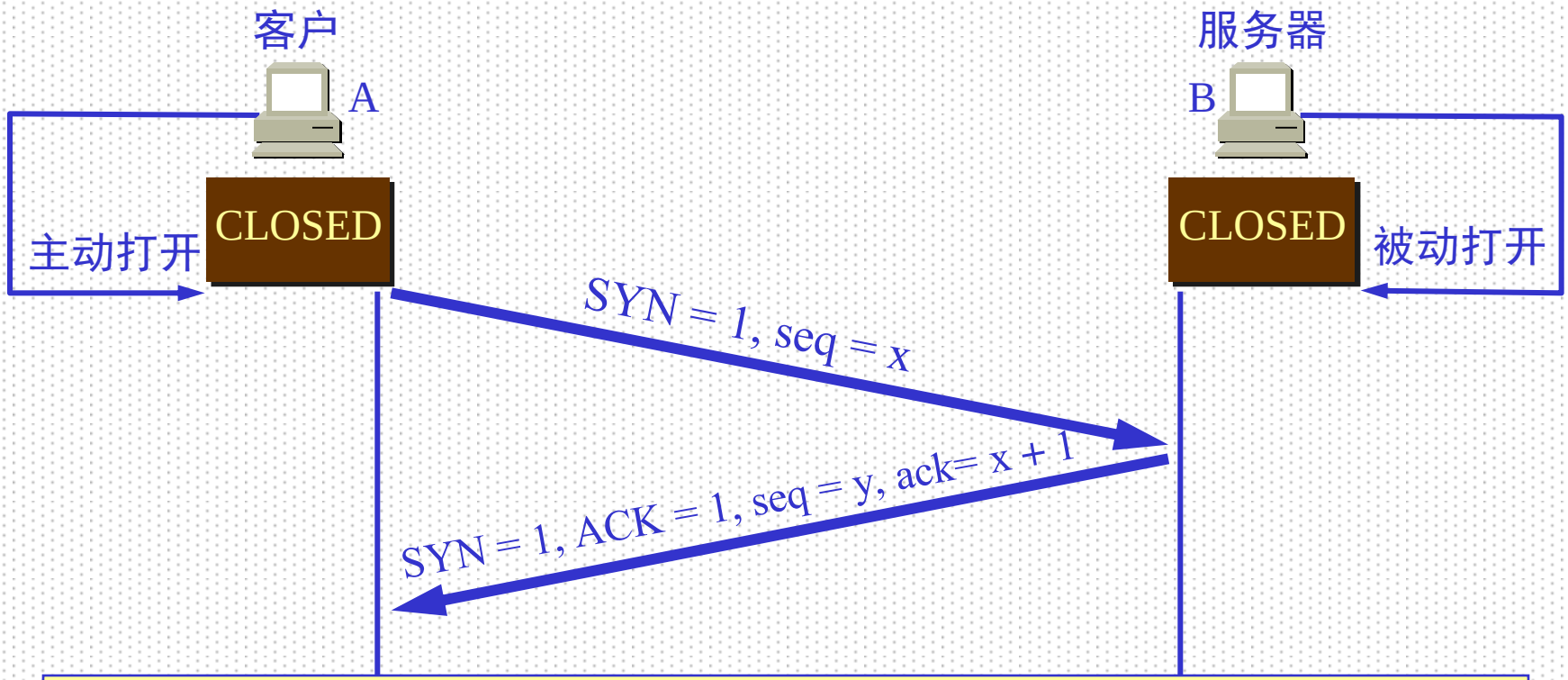
- TCP的连接建立：用三次握手建立TCP连接



A的TCP向B发出连接请求报文段，其首部中的同步位SYN=1，并选择序号seq=x，表明传送数据时的第一个数据字节的序号是x。

6.8 TCP的运输连接管理

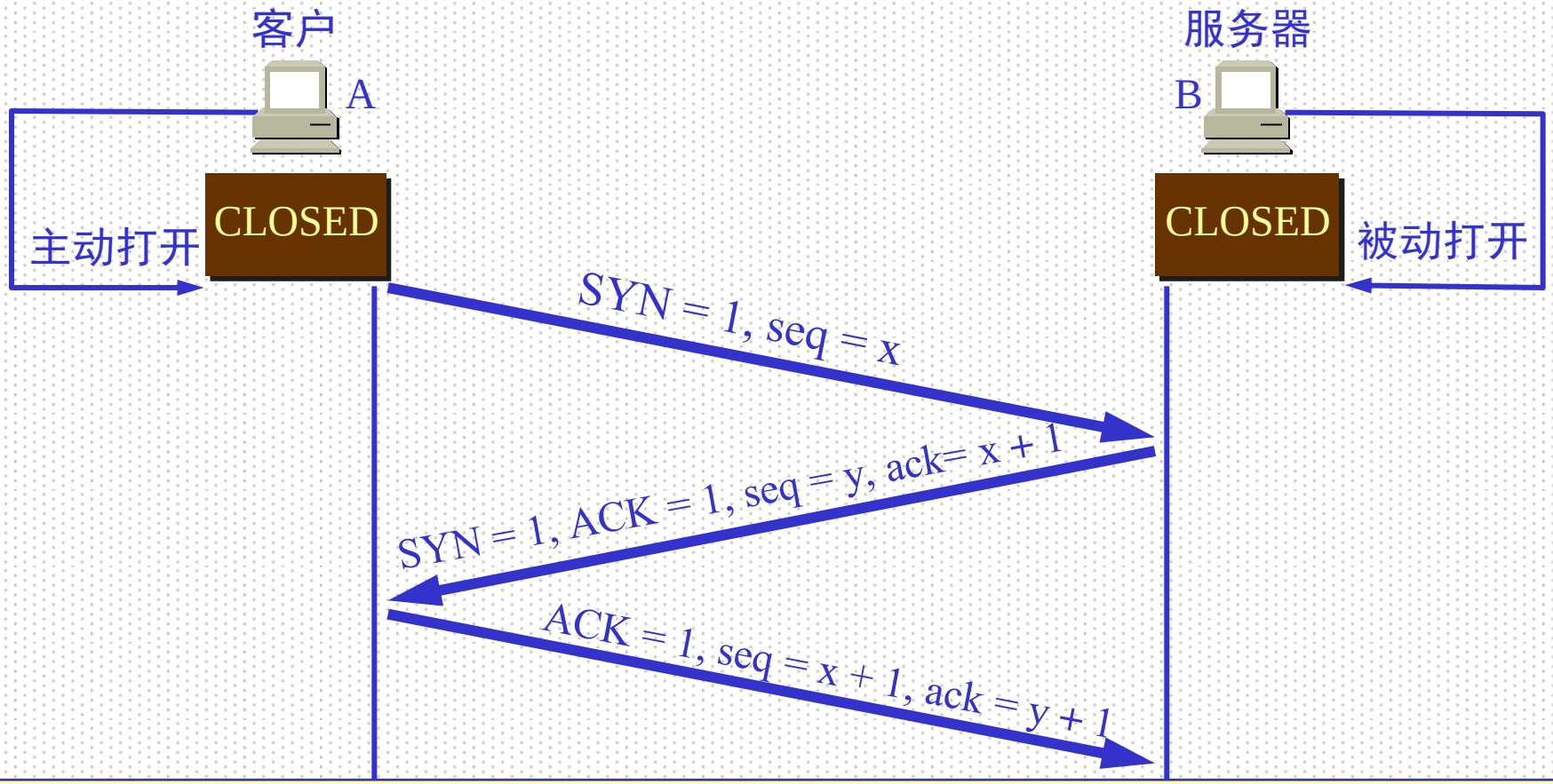
- TCP的连接建立：用三次握手建立TCP连接



