



计算机网络 (第 8 版)

第 3 章

数据链路层

计算机网络体系结构

OSI 的七层协议体系结构



(a)

TCP/IP 的四层协议体系结构



(b)

五层协议的体系结构



(c)

3.1

使用点对点信道的数据链路层

3.2

点对点协议 PPP

3.3

使用广播信道的数据链路层

3.4

扩展的以太网

3.5

高速以太网

3.1

使用点对点 信道的数据 链路层

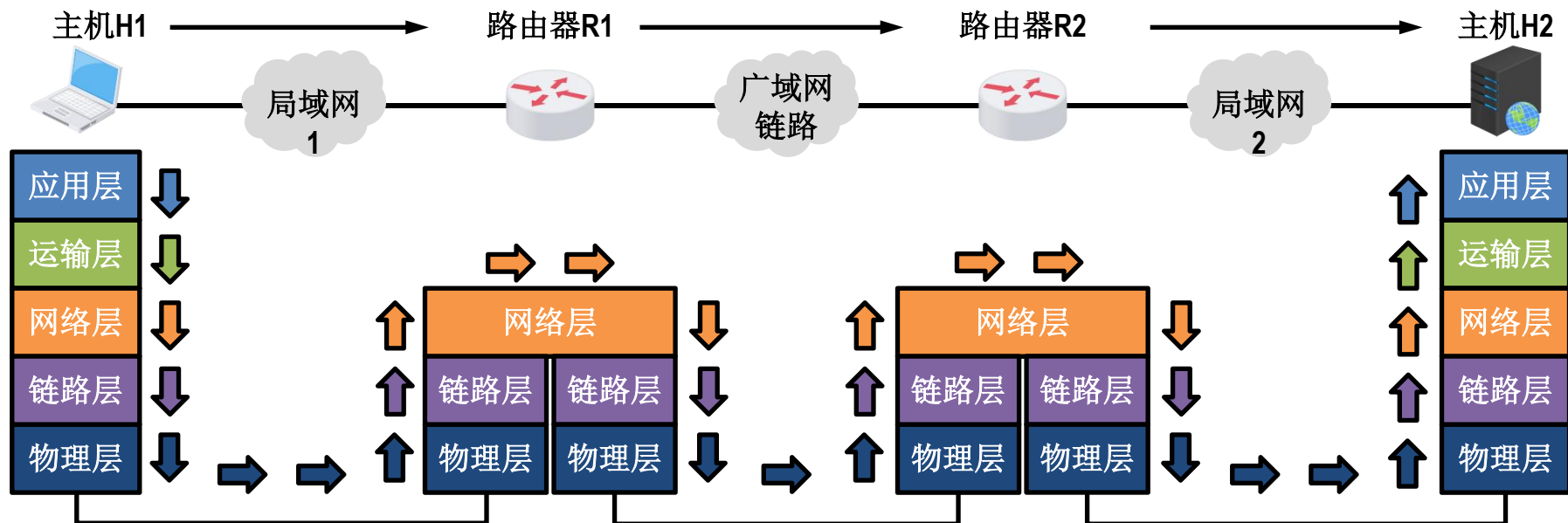
3.1.1

数据链路和帧

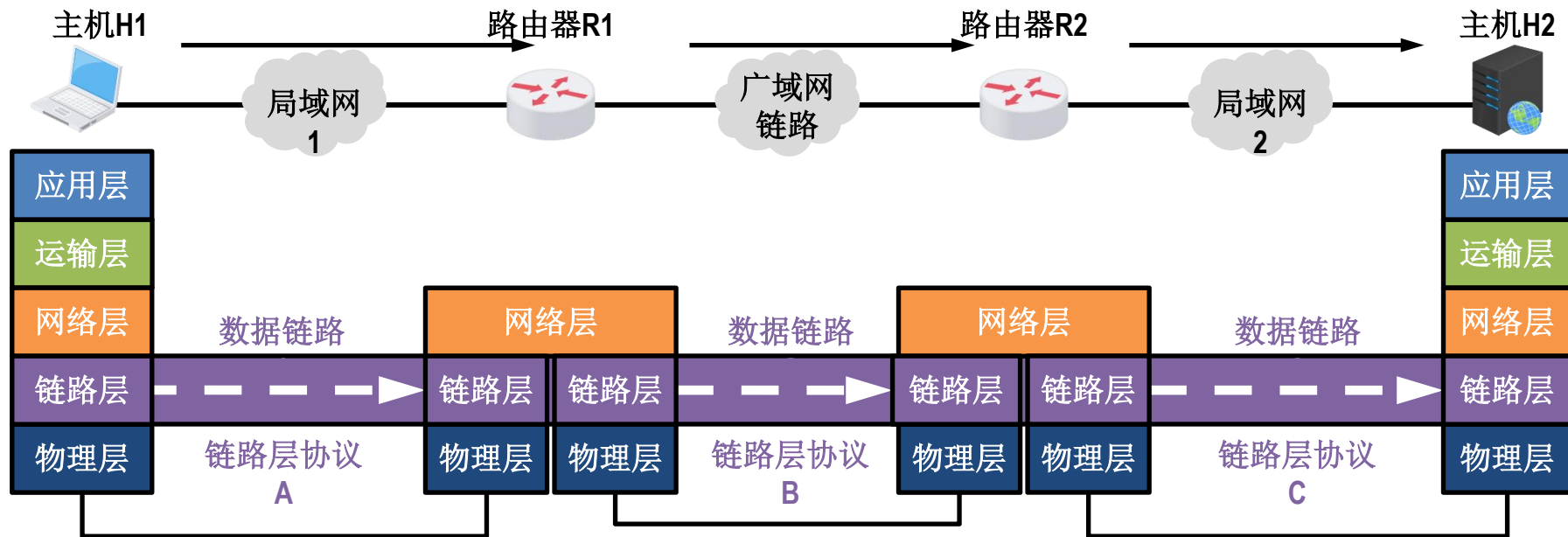
3.1.2

三个基本问题

01 数据链路层在网络体系结构中的地位



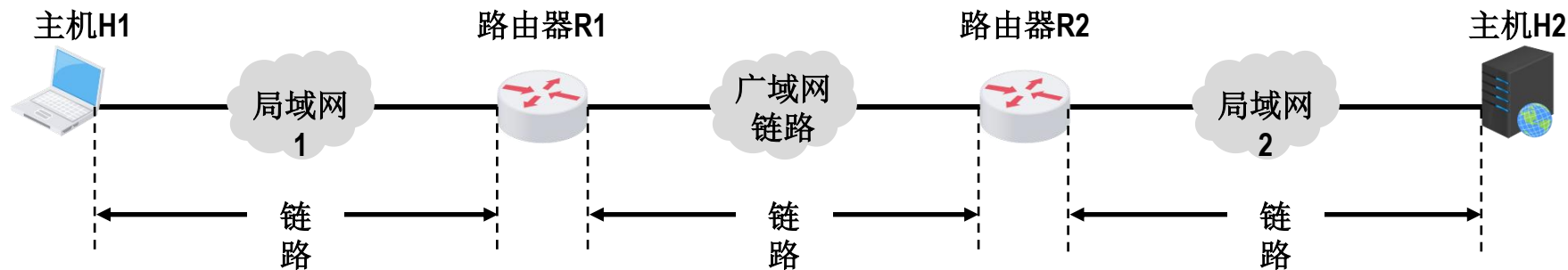
01 数据链路层在网络体系结构中的地位



