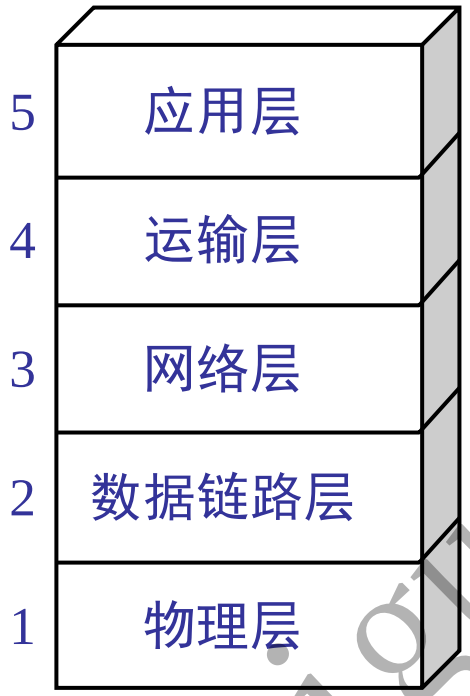
计算机网络

顾 军

计算机学院

jgu@cumt.edu.cn

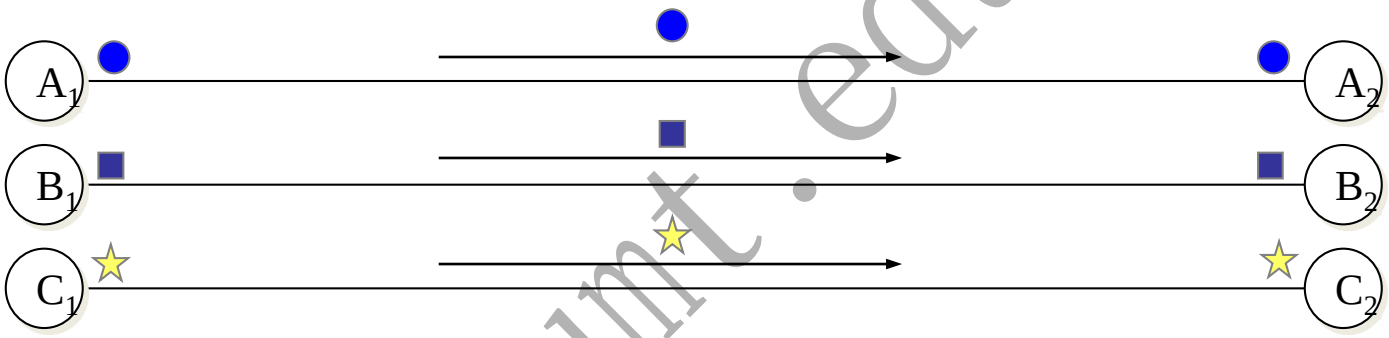
专题3：数据帧怎么到达目的结点



- 应用层(application layer)
- 运输层(transport layer)
- 网络层(network layer)
- 数据链路层(data link layer)
- 物理层(physical layer)

Q1：什么是链路和数据链路？

- 通信信道(Communication Channel)是数据传输的通路。



- 信道的作用是把携有信息的信号（电的或光的）从它的输入端传递到输出端，因此，它的最重要特征参数是信息传递能力（也叫信息通过能力）。

物理信道和逻辑信道

- 物理信道指用于传输数据信号的物理通路，它由传输介质与有关通信设备组成。
  - 根据传输介质的不同分为有线信道和无线信道
  - 按传输数据类型的不同分为数字信道和模拟信道。
- 逻辑信道指在物理信道的基础上，发送与接收数据信号的双方通过中间结点所实现的逻辑通路，用来传输数据信号。
  - 通过对物理信道的复用技术，可以构建多个逻辑通道，实现对物理信道的共享。

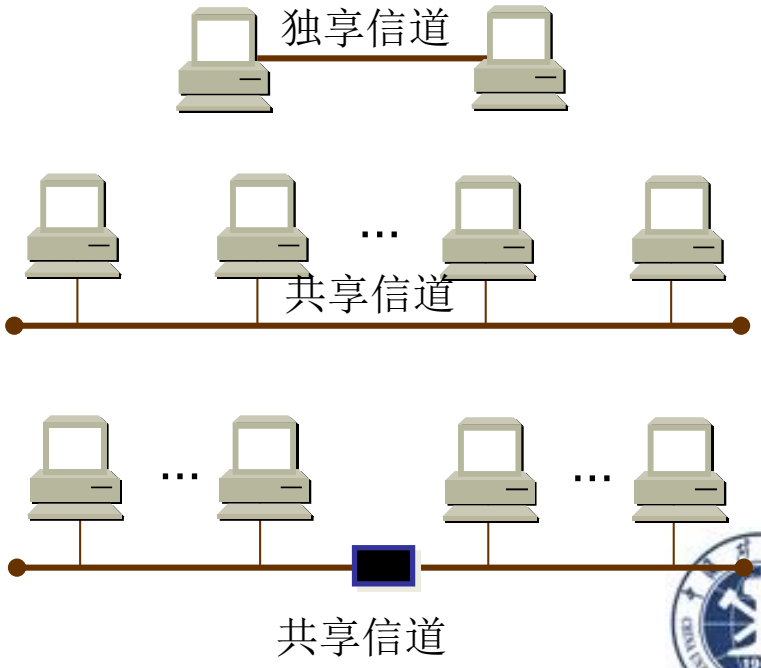
信道的实现基础是链路

- 链路(link)是一条无源的点到点的物理线路段，即相邻结点中间没有任何其他的交换结点。
  - 链路是信道的物理实现
  - 信道基本上就是在用或拟用的链路
  - 一条链路只是一条通路的一个组成部分

点到点网络环境中的两个结点。

由同一物理线路连接的共享介质环境中的结点。

位于不同物理线路上，但由工作在物理层的网络设备（中继器或集线器）互连的结点。



仅考虑物理层的信道技术是不够的

- 数据在物理通路上传输时，会因为多种因素导致传输质量下降：
  - 收发双方的接收和发送速率不匹配
  - 数据比特位的损坏或丢失
  - 共享信道时的目的结点识别
  - 数据比特流的定界
  - 等等
- 只有物理层的功能是不够的，除了物理线路外，还必须要有通信协议来控制数据的传输过程。