- B的TCP收到连接请求报文段后，如同意，则发回确认。
- B在确认报文段中应使SYN=1，使ACK=1，其确认号ack=x+1，自己选择的序号seq=y。

6.8 TCP的运输连接管理

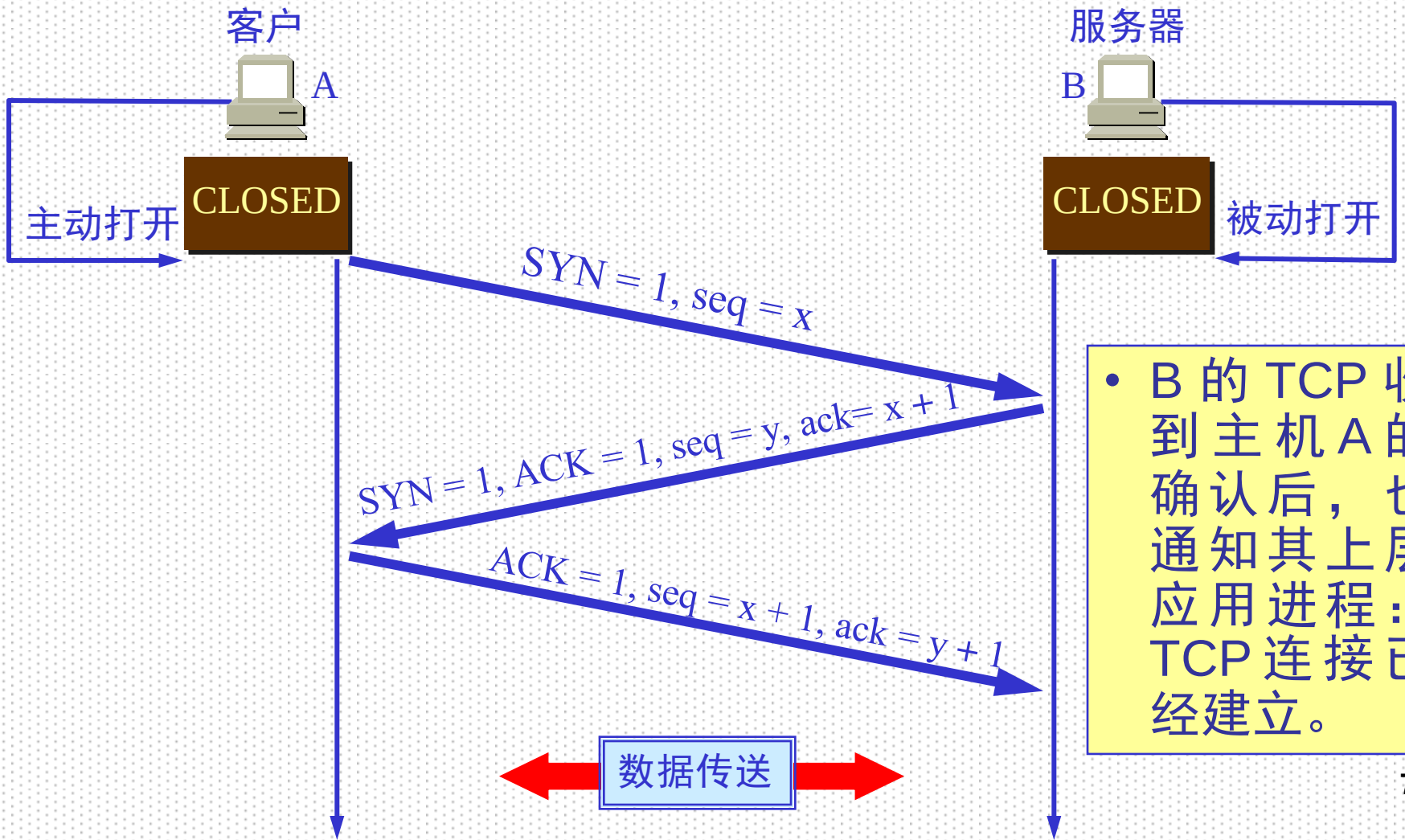
- TCP的连接建立：用三次握手建立TCP连接



- A收到此报文段后向B给出确认，其ACK=1，确认号ack=y+1。
- A的TCP通知上层应用进程，连接已经建立。

