




第5章：链路层和局域网

导引：

- ❖ 网络层解决了一个网络如何到达另外一个网络的路由问题
- ❖ 在一个网络内部如何由一个节点（主机或者路由器）到达另外一个相邻节点
 - 链路层的**点到点**传输层功能



第5章：链路层和局域网

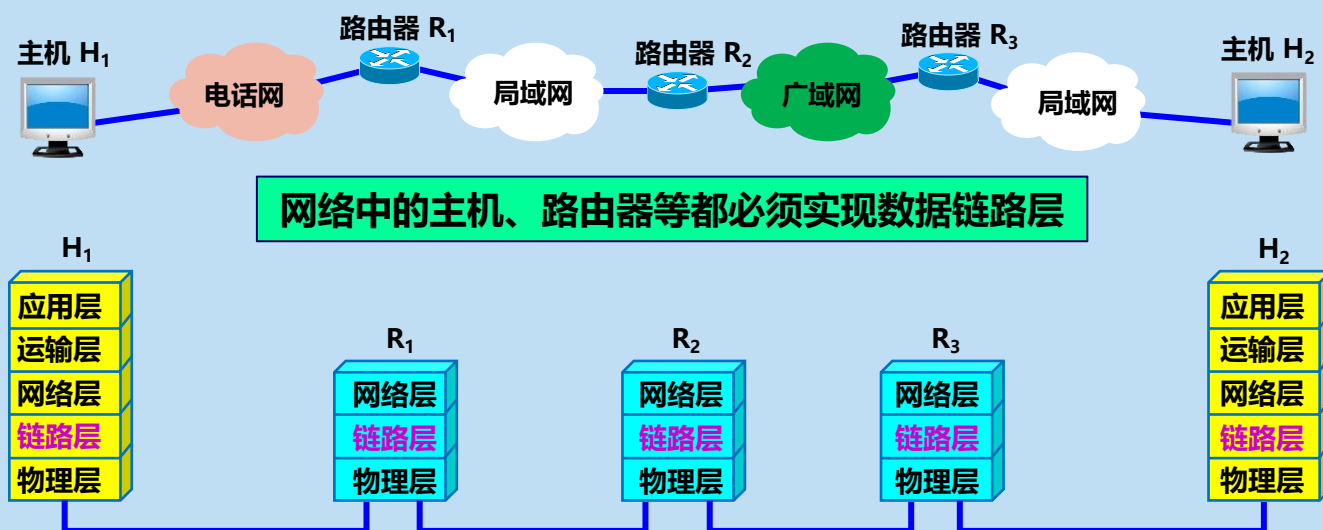
目标：

❖ 理解数据链路层服务的原理：

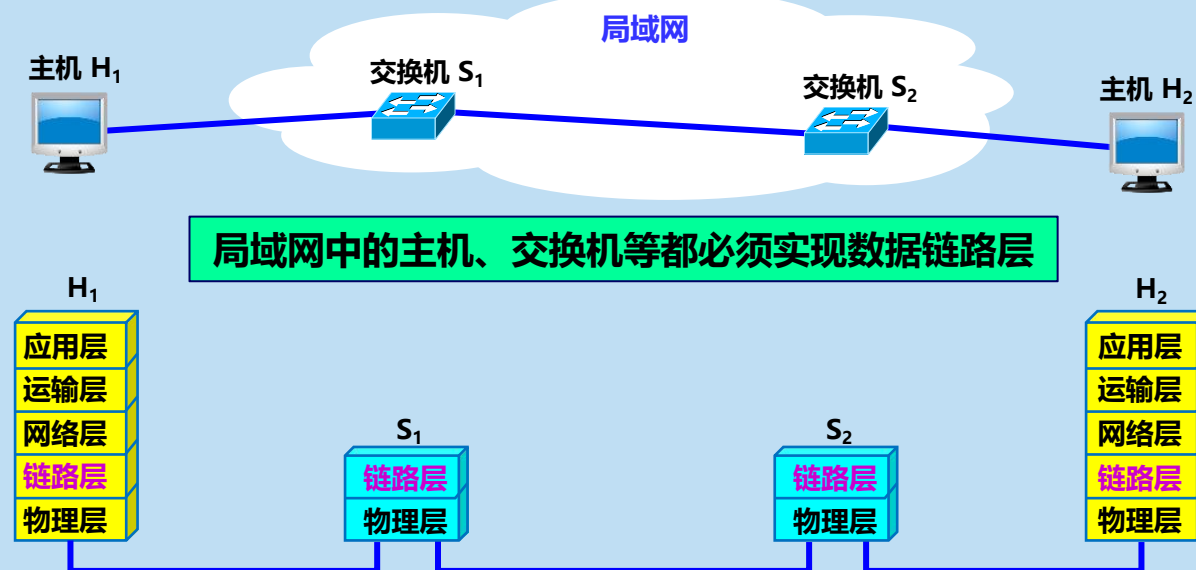
- 检错和纠错
- 共享广播信道：多点接入（多路访问）
- 链路层寻址
- LAN:以太网、WLAN、VLANs
- 可靠数据传输，流控制

❖ 实例和各种链路层技术的实现

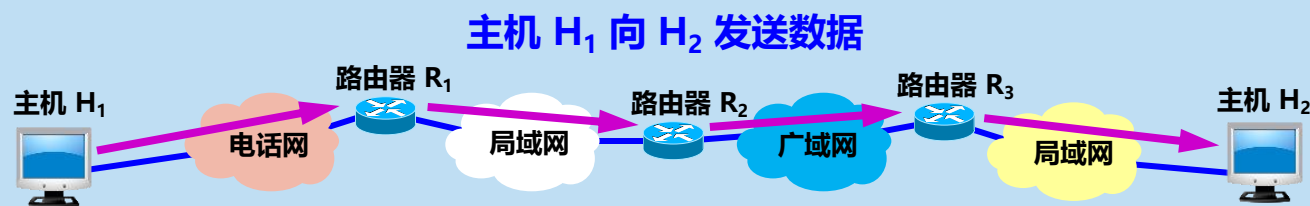
数据链路层的地位



数据链路层的地位

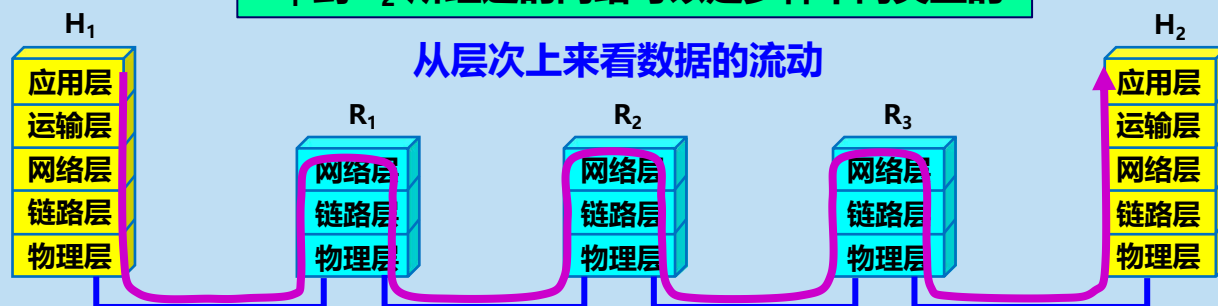


数据链路层的地位



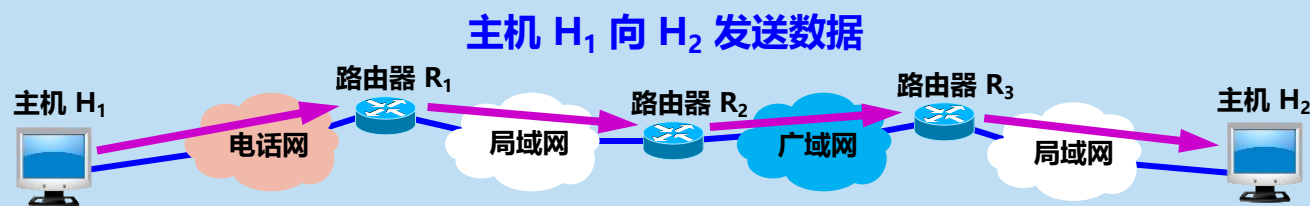
H_1 到 H_2 所经过的网络可以是多种不同类型的

从层次上来看数据的流动

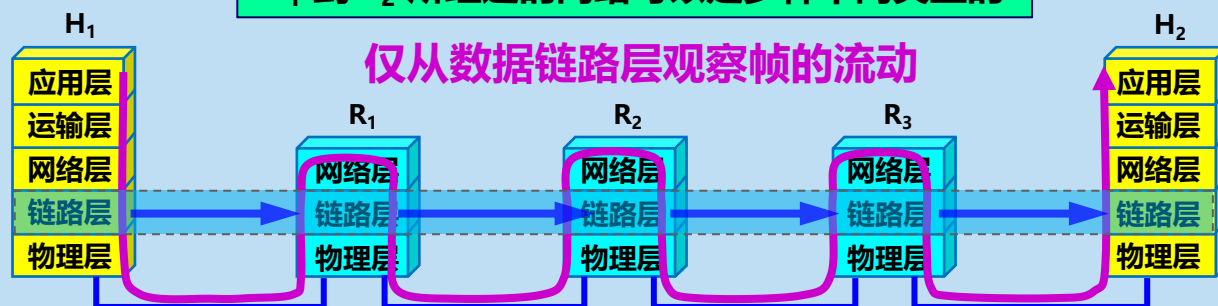


数据链路层的地位

数据链路层的地位

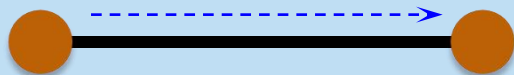


H₁ 到 H₂ 所经过的网络可以是多种不同类型的



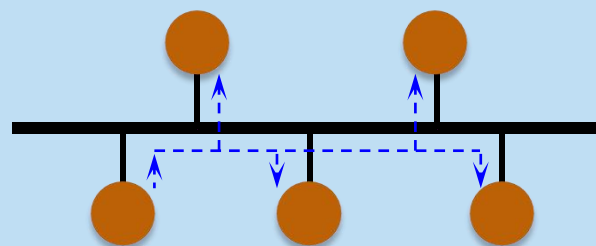
注意：不同的链路层可能采用不同的数据链路层协议

数据链路层信道类型



(a) 点对点信道

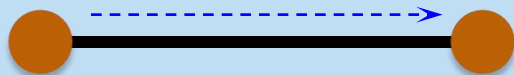
- 使用一对一的**点对点**通信方式。



(b) 广播信道

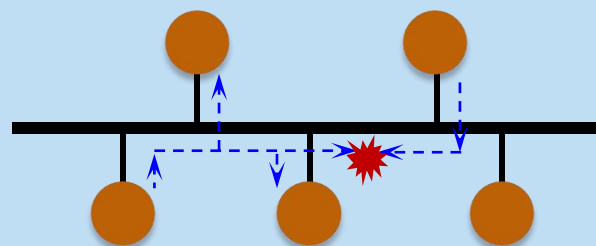
- 使用一对多的**广播**通信方式。
- 必须使用专用的**共享信道协议**来协调这些主机的数据发送。

数据链路层信道类型



(a) 点对点信道

- 使用一对一的**点对点**通信方式。



(b) 广播信道

- 使用一对多的**广播**通信方式。
- 必须使用专用的**共享信道协议**来协调这些主机的数据发送。

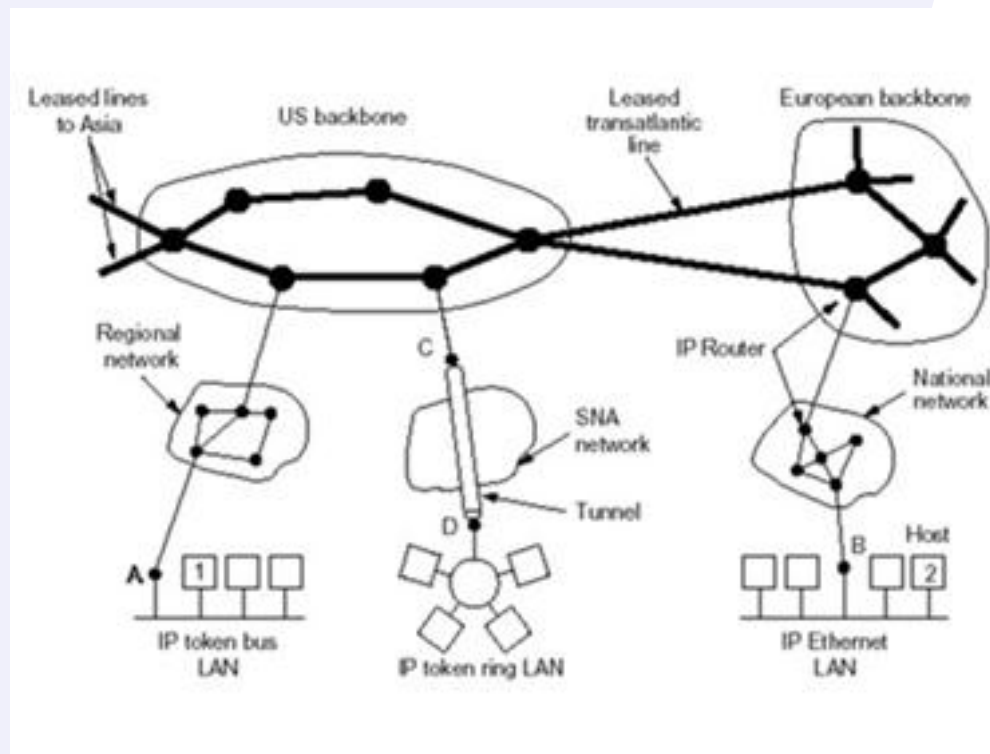
网络节点的连接方式

Q: 一个子网中的若干节点是如何连接到一起的

❖ 点到点连接

❖ 多点连接:

- 共享型介质
- 通过网络交换机



数据链路层和局域网

❑ WAN:网络形式采用点到点链路

- 带宽大、距离远（延迟大）
 - 带宽延迟积大
- 如果采用多点连接方式
 - 竞争方式：一旦冲突代价大
 - 令牌等协调方式：在其中协调节点的发送代价大

❑ 点到点链路的链路层服务实现非常简单，封装和解封

❑ LAN一般采用多点连接方式

- 连接节点非常方便
- 接到共享型介质上（或网络交换机），就可以连接所有其他节点

❑ 多点连接方式网络的链路层功能实现相当复杂

- 多点接入：协调各节点对共享性介质的访问和使用
- 竞争方式：冲突之后的协调；
- 令牌方式：令牌产生，占有和释放等

The background of the slide features a detailed map of East Asia, including parts of China, Korea, and Japan. The map is rendered in a light blue and white color scheme, with a dark blue overlay at the top. The title '内容提纲' is centered in the dark blue area.