数据链路的引入

- 数据链路(data link)：若把实现通信协议的硬件和软件加到链路上，就构成了数据链路。

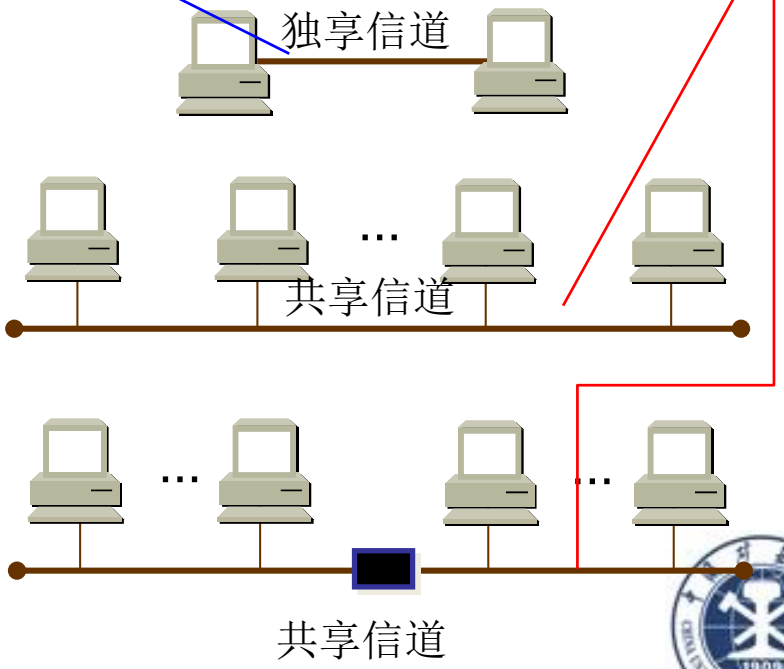
点到点链路

点到点网络环境中的两个结点。

广播链路

由同一物理线路连接的共享介质环境中的结点。

位于不同物理线路上，但由工作在物理层的网络设备（中继器或集线器）互连的结点。



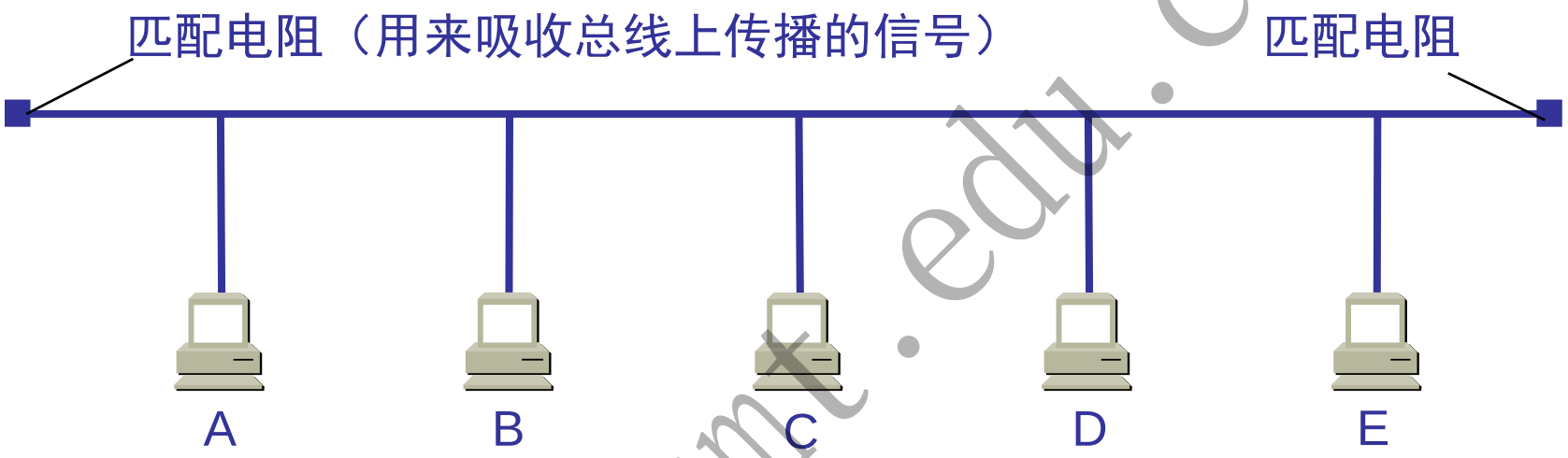
Q2：数据链路的实现方式？

- 信道的通信能力除了和介质与信号的类型有关外，还与不同网络的信道通信方式有关，由此也决定了数据链路实现方式的不同。
  - 点到点数据链路(Point-to-Point)
    - 使用一对一的点对点通信方式。
  - 广播数据链路(Broadcasting)
    - 使用一对多的广播通信方式，通过向所有站点发送分组的方式传输信息，过程比较复杂。

点对点数据链路

- 没有信道竞争，几乎不存在介质访问控制问题，但是会浪费一些带宽。
- 点到点通信的最主要应用是广域网，以及成千上万用户在家里使用MODEM和拨号电话线连接到Internet。
- HDLC(High-level Data Link Control)
  - 同步、面向位，具有差错控制、流量控制功能
  - 能实现可靠传输，但是现在已经很少使用了
- PPP(Point-to-Point Protocol)
  - Internet在点到点链路上传输网络层分组的协议

广播数据链路



- 广播网络的信道上连接的主机很多，因此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，即解决发送过程中的信道资源竞争和冲突问题。
- 无线广播电台和局域网(LAN)大多采用这种方式传播分组信息。

广播数据链路的信道分配策略

- 静态分配策略
  - 静态划分信道：用户只要分配到了信道就不会和其他用户发生冲突，如频分复用、时分复用、码分复用等，不仅控制协议简单，而且信道利用率较高。
  - 适用于网络节点数目少而固定，且每个节点都有大量数据要发送的场合。
  - 但是代价较高，不适合于局域网使用。

广播数据链路的信道分配策略

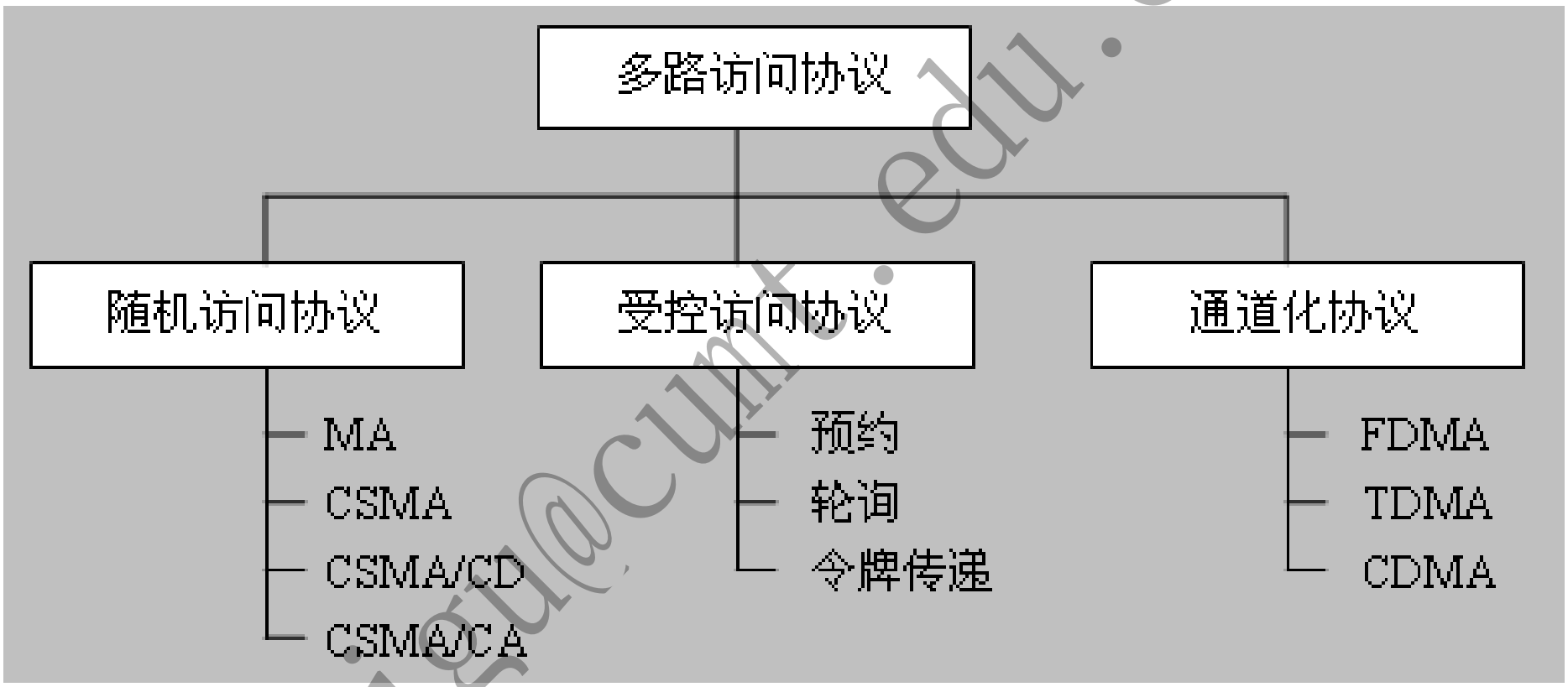
- 动态分配策略
  - 又称为动态媒体接入控制(多点接入)，信道并非固定分配给用户，而是在各站点有数据要发送时，才占用信道进行数据传输，本质上属于异步时分多路复用。
  - 适合于局域网环境，包括受控接入和随机接入。

广播链路的动态信道分配策略

- 受控接入：发送节点首先获得信道的使用权，如多点线路探询(polling)、或轮询，只有获得使用权后再发送数据，因而不会出现碰撞和冲突。
  - 当网络负载较重时，可以获得很高的信道利用率。
- 随机接入：各个节点有数据就发送，发生碰撞之后再采取措施解决。
  - 适用于负载较轻的网络，网络延迟较短。

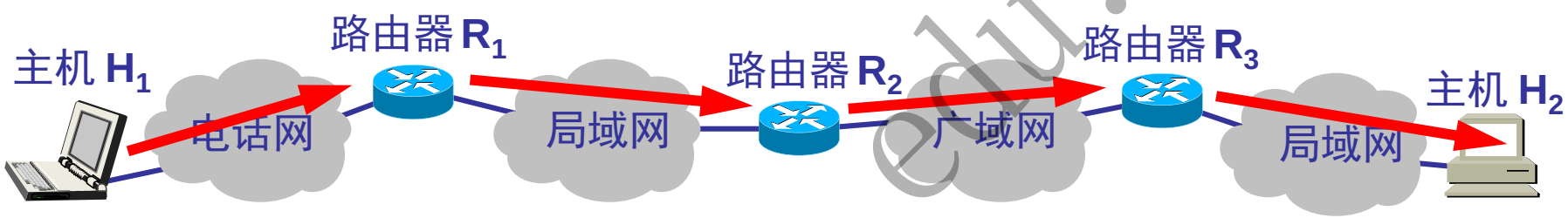
现在的有线局域网中部署了具有帧过滤功能的交换机，已经不再是传统意义上的共享信道广播式通信了，因此碰撞和冲突问题基本可以忽略了。但是在无线局域网中依然需要考虑信道冲突问题。