6.8 TCP的运输连接管理

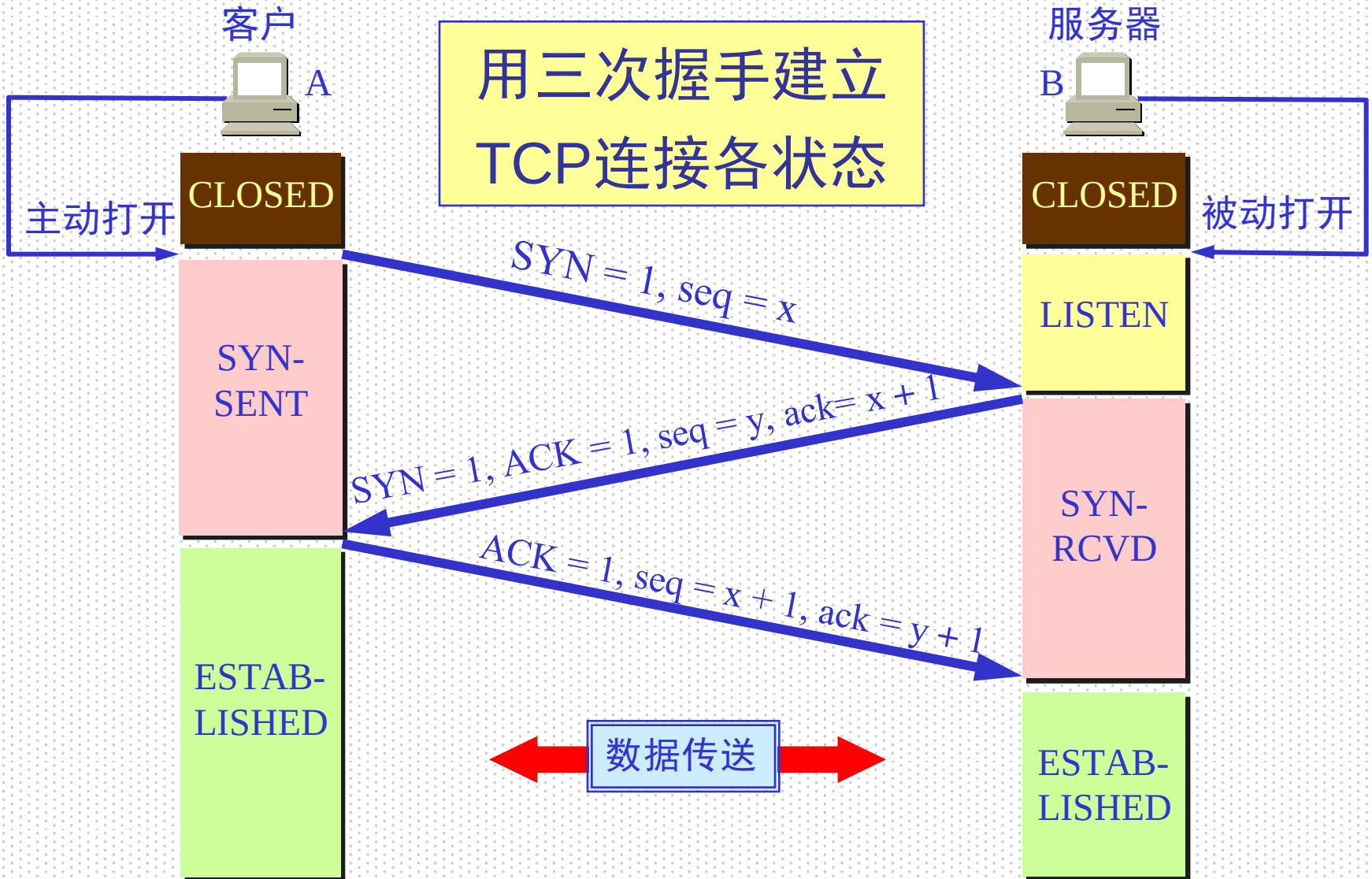
- TCP的连接建立：用三次握手建立TCP连接



- B 的 TCP 收到主机 A 的确认后，也通知其上层应用进程：TCP 连接已经建立。

6.8 TCP的运输连接管理

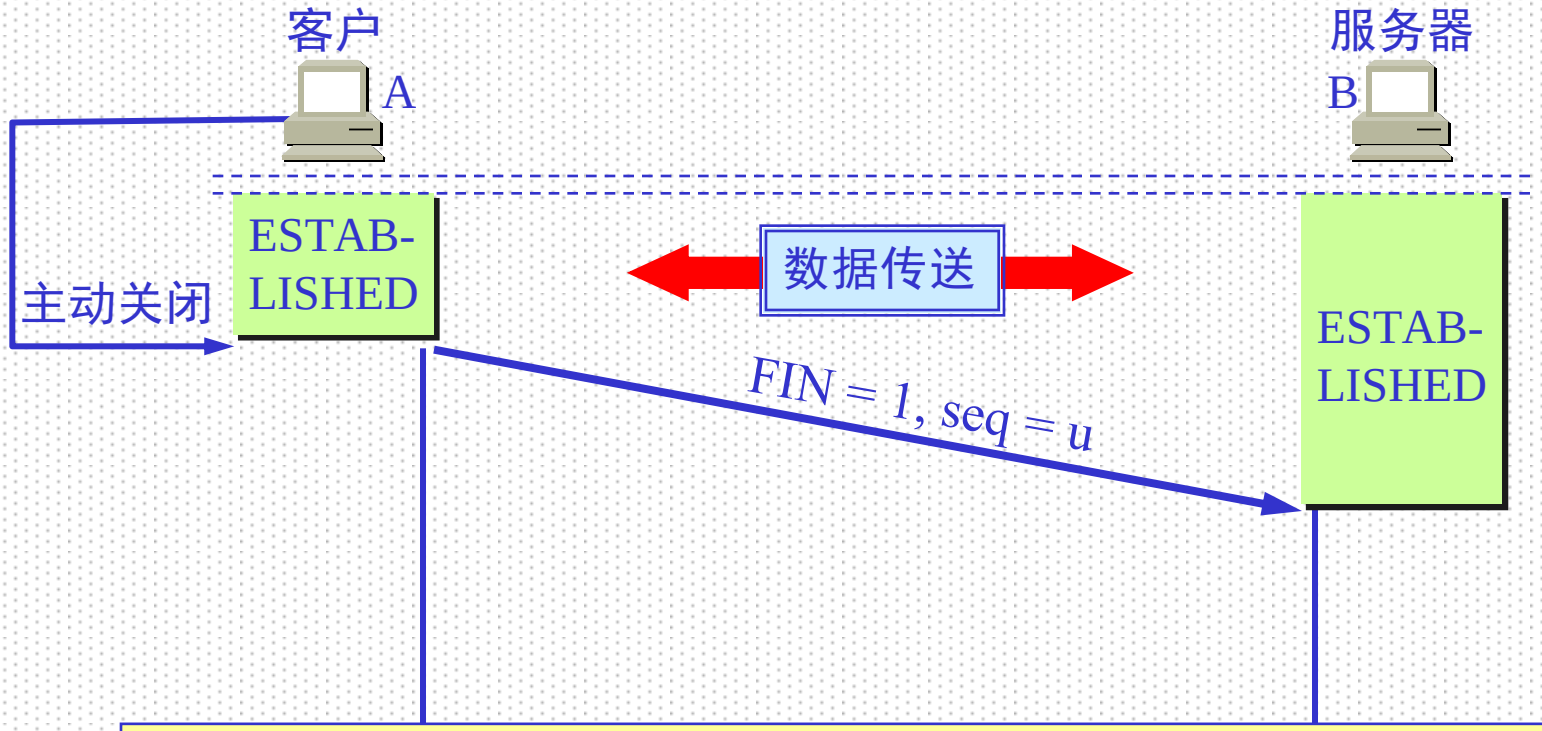
- TCP的连接建立



6.8 TCP的运输连接管理

