第三章 数据链路层

刘 轶
北京航空航天大学 计算机学院

本章内容

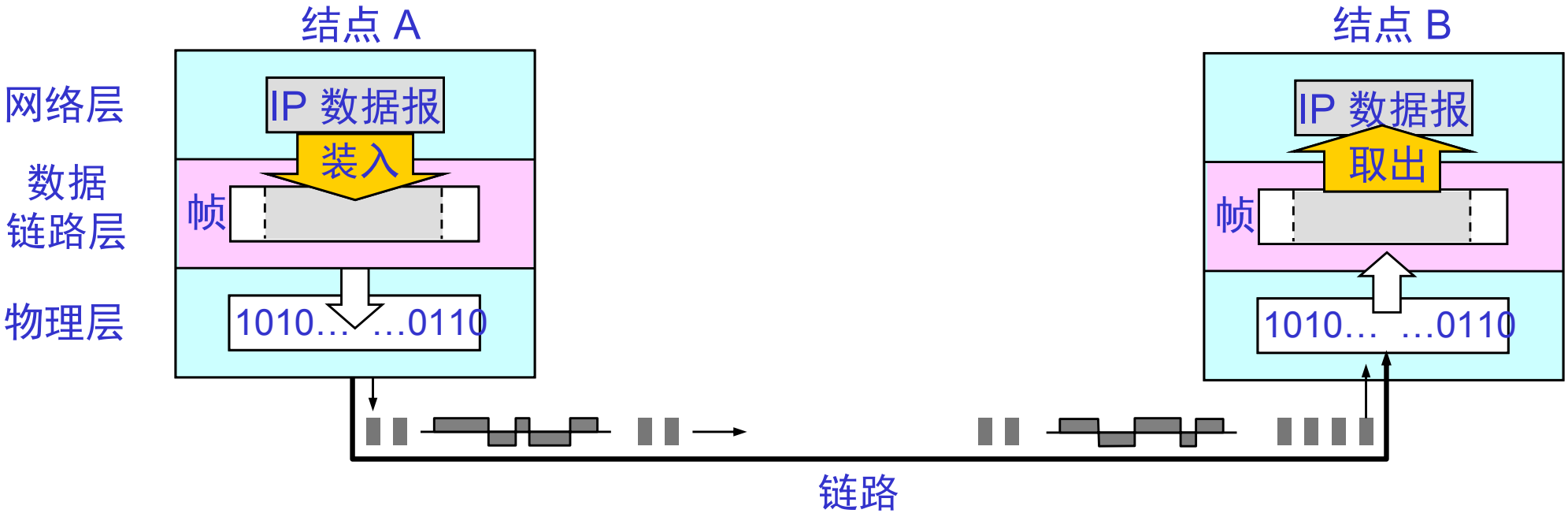
3.1 数据链路层设计要点
3.2 错误检测和纠正
3.3 基本数据链路协议
3.4 滑动窗口协议
3.5 点对点协议 PPP
3.6 介质访问控制
3.7 以太网
3.8 局域网互连

3.1 数据链路层设计要点

3.1 数据链路层设计要点

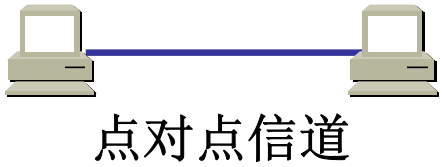
一、数据链路层概述(1/2)

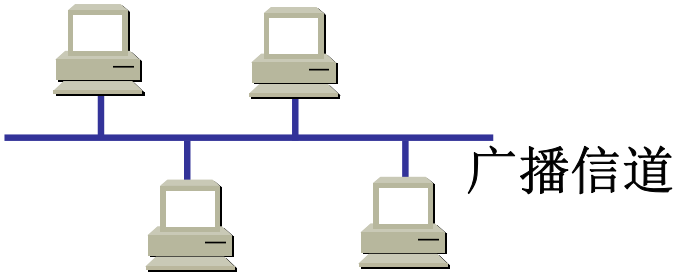
- 物理层实现了比特流的传输，数据链路层在其基础上实现帧(frame)的传输
  - 数据链路层传输的协议数据单元(PDU)是帧



3.1 数据链路层设计要点

一、数据链路层概述(2/2)





- 数据链路层使用的信道类型
  - 点对点信道
    - 使用一对一的点对点通信方式。
  - 广播信道
    - 使用一对多的广播通信方式
    - 广播信道上连接多个主机，必须采用专门的共享信道协议来协调数据发送
- 数据链路层涉及的问题
  - ① 成帧(framing)：怎样组成帧、怎样使接收方识别帧
  - ② 差错控制：帧在传输过程中出错的检测
  - ③ 流量控制及可靠传输：仅是数据链路层的选项
  - ④ 广播信道中的介质访问控制

3.1 数据链路层设计要点

二、成帧方法

- 成帧要考虑的问题：接收方如何识别帧的边界？
- 常用的成帧方法

(1) 字符计数法

在帧头部字段中指明本帧的字节数，接收方通过该字段得知该接收多少字节

(2) 字符填充的首尾定界法

定义专门的字符作为帧的起始/结束标志，并使用字符填充方式将标志字符与数据区分开来

(3) 比特填充的首尾定界法

定义专门的比特序列作为帧的起始/结束标志，并使用比特填充方式将标志序列与数据区分开来

本章PPP协议中可看到实例

(4) 物理编码违例法

使用无效的物理编码作为帧的开始/结束标志，供接收方识别

3.2 错误检测和纠正

3.2 错误检测和纠正

一、检错编码(Error detecting code)

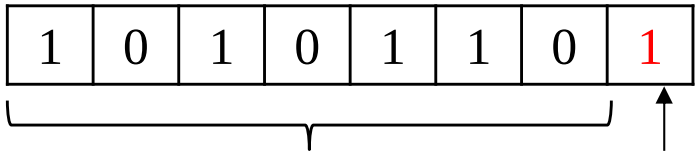
- 任何通信链路在传输数据时都可能出错
- 一般用误码率BER(Bit Error Rate)表示链路可靠性

$$\text{误码率} = \frac{\text{出错的比特数}}{\text{传送的总比特数}}$$

例如：误码率为10^{-10}表示平均每传送10^{10}个比特会出现一个比特出错

- 处理方法
  - 使用可检测并纠正错误的编码：纠错码
  - 使用可检测错误的编码 + 重传：检错码
- 常用检错编码方法
  - 奇偶校验
  - 简单累加和(校验和)
  - 循环冗余校验CRC(Cyclic Redundancy Check)

奇校验举例：



3.2 错误检测和纠正

一、检错编码(Error detecting code)

- 循环冗余校验CRC原理
  - 发送方把数据划分为组，设每组 k 个比特，在其后添加供差错检测用的n 位冗余码，(k+n)比特一起发送
  - 对数据M计算n位冗余码的过程：
    ① 用二进制的模 2 运算进行 2^n乘 M 的运算，这相当于在 M 后面添加 n 个 0
    ② 得到的 (k + n) 位的数除以事先选定好的长度为 (n + 1) 位的除数 P，得出商是 Q 而余数是 R，余数 R 比除数 P 少1 位，即 R 是 n 位
    ③ R作为冗余码，添加在数据M后面，最终发送数据：2^nM+R

    注：除数P为双方事先商定
  - 接收方对收到的(k+n)比特计算冗余码，结果为0表示传输正确，否则表示传输错误

3.2 错误检测和纠正

例：计算101001的3位CRC冗余码

已知：M=101001

k=6, n=3

设：除数P=1101

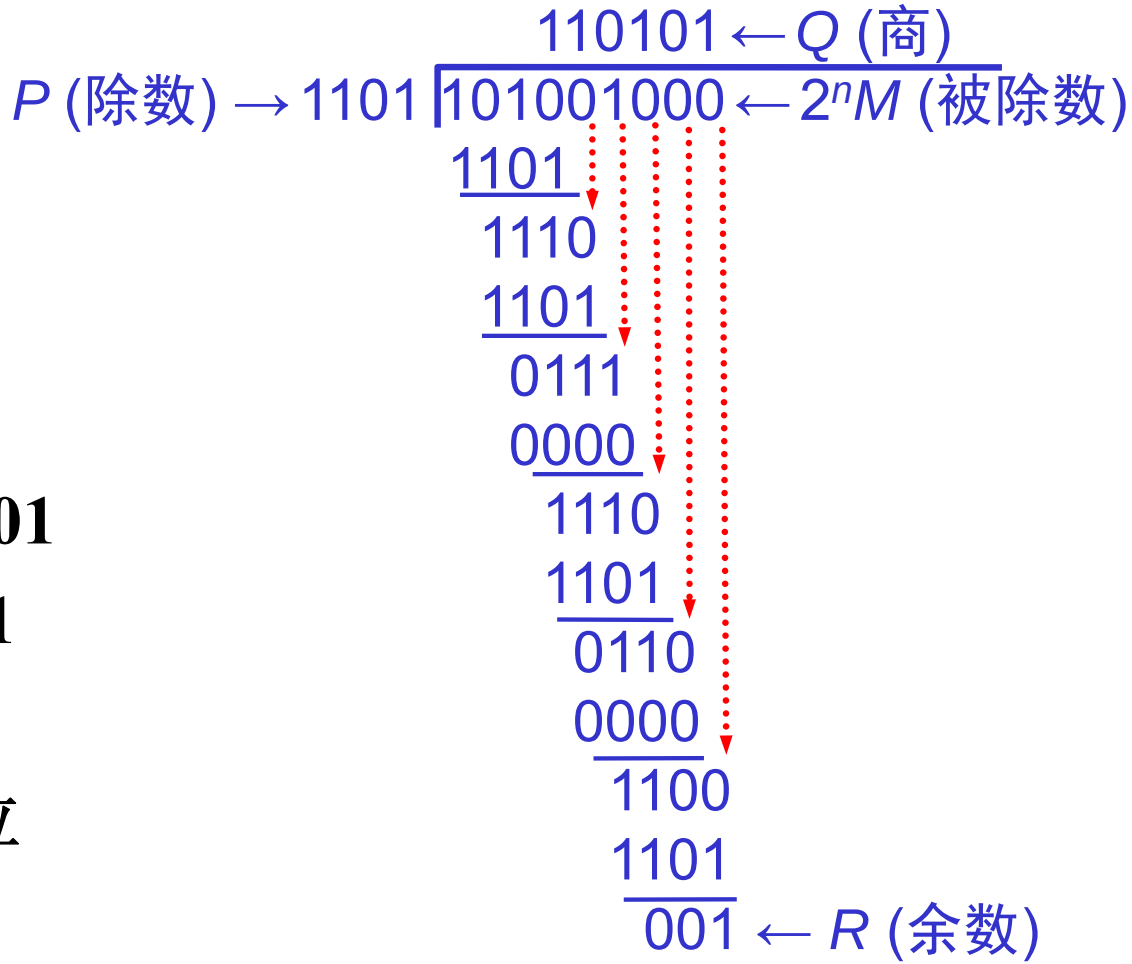
被除数：$2^nM = 101001000$

模 2 运算的结果：商 Q = 110101

余数 R = 001

发送的数据：$2^nM + R$

即：101001001，共 (k + n) 位



3.2 错误检测和纠正

一、检错编码(Error detecting code)

- 循环冗余校验CRC原理
  - 发送方把数据划分为组，设每组 k 个比特，在其后添加供差错检测用的n 位冗余码，(k+n)比特一起发送
  - 对数据M计算n位冗余码的过程：
    ① 用二进制的模 2 运算进行 2^n乘 M 的运算，这相当于在 M 后面添加 n 个 0
    ② 得到的 (k + n) 位的数除以事先选定好的长度为 (n + 1) 位的除数 P，得出商是 Q 而余数是 R，余数 R 比除数 P 少1 位，即 R 是 n 位
    ③ R作为冗余码，添加在数据M后面，最终发送数据：2^nM+R

    注：除数P为双方事先商定
  - 接收方对收到的(k+n)比特计算冗余码，结果为0表示传输正确，否则表示传输错误

3.2 错误检测和纠正

- 通常用生成多项式*P(x)*表示除数P

  例：除数P=1101的生成多项式$P(X) = X^3 + X^2 + 1$

- 目前广泛使用的生成多项式

  $CRC\text{-}16 = X^{16} + X^{15} + X^2 + 1$

  $CRC\text{-}CCITT = X^{16} + X^{12} + X^5 + 1$

  $CRC\text{-}32 = X^{32}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$

- 在网络中具体实现时，通常采用硬件电路生成**CRC校验和(checksum)**

3.2 错误检测和纠正

二、纠错编码(Error correcting code)

- 检错码只能发现数据出现了错误，无法得知哪个比特出错
- 纠错编码通过增加冗余信息使得能够检测错误发生所在，以便于纠正，又称为前向纠错(forward error correcting)
  - 海明编码

- 关于数据链路层检错/纠错的讨论
  - 通过检错码/纠错码可以做到帧的无差错接收，或者说“无比特差错”
  - 并不意味着可靠传输，其他的错误包括：
    - 帧丢失、帧重复、帧失序