多路访问协议

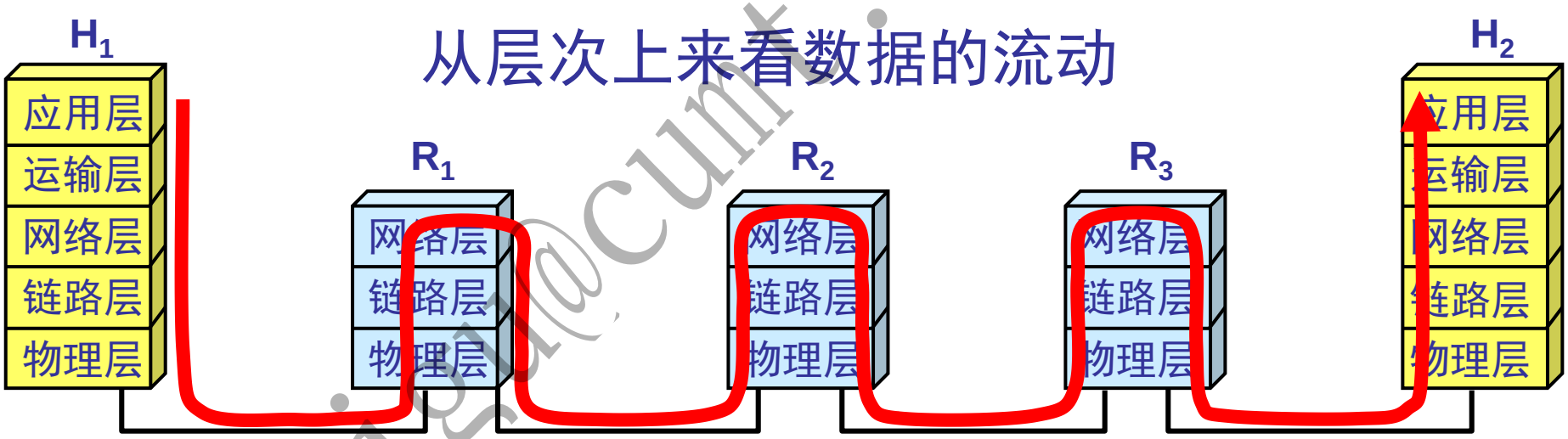


Q3：数据链路层是什么东东？

主机 H_1 向 H_2 发送数据



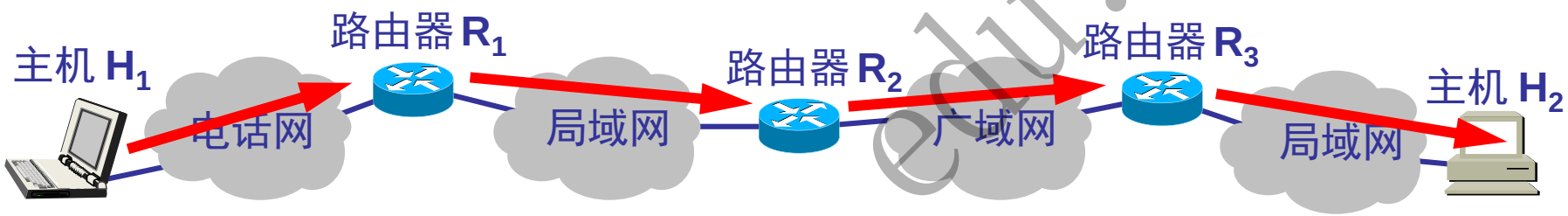
从层次上来看数据的流动



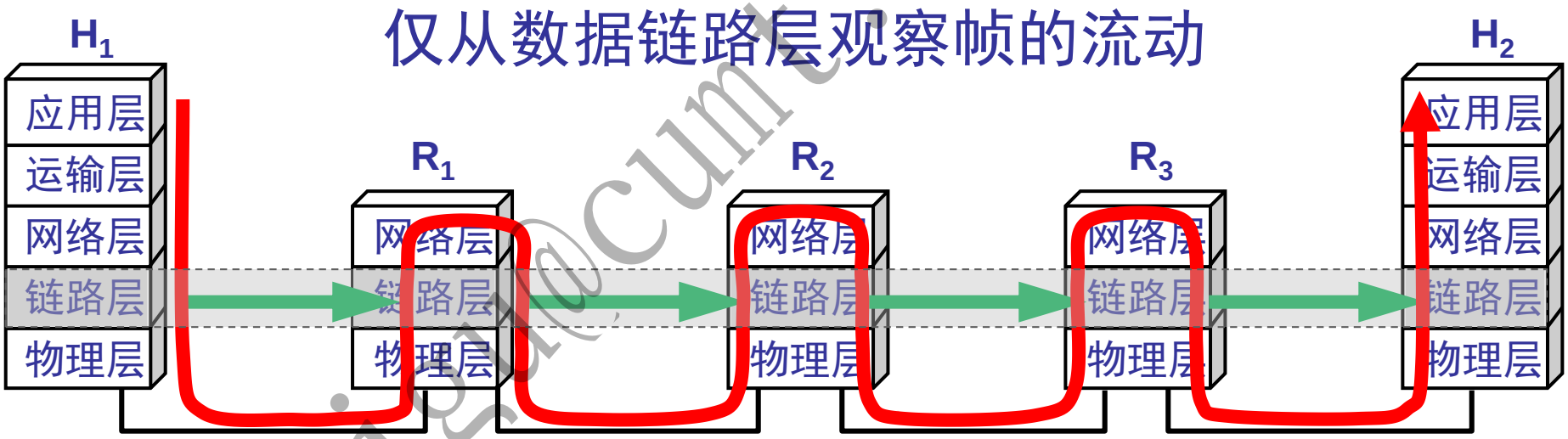
物理层的比特流是对数据链路帧的信号描述

数据链路层的简单模型

主机 H_1 向 H_2 发送数据



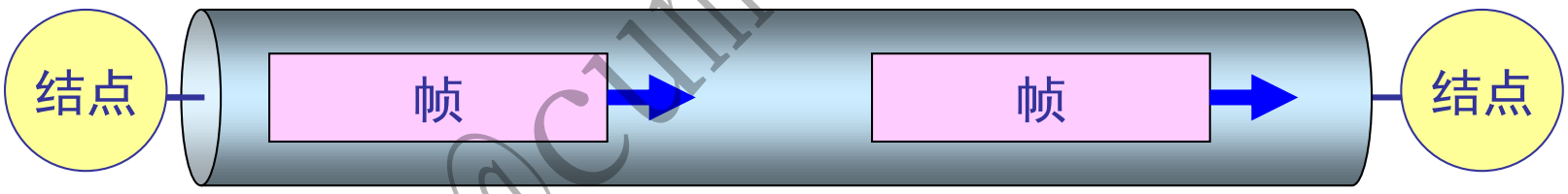
仅从数据链路层观察帧的流动



数据链路层可以不考虑物理层的实现细节

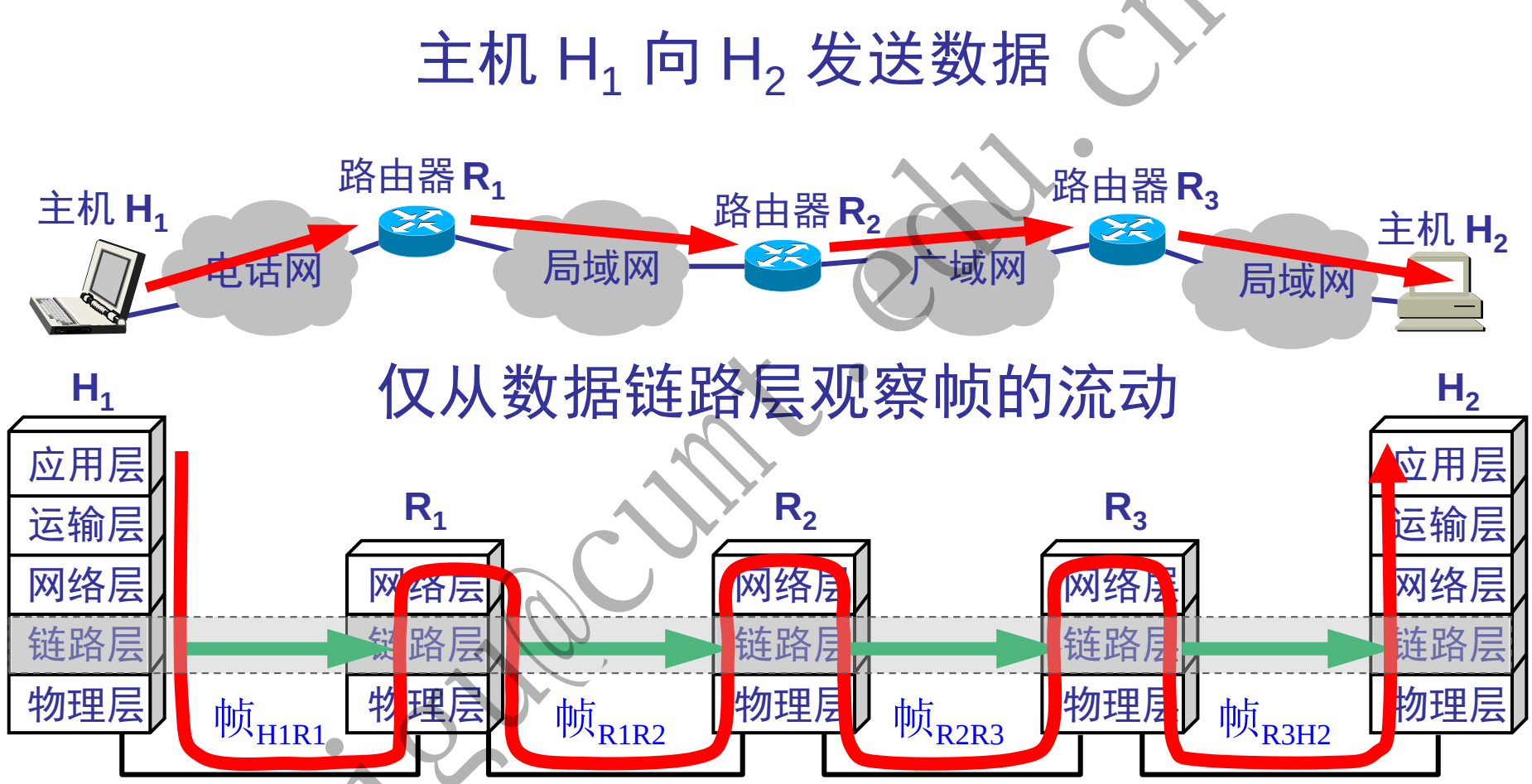
数据链路层形象描述

- 数据链路层是为了解决两个相邻结点间的无差错传输而建立的协议分层。
- 两个对等的数据链路层之间就像通过一个数字管道相连一样，而在这条数字管道上传输的数据单位是帧。