内容提纲

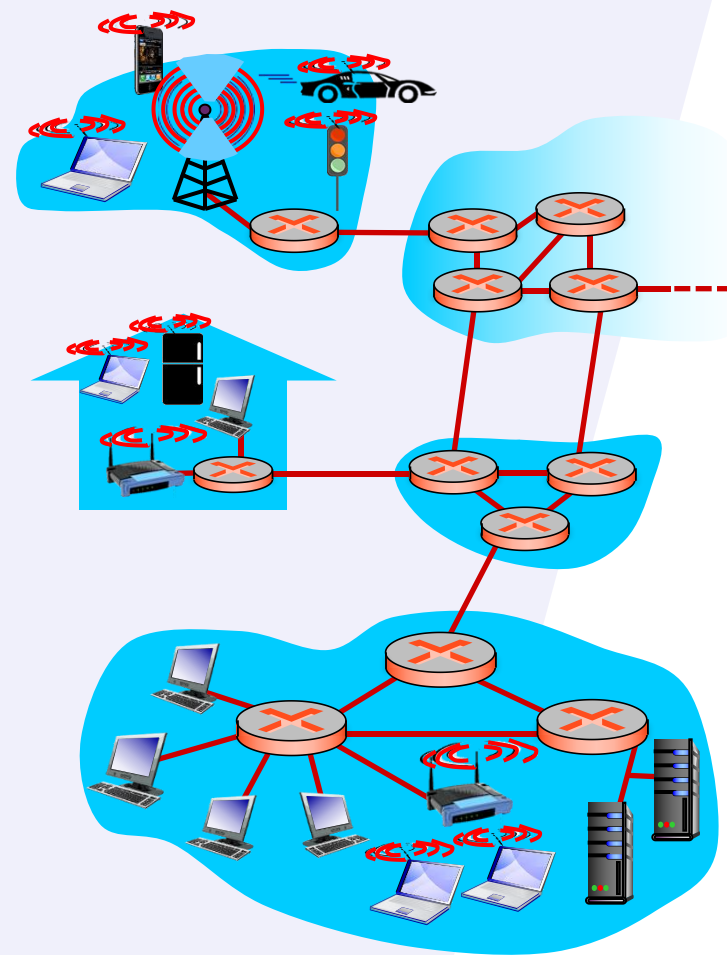
- ❖ 5.1 引论和服务
- ❖ 5.2 差错检测和纠正
- ❖ 5.3 多点访问协议
- ❖ 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches

5.1 链路层: 导论

一些术语:

- ❖ 主机和路由器是节点（网桥和交换机也是）：**nodes**
- ❖ 沿着通信路径,连接个相邻节点通信信道的是**链路: links**
 - 有线链路
 - 无线链路
 - 局域网，共享性链路
- ❖ 第二层协议数据单元帧**frame**，封装数据报

数据链路层负责从一个节点通过链路将(帧中的) 数据报发送到**相邻的物理节点**(一个子网内部的2节点)



链路层: 上下文

❖ 数据报(分组) 在不同的链路上以不同的链路协议传送:

- 第一跳链路: 以太网
- 中间链路: 帧中继链路
- 最后一跳802.11:

❖ 不同的链路协议提供不同的服务

- e.g., 比如在链路层上提供(或没有)可靠数据传送

传输类比

- ❖ 从Princeton到Lausanne
 - 轿车: Princeton to JFK
 - 飞机: JFK to Geneva
 - 火车: Geneva to Lausanne
- ❖ 旅行者=数据报 **datagram**
- ❖ 交通段=通信链路 **communication link**
- ❖ 交通模式=链路层协议: **数据链路层和局域网protocol**
- ❖ 票务代理=路由算法 **routing algorithm**

链路层提供的服务

❖ 成帧，链路接入：

- 将数据报封装在帧中，加上帧头、帧尾部
- 如果采用的是共享性介质，信道接入获得信道访问权
- 在帧头部使用“MAC”(物理)地址来标示源和目的
 - 不同于IP地址

❖ 在(一个网络内) 相邻两个节点完成可靠数据传输

- 已经学过了(第三章)
- 在低出错率的链路上(光纤和双绞线电缆) 很少使用
- 在无线链路经常使用：出错率高
 - Q: 为什么在链路层和传输层都实现了可靠性

一般化的链路层服务，不是所有的链路层都提供这些服务，一个特定的链路层只是提供其中一部分的服务

链路层服务(续)

❖ 在相邻节点间(一个子网内) 进行可靠的转发

- 在低差错链路上很少使用 (光纤,一些双绞线)
 - 出错率低, 没有必要在每一个帧中做差错控制的工作, 协议复杂
 - 发送端对每一帧进行差错控制编码, 根据反馈做相应的动作
 - 接收端进行差错控制解码, 反馈给发送端(ACK , NAK)
 - 在本层放弃可靠控制的工作, 在网络层或者是传输层做可靠控制的工作, 或者根本就不做可靠控制的工作
- 在高差错链路上需要进行可靠的数据传送
 - 高差错链路: 无线链路:

Q: 为什么要在采用无线链路的网络上, 链路层做可靠数据传输工作; 还要在传输层做端到端的可靠性工作?
 - 原因: 出错率高, 如果在链路层不做差错控制工作, 漏出去的错误 比较高; 到了上层如果需要可靠控制的数据传输代价会很大
 - 如不做local recovery工作, 总体代价大



链路层服务（续）

❖ 流量控制：

- 使得相邻的发送和接收方节点的速度匹配

❖ 错误检测：

- 差错由信号衰减和噪声引起
- 接收方检测出的错误：
- 通知发送端进行重传或丢弃帧

❖ 差错纠正：

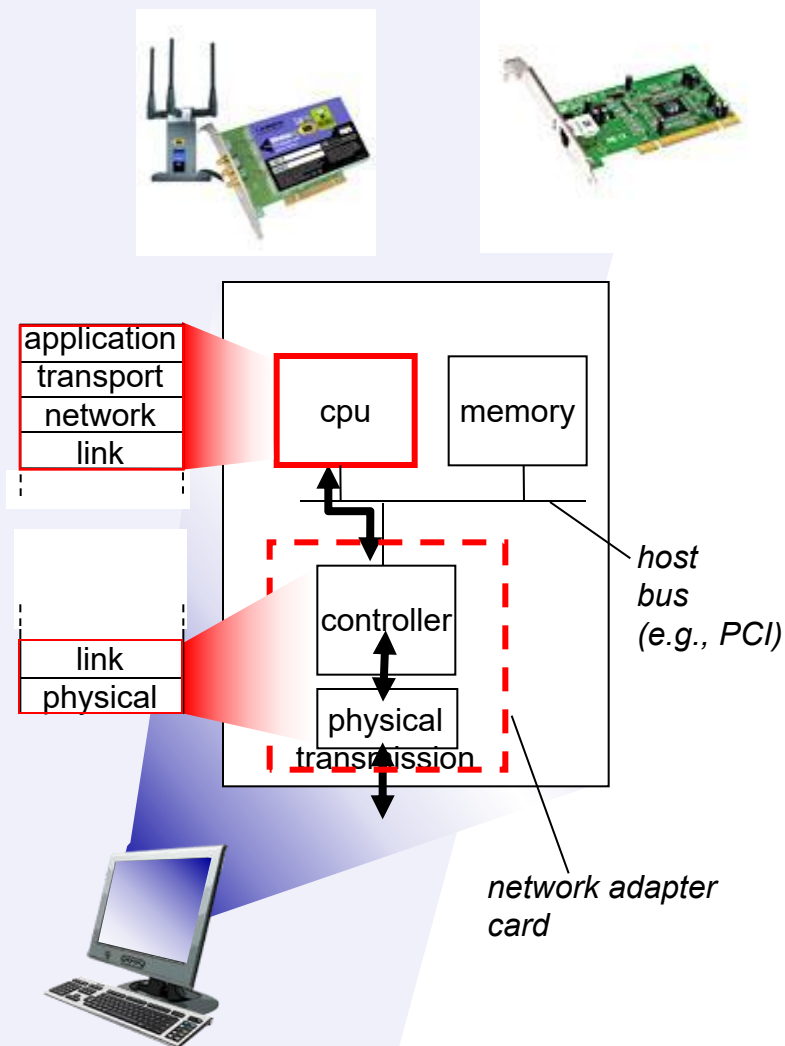
- 接收端检查和纠正bit错误，不通过重传来纠正错误

❖ 半双工和全双工：

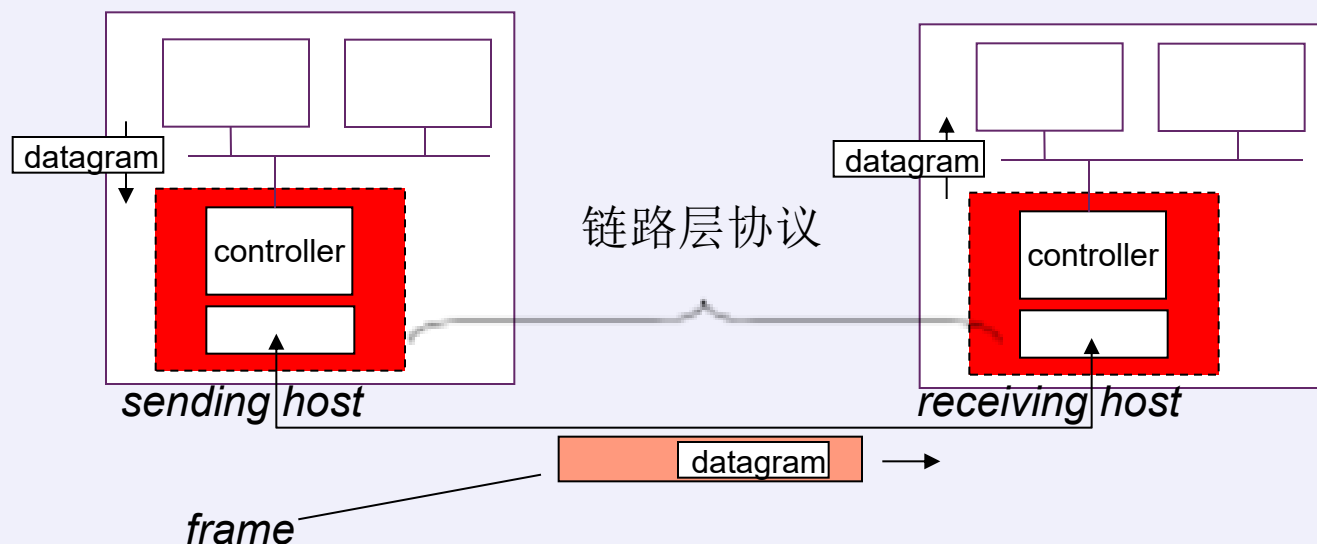
- 半双工：链路可以双向传输，但一次只有一个方向

5.1.2 链路层在哪里实现？

- ❖ 在每一个主机上
 - 也在每个路由器上
 - 交换机的每个端口上
- ❖ 链路层功能在“适配器”上实现
(**network interface card, NIC**) 或者在一个芯片组上
 - 以太网卡，802.11 网卡; 以太网芯片组
 - 实现链路层和相应的物理层功能
- ❖ 接到主机的系统总线上
- ❖ 硬件、软件和固件的综合体



适配器通信



❖ 发送方:

- 在帧中封装数据报
- 加上差错控制编码, 实现rdt和流量控制功能等

❖ 接收方

- 检查有无出错, 执行rdt和流量控制功能等
- 解封装数据报, 将至交给上层

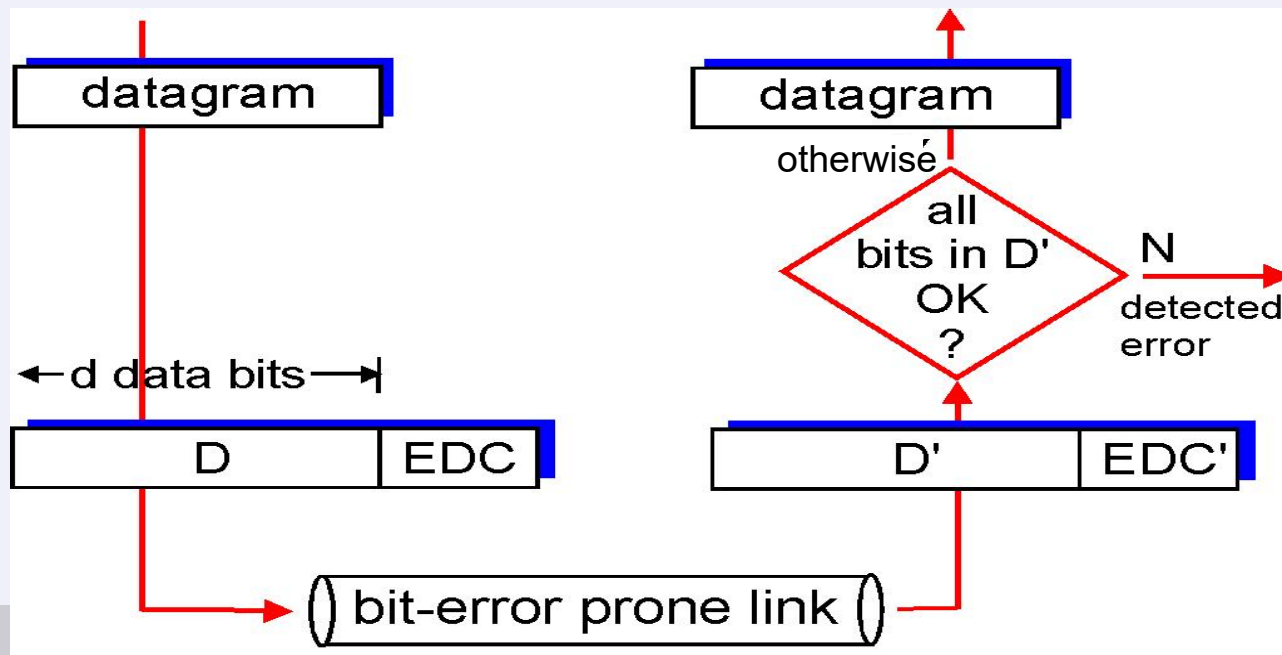
The background of the slide features a detailed map of East Asia, including parts of China, Korea, and Japan. The map is rendered in a light, textured style, showing geographical features like rivers, coastlines, and major cities. A dark purple horizontal band is overlaid on the map, containing the title text. The right side of the slide has a light blue and white geometric design element.

内容提纲

- ❖ 5.1 引论和服务
- ❖ 5.2 差错检测和纠正
- ❖ 5.3 多点访问协议
- ❖ 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches

6.2 错误检测与纠正

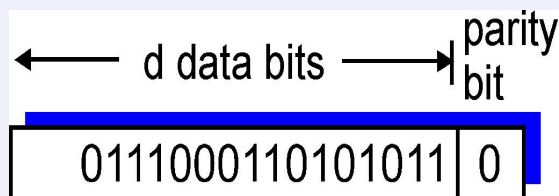
- ❖ EDC=差错检测和纠正位(冗余位)
- ❖ D=数据由差错检测保护，可以包含头部字段
- ❖ 错误检测不是100%可靠的!
 - 协议会漏检一些错误，但是很少
 - 更长的EDC字段可以得到更好的检测和纠正效果



(一) 奇偶校验

❖ 单bit奇偶校验:

- 检测单个bit级错误



校验位的计算

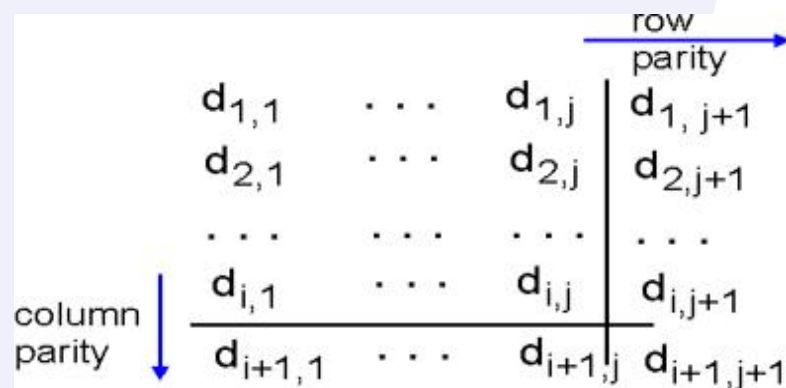
奇校验：如果数据单元中1的数量已经是奇数，则校验位设置为0；否则，校验位设置为1。

偶校验：如果数据单元中1的数量已经是偶数，则校验位设置为0；否则，校验位设置为1。

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

❖ 2维奇偶校验:

- 检测和纠正单个bit错误



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

no errors

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

parity error

*correctable
single bit error*

(二) Internet 校验和

目标: 检测在传输报文段时的错误(如位翻转),
(注: 仅用在传输层)

❖ 发送方:

- 将报文段看成16-bit整数
- 报文段的校验和: 和 (1'的补码和)
- 发送方将checksum的值放在'UDP校验和'字段

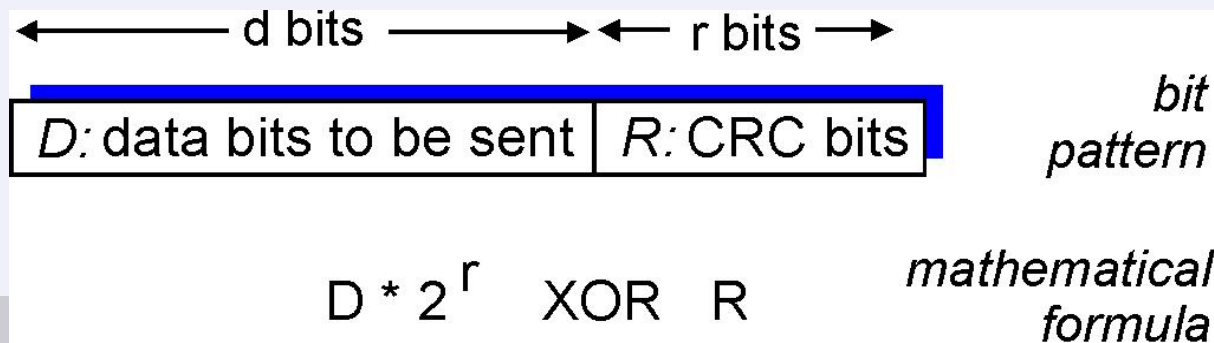
❖ 接收方:

- 计算接收到的报文段的校验和
- 检查是否与携带校验和字段值一致:
 - 不一致: 检出错误
 - 一致: 没有检出错误, 但可能还是有错误

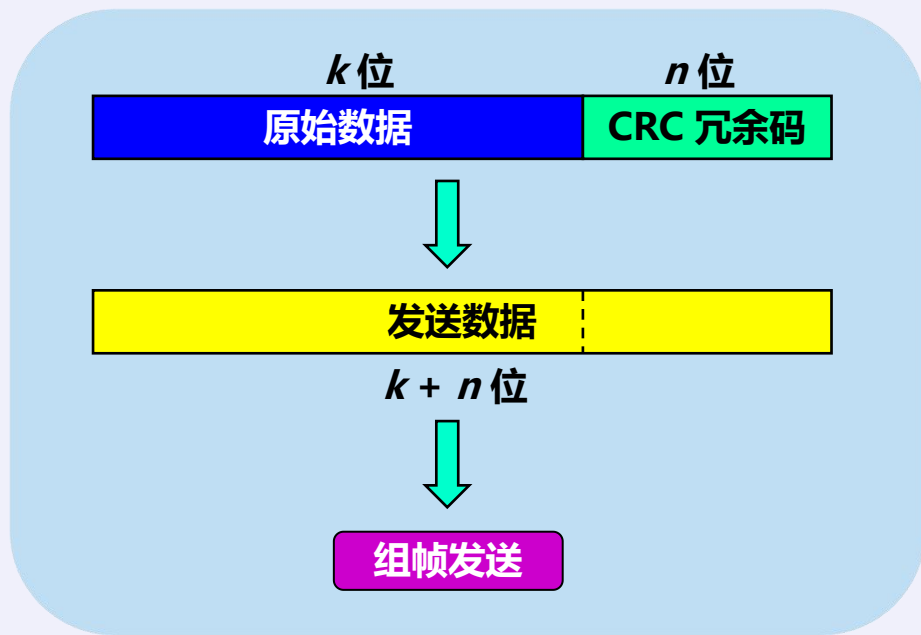
有更简单的检查方法
全部加起来看是不是全1

(三) CRC(循环冗余校验)

- ❖ 强大的差错检测码
- ❖ 将数据比特 **D**, 看成是二进制的数
- ❖ 生成多项式 **G**: 双方协商 **r+1** 位模式(**r** 次方)
 - 生成和检查所使用的位模式
- ❖ 目标:选择 **r** 位 **CRC** 附加位 **R**, 使得
 - **<D,R>** 正好被 **G** 整除 (modulo 2)
 - 接收方知道 **G**, 将 **<D,R>** 除以 **G**. 如果非 0 余数: 检查出错误!
 - 能检出所有少于 **r+1** 位的突发错误
- ❖ 实际中广泛使用(以太网、802.11 WiFi、ATM)

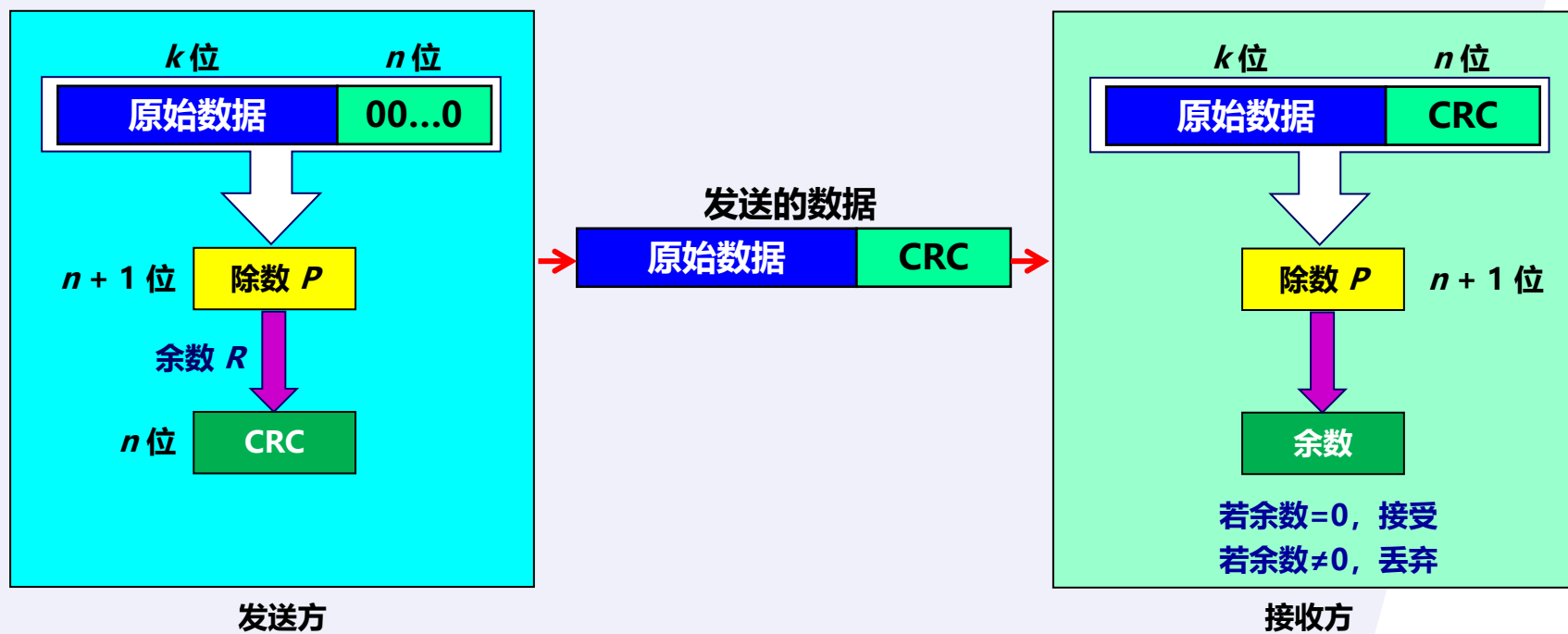


循环冗余检验 CRC (Cyclic Redundancy Check) 原理



- 在发送端，先把数据划分为组。假定每组 k 个比特。
- CRC 运算在每组 M 后面再添加供差错检测用的 n 位冗余码，然后构成一个帧发送出去。一共发送 $(k + n)$ 位。

CRC 冗余码的计算



CRC 冗余码的计算

- 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 , 余数是 R , 余数 R 比除数 P 少 1 位, 即 R 是 n 位。
- 3, 将余数 R 作为冗余码拼接在数据 M 后面, 一起发送出去。

这种为了进行检错而添加的冗余码常称为帧检验序列 FCS (Frame Check Sequence)。

广泛使用的生成多项式P(X)

$$\text{CRC-8} = X^8 + X^5 + X^4 + 1$$

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

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

$$\begin{aligned} \text{CRC-32} = & X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 \\ & + X^4 + X^2 + X + 1 \end{aligned}$$

比如CRC8中用到的位宽为8的生成多项式，
其实对应得二进制数有九位：

100110001

CRC 例子

❖ 需要:

$$D \cdot 2^r \text{ XOR } R = nG$$

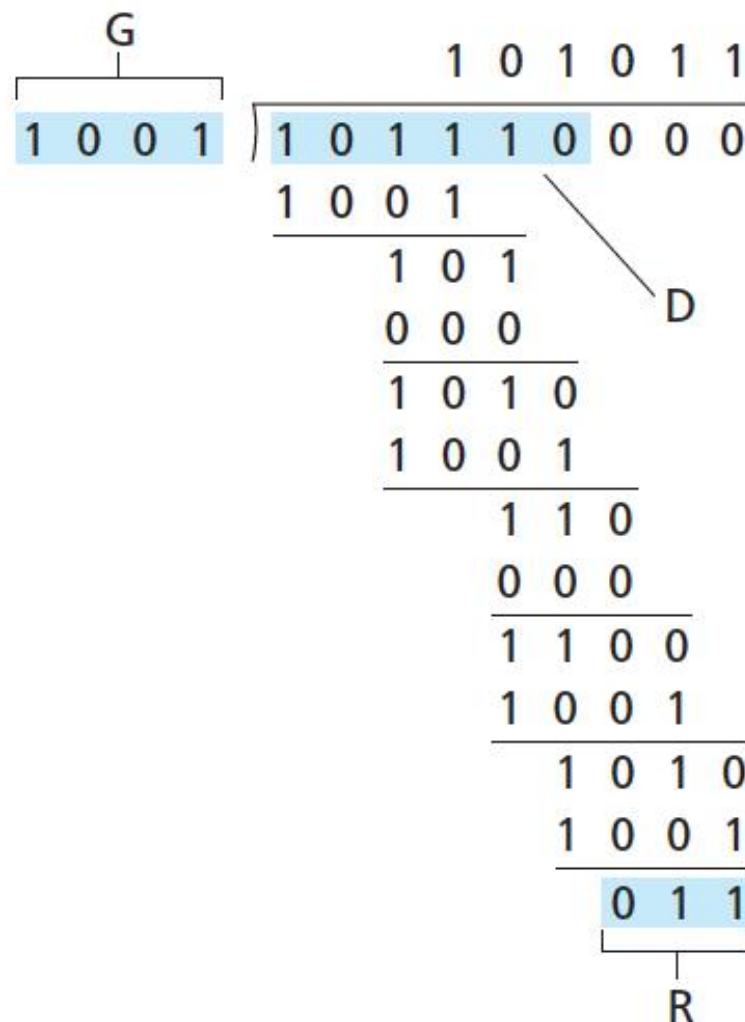
❖ 等价于:

$$D \cdot 2^r = nG \text{ XOR } R$$

❖ 等价于:

- 两边同除G
- 得到余数R

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$



CRC 冗余码的计算举例

原始数据 $M = 101001$

除数 $G = x^3 + x^2 + 1$

得到:

发送数据 = 101001**R**

每次异或结果的最左边第一位一定为**0**，这一位会被省略掉。

余数的位数比除数少**1**位，如果余数位数不够就在前面补**0**

CRC 冗余码的计算举例

原始数据 $M = 101001$

除数 $G = 1101$

得到:

发送数据 = $101001\mathbf{001}$

$$\begin{array}{r} \text{G (除数)} \rightarrow 1101 \quad \left. \begin{array}{l} 110100 \leftarrow Q \text{ (商)} \\ 101001\mathbf{000} \leftarrow 2^n M \text{ (被除数)} \end{array} \right\} \\ \begin{array}{r} 1101 \\ \hline 1110 \\ 1101 \\ \hline 0111 \\ 0000 \\ \hline 1110 \\ 1101 \\ \hline 0110 \\ 0000 \\ \hline 1100 \\ 1101 \\ \hline \mathbf{001} \end{array} \end{array}$$

$\mathbf{001} \leftarrow R \text{ (余数), 作为 FCS}$

每次异或结果的最左边第一位一定为0，这一位会被省略掉。

余数的位数比除数少1位，如果余数位数不够就在前面补0

CRC性能分析

❖ 突发错误和突发长度

❖ CRC检错性能描述

- 能够检查出所有的**1bit**错误
- 能够检查出所有的双**bits**的错误
- 能够检查出所有长度=**r**或者<**r**位的错误
- 出现长度为**r+1**的突发错误，检查不出的概率是 $\frac{1}{2^{r-1}}$
- 出现长度大于**r+1**的突发错误，检查不出的概率 $\frac{1}{2^r}$

The background of the slide features a map of East Asia, including parts of China, Korea, and Japan. The map is rendered in a light blue and white color scheme, with a dark blue overlay at the top where the title is located. The title '内容提纲' is written in white Chinese characters. Below the title, the main content is listed in a bulleted format, with the third item '5.3 多点访问协议' highlighted in red.

内容提纲

- ❖ 5.1 引论和服务
- ❖ 5.2 差错检测和纠正
- ❖ **5.3 多点访问协议**
- ❖ 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches

5.3 多路访问链路和协议

❖ 两种类型的链路(一个子网内部链路连接形式)：

■ 点对点

- 拨号访问的PPP
- 以太网交换机和主机之间的点对点链路

■ 广播 (共享线路或设备)

- 传统以太网
- HFC上行链路
- 无线局域网



shared wire (e.g.,
cabled Ethernet)



shared RF
(e.g., 802.11 WiFi)



shared RF
(satellite)



人类在酒会
(共享空气, 声音)

多路访问协议

- ❖ 单个共享的广播型链路
- ❖ 如果2个或更多站点同时传送: 冲突(**collision**)
 - 多个节点在同一个时刻发送, 则会收到2个或多个信号叠加

多路访问协议(介质访问控制协议: **MAC**)

- ❖ 分布式算法-决定节点如何使用共享信道, 即: 决定节点什么时候可以发送?
- ❖ 关于共享控制的通信必须用借助信道本身传输!
 - 没有带外的信道, 各节点使用其协调信道使用
 - 用于传输控制信息

理想的多路访问协议

给定：Rbps的广播信道

必要条件：

1. 当一个节点要发送时，可以R速率发送.
2. 当M个节点要发送，每个可以以R/M的平均速率发送
3. 完全分布的：
 - 没有特殊节点协调发送
 - 没有时钟和时隙的同步
4. 简单



MAC(媒体访问控制)协议：分类

三大类：

❖ 信道划分

- 把信道划分成小片(时间、频率、编码)
- 分配片给每个节点专用

❖ 随机访问

- 信道不划分，允许冲突
- 冲突后恢复

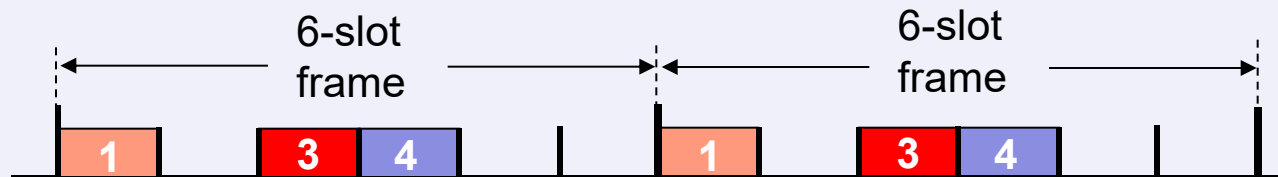
❖ 依次轮流

- 节点依次轮流
- 但是允许有很多数据传输的节点可以获得较长的信道使用权

1-1.信道划分MAC协议: TDMA

TDMA:time division multiple access

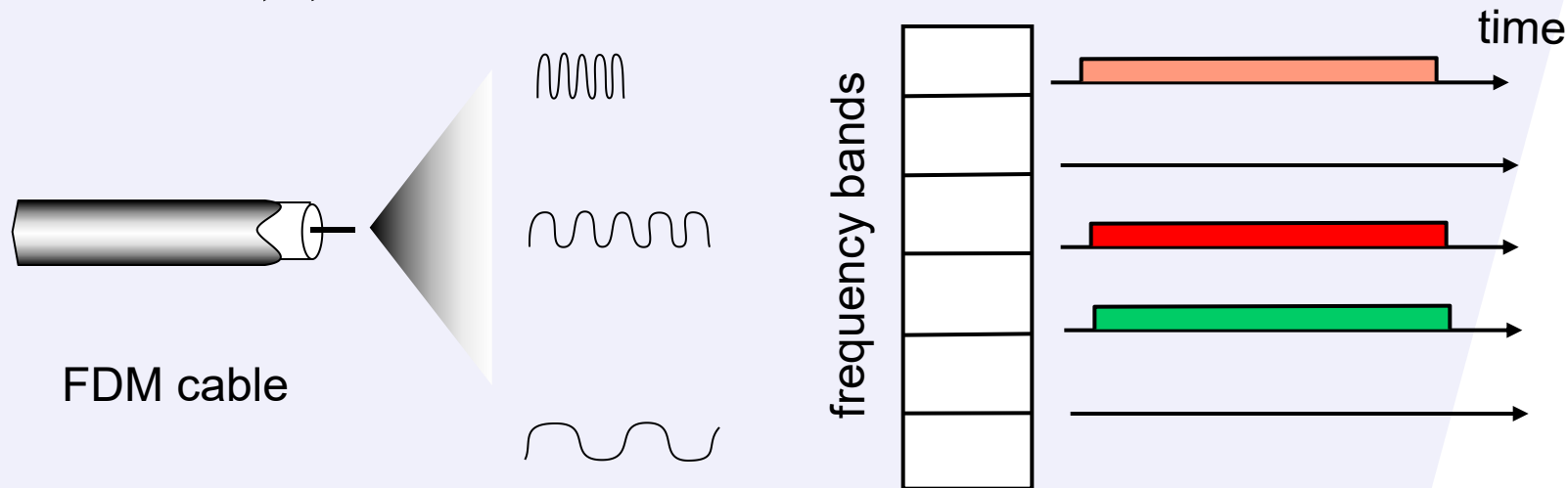
- ❖ 轮流使用信道，信道的时间分为周期
- ❖ 每个站点使用每周期中固定的时隙(长度=帧传输时间) 传输帧
- ❖ 如果站点无帧传输，时隙空闲-》浪费
- ❖ 如：6站LAN，1、3、4有数据报，时隙2、5、6空闲



1-2.信道划分MAC协议: FDMA

FDMA: frequency division multiple access

- ❖ 信道的有效频率范围被分成一个个小的频段,每个站点被分配一个固定的频段
- ❖ 分配给站点的频段如果没有被使用, 则空闲
- ❖ 例如: 6站LAN, 1、3、4有数据报, 频段2、5、6空闲