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

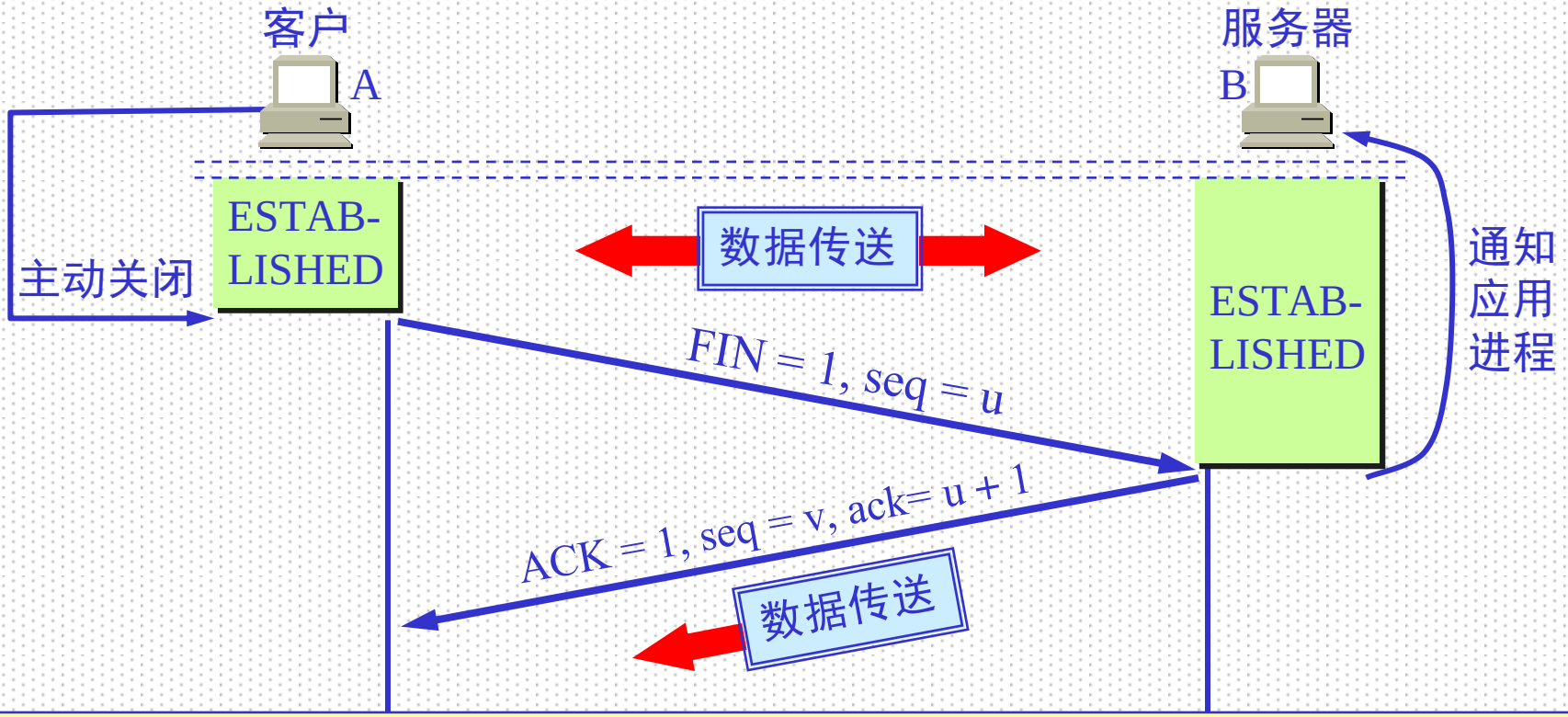
- TCP的连接释放



- 数据传输结束后，通信的双方都可释放连接。现在A的应用进程先向其TCP发出连接释放报文段，并停止再发送数据，主动关闭TCP连接。
- A把连接释放报文段首部的FIN=1，其序号seq=u，等待B的确认。