数据链路层不必考虑物理层如何实现比特传输的细节。甚至还可以更简单地设想好像是沿着两个数据链路层之间的水平方向把帧直接发送到对方

不同数据链路协议有不同帧格式



帧中的源和目的地址只与当前链路两端的结点有关

Q4：数据链路层可以干什么？

数据链路层的三个基本功能

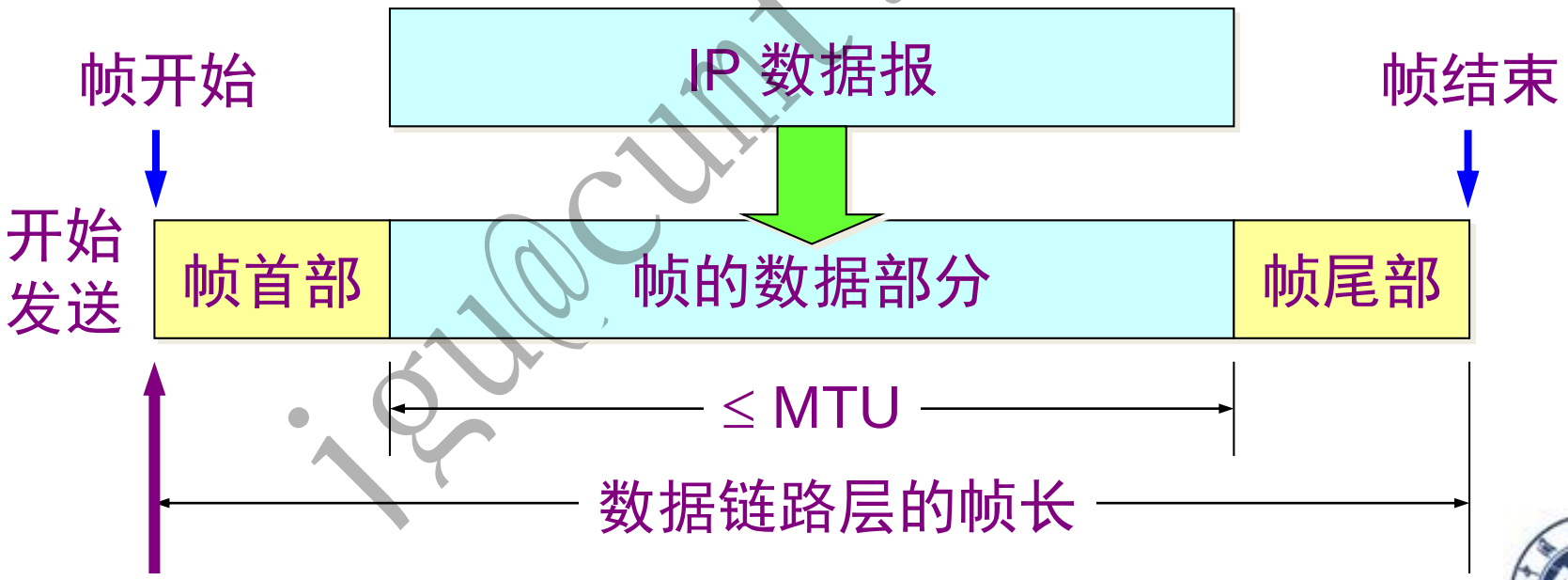
(1) 封装成帧

(2) 透明传输

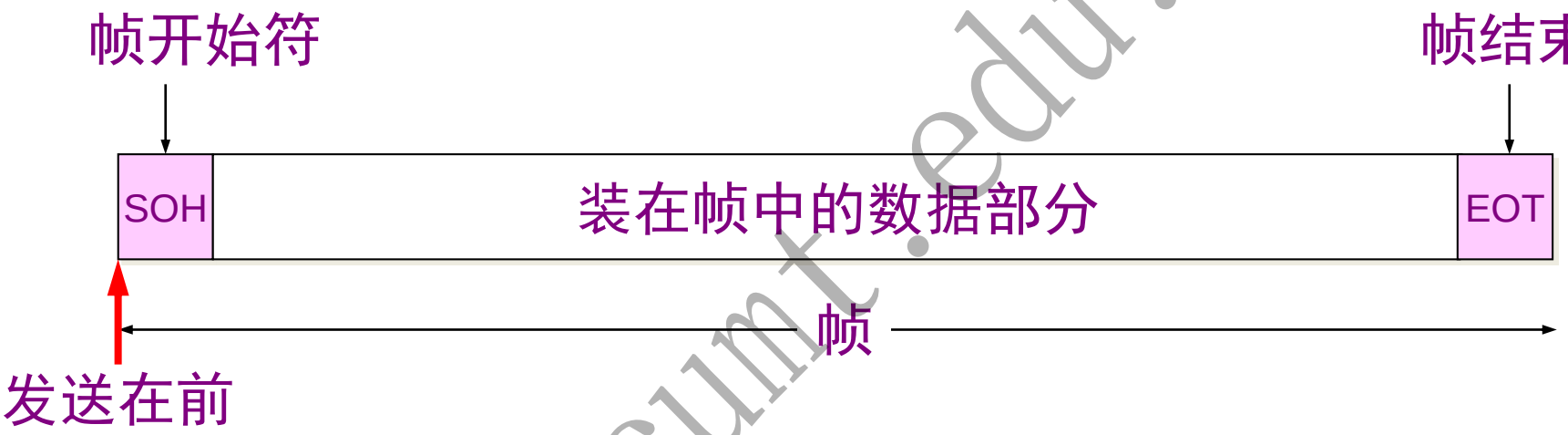
(3) 差错控制

1. 封装成帧

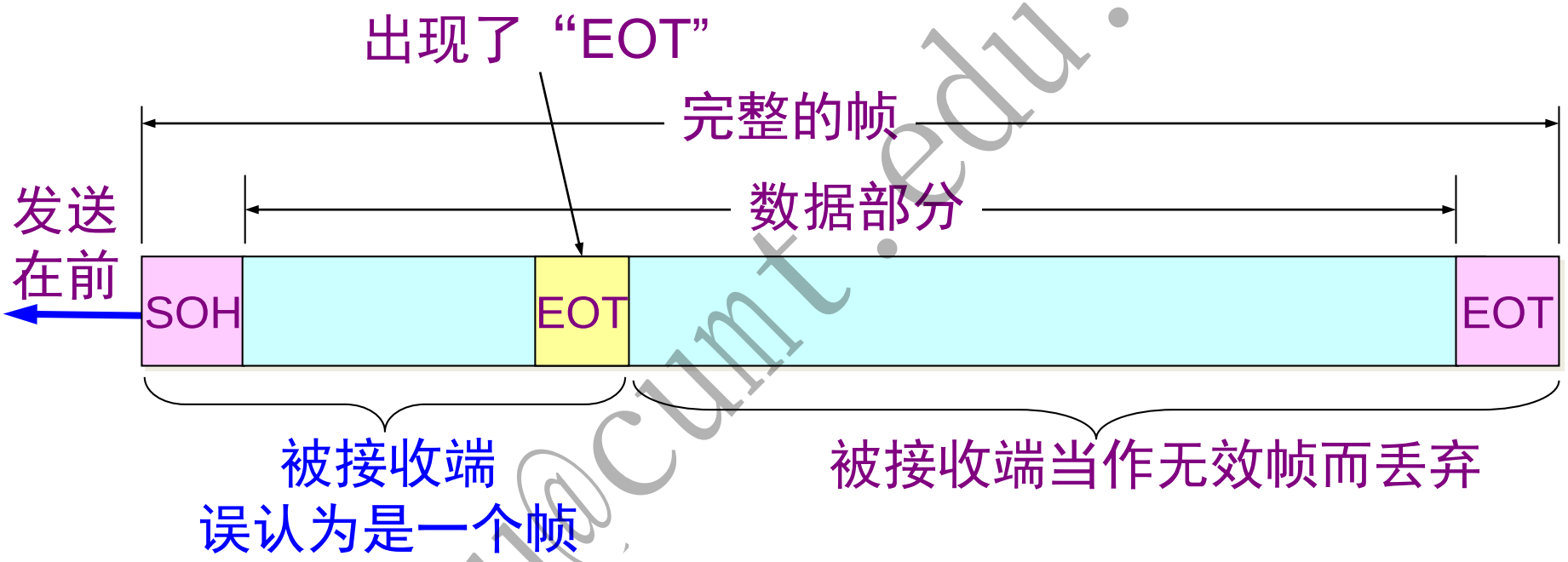
- 封装成帧(framing)就是在一段数据的前后分别添加首部和尾部，然后就构成了一个帧。确定帧的界限。