3.3 基本数据链路协议

3.3 基本数据链路协议

未考虑接收方的处理速度
设想：发送方发送>接收方处理速度

一、无限制的单工协议

- 假设前提
  - 单向传输
  - 理想信道
  - 发送方总有数据发送
  - 接收方总能及时处理所收到的数据

发送方

```
void sender1(void)
{
   frame  s;                  //待发送帧
   packet buffer;             //待发送数据包
   while ( true )
   {
      from_network_layer( &buffer ); //从网络层取包
      s.info = buffer;                //拷贝到s中发送
      to_physical_layer( &s );        //通过物理层发送
   }
}
```

接收方

```
void receiver1(void)
{
   frame  r;
   event_type event;   //等待到的事件
   while ( true )
   {
      wait_for_event( &event ); //等待帧到达
      from_physical_layer( &r );//接收帧
      to_network_layer( &r.info );//上交给网络层
   }
}
```

3.3 基本数据链路协议

考虑了接收方的处理速度
但未考虑传输出错

二、单工停-等(stop-and-wait)协议

- 按照“无限制的单工协议”，如果接收方处理帧的速度不及发送方，则帧可能丢失
- 解决方法：增加流量控制(flow control)机制，得到单工停等协议
  - 接收方每收到一帧，都向发送方返回一个应答帧
  - 发送方每发送一帧，都等待来自接收方的应答帧，之后才发送下一帧

发送方

```
void sender2(void)
{  frame  s;                    //待发送帧
   packet buffer;               //待发送数据包
   event_type event;            //等待到的事件
   while ( true )
   {
      from_network_layer( &buffer ); //从网络层取包
      s.info = buffer;                //拷贝到s中发送
      to_physical_layer( &s );        //通过物理层发送
      wait_for_event( &event );       //等待应答帧
   }
}
```

接收方

```
void receiver2(void)
{
   frame  r, s;
   event_type event;   //等待到的事件
   while ( true )
   {
      wait_for_event( &event ); //等待帧到达
      from_physical_layer( &r );//接收帧
      to_network_layer( &r.info );//上交给网络层
      to_physical_layer( &s );     //发送应答帧
   }
}
```

3.3 基本数据链路协议

三、有噪声信道的单工协议

- 在有噪声信道中，帧在传输过程中可能出错
- 解决方法　**---ARQ(Automatic Repeat reQuest)协议**
  - 校验和：使接收方能够检测帧是否出错
  - 确认帧：使发送方知道帧已被正确接收
  - 超时重发：发送方在规定时间内未收到确认帧，则重发帧
  - 帧序号：保证接收方不会重复接收帧
- 协议设计要考虑的三种情形

① 数据帧被正确接收

接收方返回确认帧，发送方收到后继续发送下一帧

② 数据帧出错或丢失

接收方未收到帧或校验出错丢弃该帧，发送方等待确认帧超时后，重发数据帧

③ 确认帧出错或丢失

发送方未收到有效的确认帧，重发数据帧，接收方收到后，检查帧序号重复，不上交该帧，只返回确认帧

发送方

```
void sender3(void)
{  …
   next_frame_to_send = 0;            //帧序号清0
   from_network_layer( &buffer ); //从网络层取包
   while ( true )
   {
      s.info = buffer;                    //拷贝到s中发送
      s.seq = next_frame_to_send;     //帧序号
      to_physical_layer( &s );          //通过物理层发送
      start_timer( s.seq );               //启动计时器
      wait_for_event( &event );         //等待应答帧
      if  ( event == 应答帧到达 )
      {
         from_physical_layer( &s );    //接收应答帧
         if  ( s.ack == next_frame_to_send ) //检查帧序号
         {
            stop_timer( s.ack );          //停止计时器
            from_network_layer( &buffer ); //取下一帧
            inc( next_frame_to_send ); //0,1切换
          }
       }
   }
}
```

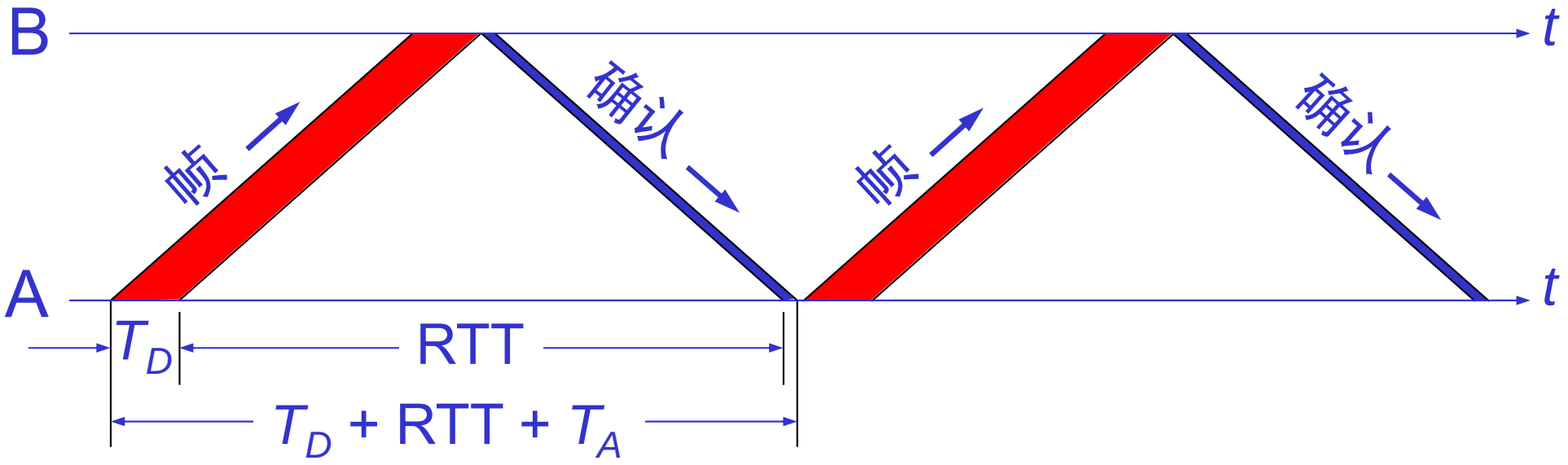
接收方

```
void receiver3(void)
{
   frame_expected = 0;   //期待的帧序号
   while ( true )
   {
      //可能的事件：帧到达、校验错
      wait_for_event( &event );
      if ( event == 帧到达 )  //收到有效帧
      {
         from_physical_layer( &r );//接收帧
         if ( r.seq == frame_expected )//帧序号
         {
            to_network_layer( &r.info );//上交
            inc( frame_expected );        //0,1切换
          }
         s.ack = 1 – frame_expected;//应答序号
         to_physical_layer( &s );     //发送应答帧
       }
    }
}
```

- 传输链路存在时延，而**ARQ**协议在同一时刻仅有一个帧在链路上传输(数据帧或确认帧)，其对信道的利用率较低
  - 请回忆第**1**章介绍的时延带宽积概念
- **ARQ**协议的信道利用率过低(尤其当传输时延较长时)

$$U = \frac{T_D}{T_D + RTT + T_A}$$

T_D：发送数据帧的时间
RTT：往返时延/环路时延
T_A：发送应答帧的时间



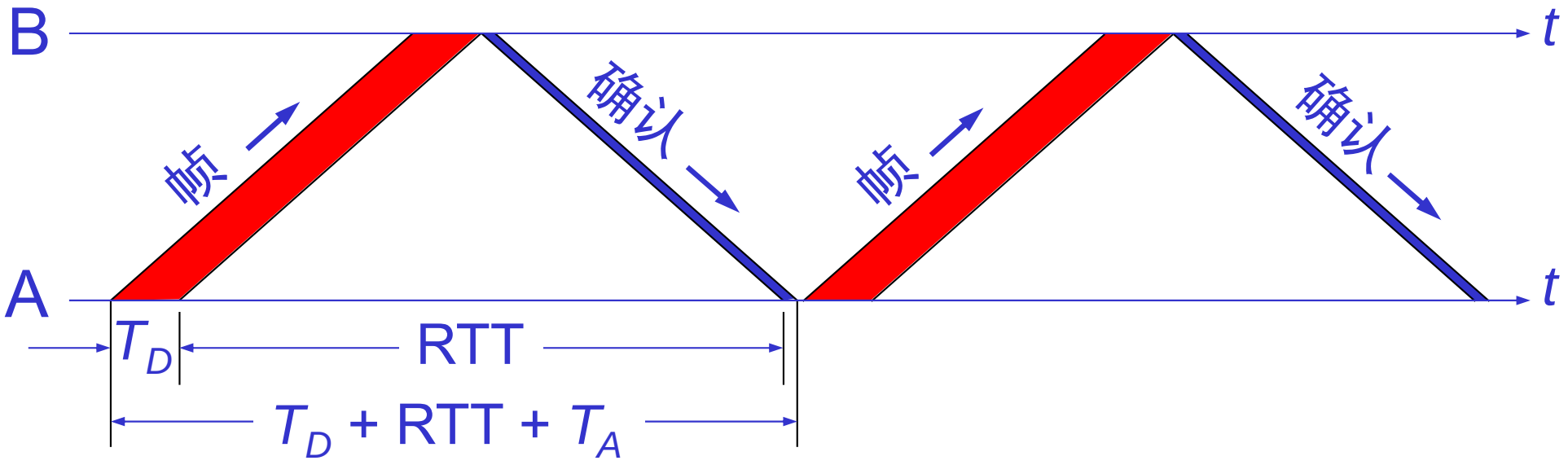
3.4 滑动窗口协议

一、简介

- 传输链路存在时延，而**ARQ**协议在同一时刻仅有一个帧在链路上传输(数据帧或确认帧)，其对信道的利用率较低
  - 请回忆第**1**章介绍的时延带宽积概念
- **ARQ**协议的信道利用率过低(尤其当传输时延较长时)

$$U = \frac{T_D}{T_D + RTT + T_A}$$

T_D：发送数据帧的时间
RTT：往返时延/环路时延
T_A：发送应答帧的时间



3.4 滑动窗口协议

二、滑动窗口协议原理(sliding window protocol)

- 滑动窗口协议的基本思想
  - 允许发送方连续发送多个帧
  - 通过滑动窗口实现流量控制
    - 每个待发送的帧都有一个序列号
    - 发送方维护一个发送窗口，它包含一组序列号，对应允许它发送的帧
    - 接收方维护一个接收窗口，对应允许它接收的帧
- 问题：发送窗口和接收窗口的宽度如何确定？
  - 接收窗口宽度：
  - 发送窗口宽度：

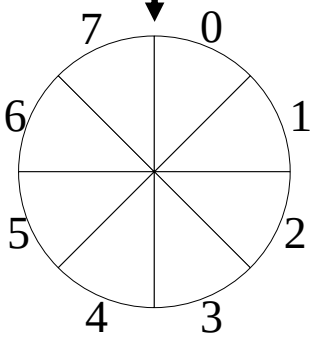
3.4 滑动窗口协议

二、滑动窗口协议原理

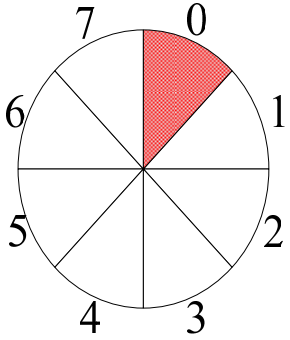
- 发送方
  - 发送窗口内的序列号代表允许它发送的帧
    - 窗口内最大的序列号称为窗口上边界，或窗口上沿、前沿
    - 窗口内最小的序列号称为窗口下边界，或窗口下沿、后沿
  - 每当从网络层得到一个数据包，将其组成帧发出后，发送窗口的上边界+1
  - 发送窗口下边界的帧被接收方确认后，下边界+1
- 接收方
  - 接收窗口内的序列号代表它可以接收的帧
  - 收到的帧序列号等于窗口下边界时，将该帧上交网络层，并返回确认帧，同时整个窗口向前移动1个位置
  - 如果收到帧序列号落在接收窗口之外，则将其丢弃
  - 注意：接收窗口总是保持固定大小