6.8 TCP的运输连接管理

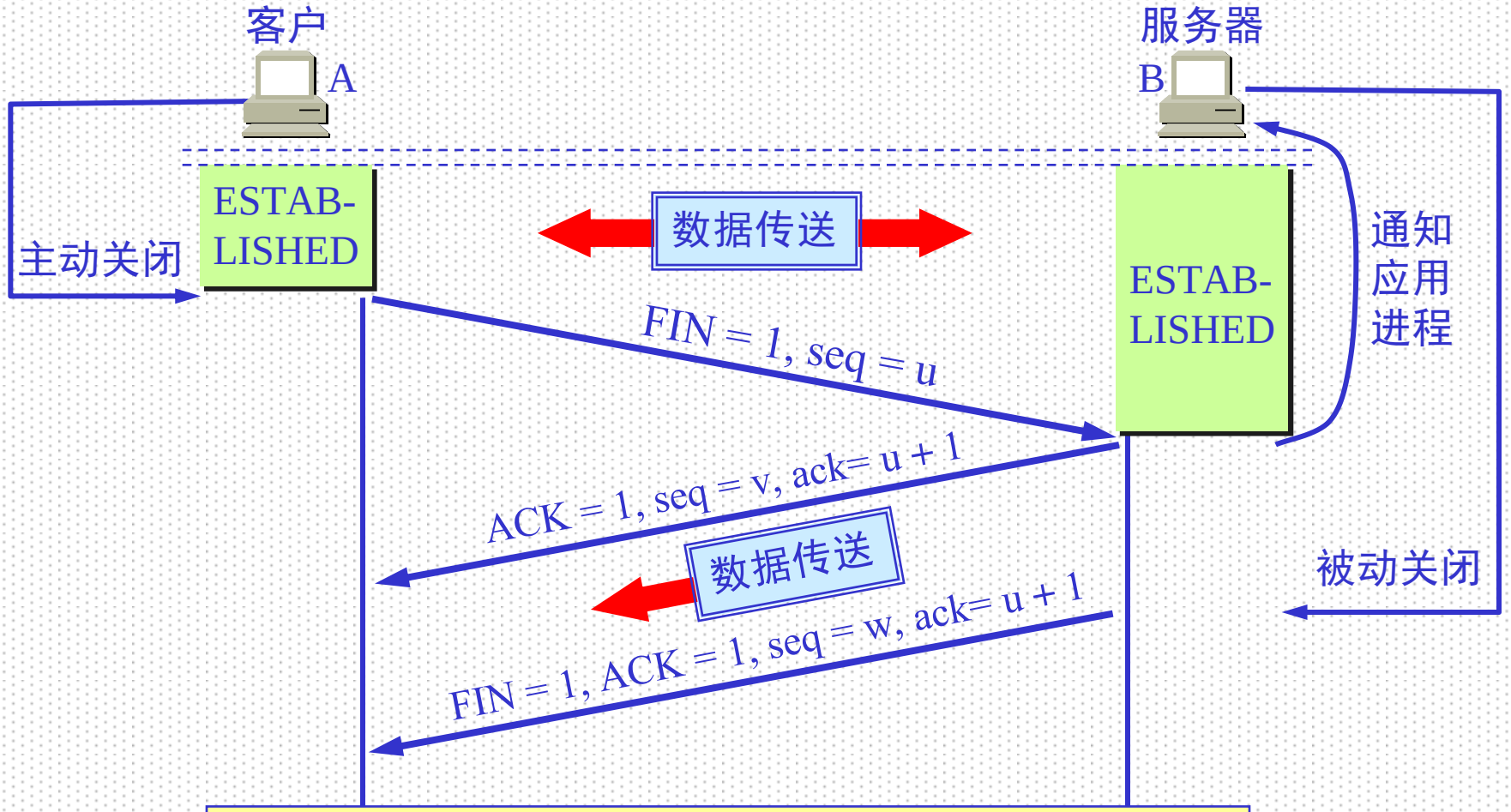
- TCP的连接释放



- B发出确认，确认号ack=u+1，而这个报文段自己的序号seq=v。
- TCP服务器进程通知高层应用进程。
- 从A到B这个方向的连接就释放了，TCP连接处于半关闭状态。B若发送数据，A仍要接收。