1-3.信道划分MAC协议:码分多路访问(CDMA)

❖ CDMA (code division multiple access):

- 所有站点在整个频段上同时进行传输, 采用编码原理加以区分
- 完全无冲突
- 假定:信号同步很好,线性叠加

❖ 比方

- TDM:不同的人在不同的时刻讲话
- FDM:不同的组在不同的小房间里通信
- CDMA:不同的人使用不同的语言讲话

2.随机访问协议

- ❖ 当节点有帧要发送时
 - 以信道带宽的全部Rbps发送
 - 没有节点间的预先协调
- ❖ 两个或更多节点同时传输，会发生→冲突碰撞“collision”
- ❖ 随机存取协议规定：
 - 如何检测冲突
 - 如何从冲突中恢复(如：通过稍后的重传)
- ❖ 随机MAC协议：
 - 时隙ALOHA
 - ALOHA
 - CSMA, CSMA/CD, CSMA/CA

2.1 时隙ALOHA

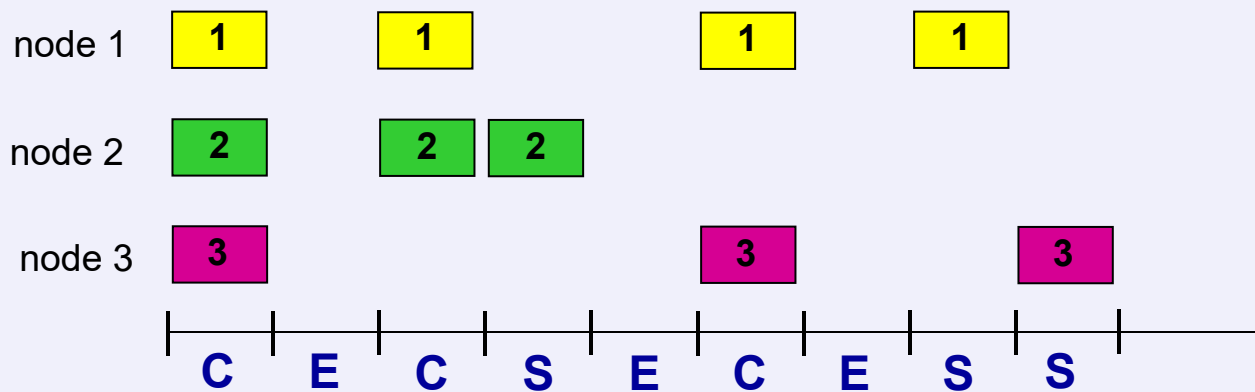
假设：

- ❖ 所有帧是等长的
- ❖ 时间被划分成相等的时隙，每个时隙可发送一帧
- ❖ 节点只在时隙开始时发送帧
- ❖ 节点在时钟上是同步的
- ❖ 如果两个或多个节点在一个时隙传输，所有的站点都能检测到冲突

运行：

- ❖ 当节点获取新的帧，在下一个时隙传输
- ❖ 传输时没有检测到冲突，成功
 - 节点能够在下一时隙发送新帧
- ❖ 检测时如果检测到冲突，失败
 - 节点在每一个随后的时隙以概率 p 重传帧直到成功

2.1 时隙ALOHA



优点

- ❖ 节点可以以信道带宽全速连续传输
- ❖ 高度分布：仅需要节点之间在时隙上的同步
- ❖ 简单

缺点

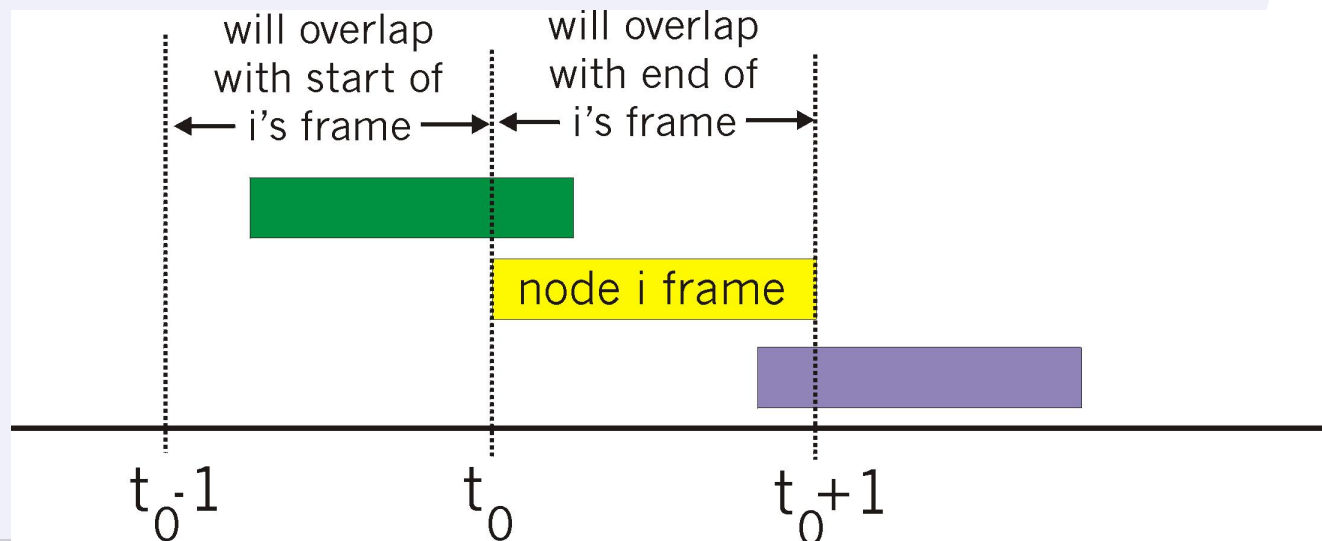
- ❖ 存在冲突，浪费时隙
- ❖ 即使有帧要发送，仍然有可能存在空闲的时隙
- ❖ 节点检测冲突的时间 $<$ 帧传输的时间
 - 必须传完
- ❖ 需要时钟上同步

时隙ALOHA的效率(Efficiency)

- ❖ 假设N个节点，每个节点都有很多帧要发送，在每个时隙中的传输概率是p
- ❖ 一个节点成功传输概率是 $p(1-p)^{N-1}$
- ❖ 任何一个节点的成功概率是 $= Np(1-p)^{N-1}$
- N个节点的最大效率：
 - ❖ 求出使 $f(P)=Np(1-p)^{N-1}$ 最大的 p^*
 - ❖ 代入 P^* 得到最大 $f(p^*)=Np^*(1-p^*)^{N-1}$
 - ❖ N为无穷大时的极限为
 $1/e=0.37$

2.2 纯ALOHA(非时隙)

- ❖ 无时隙ALOHA: 简单、无须节点间在时间上同步
- ❖ 当有帧需要传输: 马上传输
- ❖ 冲突的概率增加:
 - 帧在 t_0 发送, 和其它在 $[t_0-1, t_0+1]$ 区间内开始发送的帧冲突
 - 和当前帧冲突的区间(其他帧在此区间开始传输)增大了一倍



纯ALOHA的效率



$$P(\text{指定节点成功}) = P(\text{节点传输}) \cdot$$

$$P(\text{其它节点在}[t_0-1, t_0]\text{不传}) \cdot$$

$$P(\text{其它节点在}[t_0, t_0+1]\text{不传})$$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1} = p \cdot (1-p)^{2(N-1)}$$

选择最佳的 p 、 N 趋向无穷大...

$$= 1/(2e) = 17.5\%$$

效率比时隙ALOHA更差了!

2.3 CSMA(载波侦听多路访问)

❖ Aloha: 如何提高ALOHA的效率

- 发之前不管有无其他节点在传输

❖ CSMA: 在传输前先侦听信道:

- 如果侦听到信道空闲, 传送整个帧
- 如果侦听到信道忙, 推迟传送

- 人类类比: 不要打断别人正在进行的说话!

CSMA冲突

冲突仍然可能发生:

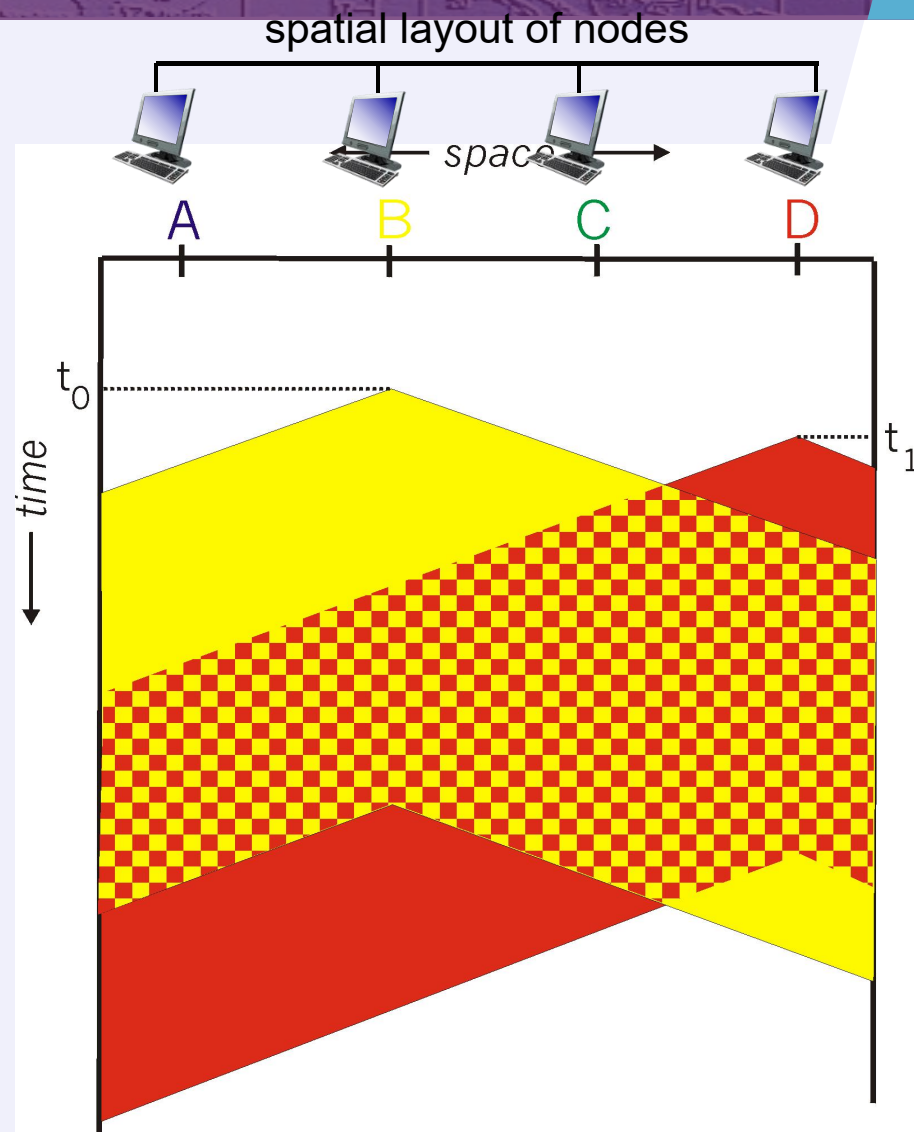
- ❖ 由传播延迟造成: 两个节点可能侦听不到正在进行的传输

冲突:

- ❖ 整个冲突帧的传输时间都被浪费了, 是无效的传输(红黄区域)

注意:

- ❖ 传播延迟(距离)决定了冲突的概率



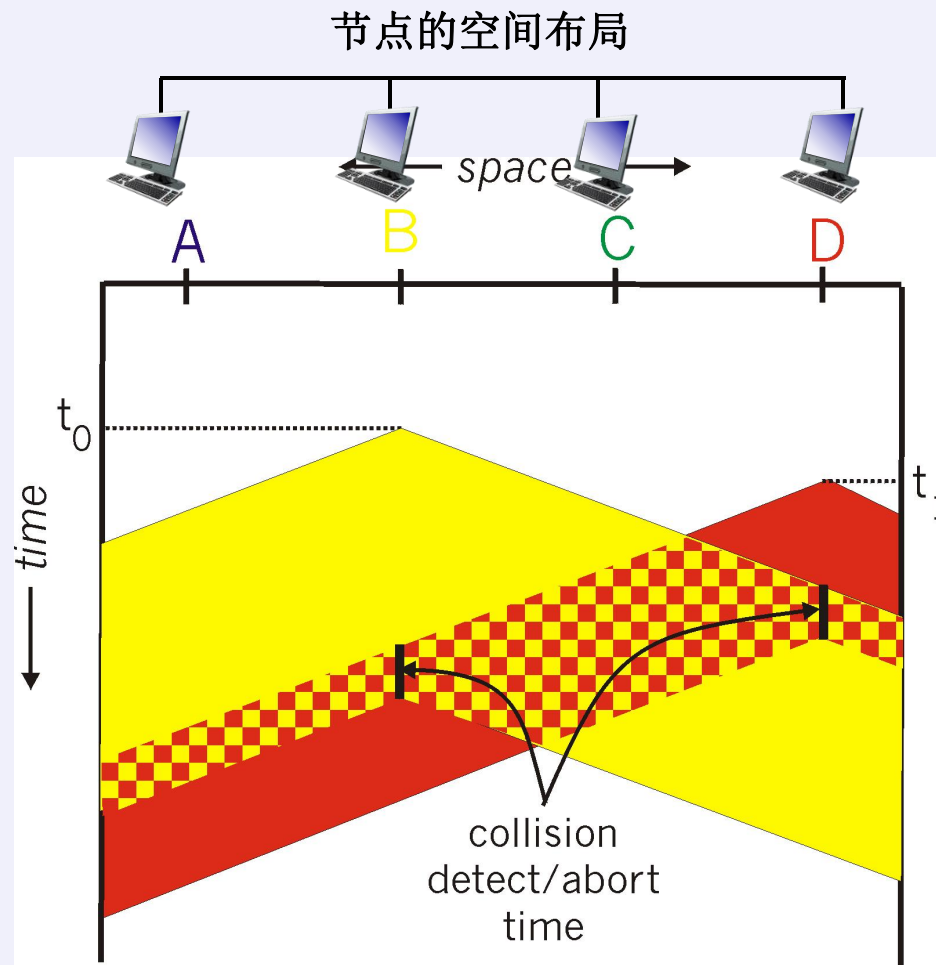
CSMA/CD(冲突检测)

CSMA/CD:

- 载波侦听CSMA: 和在CSMA中一样发送前侦听信道
- 没有传完一个帧就可以在短时间内检测到冲突
- 冲突发生时则传输终止, 减少对信道的浪费
- ❖ 冲突检测CD技术, 有线局域网中容易实现:
 - 检测信号强度, 比较传输与接收到的信号是否相同
- ❖ 人类类比: 礼貌的对话人
- ❖ **CSMA/CD冲突避免的方法: 先听后发、边听边发、随机延迟后重发**

CSMA/CD（冲突检测）

对信道的浪费减少



以太网CSMA/CD算法

1. 适配器获取数据报，创建帧

2. 发送前：侦听信道CS

- 1) 闲：开始传送帧
- 2) 忙：一直等到闲再发送

3. 发送过程中，冲突检测CD

- 1) 没有冲突：成功
- 2) 检测到冲突：放弃，之后尝试重发

4. 发送方适配器检测到冲突，除放弃外，还发送一个Jam信号，所有听到冲突的适配器也是如此
强化冲突：让所有站点都知道冲突

5. 如果放弃，适配器进入指数退避状态：

- 在第m次失败后，适配器随机选择一个 $\{0, 1, 2, \dots, 2^m - 1\}$ 中选择一个K，等待 $K \times 512$ 比特时间，然后转到步骤2
- n取值最大值在10以内

❖ **Binary exponential backoff** 二进制指数退避算法

以太网CSMA/CD算法

指数退避:

- ❖ 目标：适配器试图适应当前负载，在一个变化的碰撞窗口中随机选择时间点尝试重发
 - 高负载：重传窗口时间大，减少冲突，但等待时间长
 - 低负载：使得各站点等待时间少，但冲突概率大
- ❖ 首次碰撞：在 $\{0, 1\}$ 选择 K ；延迟 $K \times 512$ 比特时间
- ❖ 第2次碰撞：在 $\{0, 1, 2, 3\}$ 选择 K
- ❖ 第10次碰撞：在 $\{0, 1, 2, 3, \dots, 1023\}$ 选择 K
随机数范围： $0 \sim 2^m - 1$ ($m \leq 10$)