3.4 滑动窗口协议

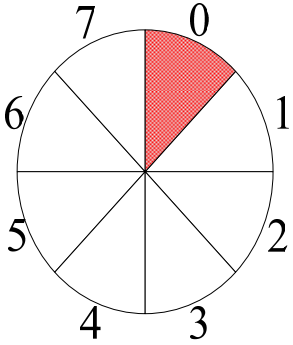
发送方

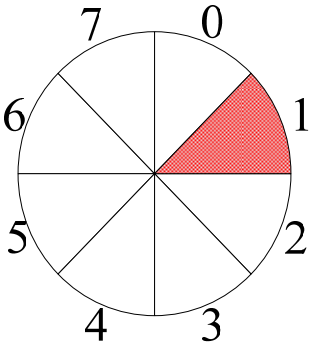


初始状态

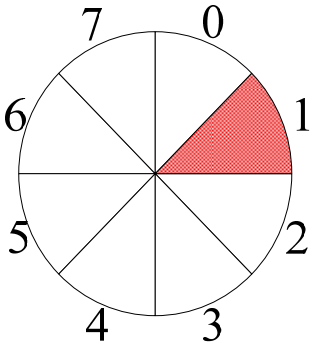


发出0号帧

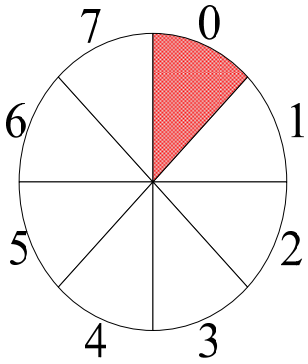




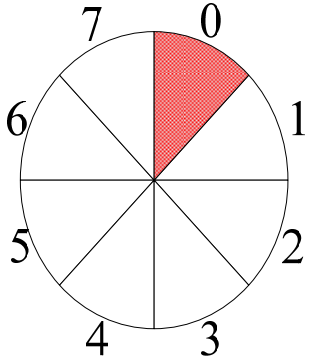
收到ACK0
发出1号帧



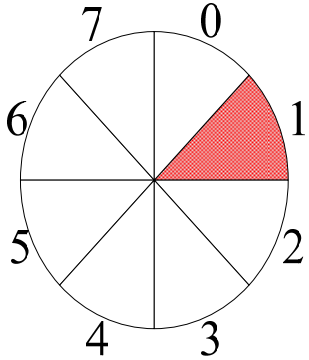
接收方



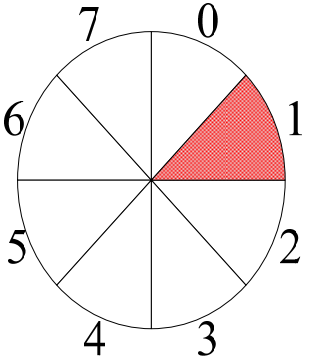
初始状态

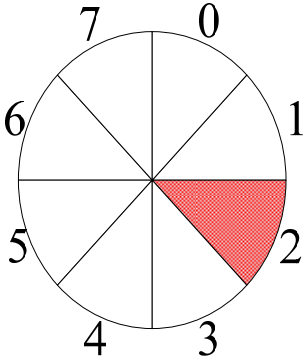


可收0号帧



收到0号帧
返回ACK0





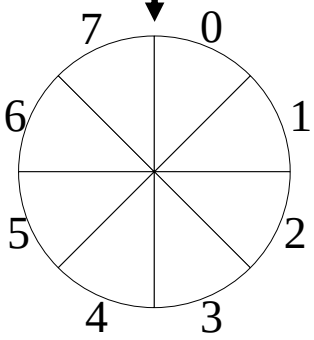
收到1号帧
返回ACK1

假设：发送窗口$W_T = 1$，接收窗口$W_r = 1$

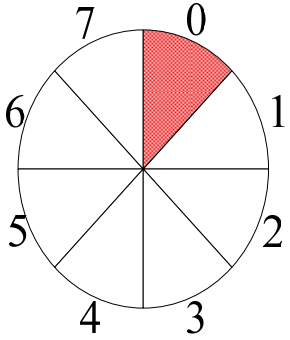
窗口最大尺寸为1的滑动窗口协议称为1位滑动窗口协议，即为ARQ协议

3.4 滑动窗口协议

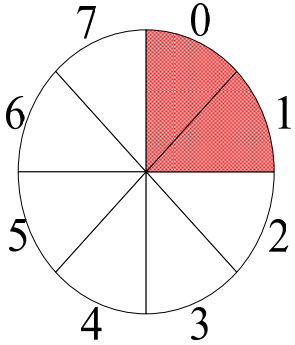
发送方



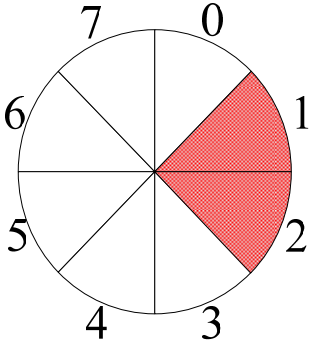
初始状态



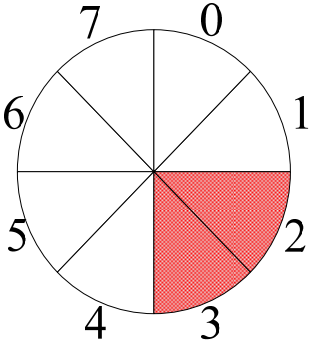
发出0号帧



发出1号帧

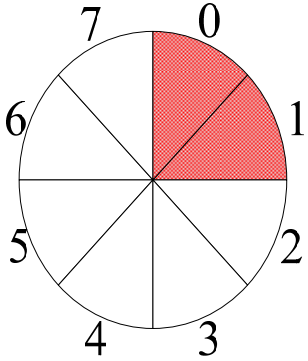


收到ACK0
发出2号帧

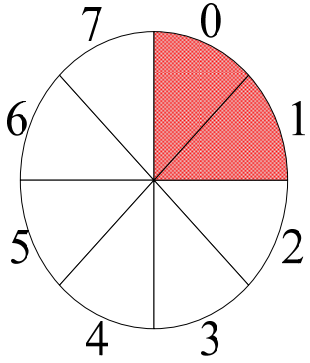


收到ACK1
发出3号帧

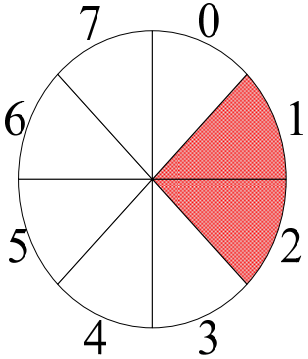
接收方



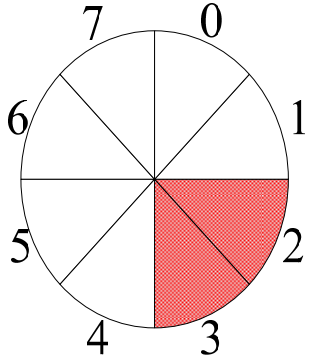
初始状态



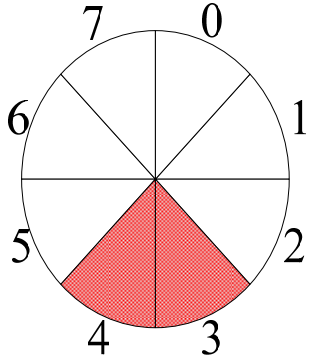
可收0,1号帧



收到0号帧
返回ACK0



收到1号帧
返回ACK1

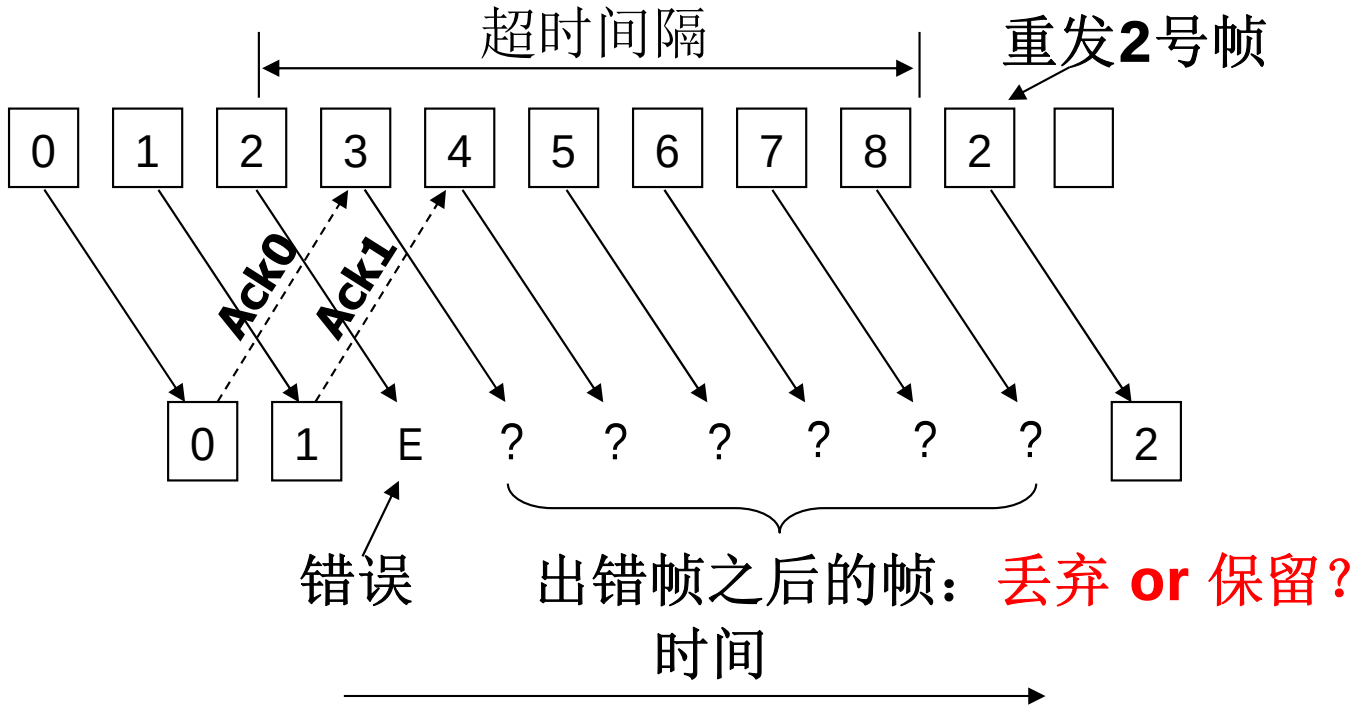


收到2号帧
返回ACK2

假设：发送窗口 $W_T = 2$，接收窗口 $W_r = 2$

3.4 滑动窗口协议

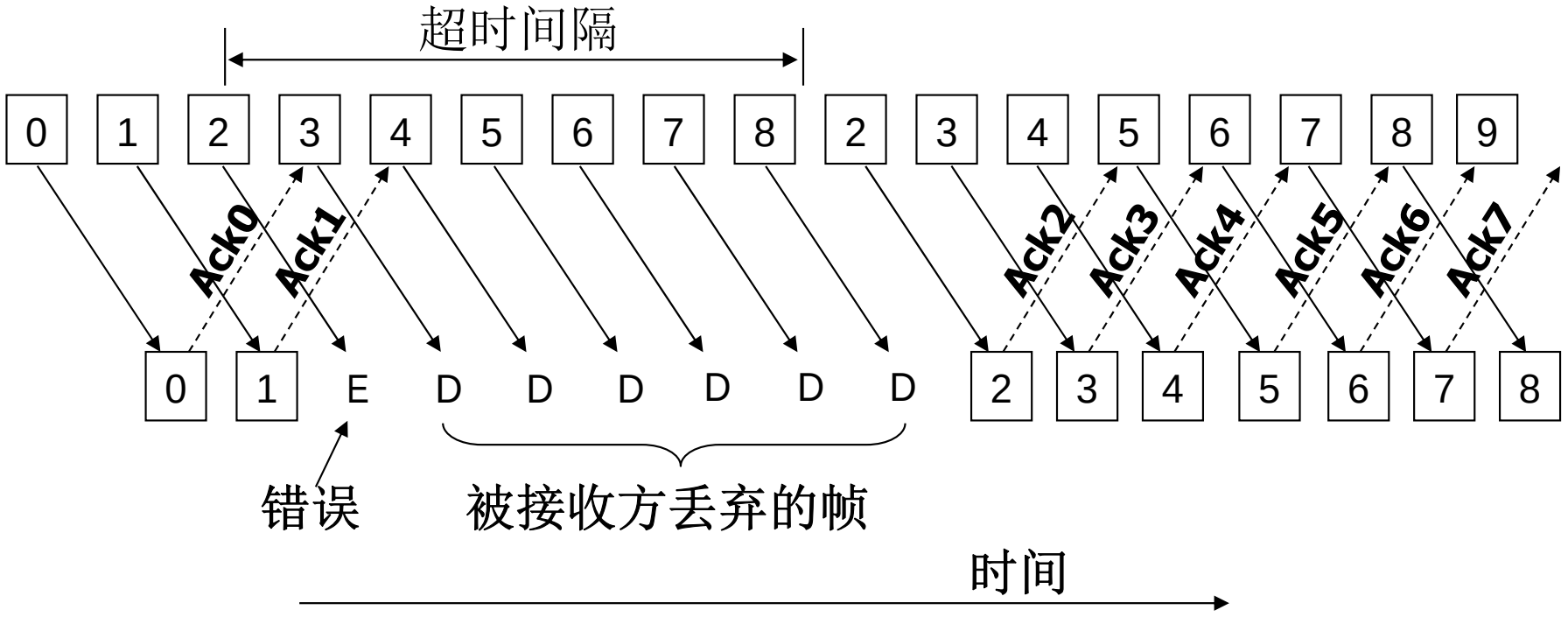
- 一个问题：连续发送的多个帧中，某一个帧出错或丢失，但后续帧已被正确接收，如何处理？
  - 必须遵守的规则：数据链路层需按顺序向网络层上交帧
  - 方案一：出错帧后的帧丢弃，从出错帧开始重新发送 →后退N帧
  - 方案二：出错帧后的帧保留，只重发出错帧 →选择性重传