6.8 TCP的运输连接管理

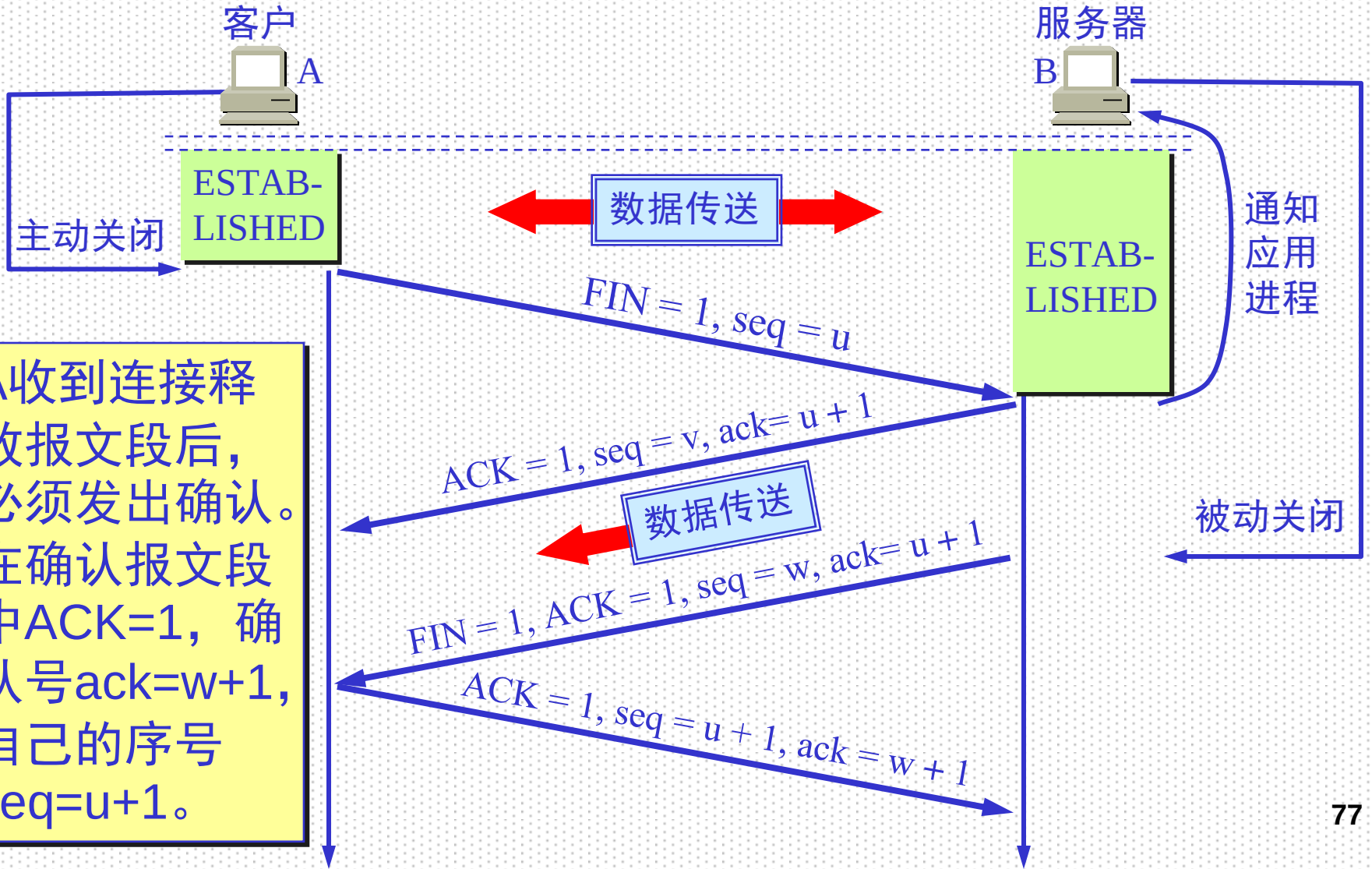
- TCP的连接释放



- 若B已经没有要向A发送的数据，其应用进程就通知TCP释放连接。

6.8 TCP的运输连接管理

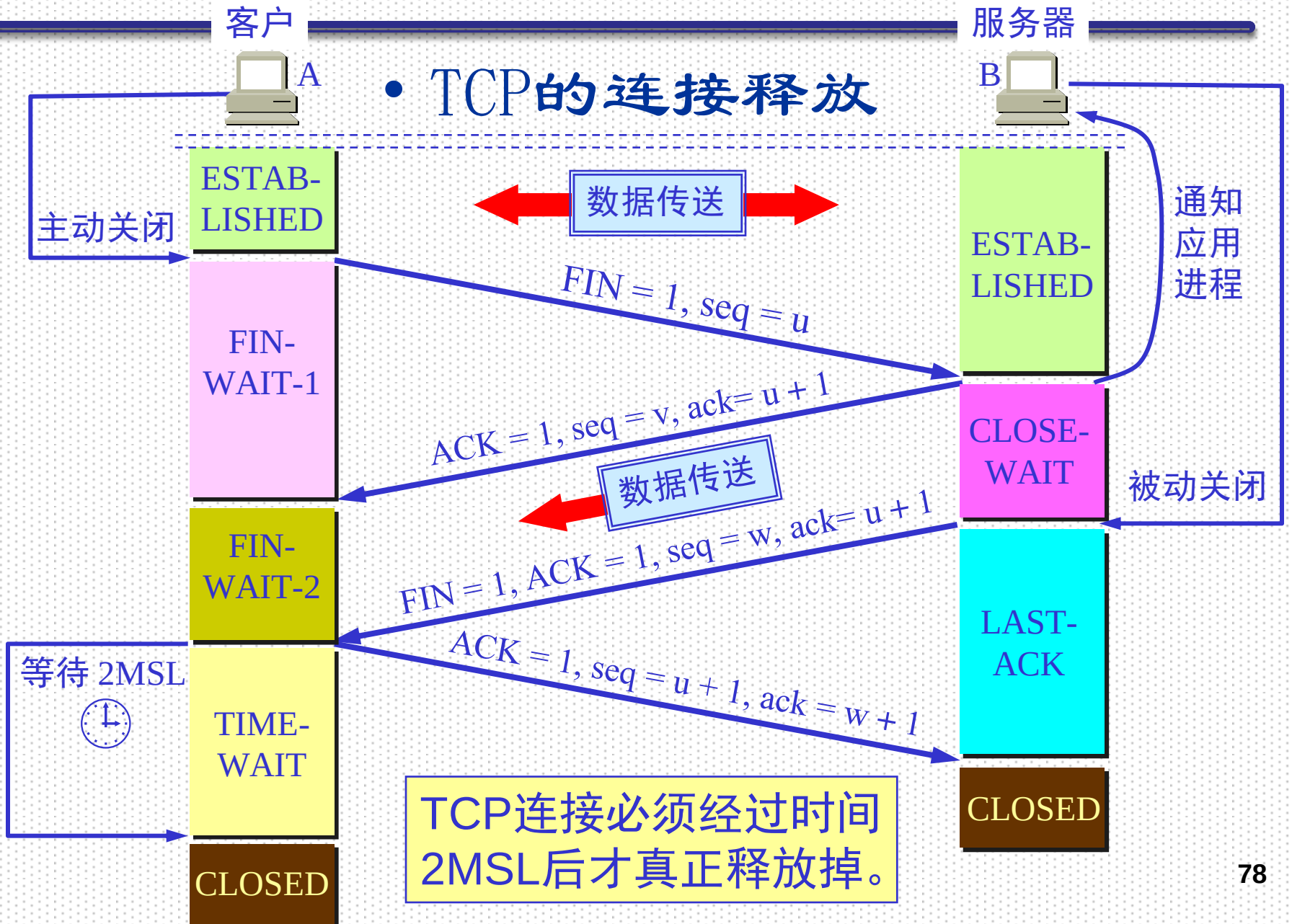
- TCP的连接释放



- A收到连接释放报文段后，必须发出确认。
- 在确认报文段中ACK=1，确认号ack=w+1，自己的序号seq=u+1。

6.8 TCP的运输连接管理

• TCP的连接释放



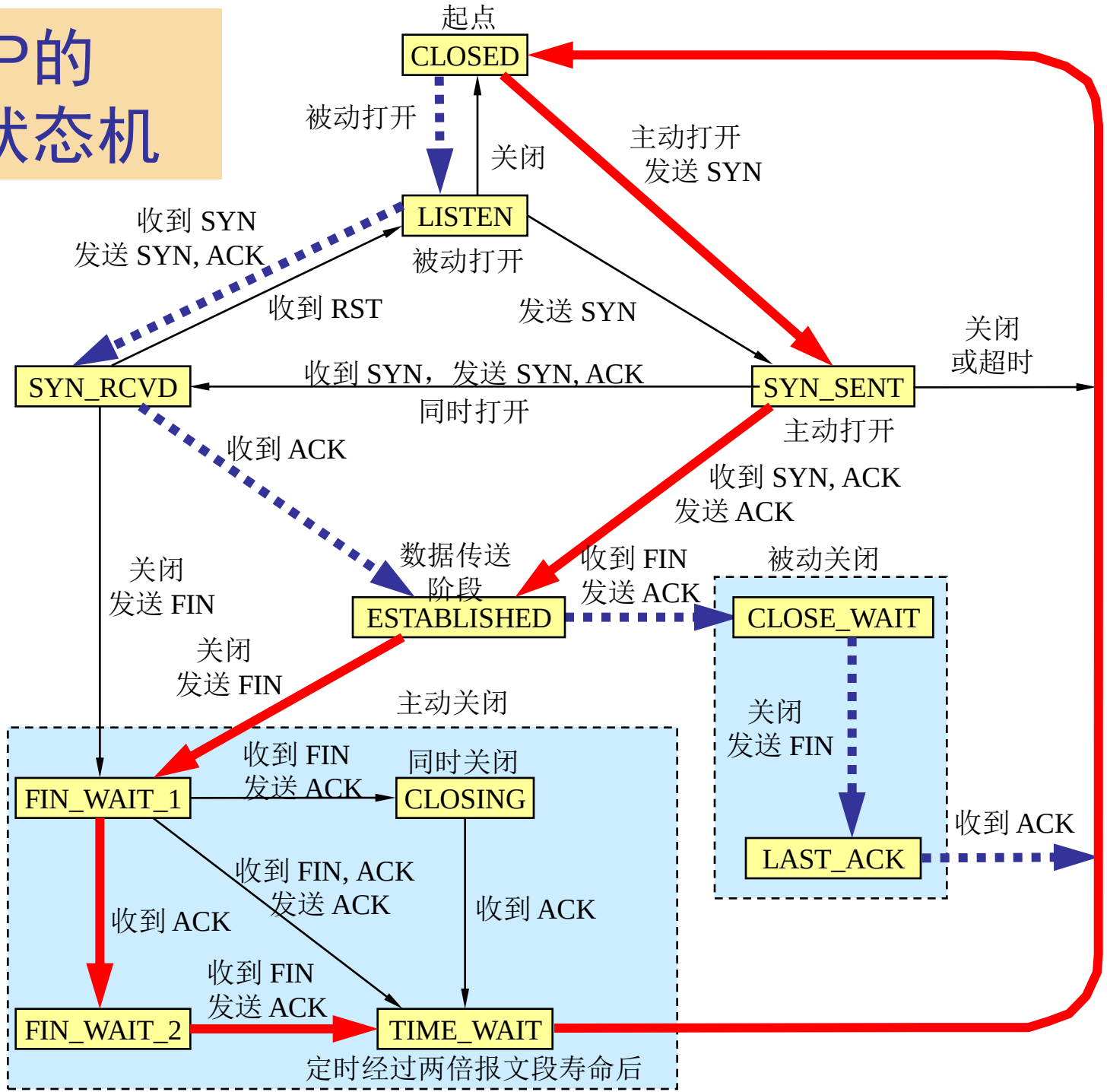
6.8 TCP的运输连接管理

- TCP的连接释放
  - A必须等待2MSL（Maximum Segment Lifetime，报文最大生存时间）的时间
    - 为了保证A发送的最后一个ACK报文段能够到达B