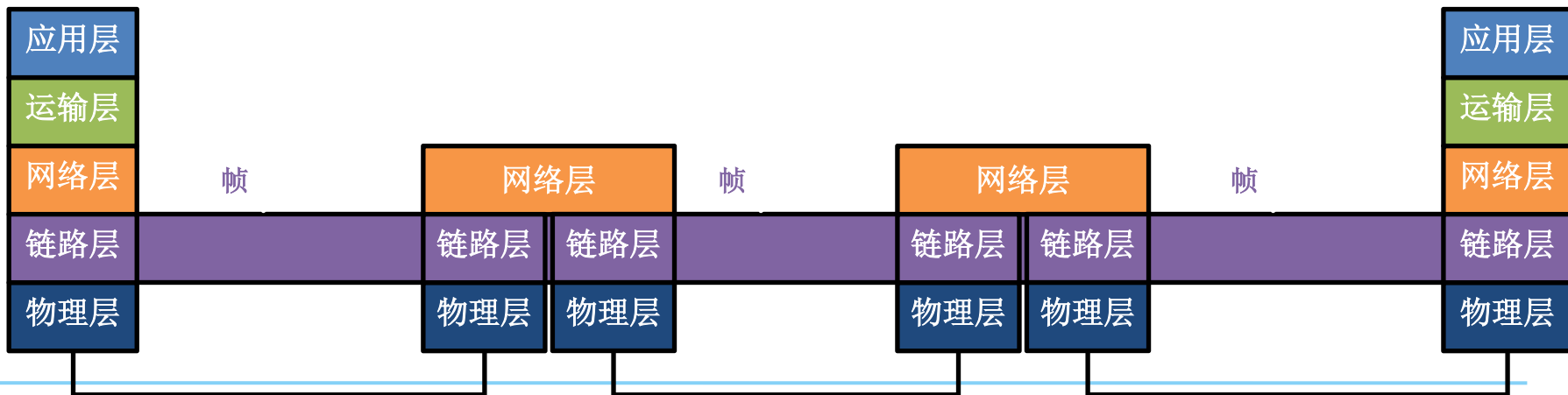
02

链路、数据链路和帧

- 链路（**Link**）是指从一个节点到相邻节点的一段物理线路（有线或无线），而中间没有任何其他的交换节点。
- 数据链路（**Data Link**）是基于链路的。当在一条链路上传送数据时，除需要链路本身，还需要一些必要的通信协议来控制这些数据的传输，把实现这些协议的硬件和软件加到链路上，就构成了数据链路。

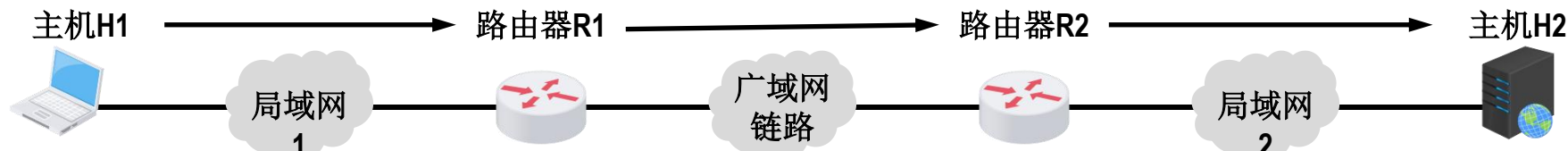


- 链路（Link）是指从一个节点到相邻节点的一段物理线路（有线或无线），而中间没有任何其他的交换节点。
- 数据链路（Data Link）是基于链路的。当在一条链路上传送数据时，除需要链路本身，还需要一些必要的通信协议来控制这些数据的传输，把实现这些协议的硬件和软件加到链路上，就构成了数据链路。