3.4 滑动窗口协议

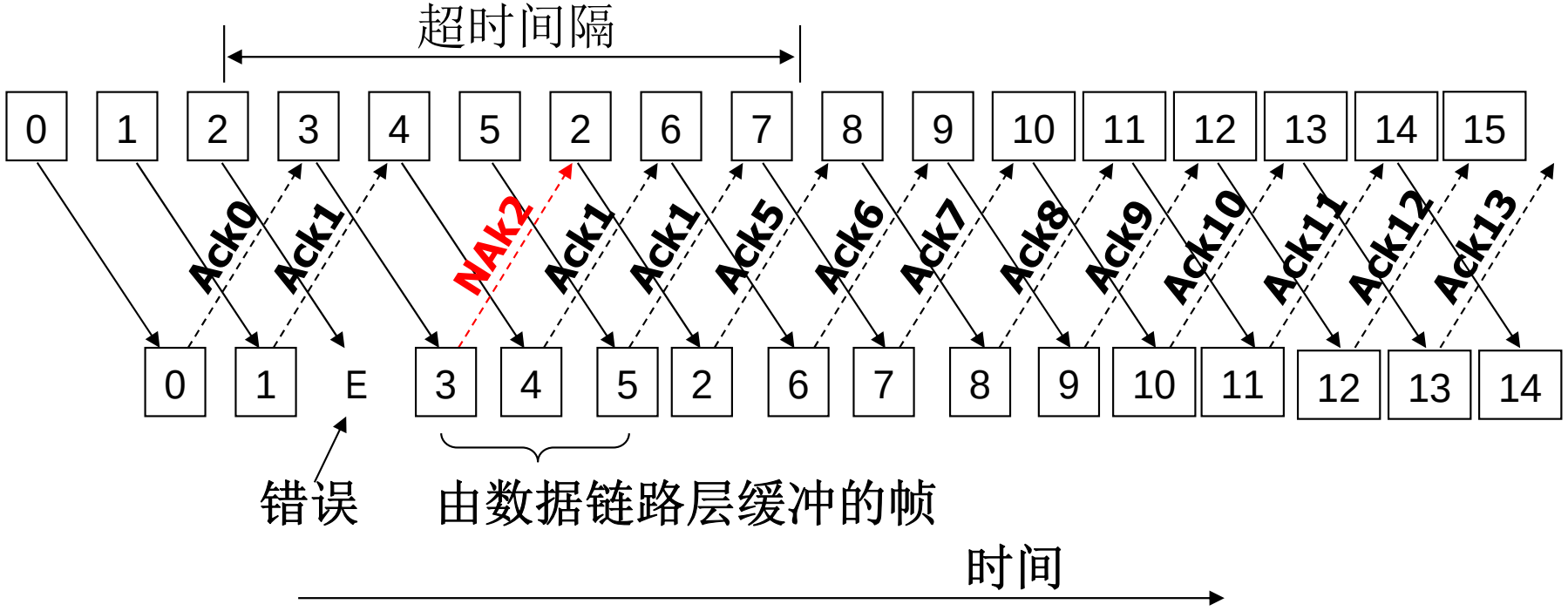
二、后退N帧的滑动窗口协议(Go back N)

- 当某帧出错时，该帧之后的帧全被丢弃，从出错帧开始重新发送
- 实际上此时接收窗口宽度为1



三、选择性重传的滑动窗口协议(selective repeat)

- 当某帧出错时，只选择性地重发该帧，该帧之后发送的帧由接收方数据链路层缓冲，收到重发的出错帧后上交给网络层
- 当接收方检测到出错帧时，发送一个否定的确认(**NAK, Negative Acknowledgement**)
  - 带来的好处：发送方可以尽快重发出错帧，而不必等到超时
  - 问：如果数据帧不是出错而是丢失？如果确认帧出错或丢失？
- 该协议要求接收方能够临时性缓冲接收窗口内的帧
- ***ACKn*表示*n*及*n*以前的帧均已被正确接收**



3.4 滑动窗口协议

2009年的一道考研题：

数据链路层采用后退N帧(GBN)协议，发送方已经发送了编号为0—7的帧。当计时器超时时，若发送方只收到0、2、3号帧的确认，则发送方需要重发的帧数是

A. 2　　B. 3　　C. 4　　D. 5

3.5 点对点协议 PPP

- 早期的数据链路层协议：**HDLC(High-level Data Link Control)**

3.5 点对点协议 PPP

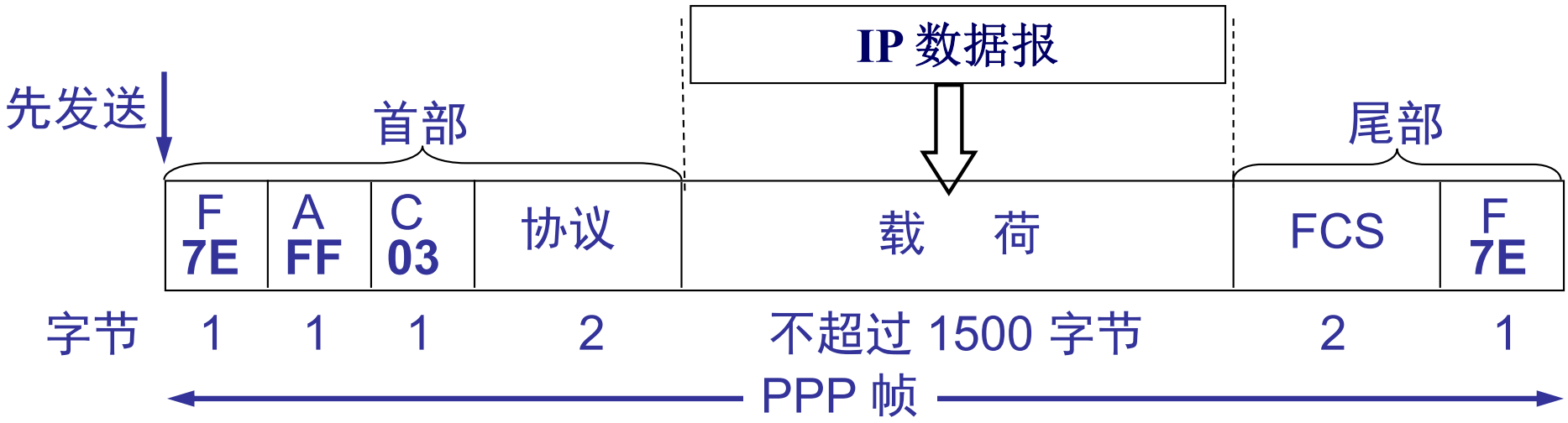
一、PPP 协议的特点

- **PPP (Point-to-Point Protocol)：一种数据链路层协议，广泛应用于点到点链路的数据传输**
  - **拨号上网、光纤传输、…**
- **1992 年制订，1993和 1994 年修订，成为 Internet标准RFC 1661**
- **PPP 协议有三个组成部分**
  - **将 IP 数据报封装到串行链路的方法**
  - **链路控制协议 LCP (Link Control Protocol)**
  - **网络控制协议 NCP (Network Control Protocol)**

3.5 点对点协议 PPP

二、PPP的帧格式

- PPP 是面向字节的，所有的 PPP 帧的长度都是整数字节
- 标志字段F：=0x7E (二进制：01111110)
- 地址字段A：置为 0xFF，实际上不起作用
- 控制字段C：通常置为 0x03
- 协议字段：2字节，用于识别信息字段(又称为载荷,payload)的类型
  - 0x0021：PPP 帧的信息字段是IP 数据报
  - 0xC021：信息字段是 PPP 链路控制数据(LCP)
  - 0x8021：信息字段是网络控制数据(NCP)
- 校验字段FCS：2字节的CRC校验



3.5 点对点协议 PPP

- **PPP的透明传输问题(帧边界识别)**
  - 同步传输(如**SONET/SDH**)：零比特填充
  - 异步传输：字符填充
- 字符填充
  - 信息字段(载荷)中的每一个**0x7E → 0x7D, 0x5E**
  - 信息字段中的每一个**0x7D → 0x7D, 0x5D**
  - 信息字段中的每一个**ASCII** 码控制字符(小于 **0x20** 的字符)前面加入 **0x7D**，且编码改变
    - 例：**0x03 → 0x7D, 0x23**
  - 问：接收端如何处理？

3.5 点对点协议 PPP

注意：只要有连续5个1，无论后面是0或1，都填入0

- 零比特填充
  - 发送端：只要发现有 5 个连续 1，则立即填入一个 0
  - 接收端：对帧中的比特流进行扫描，每当发现 5 个连续1时，就把这 5 个连续 1 后的一个 0 删除

信息字段中出现了和标志字段 F 完全一样的 8 比特组合

0 1 0 0 1 1 1 1 1 1 0 0 0 1 0 1 0

会被误认为是标志字段 F

发送端在 5 个连 1 之后填入 0 比特再发送出去

0 1 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0

发送端填入 0 比特

在接收端把 5 个连 1 之后的 0 比特删除

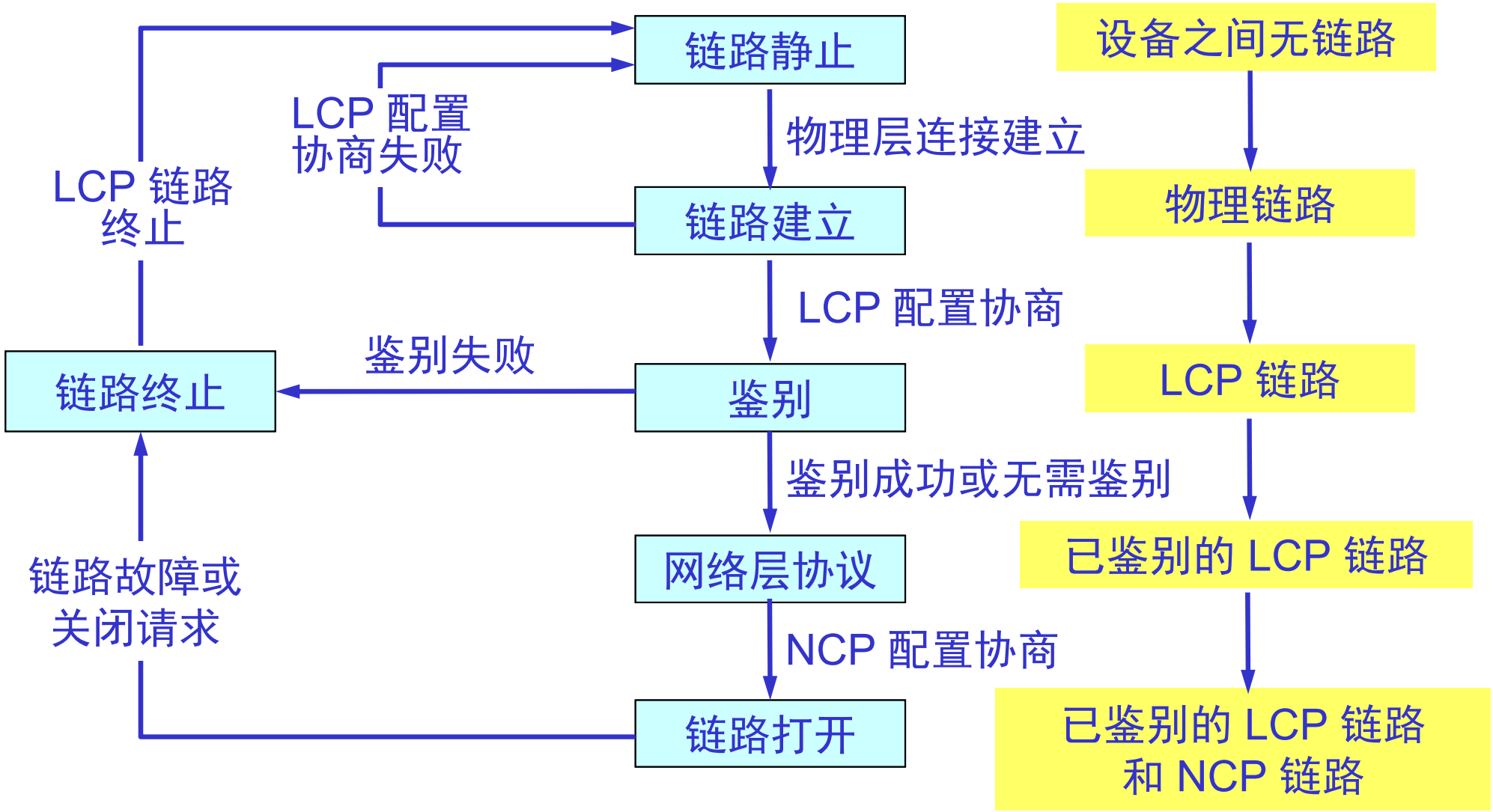
0 1 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0