    - 防止“已失效的连接请求报文段”出现在本连接中。A发送完最后一个ACK报文段后，再经过2MSL的时间，就可以使本连接持续的时间内所产生的所有报文段，都从网络中消失。这样就可以使下一个新的连接中不会出现这种旧的连接请求报文段

6.8 TCP的运输连接管理

- TCP的有限状态机
  - TCP有限状态机图中每一个方框都是TCP可能具有的状态
  - 每个方框中的大写英文字符串是TCP标准所使用的TCP连接状态名。状态之间的箭头表示可能发生的状态变迁
  - 箭头旁边的字，表明引起这种变迁的原因，或表明发生状态变迁后又出现什么动作
  - 图中有三种不同的箭头
    - 粗实线箭头表示对客户进程的正常变迁
    - 粗虚线箭头表示对服务器进程的正常变迁
    - 另一种细线箭头表示异常变迁

TCP的有限状态机


起点
CLOSED
被动打开
关闭
主动打开
发送 SYN
LISTEN
收到 SYN
发送 SYN, ACK
被动打开
收到 RST
发送 SYN
关闭
或超时
SYN_RCVD
收到 SYN，发送 SYN, ACK
同时打开
SYN_SENT
主动打开
收到 ACK
收到 SYN, ACK
发送 ACK
关闭
发送 FIN
数据传送
阶段
收到 FIN
发送 ACK
被动关闭
ESTABLISHED
CLOSE_WAIT
关闭
发送 FIN
主动关闭
关闭
发送 FIN
收到 FIN
发送 ACK
同时关闭
FIN_WAIT_1
CLOSING
收到 FIN, ACK
发送 ACK
收到 ACK
收到 ACK
收到 ACK
LAST_ACK
收到 FIN
发送 ACK
FIN_WAIT_2
TIME_WAIT
定时经过两倍报文段寿命后

本章知识点小结

- 传输层的功能
- 端口及套接字
- TCP协议
- UDP协议
- 客户服务器方式
- 三次握手
- 思考题
  - P231：6.1、6.2、6.4、6.5、6.6、6.7、6.14、6.15、6.19、6.25