- 帧（Frame）是数据链路层对等实体之间在水平方向进行逻辑通信的协议数据单元PDU。

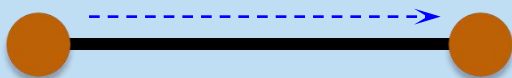


02 链路、数据链路和帧

帧（Frame）是数据链路层对等实体之间在水平方向进行逻辑通信的协议数据单元PDU。

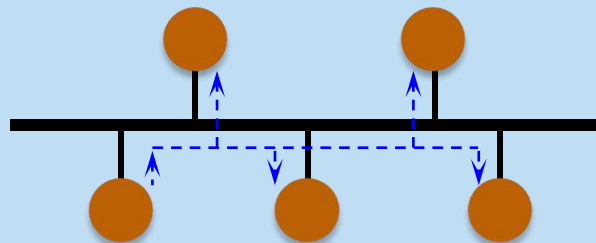


数据链路层信道类型



(a) 点对点信道

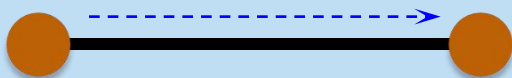
- 使用一对一的**点对点**通信方式。



(b) 广播信道

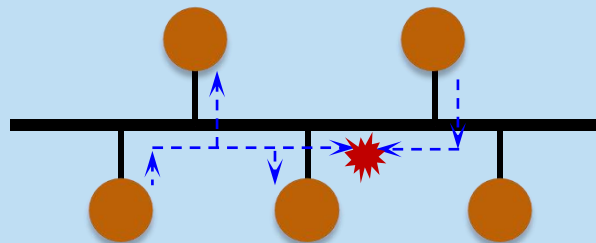
- 使用一对多的**广播**通信方式。
- 必须使用专用的**共享信道协议**来协调这些主机的数据发送。