接收端删除填入的 0 比特

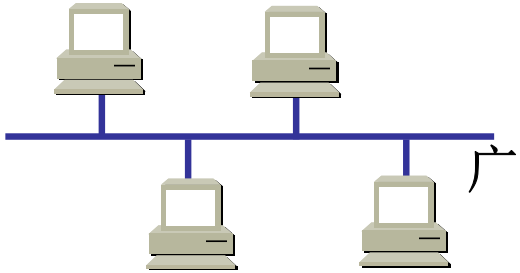
3.5 点对点协议 PPP

- PPP支持两种身份认证协议
  - PAP(Password Authentication Protocol)
  - CHAP(Challenge-Handshake Authentication Protocol)

三、PPP 协议的工作状态



点对点信道



广播信道

注意：还有其他类型的广播式信道

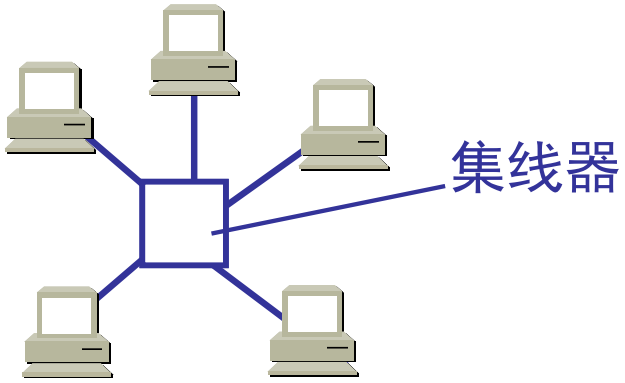
3.6 介质访问控制

media：介质、媒体
access：访问、接入
media access control：介质访问控制、媒体接入控制

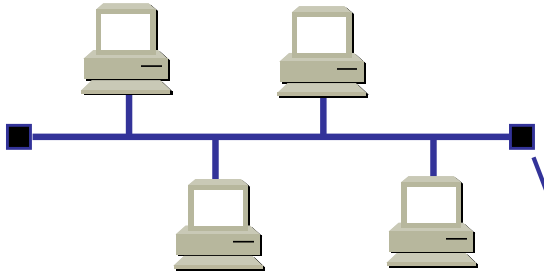
3.6 介质访问控制

一、局域网的数据链路层

- 局域网的主要特点
  - 网络为一个单位所拥有，且地理范围和站点数目均有限

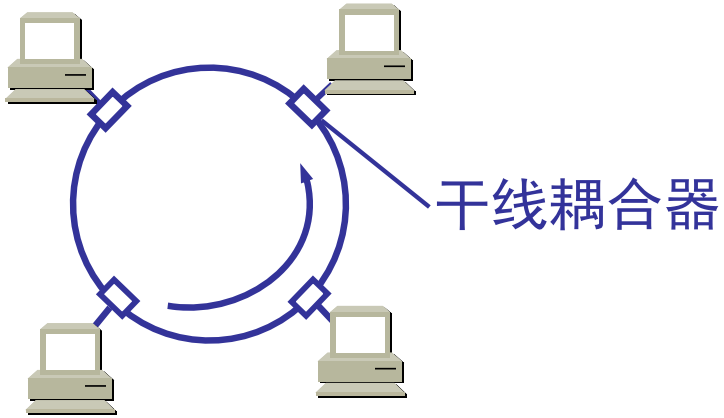


星形网

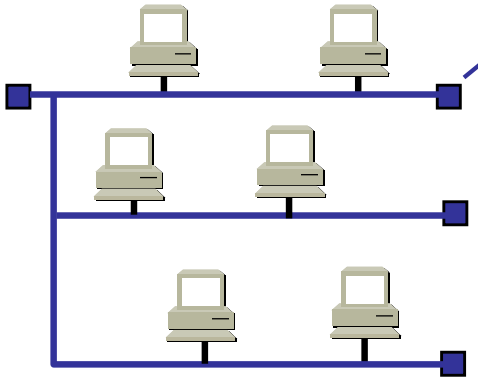


总线网

匹配电阻



环形网



树形网

局域网拓扑
(多为共享介质)

3.6 介质访问控制

- 局域网要解决的重要问题：介质访问控制技术 **(Media Access Control)**

  – 问题缘由：局域网通常使用广播信道

  (1) 静态划分信道
    - 频分复用
    - 时分复用
    - 波分复用
    - 码分复用

  (2) 动态介质访问控制(多点访问)
    - 随机访问：用户可随机发送信息，可能产生碰撞(冲突)
      – 典型协议：**ALOHA**、**CSMA**、**CSMA/CD**、**CSMA/CA**等
    - 受控访问：如多点线路探询(polling)，或轮询 → 局域网中使用较少

3.6 介质访问控制

二、CSMA/CD协议 ← 以太网的核心，非常重要

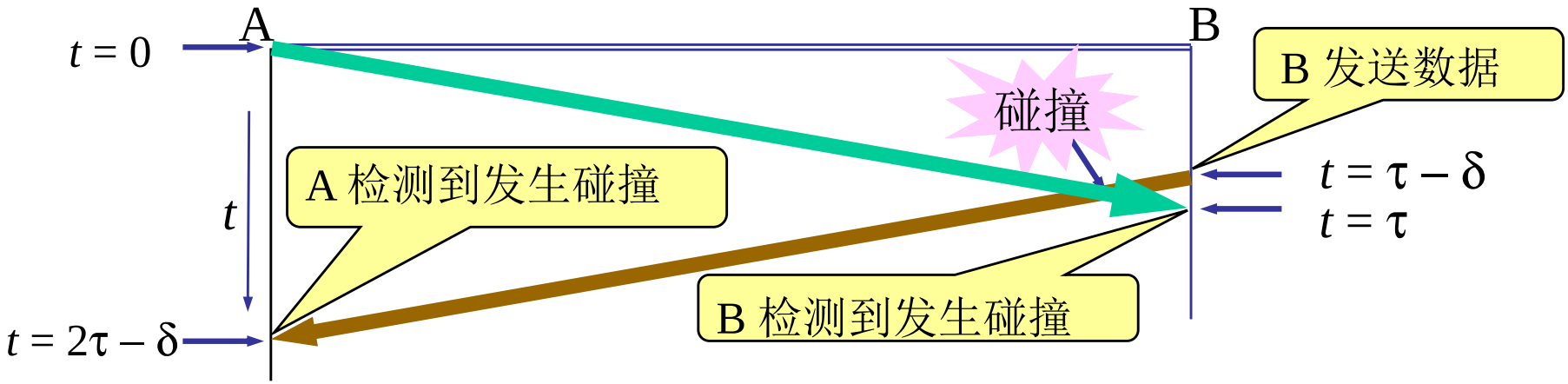
- **Carrier Sense Multiple Access with Collision Detection**
  载波监听多点访问/碰撞检测
  - 注意：collision又被译为“冲突”
  - 是一种随机访问协议
- **CSMA/CD的几个核心概念**
  ① 载波监听(carrier sense)
  ② 碰撞检测(collision detection)
  ③ 碰撞强化
  ④ 碰撞退避

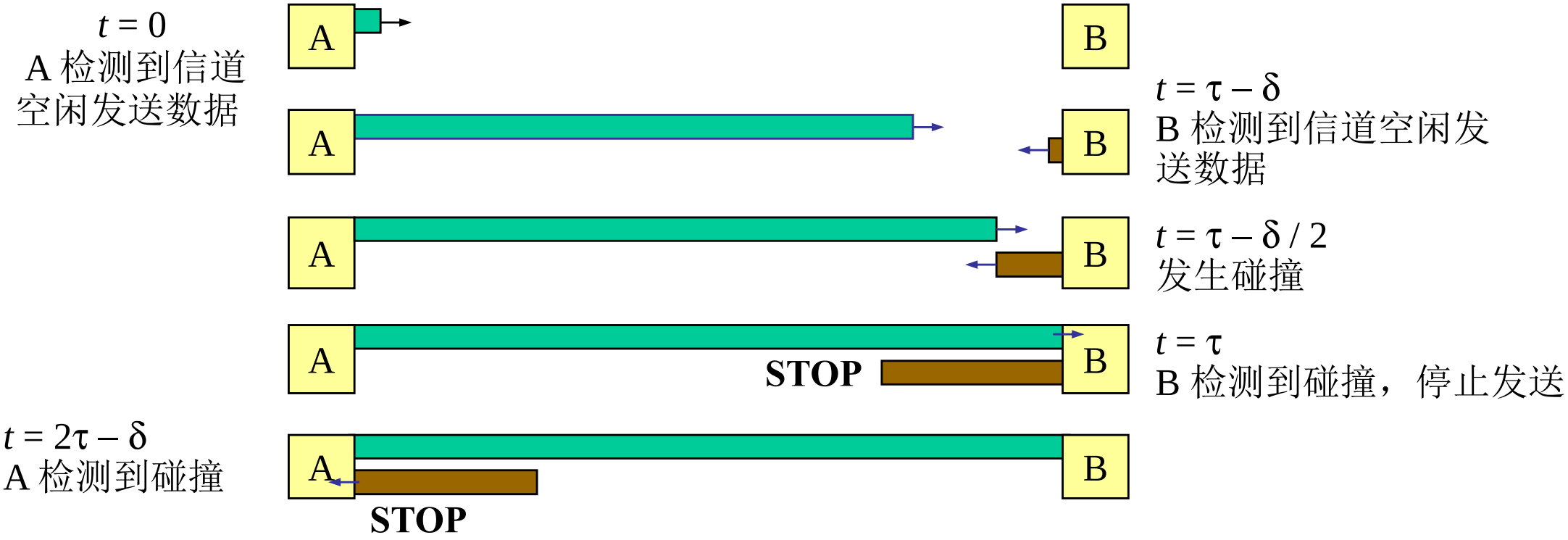
3.6 介质访问控制

二、CSMA/CD协议

- 载波监听
  - 结点在发送数据之前先检测一下总线上是否有其他结点正在发送数据，如有则暂时不要发送数据，以免发生碰撞 → “发送前先听”
- 碰撞检测
  - 结点边发送数据边检测信道上是否发生了碰撞(监听总线上传输的信号) → “边发送边听”
  - 由于线路的传播时延，单纯靠载波监听并不能完全避免碰撞 → 碰撞仍有可能发生
  - 在发生碰撞时，两个或更多的信号在总线上相互叠加，导致无法识别

- 设单程端-端传播时延为τ
- 1km长的同轴电缆上，电磁波的端-端传播时延约为5us
- 站点A在时刻t检测到总线空闲后发送数据，在数据到达站点B之前，B也检测到总线空闲并发送数据，从而引发碰撞





3.6 介质访问控制

- 碰撞强化
  - 发送方检测到碰撞后，立即停止发送，并发送**32**或**48bit**的人为干扰信号**(jamming signal)**，以便让所有用户都知道已经发生了碰撞
- 碰撞退避
  - 碰撞后，结点等待一段时间，重新开始载波检测和发送操作
  - 为避免退避后再次碰撞，冲突各方的等待时间应各不相同
  - 以太网采用截断二进制指数退避算法**(truncated binary exponential backoff)**
    - 退避时间：$T = 2\tau \times$ *倍数*
    - 倍数：在**0, 1, …, 2^k-1**中取随机数，**k = min(**重传次数**, 10)**
    - 重传次数超过**16**后，丢弃该帧，并向上层报告

问：退避算法的效果是怎样的？

3.6 介质访问控制

2010年的一道考研题：

某局域网采用CSMA/CD协议实现介质访问控制，数据传输率为100M/S，主机甲和主机乙的距离为2km，信号传播速度是200000km/s。请回答下列问题，并给出计算过程。

(1) 若主机甲和主机乙发送数据时发生冲突，则从开始发送数据时刻起，到两台主机均检测到冲突时刻为止，最短经过多长时间？最长经过多长时间？(假设主机甲和主机已发送数据时，其它主机不发送数据)

求解：

甲乙间单向传输时延$\tau = 2/200000 = 10us$

最长：甲发送的帧即将到达乙时，乙开始发送数据造成冲突，甲检测到该冲突的时间 $= 2\tau = 20us$

最短：甲乙同时开始发送信息，在总线中间点发生冲突，冲突信号向甲乙方向扩散，总时间长度 $= \tau = 10us$

3.6 介质访问控制

- 争用期
  - 一个站点开始发送数据后，最多经过时间 **2τ**(两倍的端-端时延)就可知道是否发生了碰撞
  - 以太网的端到端往返时延 **2τ**称为争用期，或碰撞窗口
  - 如果经过争用期还没有检测到碰撞，就可以肯定这次发送不会发生碰撞
- 以太网的争用期
  - 以太网的争用期长度：**51.2 us**
  - 对于 **10 Mb/s** 以太网，在争用期内可发送**512 bit**，即 **64** 字节
  - 在发送数据时，若前 **64** 字节未发生碰撞，就不会发生碰撞
  - 据此规定以太网帧长≥ **64** 字节，长度小于 **64** 字节的帧为无效帧

3.6 介质访问控制

2009年的一道考研题：

在一个采用CSMA/CD协议的网络中，传输介质是一根完整的电缆，传输速率为1Gbps，电缆中的信号传播速度是200000km/s。若最小数据帧长度减少800比特，则最远的两个站点之间的距离至少需要：