注：对100Mbps以太网512比特时间 就是5.12us

CSMA/CD效率

❖ T_{prop} = LAN上2个节点的最大传播延迟

❖ t_{trans} = 传输最大帧的时间

$$\text{efficiency} = \frac{1}{1 + 5t_{\text{prop}}/t_{\text{trans}}}$$

❖ 效率变为1

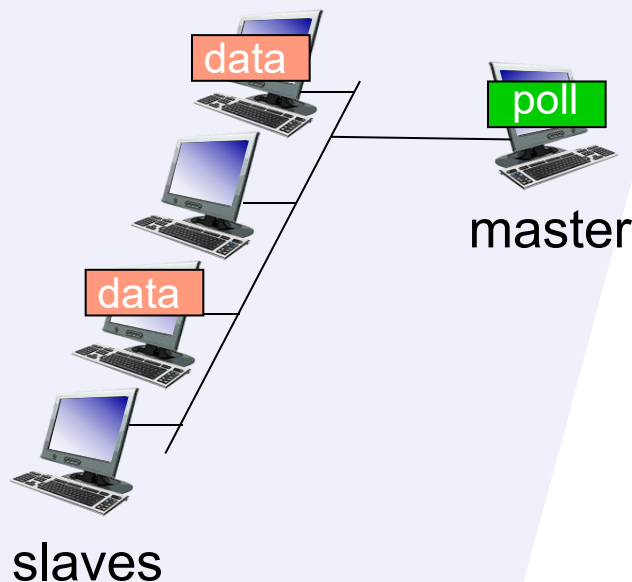
- 当 t_{prop} 变成0时
- 当 t_{trans} 变成无穷大时

❖ 比ALOHA更好的性能，而且简单，廉价，分布式！

3、轮流MAC协议

轮询协议:

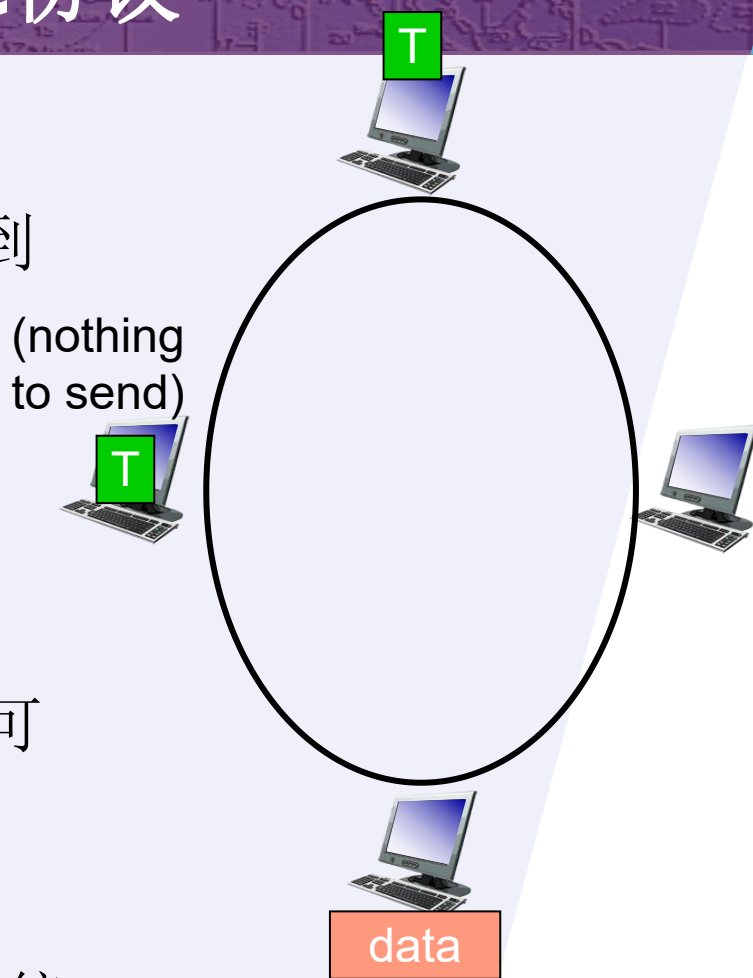
- ❖ 主节点邀请从节点依次传送
- ❖ 从节点一般比较“沉默”
- ❖ 缺点:
 - 轮询开销: 轮询本身消耗信道带宽
 - 等待时间: 每个节点需等到主节点轮询后开始传输, 即使只有一个节点, 也需要等到轮询一周后才能够发送
 - 单点故障: 主节点失效时造成整个系统无法工作



轮流MAC协议

令牌传递协议:

- ❖ 控制令牌(token)循环从一个节点到下一个节点传递
- ❖ 令牌报文: 特殊的帧
- ❖ 缺点:
 - 令牌开销: 本身消耗带宽
 - 延迟: 只有等到抓住令牌, 才可传输
 - 单点故障 (token):
 - 令牌丢失系统级故障, 整个系统无法传输
 - 复杂机制重新生成令牌



MAC协议总结

- ❖ 多点接入问题：对于一个共享型介质，各个节点如何协调对它的访问和使用？
- ❖ 信道划分：按时间、频率或者编码
 - TDMA、FDMA、CDMA
- ❖ 随机访问 (动态)
 - ALOHA, S-ALOHA, CSMA, CSMA/CD
 - 载波侦听：在有些介质上很容易 (wire: 有线介质), 但在有些介质上比较困难 (wireless: 无线)
 - CSMA/CD : 802.3 Ethernet网中使用
- ❖ 依次轮流协议
 - 集中：由一个中心节点轮询；分布：通过令牌控制
 - 蓝牙、FDDI、令牌环

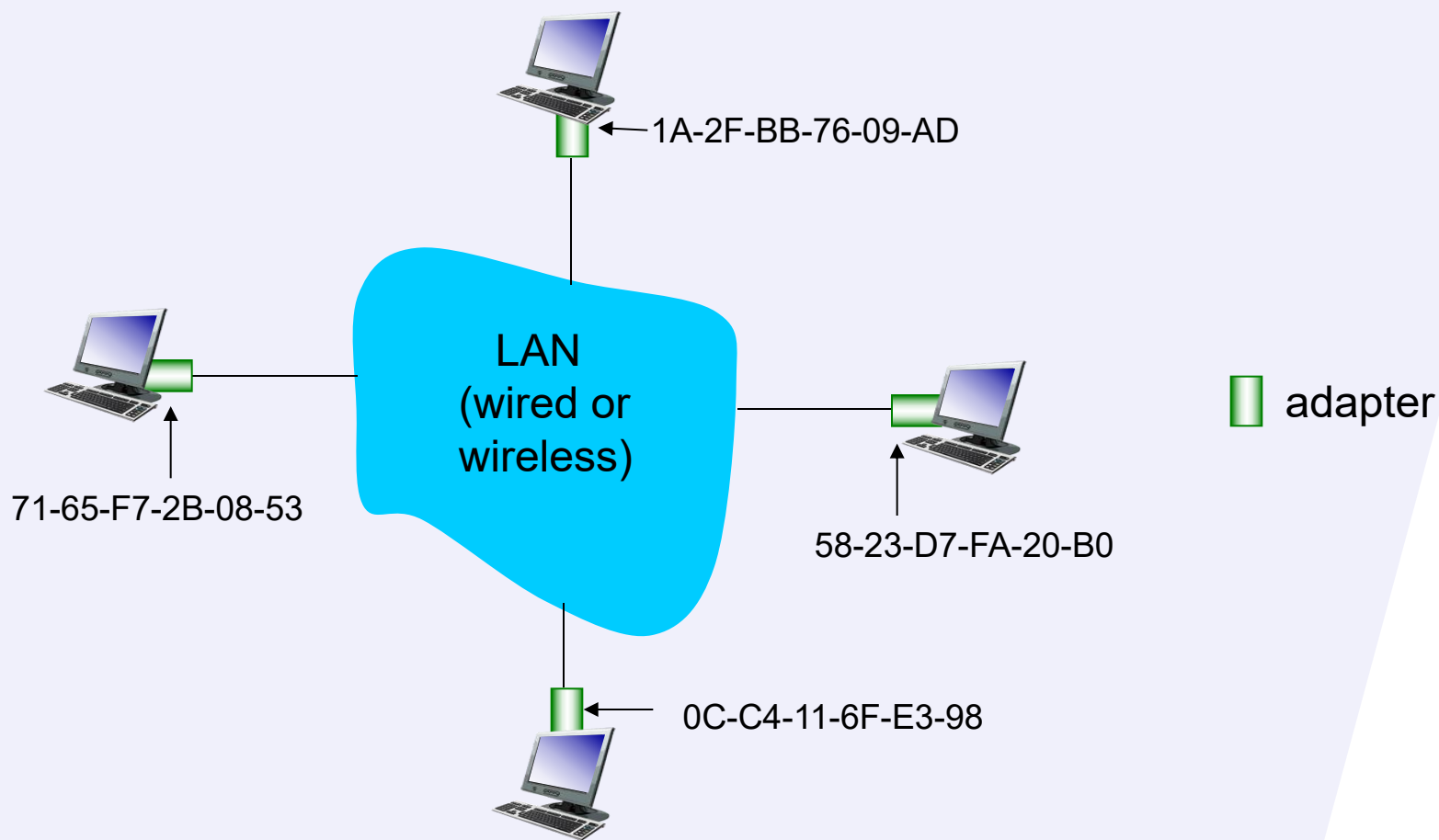


内容提纲

- ❖ 5.1 引论和服务
- ❖ 5.2 差错检测和纠正
- ❖ 5.3 多点访问协议
- ❖ 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches

5.4.1 LAN 地址和ARP

❖ 局域网上每个适配器都有一个唯一的LAN地址

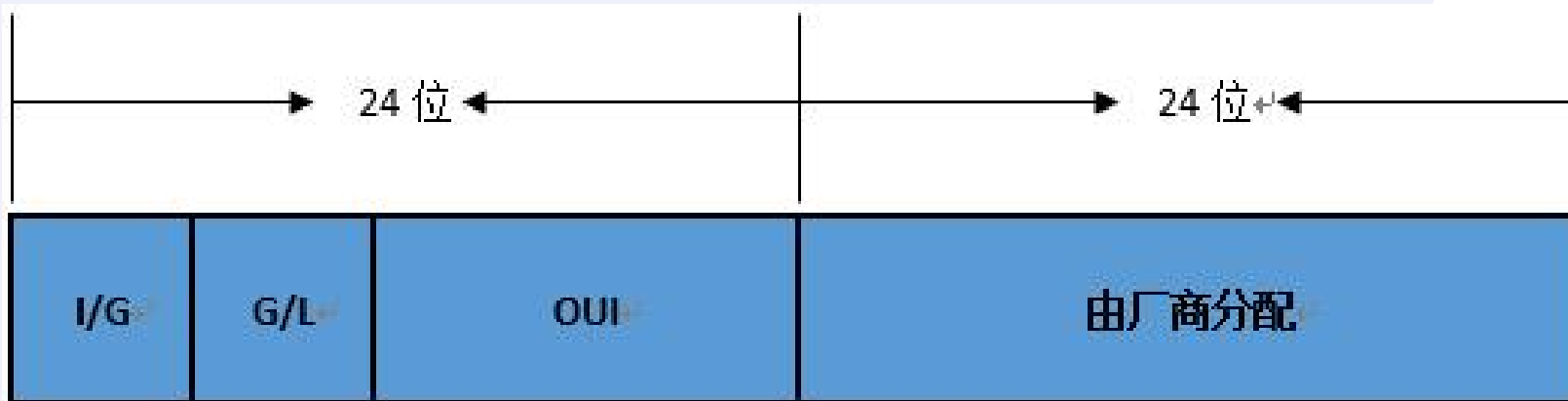


LAN 地址和ARP (续)

- ❖ MAC地址由**IEEE**管理和分配
- ❖ 制造商购入MAC地址空间(保证唯一性)
- ❖ 类比:
 - (a)MAC地址: 身份证号
 - (b)IP地址: 通讯地址
- ❖ MAC平面地址 → 支持移动
 - 可以将网卡连接到其它网络
- ❖ IP地址有层次-不能移动
 - 依赖于节点连接的IP子网, 与子网的网络号相同 (有与其相连的子网相同的网络前缀)

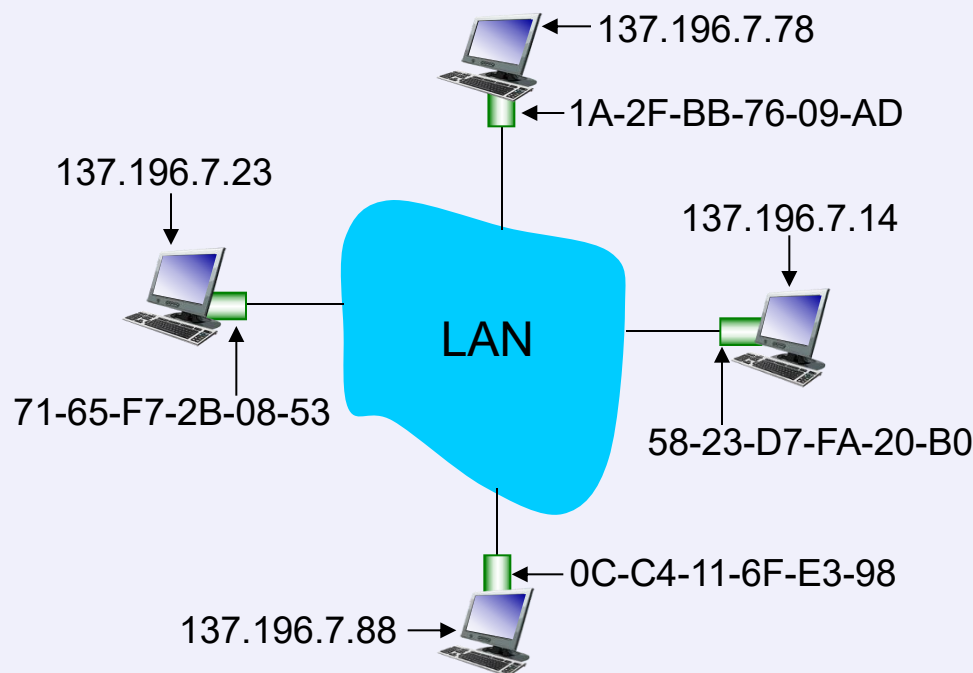
MAC地址

- ❖ 在数据链路层，数据帧通常依赖于**MAC地址**来进行数据交换，它如同公网**IP地址**一样要求具有全球唯一性，这样就可以识别每一台主机。
- ❖ **MAC地址**，英文全称**Medium Access Control**，直译为介质访问控制，它通常被固化在每个以太网网卡（**NIC, Network Interface Card**）。**MAC（硬件）地址**长**48位（6字节）**，采用十六进制格式，下图说明了**48位**的**MAC地址**及其组成部分。



ARP: Address Resolution Protocol

问题: 已知B的IP地址,
如何确定B的MAC地址?