- 首部和尾部的一个重要作用就是进行帧定界。



用控制字符进行帧定界的方法举例



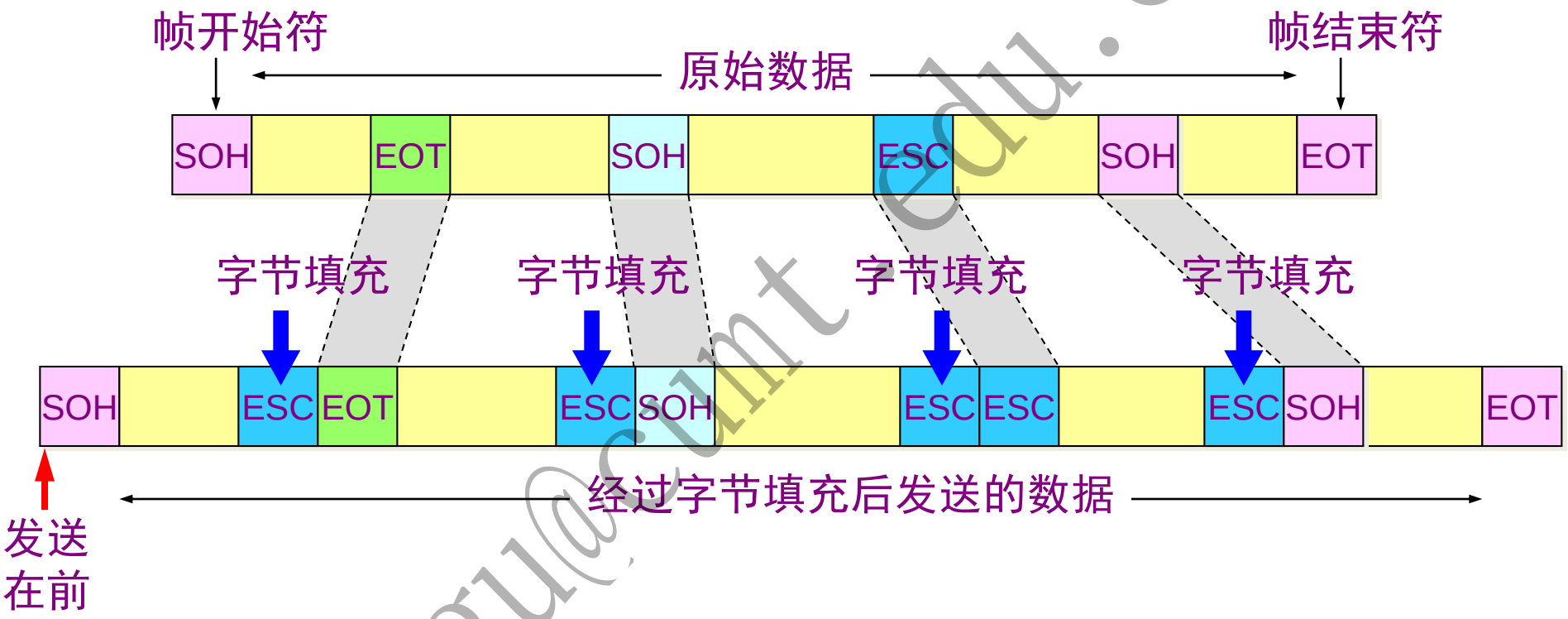
2. 透明传输



解决透明传输问题

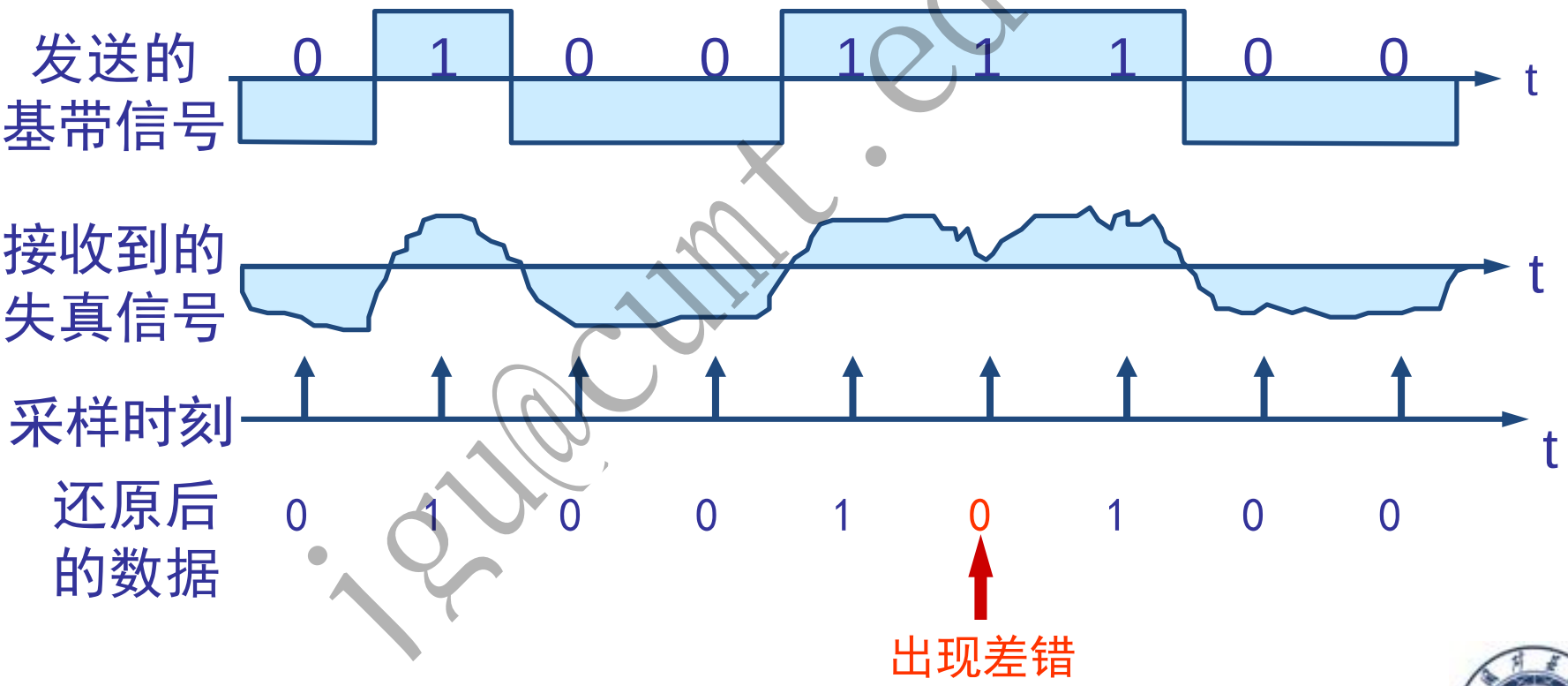
- 字节填充(byte stuffing)或字符填充(character stuffing)
  - 发送端的数据链路层在数据中出现的控制字符“SOH”或“EOT”的前面插入一个转义字符“ESC”(其十六进制编码是1B)。
  - 接收端的数据链路层在将数据送往网络层之前删除插入的转义字符。
  - 如果转义字符也出现数据当中，那么应在转义字符前面插入一个转义字符。
  - 当接收端收到连续的两个转义字符时，就删除其中前面的一个。