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。

3.1

使用点对点 信道的数据 链路层

3.1.1

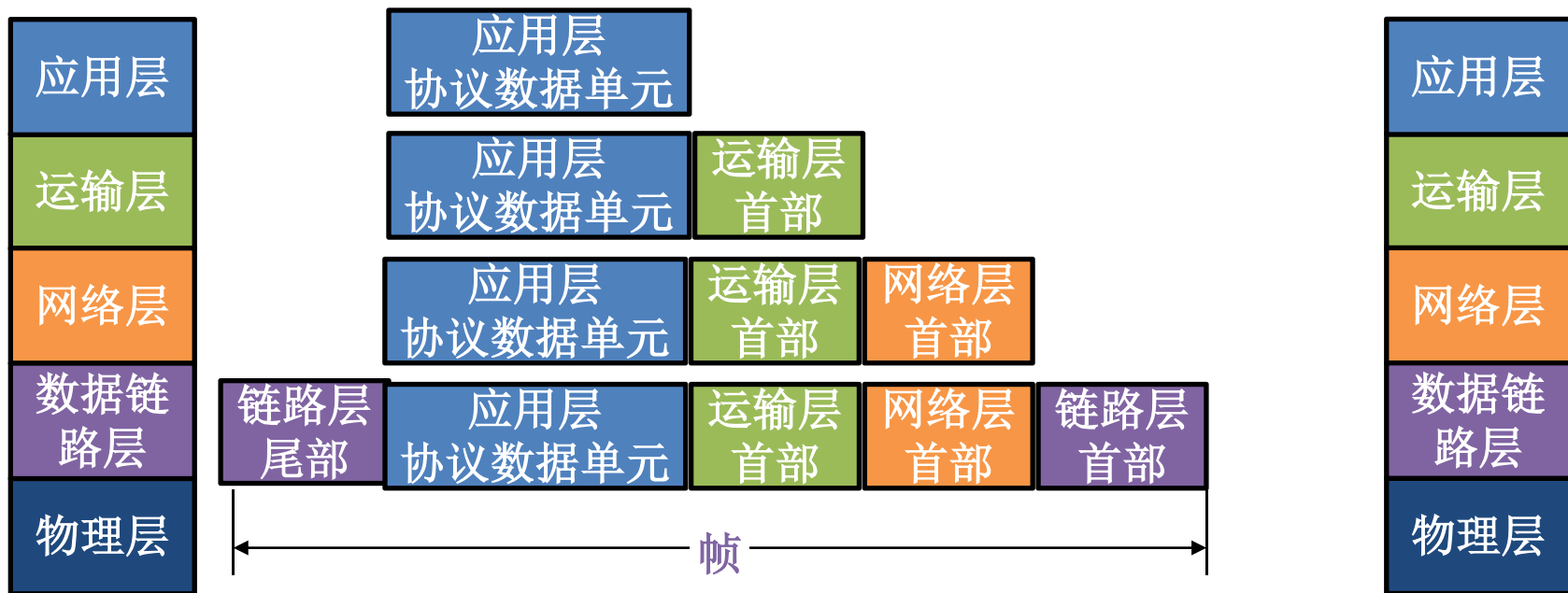
数据链路和帧

3.1.2

三个基本问题

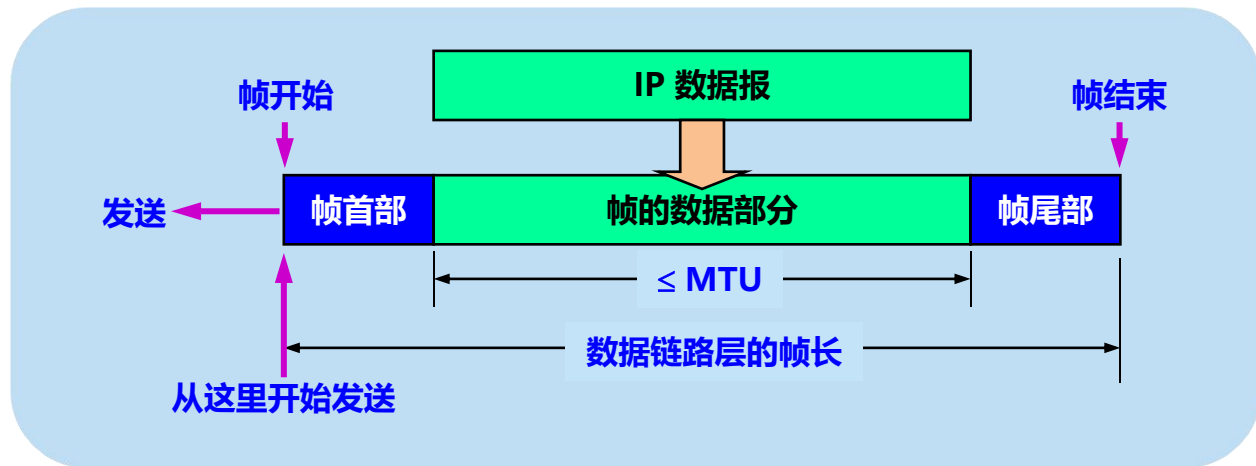
3.1.2 三个基本问题

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): 在一段数据的前后分别添加首部和尾部, 构成一个帧。