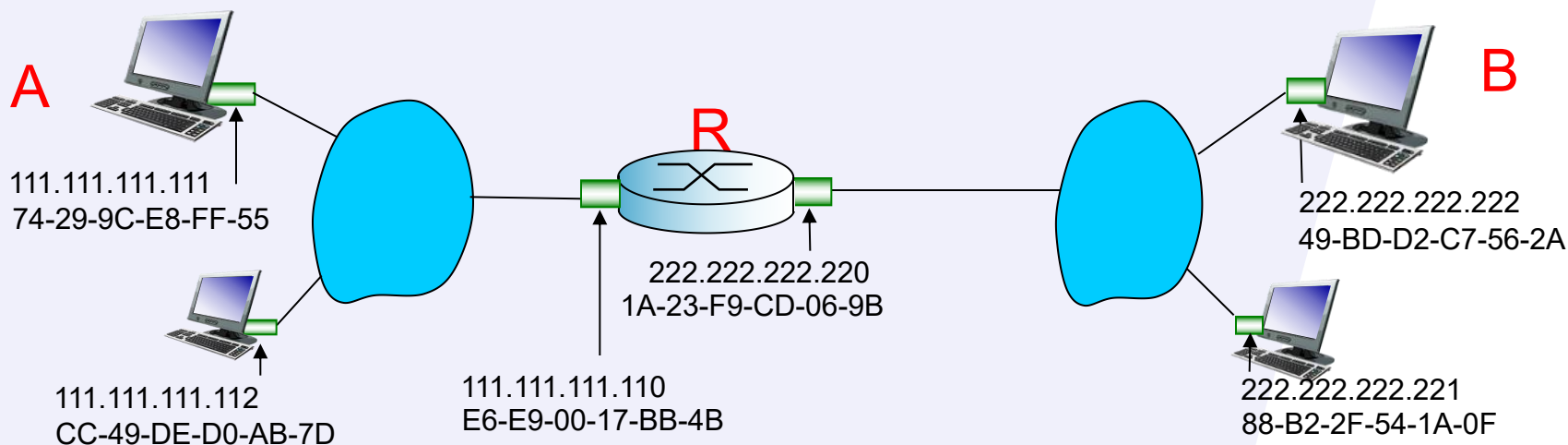
- ❖ 在LAN上的每个IP节点都有一个**ARP表**
- ❖ ARP表: 包括一些LAN节点IP/MAC地址的映射
- ❖ **<IP address; MAC address;TTL>**
 - TTL时间是指地址映射失效的时间
 - 典型是20min

ARP协议：在同一个LAN (局域网)

- ❖ A要发送帧给B(B的IP地址已知), 但B的MAC地址不在A的ARP表中
- ❖ A广播包含B的IP地址的ARP查询包
 - Dest MAC address=FF-FF-FF-FF-FF-FF
 - LAN上的所有节点都会收到该查询包
- ❖ B接收到ARP包, 回复A自己的MAC地址
 - 帧发送给A
 - 用A的MAC地址(单播)
- ❖ A在自己的ARP表中, 缓存IP-to-MAC地址映射关系, 直到信息超时
 - 软状态: 靠定期刷新维持的系统状态
 - 定期刷新周期之间维护的状态信息可能和原有系统不一致
- ❖ ARP是即插即用的
 - 节点自己创建ARP的表项
 - 无需网络管理员的干预

路由到其他LAN

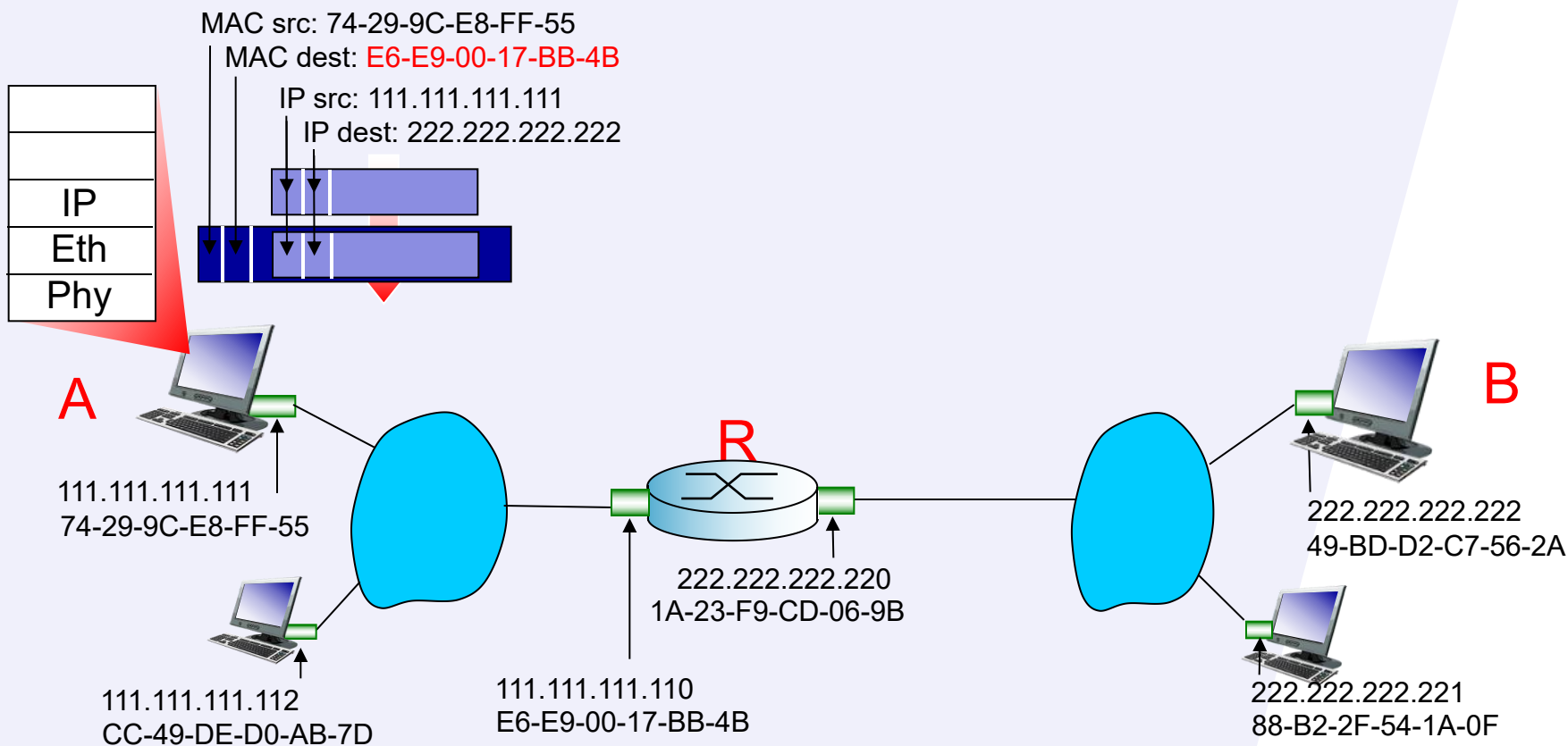
- ❖ 发送数据报：由主机A通过R到B，假设A知道B的IP地址
 - 在R上有两个ARP表，分别对应两个LAN
 - 在源主机的路由表中，发现到目标主机的下一跳时111.111.111.110
 - 在源主机的ARP表中，发现其MAC地址是E6-E9-00-17-BB-4B,



路由到其他LAN

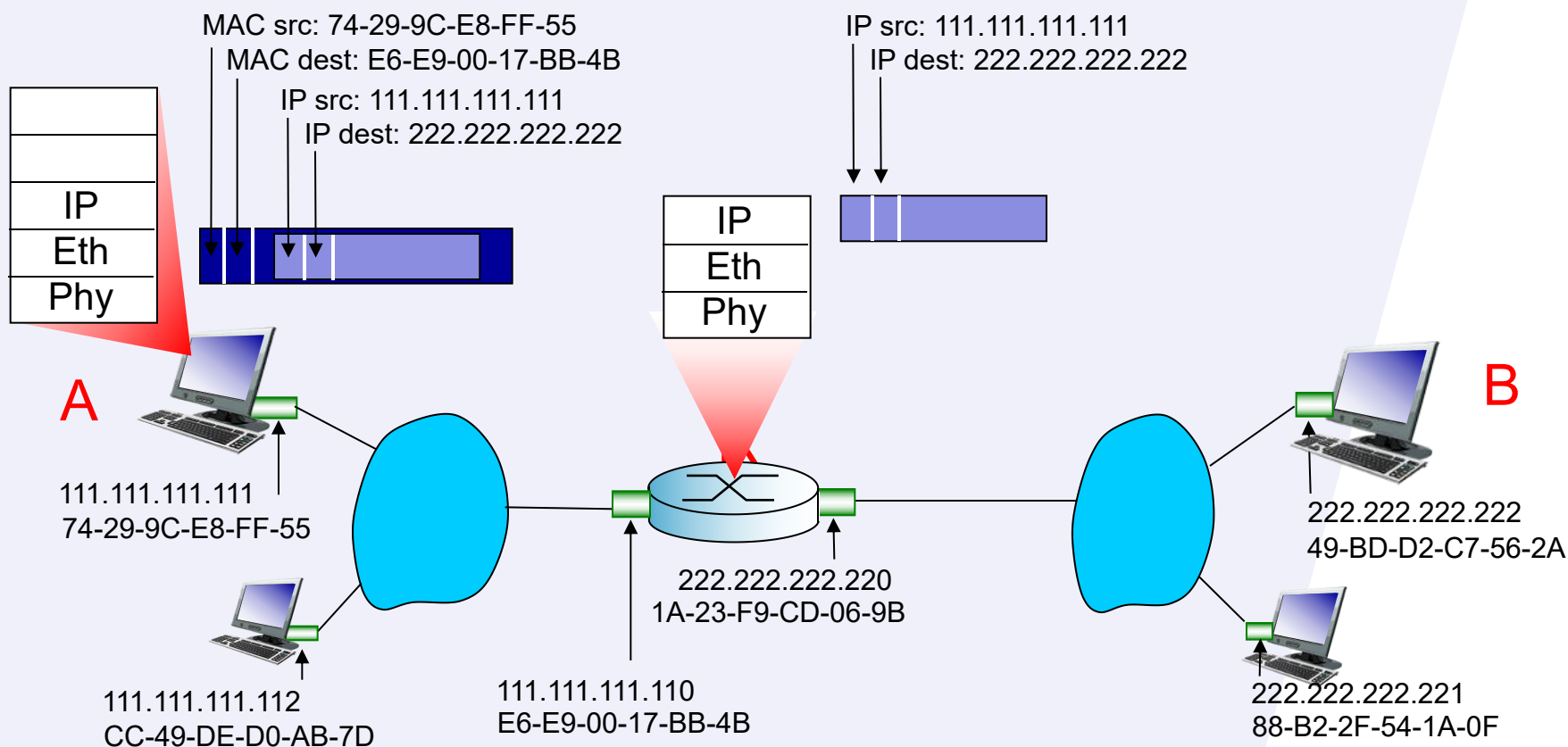
❖ 帧从A发送到R

- 帧被R接收到，从中提取出IP分组，交给上层IP协议实体



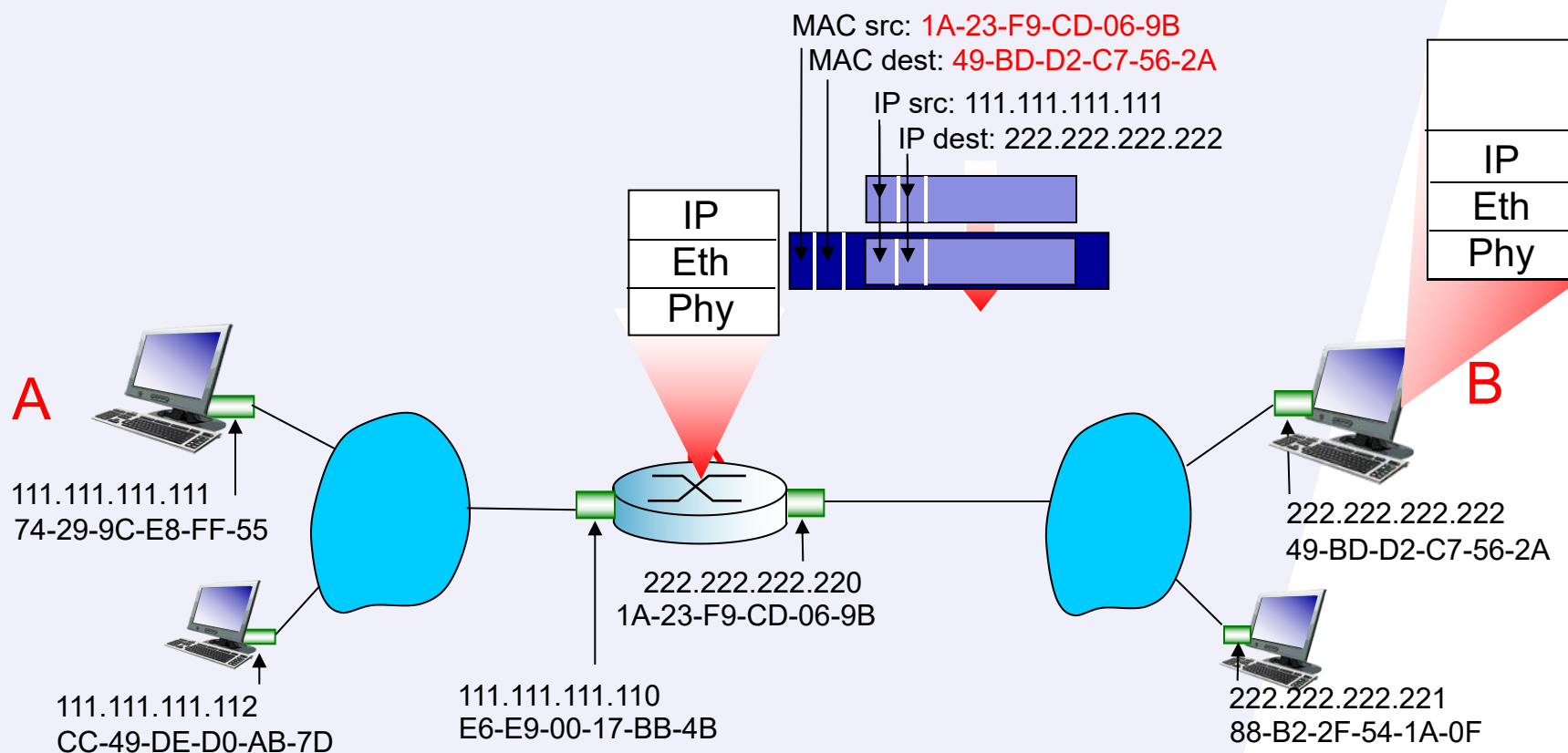
路由到其他LAN

- ❖ R转发数据报，数据报源IP地址为A，目标IP地址为B
- ❖ R创建一个链路层的帧，目标MAC地址为B，帧中包含A到B的IP数据报



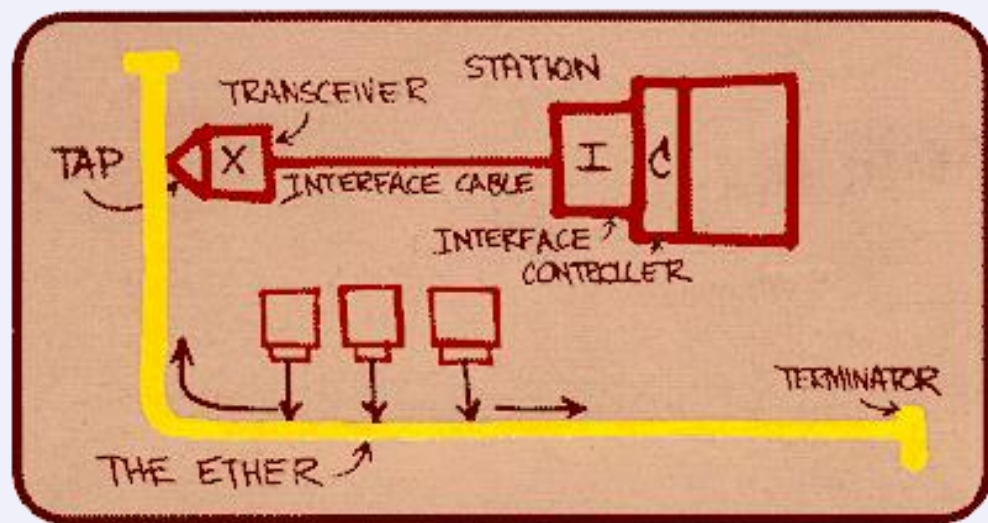
路由到其他LAN

- ❖ R转发数据报，数据报源IP地址为A，目标IP地址为B
- ❖ R创建一个链路层的帧，目标MAC地址为B，帧中包含A到B的IP数据报



5.4.2 以太网

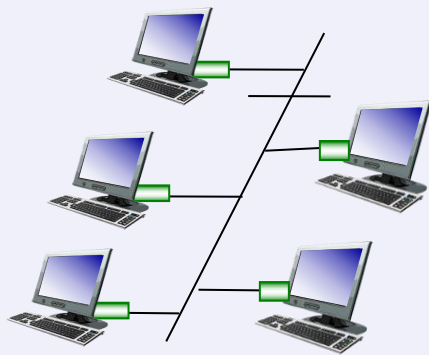
- ❖ 目前最主流的LAN技术：98%占有率
- ❖ 廉价：30元RMB 100Mbps！
- ❖ 最早广泛应用的LAN技术
- ❖ 比令牌网和ATM网络简单、廉价
- ❖ 带宽不断提升：10M, 100M, 1G, 10G