用字节填充法解决透明传输的问题



3. 差错检测

- 在传输过程中可能会产生比特差错：1 可能会变成 0 而 0 也可能变成 1。



差错产生的原因

- 热噪声是由传输介质导体的电子热运动产生的
  - 热噪声时刻存在，幅度较小且强度与频率无关，但频谱很宽，是一类随机噪声。
  - 热噪声引起的差错称随机差错。
  - 热噪声往往是孤立的、个别的。
- 与热噪声相比，冲击噪声幅度较大，是引起传输差错的主要原因。
  - 冲击噪声的持续时间要比数据传输中的每比特发送时间要长（如外界磁场的变换、电源开关的跳变等），因而冲击噪声会引起相邻多个数据位出错。
  - 冲击噪声引起的传输差错称为突发差错。
  - 突发的、连续的

差错控制

- 在一段时间内，传输错误的比特占所传输比特总数的比率称为误码率 BER (Bit Error Rate)。
  - 误码率与信噪比有很大的关系。
- 为了保证数据传输的可靠性，在计算机网络传输数据时，必须采用各种差错检测措施。
- 差错控制(error control)是指在数据通信过程中能发现或纠正差错，将差错限制在尽可能小的允许范围内。
  - 在数字通信中利用编码方法对传输中产生的差错进行控制，以提高数字消息传输的准确性。
  - 差错检测是差错控制的基础。
  - 差错控制编码又可分为检错码和纠错码。

奇偶校验码

- 奇偶校验码是一种最简单的检错码。其原理是：通过增加冗余位来使得码字中“1”的个数保持为奇数（奇校验）或偶数（偶校验）。
  - 例如，偶校验：11010100，11011011
- 在实际使用时，奇偶校验可分为以下三种方式。

(1) 垂直奇偶校验

(2) 水平奇偶校验

(3) 水平垂直奇偶校验

垂直奇偶校验码

- 垂直奇偶校验码把数据分成若干组，一组数据排成一行，再加一行校验码。
  - 针对每一列采用奇校验 或 偶校验
- 例: 有32位数据10100101 00110110 11001100 10101011

垂直奇校验	垂直偶校验	
10100101	10100101	数据
00110110	00110110	
11001100	11001100	
10101011	10101011	
00001011	11110100	校验

水平奇偶校验码

- 水平奇偶校验码对每一个数据的编码添加校验位，使信息位与校验位处于同一行。
- 例: 有32位数据10100101 00110110 11001100 10101011

	水平奇校验		水平偶校验	
数据	10100101	1	10100101	0
	00110110	1	00110110	0
	11001100	1	11001100	0
	10101011	0	10101011	1
		校验		校验

水平垂直奇偶校验码

- 水平垂直奇偶校验码就是同时用水平校验和垂直校验。
- 例: 有32位数据10100101 00110110 11001100 10101011

	水平奇校验	奇水平	水平偶校验	偶水平	
数据	10100101	1	10100101	0	
	00110110	1	00110110	0	
	11001100	1	11001100	0	
	10101011	0	10101011	1	
	00001011	0	11110100	1	校验

奇偶校验码举例

	S	Y	S	T	E	M	校验字符
	字符1	字符2	字符3	字符4	字符5	字符6	水平校验
Bit0	1	1	1	0	1	1	1
Bit1	1	0	1	0	0	0	0
Bit2	0	0	0	1	1	1	1
Bit3	0	1	0	0	0	1	0
Bit4	1	1	1	1	0	0	0
Bit5	0	0	0	0	0	0	0
Bit6	1	1	1	1	1	1	0
垂直校验	0	0	0	1	0	0	0

循环冗余检验

- CRC(Cyclic Redundancy Code)主要用于串行传输、网络同步通信及磁表面存储等场合，即给信息码右边加上几位校验码，以增加整个编码系统的码距和查错纠错能力。
- CRC可以用于对任意长度消息的编码，能提供出色的检错能力。
- 尽管CRC具有复杂的数学基础，但CRC计算过程可以用硬件加速完成，而且所需硬件非常简单，只是一个带有异或门的移位寄存器而已，其中在某些比特之间插入的异或门用来执行异或运算。

循环冗余码CRC

- 在发送端，先把数据划分为组。假定每组 k 个比特。
- 假设待传送的一组数据 $M = 101001$（现在 $k = 6$）。我们在 M 的后面再添加供差错检测用的 n 位冗余码一起发送。

模2运算规则

- 模2加减: 异或逻辑　　0⊕0=0;　0⊕1=1;

  1⊕0=1;　1⊕1=0.

- 模2乘:

$$\begin{array}{r} 1010 \\ \times\quad 101 \\ \hline 1010 \\ 0000 \\ 1010 \\ \hline 100010 \end{array}$$

模2除:

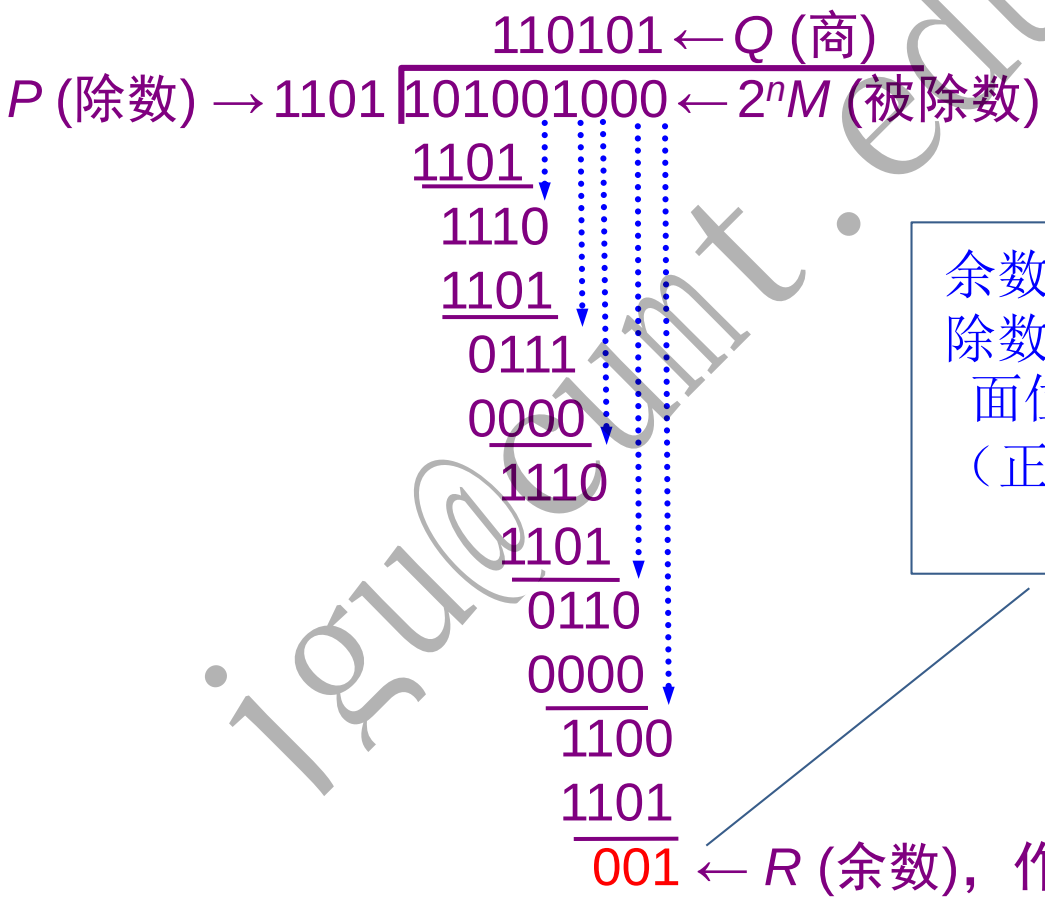
$$\begin{array}{r} \quad 101 \\ \hline 101\,/\,10000 \\ \quad 101 \\ \hline \quad 010 \\ \quad 000 \\ \hline \quad 100 \\ \quad 101 \\ \hline \quad 01 \end{array}$$

冗余码的计算

- 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。
- 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。