A. 增加160m　　B. 增加80m　　C. 减少160m　　D. 减少80m √

求解：

设总线长度为L km

单向传输时延τ = L/200000 秒；2 τ=L/10^5秒

传输速率1Gbps时，800bit的发送时间=800/10^9秒

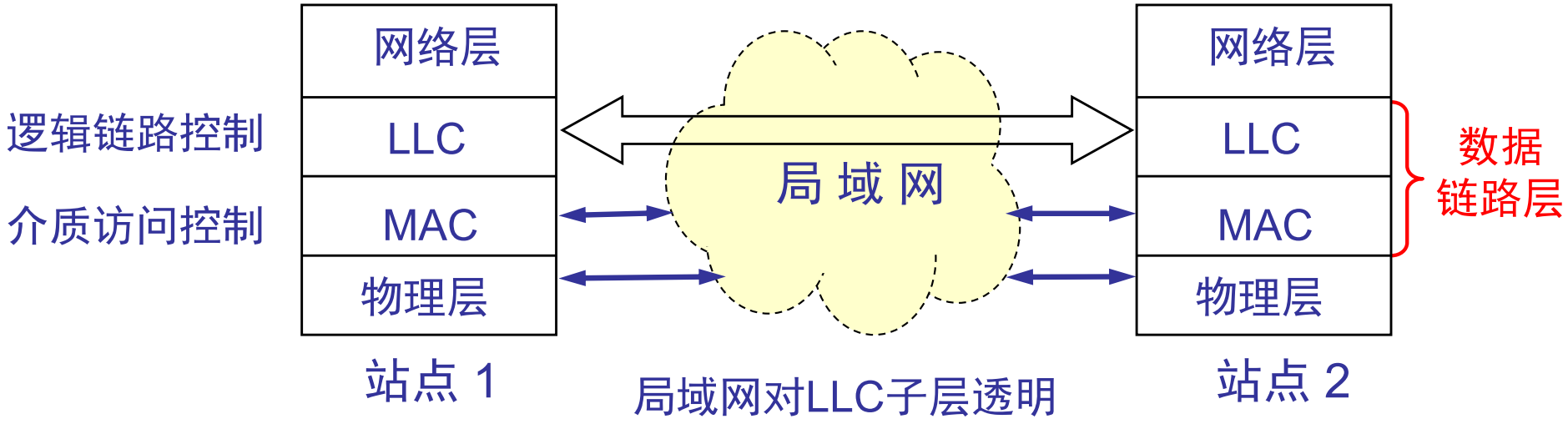
总线应减少的长度=(800/10^9)*10^5 = 0.08km = 80m

3.6 介质访问控制

- **CSMA/CD协议的优缺点讨论**
  - 网络负载较轻时效率高
  - 硬软件实现简单、灵活
  - 网络负载较重时，碰撞发生概率增大，网络效率较低
  - 由于存在多次冲突的可能，数据从发送方到达接收方的时间没有保证 → 实时性较差

三、局域网技术标准(1/2)

- **IEEE802标准将局域网的数据链路层分为两个子层**
  - **LLC(Logic Link Control)逻辑链路控制子层**
  - **MAC(Media Access Control)介质访问控制子层**



3.6 介质访问控制

三、局域网技术标准(2/2)

- **IEEE802**系列局域网标准
  - **IEEE 802.1a** 综述与体系结构
  - **IEEE 802.1b** 寻址、互联、管理
  - **IEEE 802.2** 逻辑链路控制(LLC)
  - **IEEE 802.3** CSMA/CD介质访问控制(MAC)与物理层技术规范
    - **IEEE 802.3u** 快速以太网(Fast Ethernet)
    - **IEEE 802.3z** 千兆以太网(Gigabit Ethernet)
  - **IEEE 802.4** Token Bus介质访问控制与物理层技术规范
  - **IEEE 802.5** Token Ring介质访问控制与物理层技术规范
  - **IEEE 802.11** 无线局域网介质访问控制与物理层技术规范

注：令牌环网(token ring)和令牌总线网(token bus)使用一种称为令牌(token)的控制标志，当网络中结点持有该令牌时，才能够发送数据

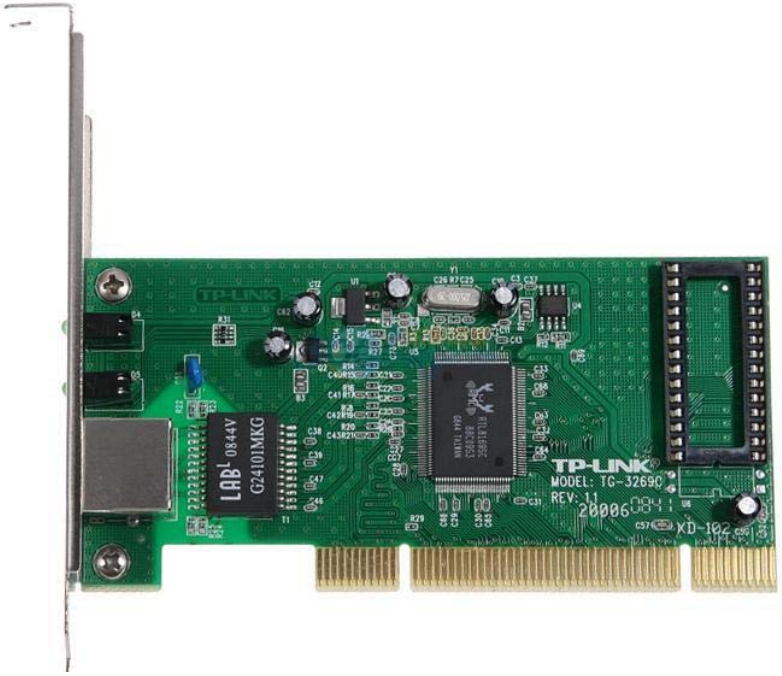
3.7 以太网

3.7 以太网

一、以太网简介

- **Ether：以太**
- **1975年，Xerox公司Palo Alto研究中心(PARC)研制**
- **1980年，DEC、Intel和Xerox公司联合推出以太网标准DIX v1**
- **1982年，DIX v2**
- **1983年，IEEE802.3标准**


TP-LINK
TL-HP8MU


TP-LINK
MODEL: TG-3269C
REV: 1.0
XD-102


1000
10
FDX
100
TP-LINK

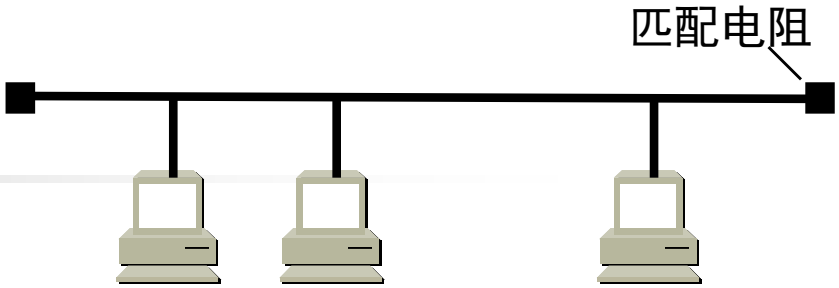
以太网集线器

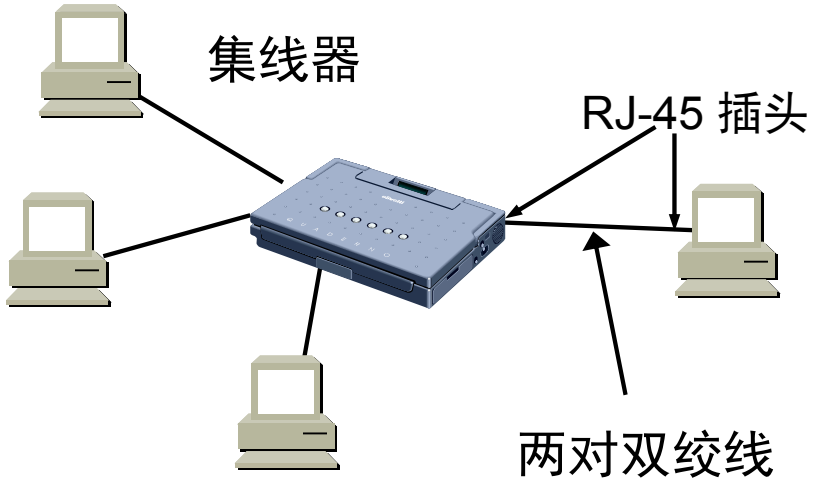
•以太网卡

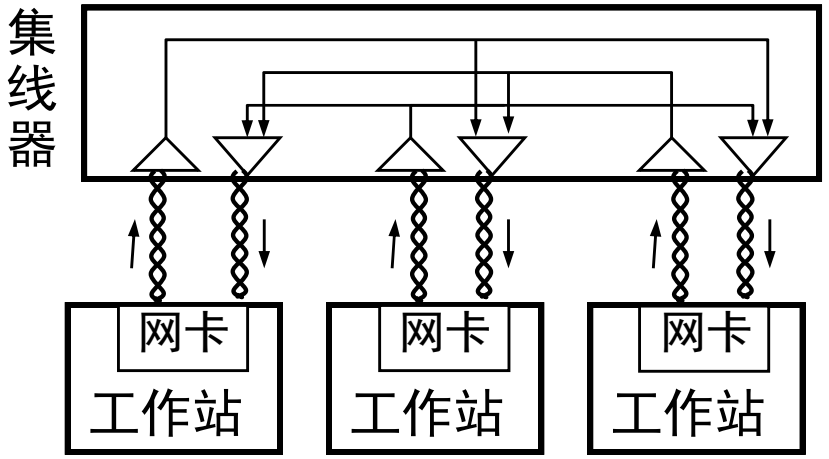
3.7 以太网

二、以太网物理层

- 最初的以太网为总线结构，采用**50Ω**同轴电缆，传输速率**10Mbps**
  - 缺点：总线上单点故障会导致全网瘫痪，网络中结点数较多时可靠性较差且维护困难；同轴电缆成本较高
- 后发展为采用更便宜和灵活的非屏蔽双绞线，使用集线器**(HUB)**连接各个结点，物理上呈星形结构
  - 此种技术称为**10Base-T**
    - **10**：**10Mbps**
    - **Base**：基带传输
    - **T**：双绞线**(Twisted pair)**
  - 使用集线器的以太网在逻辑上仍是一个总线网
  - 集线器很像一个多接口的转发器，工作在物理层



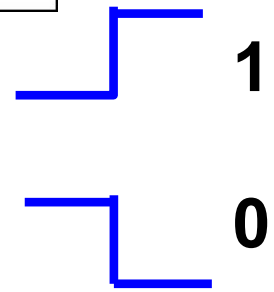


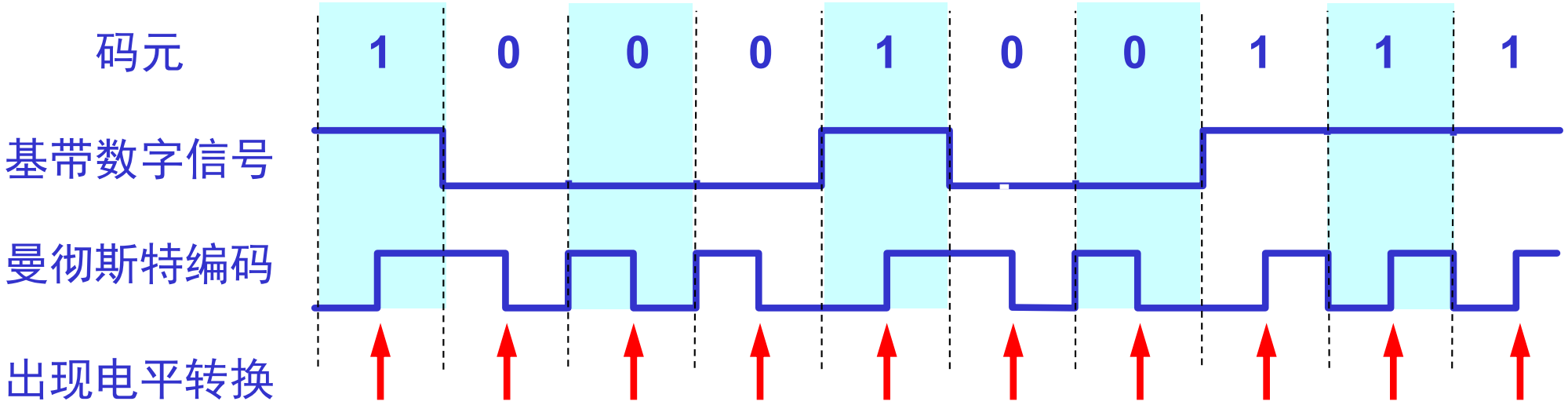


3.7 以太网

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

- 采用曼彻斯特(Manchester)编码
  - 用电平的跳变表示0或1，即每个码元都有电平跳变





三、以太网的MAC层

- 以太网采用CSMA/CD介质访问控制协议
- MAC地址
  - 结点发送数据时，以太网总线结构，总线上的所有结点都能收到帧
  - 按照IEEE802.3标准，给每个结点分配唯一的MAC地址
  - MAC地址为48 bit，高24bit为厂商标识符，低24位由厂商自行分配，须保证每个网络接口具有全球唯一的MAC地址