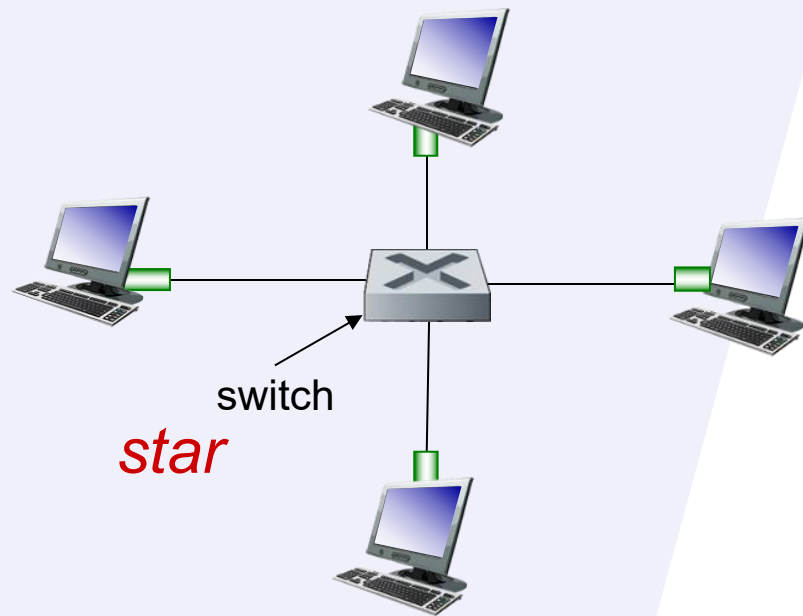
以太网：物理拓扑

❖ 总线

❖ 星型



bus: 同轴电缆



以太网：物理拓扑

❖ 总线：在上个世纪90年代中期很流行

- 所有节点在一个碰撞域内，一次只允许一个节点发送
- 可靠性差，如果介质破损，截面形成信号的反射，发送节点误认为是冲突，总是冲突

❖ 星型：目前最主流

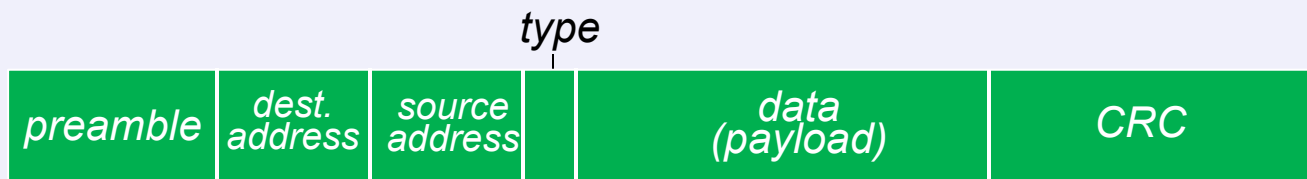
- 连接选择: **hub**或者**switch**
- 现在一般是交换机在中心
- 每个节点以及相连的交换机端口使用(独立的) 以太网协议(不会和其他节点的发送产生碰撞)

以太帧结构

❖ 发送方适配器在以太网帧中封装IP数据报，或其他网络层协议数据单元

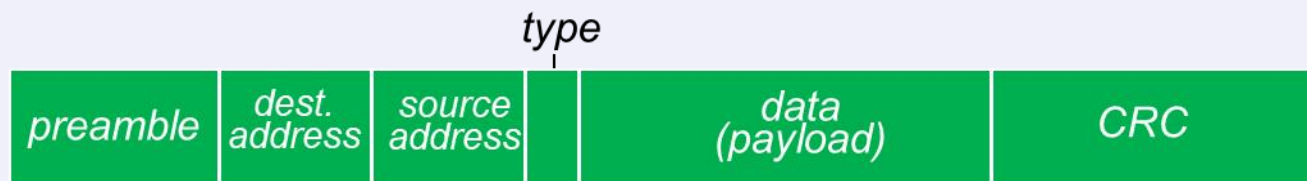
❖ 前导码（8字节）：

- 7B 10101010 + 1B 10101011
- 用来同步接收方和发送方的时钟速率
- 使得接收方将自己的时钟调到发送端的时钟
- 从而可以按照发送端的时钟来接收所发送的帧

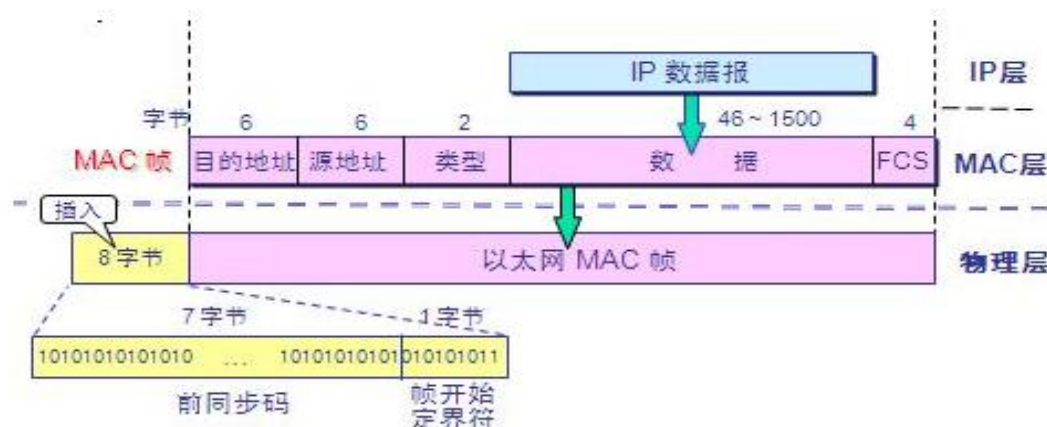


以太帧结构（续）

- ❖ **地址**：6字节源MAC地址，目标MAC地址
 - 如：帧目标地址=本站MAC地址，或是广播地址，接收，递交帧中的数据到网络层
 - 否则，适配器忽略该帧
- ❖ **类型**：指出高层协(大多情况下是IP，但也支持其它网络层协议Novell IPX和AppleTalk)
- ❖ **CRC**：在接收方校验
 - 如果没有通过校验，丢弃错误帧



以太网帧结构



以太网(IEEE 802.3)帧格式:

- 1、前导码 (前同步码) : 7字节0x55,一串1、0间隔,用于信号同步
- 2、帧开始定界符: 1字节0xD5(10101011), 表示一帧开始
- 3、DA(目的MAC): 6字节
- 4、SA(源MAC): 6字节
- 5、类型/长度: 2字节, 0 ~ 1500保留为长度域值, 1536 ~ 65535保留为类型域值(0x0600 ~ 0xFFFF)
- 6、数据: 46 ~ 1500字节
- 7、帧校验序列(FCS): 4字节, 使用CRC计算从目的MAC到数据域这部分内容而得到的校验和。

以太网：无连接、不可靠的服务

- ❖ **无连接**：帧传输前，发送方和接收方之间没有握手
- ❖ **不可靠**：接收方适配器不发送ACKs或NAKs给发送方
 - 递交给网络层的数据报流可能有gap
 - 如上层使用像传输层TCP协议这样的rdt，gap会被补上(源主机，TCP实体)
 - 否则，应用层就会看到gap
- ❖ 以太网的MAC协议：采用**二进制退避CSMA/CD**介质访问控制形式



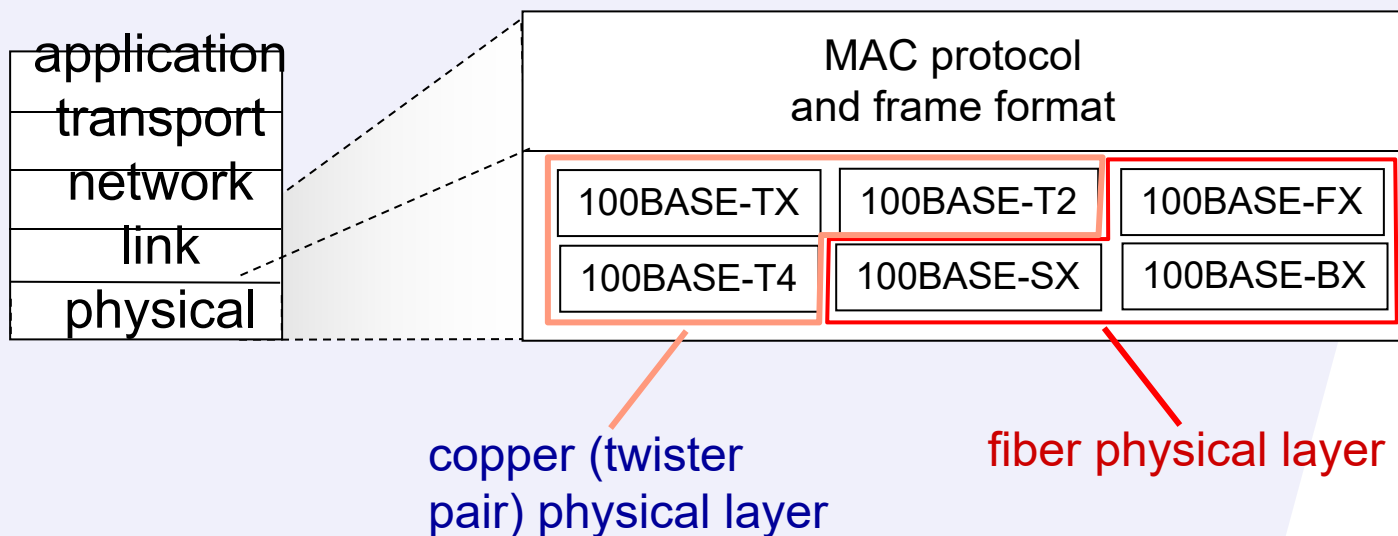
以太网使用CSMA/CD

- 没有时隙
- NIC如果侦听到其它NIC在发送就不发送：载波侦听
carrier sense
- 发送时，适配器当侦听到其它适配器在发送就放弃对当前帧的发送，冲突检测
collision detection
- 冲突后尝试重传，重传前适配器等待一个随机时间，随机访问**random access**

802.3 以太网标准：链路和物理层

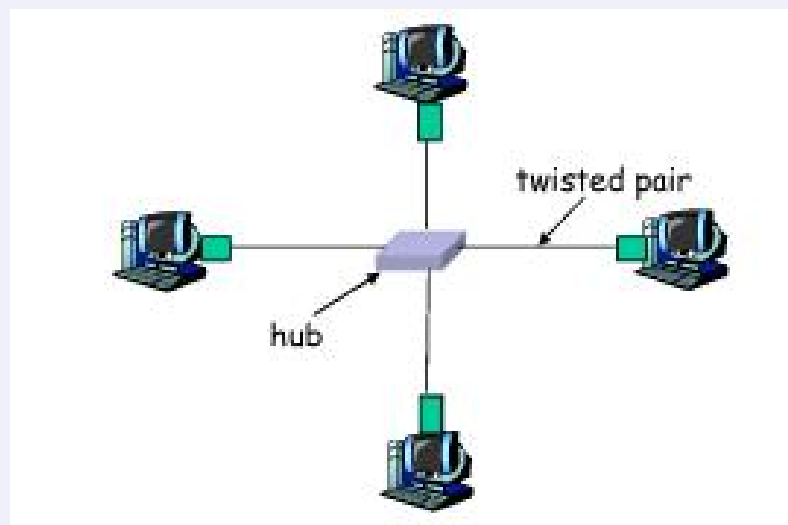
❖ 很多不同的以太网标准

- 相同的MAC协议(介质访问控制)和帧结构
- 不同的速率：2Mbps、10Mbps、100Mbps、1Gbps、10G bps
- 不同的物理层标准
- 不同的物理层媒介：光纤，同轴电缆和双绞线



10BaseT and 100BaseT

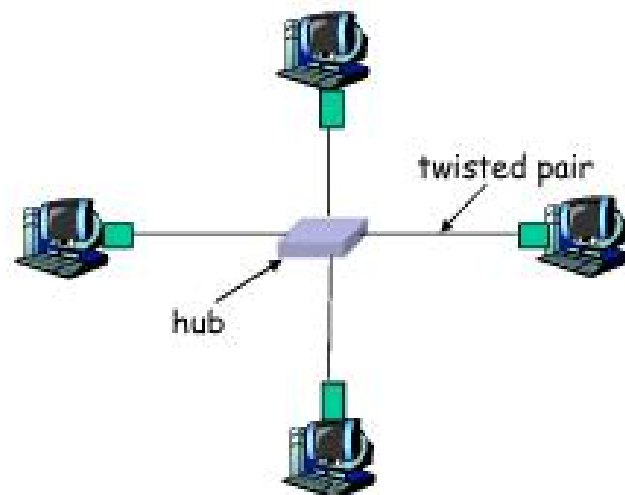
- ❖ 100Mbps速率也被称之为“fast ethernet”
- ❖ T代表双绞线
- ❖ 节点连接到HUB上:“star topology”物理上星型，
 - 逻辑上总线型，盒中共享总线
- ❖ 节点和HUB间的最大距离是100m



1、Hubs集线器

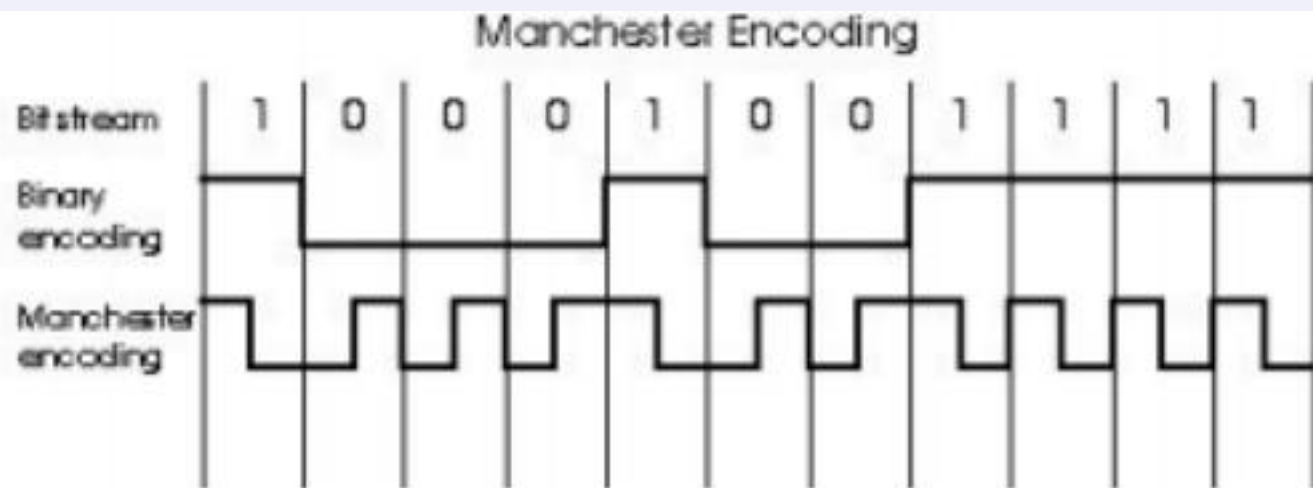
❖ Hubs本质上是物理层的中继器:

- 从一个端口收，转发到所有其他端口
- 速率一致
- 没有帧的缓存
- 在hub端口上没有CSMA/CD机制:适配器检测冲突
- 提供网络管理功能



Manchester 编码

- ❖ 在10BaseT中使用
- ❖ 每一个bit的位时中间有一个信号跳变
- ❖ 允许在接收方和发送方节点之间进行时钟同步
 - 节点间不需要集中的和全局的时钟
- ❖ 10Mbps，使用20M带宽，效率50%



Hub: 集线器

- ❖ 网段(LAN segments)：可以允许一个站点发送的网络范围
 - 在一个碰撞域，同时只允许一个站点在发送
 - 如果有2个节点同时发送，则会碰撞
 - 通常拥有相同的前缀，比IP子网更详细的前缀
- ❖ 所有以hub连到一起的站点处在一个网段，处在一个碰撞域
 - 骨干hub将所有网段连到了一起
- ❖ 通过hub可扩展节点之间的最大距离
- ❖ 通过HUB,不能将10BaseT和100BaseT的网络连接到一起

2、交换机

❖ 链路层设备：扮演主动角色(端口执行以太网协议)

- 对帧进行**存储和转发**
- 对于到来的帧，检查帧头，根据目标MAC地址进行**选择性**转发
- 当帧需要向某个(些)网段进行转发，需要使用CSMA/CD进行接入控制
- 通常一个交换机端口一个独立网段

❖ 透明：主机对交换机的存在可以不关心

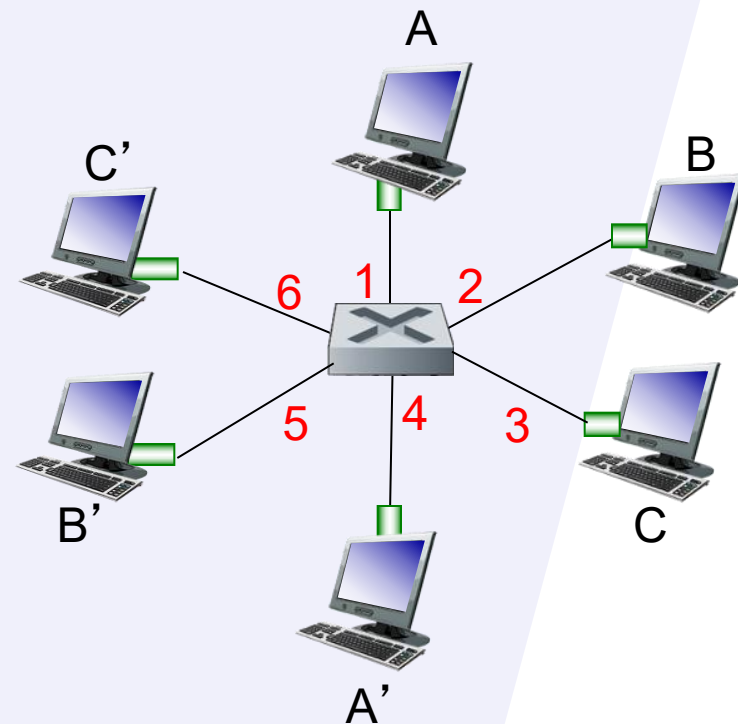
- 通过交换机相联的各节点好像这些站点是直接相联的一样
- **有MAC地址；无IP地址**