冗余码的计算举例1

- 现在 $k = 6, M = 101001$。
- 设 $n = 3$, 除数 $P = 1101$，被除数是 $2^nM = 101001000$。



余数的位数一定只能是比除数位数少一位，哪怕前面位是0，甚至是全为0（正好整除），也都不能省略。

冗余码的计算举例1

- 模 2 运算的结果是：商 $Q = 110101$，

  余数 $R = 001$。

- 把余数 R 作为冗余码添加在数据 M 的后面发送出去。发送的数据是：$2^nM + R$

  即：101001001，共 $(k + n)$ 位。

冗余码的计算举例2

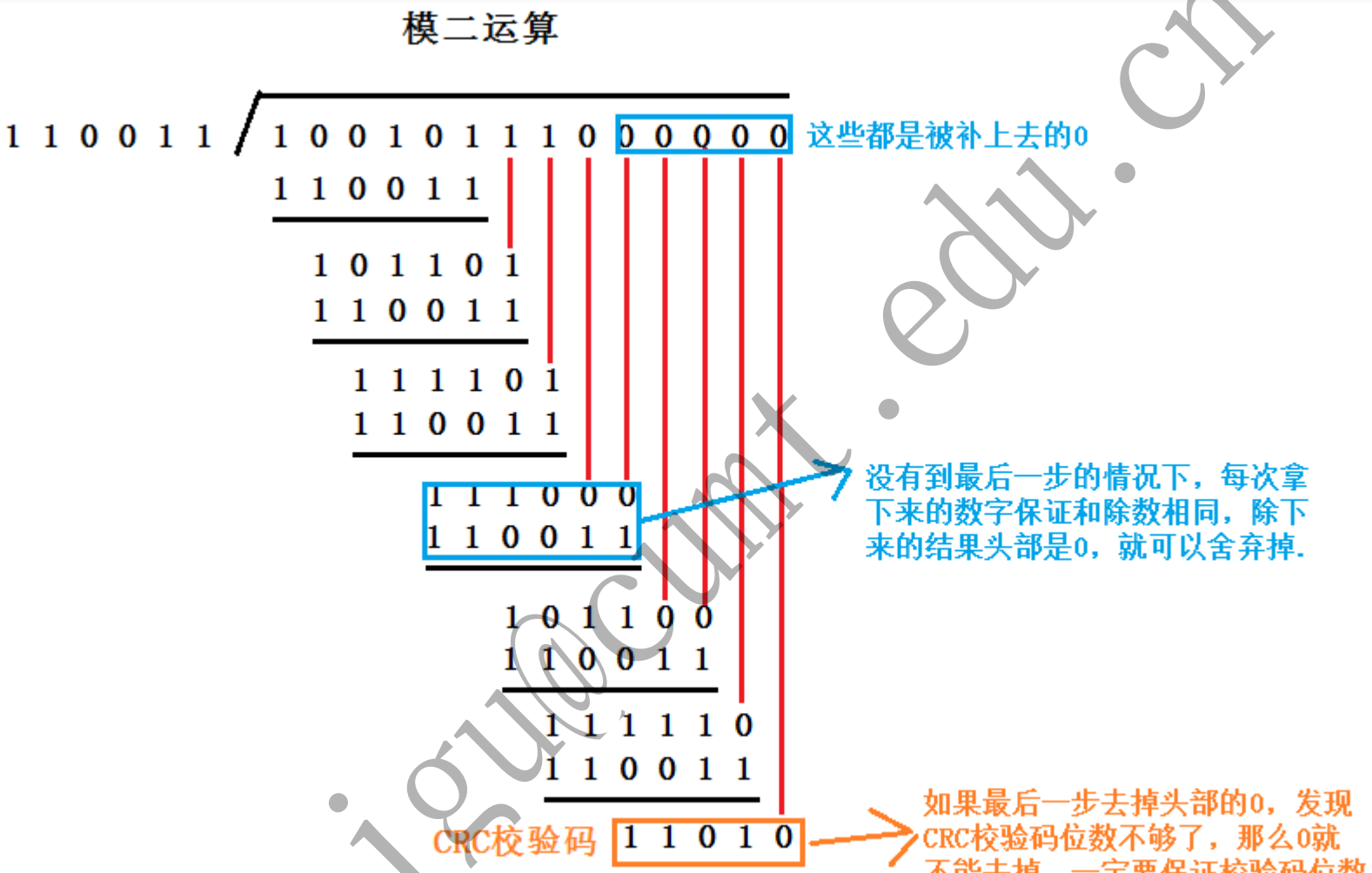
假设$P(X) = X^5 + X^4 + X + 1$，要发送的二进制序列为*100101110*，求*CRC*校验码是多少?