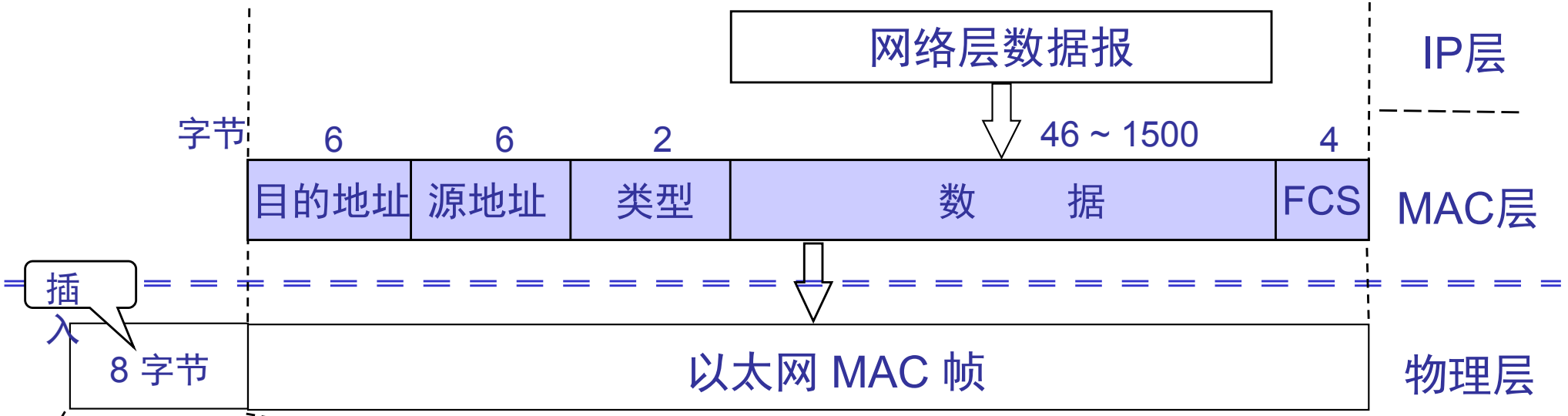
示例：`Physical Address. . . . . : 00-26-2D-FF-A8-0E`

- 适配器每收到一个帧就检查帧中的目的MAC地址，如果是发往本站的帧则进行处理，否则丢弃
  - “发往本站的帧”包括以下三种帧：
    - 单播(unicast)帧(一对一)
    - 广播(broadcast)帧(一对全体)，MAC地址为全1表示广播
    - 组播/多播(multicast)帧(一对多)

3.7 以太网

以太网帧最大长度=？最小长度=？

- 以太网的帧格式(5个字段)
  - 目的地址、源地址：各6字节的MAC地址
  - 类型：2字节，标明上层协议类型，例：0x0800表示IP包
  - 数据：网络层数据报，长度46～1500字节
  - FCS：4字节，帧校验序列，采用CRC校验



- 每个帧在发送时前面插入8字节，其中前7字节用于比特同步，后1字节为帧起始标志
- 注意：以太网采用曼彻斯特编码，每个比特均有电平跳变，无电平跳变意味着总线空闲

3.7 以太网

四、高速以太网(1/3)

- 快速以太网(Fast Ethernet)
  - IEEE802.3u标准
  - 传输速率100Mb/s
  - 仍使用 IEEE 802.3 CSMA/CD 协议(全双工模式时不用)
  - MAC 帧格式仍然是 802.3 标准规定的
  - 最短帧长不变，但将一个网段的最大电缆长度减小到100m
  - 线缆标准
    - 100BASE-TX：使用2 对UTP 5 类线或屏蔽双绞线 STP
    - 100BASE-FX：使用1对光纤
    - 100BASE-T4：使用4对 UTP3类线或5类线

四、高速以太网(2/3)

- 千兆以太网(Gigabit Ethernet)
  - IEEE802.3z标准
  - 传输速率1Gb/s
  - 仍使用 IEEE 802.3 CSMA/CD 协议(全双工模式时不用)
  - 线缆标准
    - 1000BASE-X：基于光纤
      - 1000BASE-SX　SX表示短波长
      - 1000BASE-LX　LX表示长波长
      - 1000BASE-CX　CX表示铜线
    - 1000BASE-T：使用 4对5类双绞线UTP

四、高速以太网(3/3)

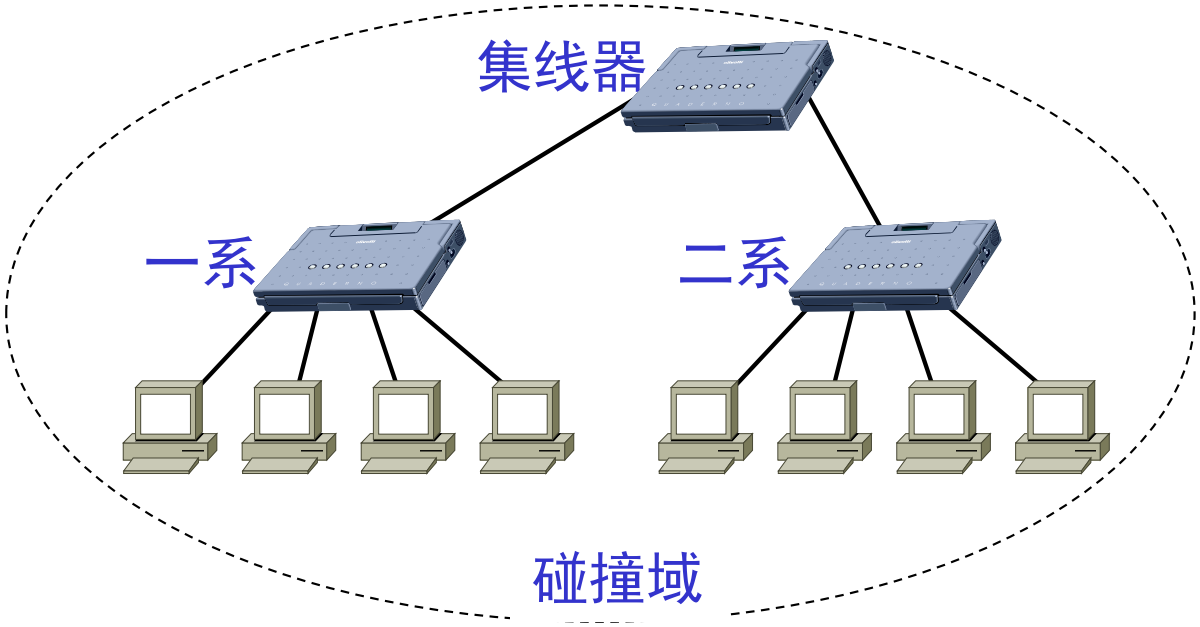
- 万兆以太网(**10Gbps以太网**)
  - **IEEE802.3ae标准**
  - 传输速率**10Gb/s**
  - 帧格式不变
  - 只使用光纤
  - 只工作在全双工模式，不存在争用，因此不用**CSMA/CD**
- 更高速的以太网
  - **IEEE802.3ba: 40GE，传输速率40Gb/s**
  - **IEEE802.3bm: 100GE，传输速率100Gb/s**

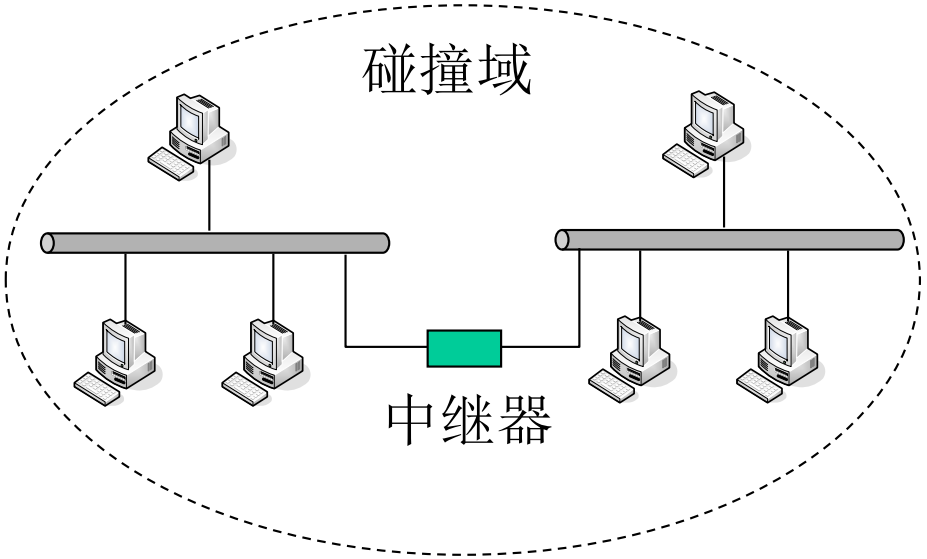
3.8 局域网互连

3.8 局域网互连

一、在物理层扩展局域网

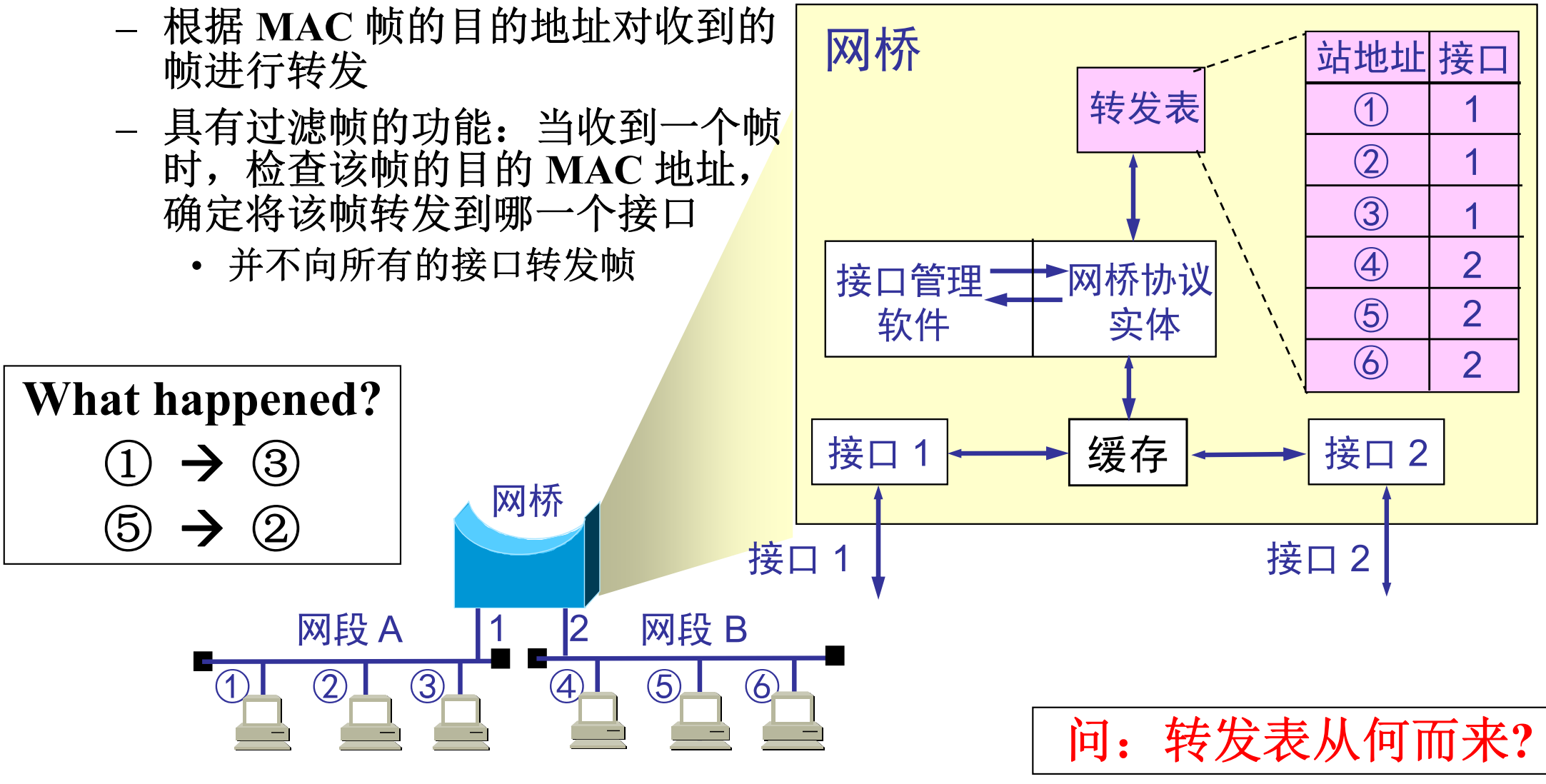
- 使用中继器(repeater)或集线器(HUB)可实现局域网在物理层的互连
  - 中继器、集线器工作在网络的哪一层？
- 优点：可以方便地实现网络的扩展，且成本较低
- 缺点：碰撞域增大，碰撞发生概率增大，可能影响网络性能