❖ 即插即用，自学习：

- 交换机无需配置

交换机：多路同时传输

- ❖ 主机有一个专用和直接到交换的连接
- ❖ 交换机缓存到来的帧
- ❖ 对每个帧进入的链路使用以太网协议，没有碰撞；全双工
 - 每条链路都是一个独立的碰撞域
 - MAC协议在其中的作用弱化了
- ❖ 交换：A-to-A'和 B-to- B'可以同时传输，没有碰撞



switch with six interfaces
(1,2,3,4,5,6)

交换机转发表

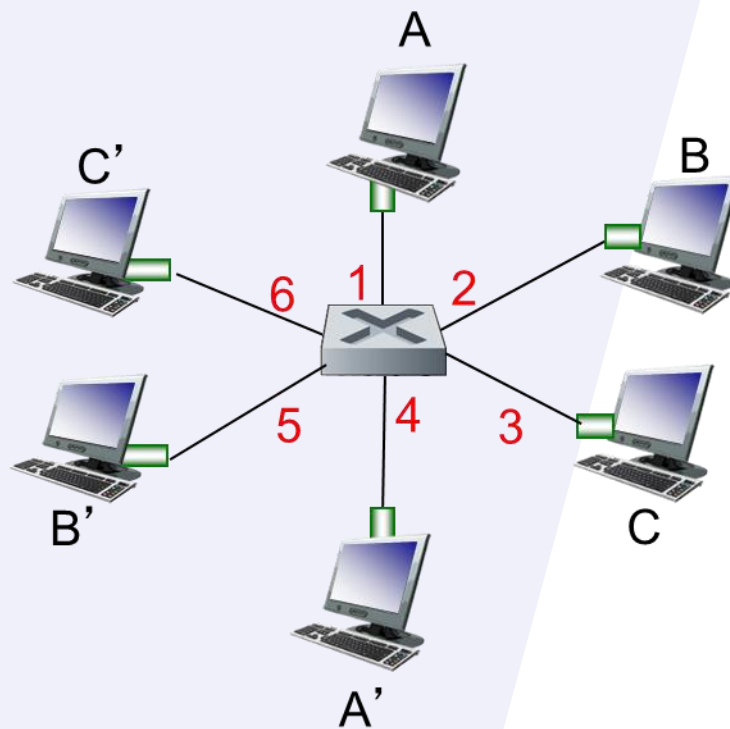
❖ Q: 交换机如何知道通过接口1到达 A, 通过接口5到达 B'?

❖ 每个交换机都有一个交换表 **switch table**, 每个表项:

- (主机的MAC地址,到达该MAC地址经过的接口, 时戳)
- 比较像路由表!

❖ Q: 每个表项是如何创建的? 如何维护的?

- 有点像路由协议?

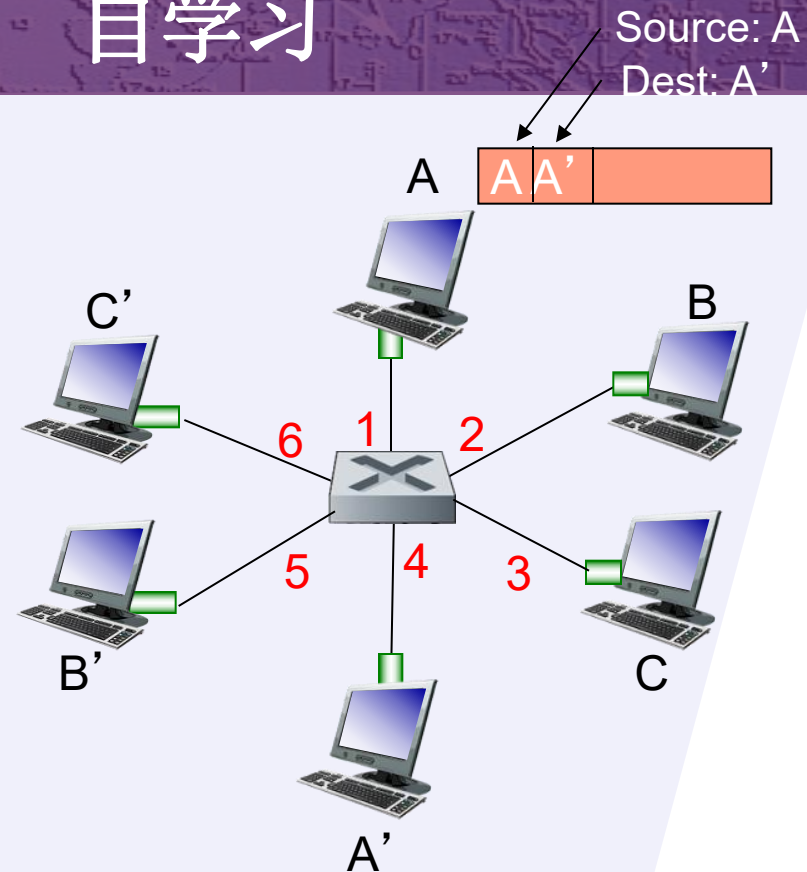


switch with six interfaces
(1,2,3,4,5,6)

交换机：自学习

❖ 交换机通过学习得到哪些主机(**mac**地址) 可以通过哪些端口到达

- 当接收到帧，交换机学习到发送站点所在的端口(网段)
- 记录发送方**MAC**地址/进入端口映射关系，在交换表中



MAC addr	interface	TTL
A	1	60

*Switch table
(initially empty)*

交换机：过滤 / 转发

当交换机收到一个帧：

1. 记录进入链路，发送主机的**MAC**地址
2. 使用目标**MAC**地址对交换表进行索引

3. **if** entry found for destination
 then{

if dest on segment from which frame arrived
 then drop the frame

else forward the frame on interface indicated

 }

else flood

过滤

转发

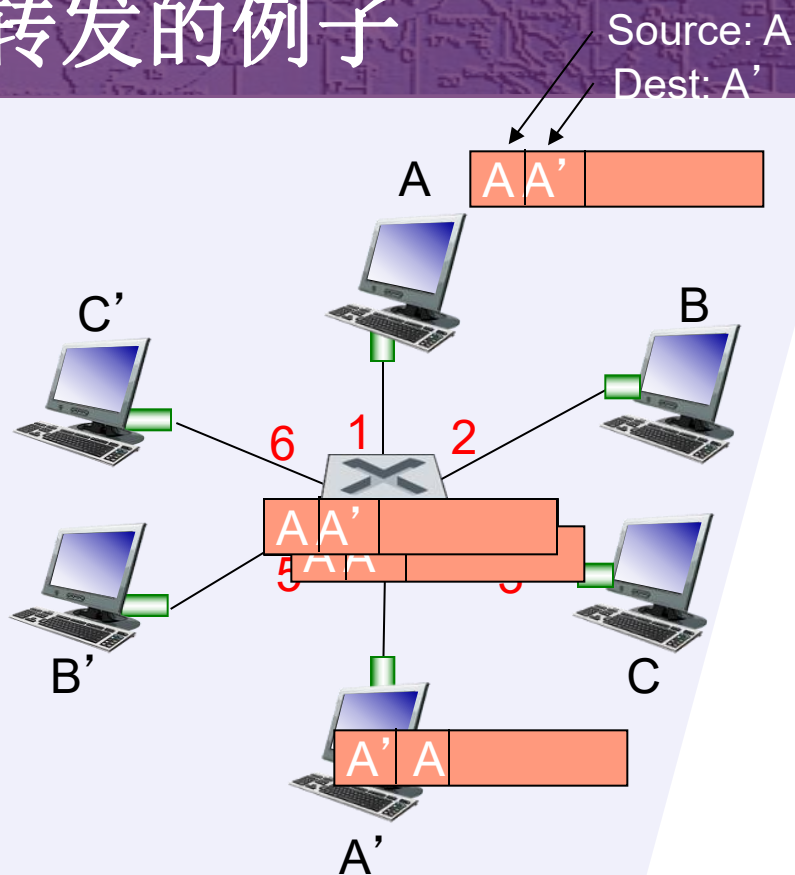
泛洪：除了帧到达的网段，向所有网络接口

自学习，转发的例子

❖ 帧的目标：A', 不知道其位置在哪：
泛洪

flood

❖ 知道目标A对应的链路：选择性发送到那个端口

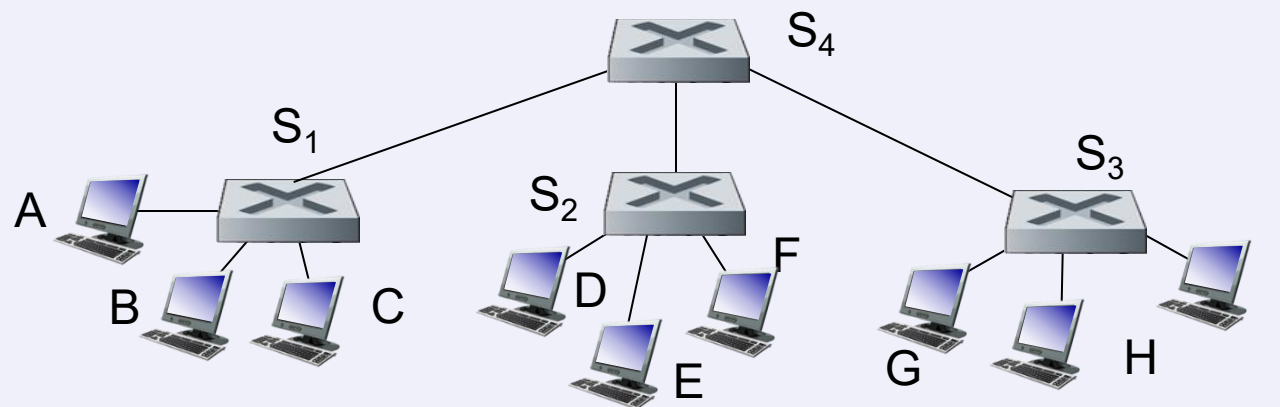


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

交换机级联

❖ 交换机可被级联到一起

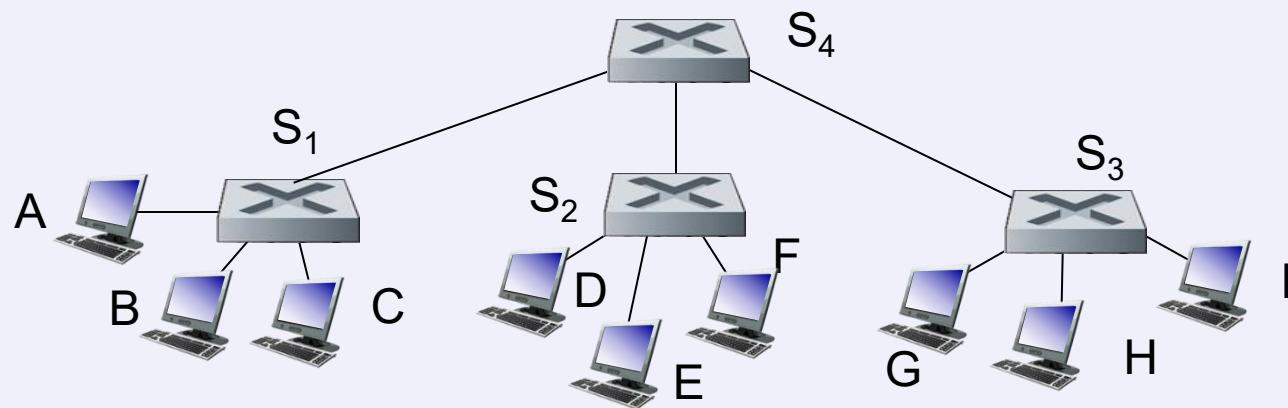


Q: A to G的发送—交换机S1 如何知道经过从S4和S3最终达到F?

A: 自学习! (和在一个交换机联接所有站点一样!)

多交换机自学习的例子

❖ 假设C向I发送帧，I给C应答



Q: 显示交换表和帧在S1, S2, S3, S4 的转发

Switches vs. routers

❖ 都是存储转发设备，但层次不同

交换机：链路层设备（检查链路层头部）

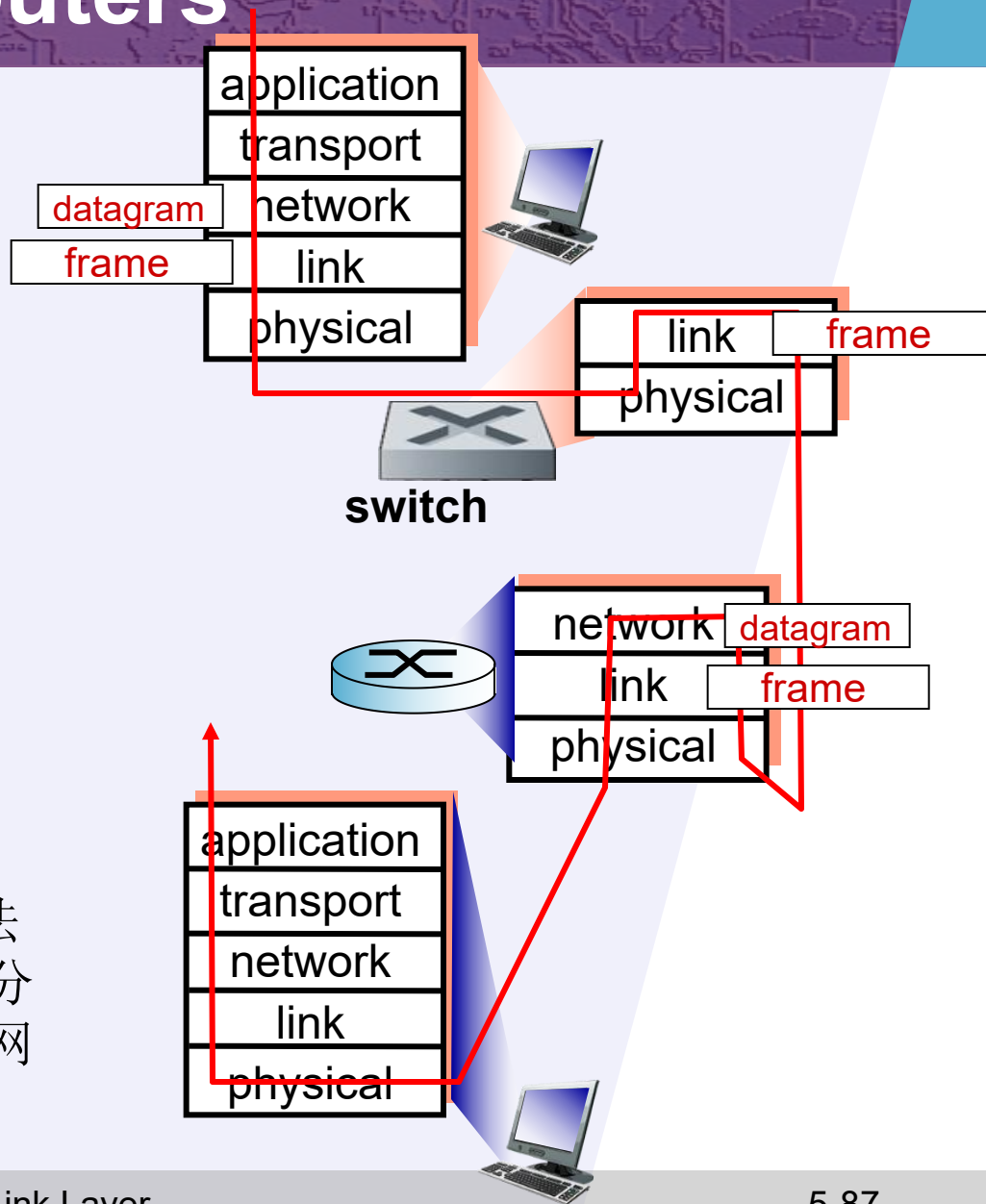
路由器：网络层设备（检查网络层的头部）

❖ 都有转发表：

交换机：维护交换表，按照MAC地址转发，执行过滤、自学习，即插即用；二层设备，速率高；

ARP表项随着站点数量增多而增

路由器维护路由表，执行路由算法可以以各种拓扑构建网络，对广播分组做限制，不是即插即用的，配置网络地址；三层设备，速率低



The background of the slide features a detailed map of East Asia, including parts of China, Korea, and Japan. The map is rendered in a muted, sepia-like tone. Overlaid on the map is a dark purple horizontal band that serves as a background for the title. The title 'Exercises' is written in a large, white, sans-serif font, centered within this band. Below the band, the main content area is a light lavender color, which transitions into a white area at the bottom right corner.

Exercises

❖ 完成《第五章课后练习与习题》文档中列出的题目