（1）*P*(x)多项式转换为二进制串为：110011

（2）CRC校验码的位数为多项式二进制串减一位，所以为5位

（3）给要发送的二进制序列上补充5位 -> *100101110*00000

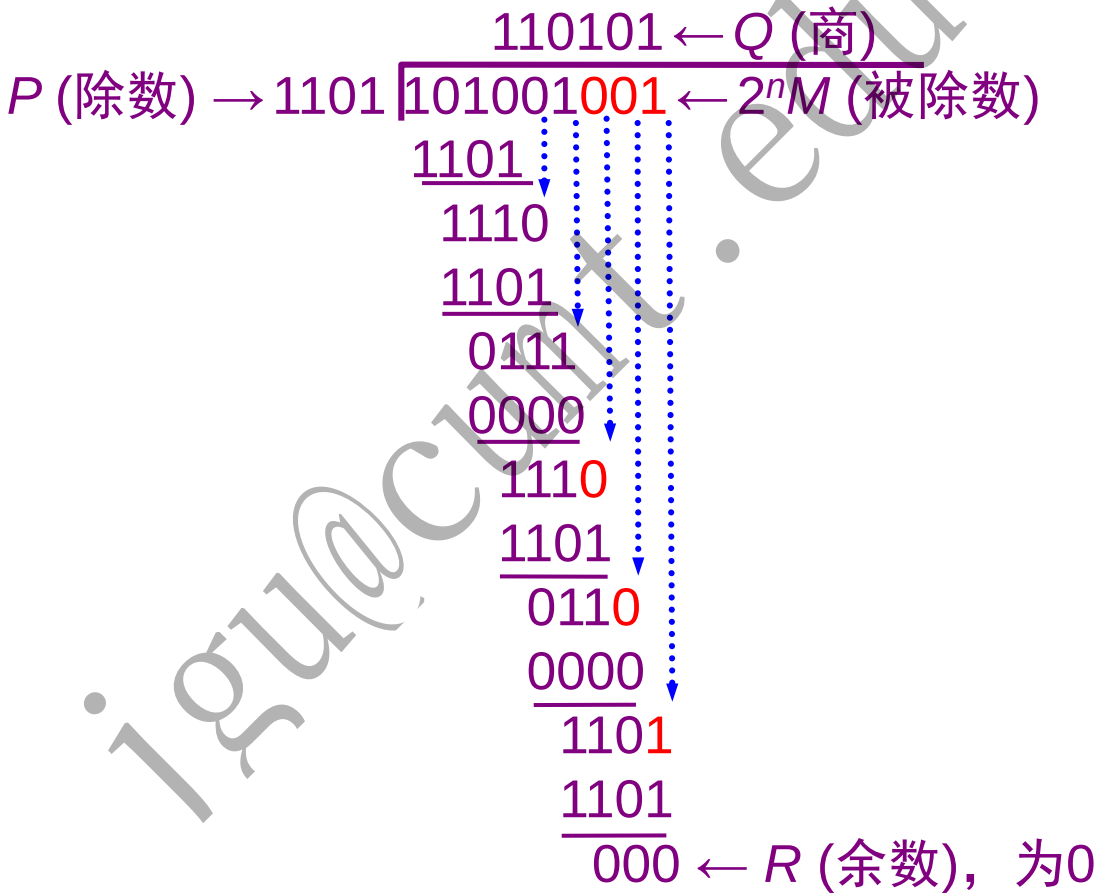
冗余码的计算举例2



*100101110*00000 -> *100101110*11010

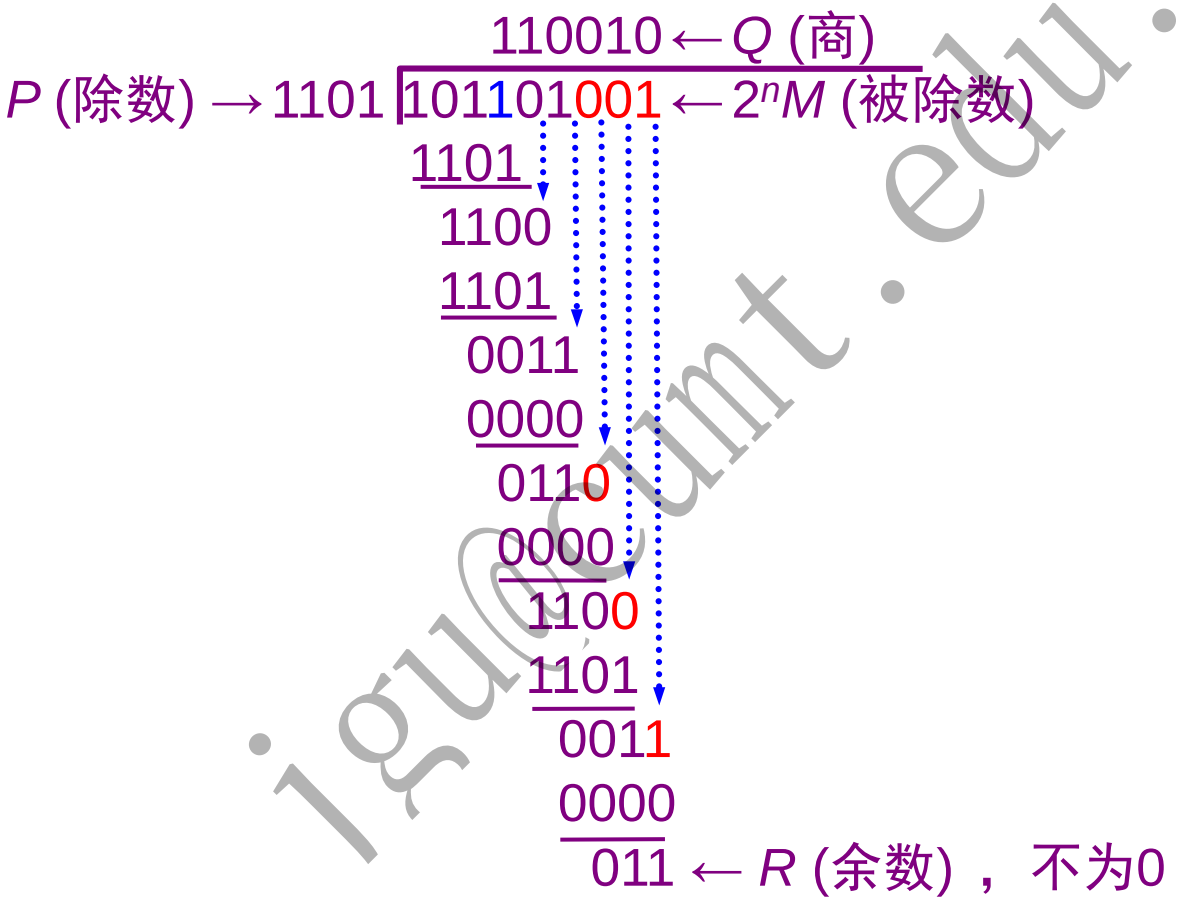
接收端对收到的每一帧进行 CRC 检验

(1) 若得出的余数 $R = 0$，则判定这个帧没有差错，就接受(accept)。



接收端对收到的每一帧进行 CRC 检验

(2) 若余数 $R \neq 0$，则判定这个帧有差错，就丢弃。



生成多项式G(X)

- 显然，这种检测方法并不能确定究竟是哪一个或哪几个比特出现了差错。
- 但是，只要经过严格的挑选，并使用位数足够多的除数 *P*，那么出现检测不到的差错的概率就很小很小，这样的P称为生成多项式*G*(x)。
- 生成多项式的选择是能够产生具有良好检错能力的CRC的关键。
- 生成多项式*G*(*x*)是接收方与发送方的约定，在整个传输过程中始终不变。

常用生成多项式

码元距离(海明距离)：两个码组中对应码位上码元不同的个数，反映的是码组之间的差异程度。

N	K	码距d	G(x)多项式	G(x)
7	4	3	x^3+x+1	1011
7	4	3	x^3+x^2+1	1101
7	3	4	$x^4+x^3+x^2+1$	11101
7	3	4	x^4+x^2+x+1	10111
15	11	3	x^4+x+1	10011
15	7	5	$x^8+x^7+x^6+x^4+1$	111010001
31	26	3	x^5+x^2+1	100101
31	21	5	$x^{10}+x^9+x^8+x^6+x^5+x^3+1$	11101101001
63	57	3	x^6+x+1	1000011
63	51	5	$x^{12}+x^{10}+x^5+x^4+x^2+1$	1010000110101
1041	1024		$x^{16}+x^{15}+x^2+1$	1100000000000101

- 00和01两组码的码距为1
- 011和100的码距为3
- 11000与10011之间的距离为3
- 码字10011001和11110101之间的码距为4

常用生成多项式

码元距离(海明距离)：两个码组中对应码位上码元不同的个数，反映的是码组之间的差异程度。。

N	K	码距d	G(x)多项式	G(x)
7	4	3	x^3+x+1	1011
7	4	3	x^3+x^2+1	1101
7	3	4	$x^4+x^3+x^2+1$	11101
7	3	4	x^4+x^2+x+1	10111
15	11	3	x^4+x+1	10011
15	7	5	$x^8+x^7+x^6+x^4+1$	111010001
31	26	3	x^5+x^2+1	100101
31	21	5	$x^{10}+x^9+x^8+x^6+x^5+x^3+1$	11101101001
63	57	3	x^6+x+1	1000011
63	51	5	$x^{12}+x^{10}+x^5+x^4+x^2+1$	1010000110101
1041	1024		$x^{16}+x^{15}+x^2+1$	1100000000000101

- 码字集中所有码字之间码距的最小值称为最小码距
- 最小码距的值越大，从一个编码错成另一个编码的可能性越小，则其检错、纠错能力也就越强。

CRC又叫（N, K）码

- N位：整个编码长度
- K位：信息码（原始报文）长度
- 校验码长度：R (= N - K) 位

$M(x)= x^3 + x= 1010$

$M(x) \cdot x^3= x^6 + x^5= 1010000$

$P(x)= x^3 + x +1= 1011$

$$\frac{M(x) \cdot x^3}{P(x)} = \frac{1010000}{1011} = 1001 + \frac{011}{1011}$$

$M(x) \cdot x^3 + R(x) = 1010000 + 011$

$= 1010011$

编好的循环校验码称为(7,4)码

通信与网络中常用的CRC

- 在数据通信与网络中，通常待校验的信息码长度k相当大，可能是由一千甚至数千数据位构成一帧，再采用CRC码产生r位的校验位。
- 此时，只能检测出错误，而不能纠正错误。
- 当接收方检测到CRC码字出错，通过要求重发的方式来实现纠错。
- 因此，计算机网络在数据链路层传送的帧中，广泛使用CRC检测传输差错，再在传输方案中增加ARQ机制即可确保消息正确传递。

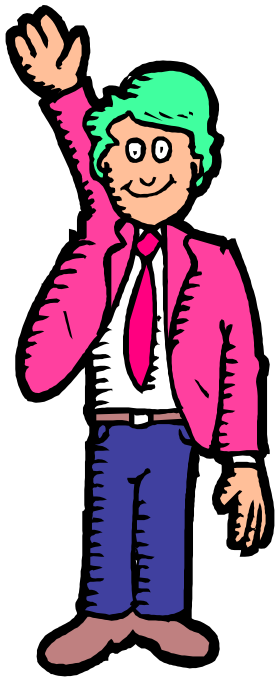
标准CRC生成多项式具有极高的检错率

名称	生成多项式	16进制简记式*	标准引用
CRC-4	x^4+x+1	3	ITU G.704
CRC-8	$x^8+x^5+x^4+1$	0x31	
CRC-8	$x^8+x^2+x^1+1$	0x07	
CRC-8	$x^8+x^6+x^4+x^3+x^2+x^1$	0x5E	
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$	0x80F	
CRC-16	$x^{16}+x^{15}+x^2+1$	0x8005	IBM SDLC
CRC16-CCITT	$x^{16}+x^{12}+x^5+1$	0x1021	ISO HDLC, ITU X.25, V.34/V.41/V.42, PPP-FCS
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	0x 04C11DB7	ZIP, RAR, IEEE 802 LAN/FDDI, IEEE 1394, PPP-FCS
CRC-32c	$x^{32}+x^{28}+x^{27}+\ldots+x^8+x^6+1$	0x1EDC6F41	SCTP

最高次幂省略，其余位次对应幂次

帧检验序列 FCS

- 在数据后面添加上的冗余码称为帧检验序列 FCS (Frame Check Sequence)。
- 循环冗余检验 CRC 和帧检验序列 FCS并不等同。
  - CRC 是一种常用的检错方法，而 FCS 是添加在数据后面的冗余码。
  - FCS 可以用 CRC 这种方法得出，但 CRC 并非用来获得 FCS 的唯一方法。



THANK YOU!