二、在数据链路层扩展局域网

- 使用网桥(bridge)可实现局域网在数据链路层的互连
  - 网桥工作在网络的哪一层?
- 网桥的基本工作原理
  - 根据 MAC 帧的目的地址对收到的帧进行转发
  - 具有过滤帧的功能：当收到一个帧时，检查该帧的目的 MAC 地址，确定将该帧转发到哪一个接口
    - 并不向所有的接口转发帧

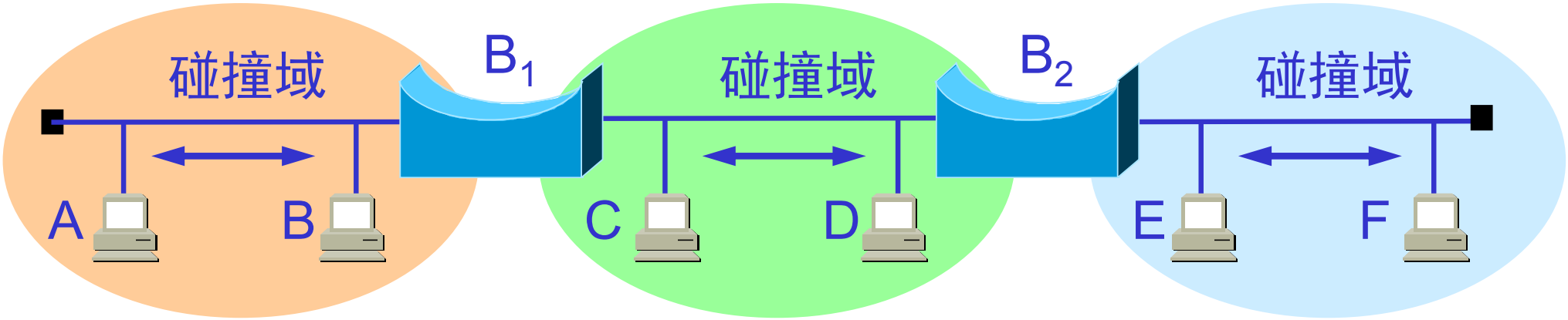


问：转发表从何而来?

3.8 局域网互连

- 用网桥实现局域网互连的优点
  - 过滤通信量、增大吞吐量
    - 各网段是独立的碰撞域
  - 扩大了物理范围
  - 提高了可靠性
  - 可互连不同物理层、不同 MAC 子层和不同速率的局域网

- 用网桥实现局域网互连的局限性
  - 存储转发增加了时延
  - 在MAC 子层并没有流量控制功能
  - 网桥只适合于用户数不太多(不超过几百个)和通信量不太大的局域网
    - 广播风暴：因某些站点频繁发送广播帧产生网络拥塞　　(问：何谓广播帧？)

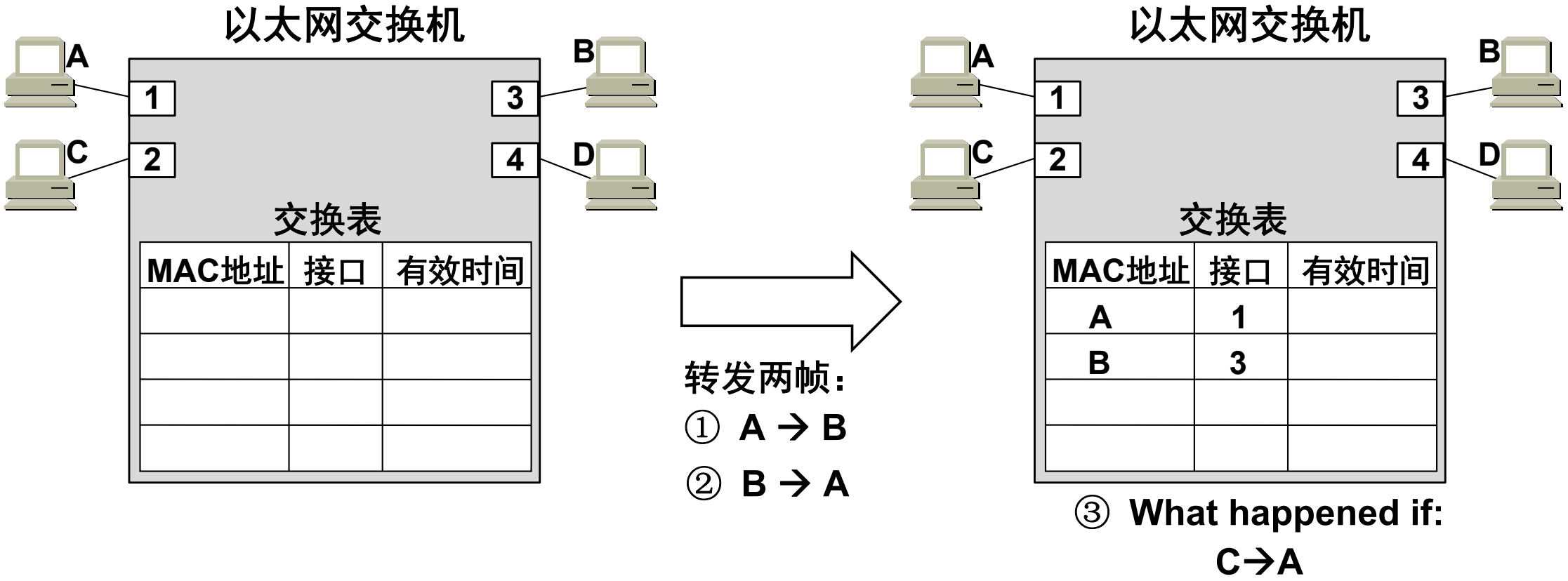


3.8 局域网互连

- 多接口网桥—交换机
  - 交换机(switch)早期称为交换式集线器(switching HUB)
  - 通常有十几到几十个网络接口，每个接口可连接主机或其他交换机
  - 其内部工作原理为网桥，每一个网络接口被视为一个网段，因此交换机是一种多接口网桥
    - 普通的交换机工作在哪一层？ (注意：有三层交换机)
  - 与集线器相比
    - 交换机工作在数据链路层，每个接口为一个网段(碰撞域)，可以大幅提高网络性能
    - 集线器由所有接口共享传输介质的带宽，而交换机为每个接口独享带宽
  - 交换机按照自学习算法(self-learning)建立转发表
    - 原理：若从 A 发出的帧从接口 x 进入了某网桥，那么从这个接口出发沿相反方向一定可把一个帧传送到 A

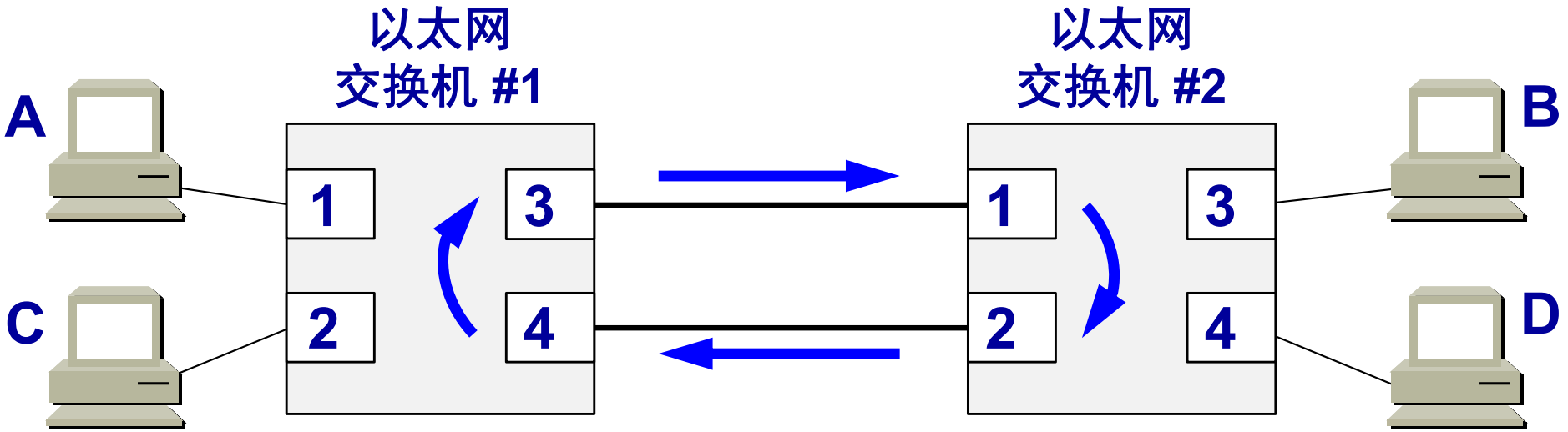
交换机的自学习过程：

- 初始时转发表为空，通过自学习逐步建立
- 交换机收到一个帧后的处理过程
  - 首先进行自学习，检查帧中源地址在交换表中是否存在，如不存在，则在交换表添加一项，记下源地址和进入交换机的接口
  - 然后转发帧，根据收到的帧中的目的地址在交换表中查找
    - 如果找到，则对应的接口即为转发接口
    - 如果未找到，则向除进入接口外的所有其他接口转发



3.8 局域网互连

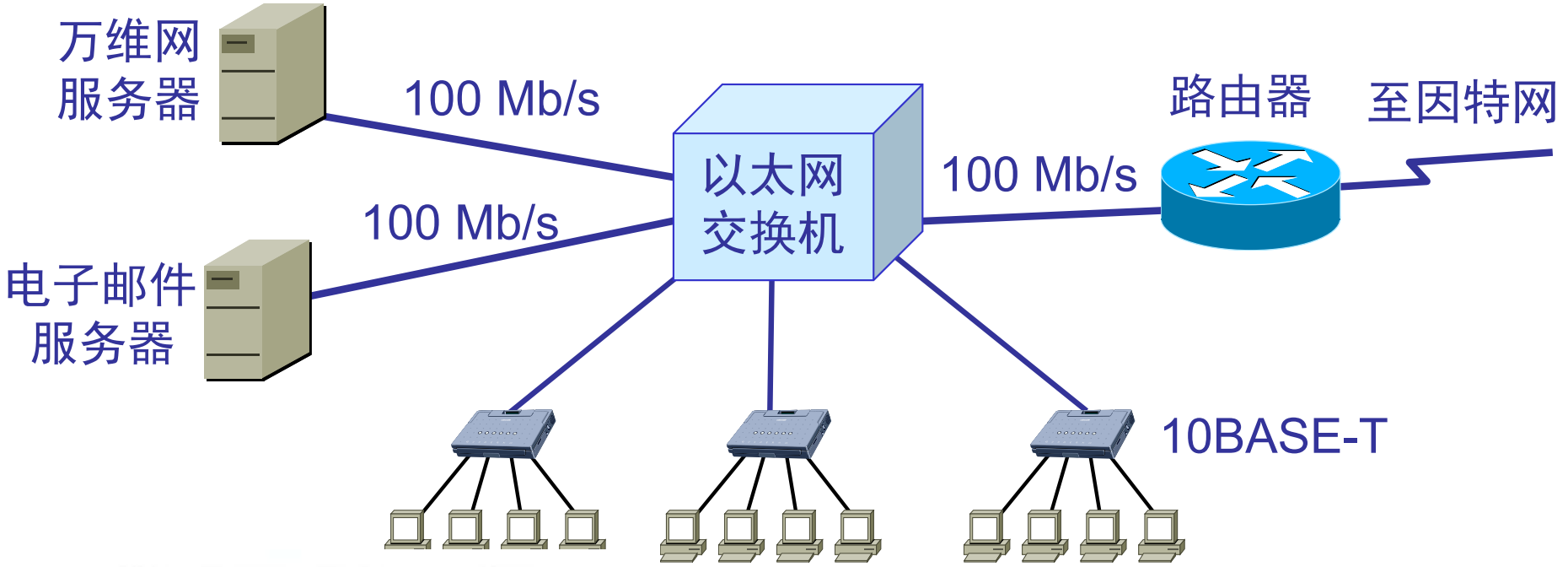
- 透明网桥还使用生成树算法(spanning tree)
  - 用途：当多个局域网互连形成环路时，避免帧无休止转发
  - 方法：建立生成树
    - 互连在一起的网桥彼此通信后，能找出网络拓扑的一个子集。在该子集中，整个连通的网络中不存在回路，即在任何两个站之间只有一条路径
    - 考虑到网络拓扑动态变化，生成树需定期更新



两台交换机之间使用两条链路互连，形成了环路，如不加处理，帧可能无休止转发

- 假设初始时A→B发送一帧

交换机连网举例



企业级交换机

6个插槽，最多240个端口，四层交换

TP-Link TL-SL3428

24个10/100 + 4个 10/100/1000端口

3.8 局域网互连

- 虚拟局域网(VLAN)
  - VLAN(Virtual LAN)，在现有局域网基础上，通过将网络站点分组，构成若干逻辑上独立的虚拟局域网
  - 注意：帧不会在两个VLAN之间自动转发，包括广播帧
  - VLAN的用途：便于管理、控制广播风暴、安全性、…