- 首部和尾部的一个重要作用就是进行**帧定界** (即确定帧的界限)。

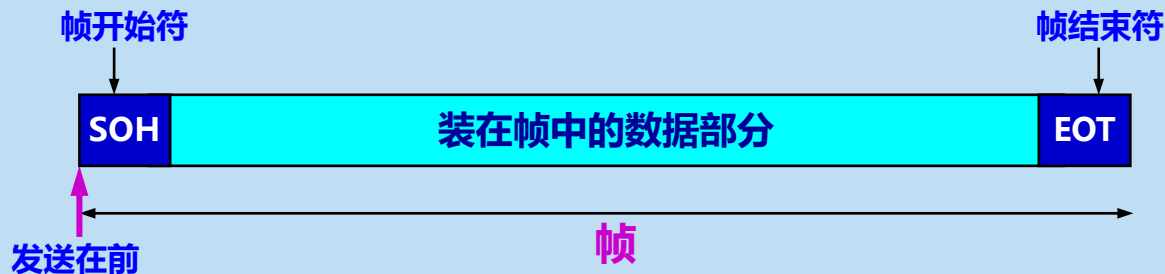


最大传送单元 MTU
(Maximum Transfer Unit): 规定了所能传送的帧的数据部分长度上限。

用帧首部和帧尾部封装成帧

用控制字符作为帧定界符

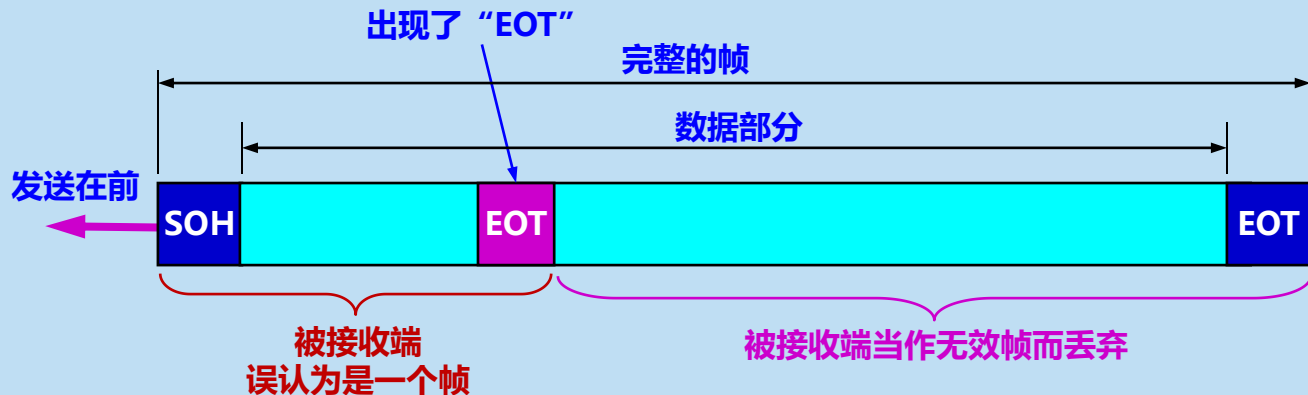
- 控制字符 SOH (Start Of Header) 放在一帧的最前面，表示帧的首部开始。
- 控制字符 EOT (End Of Transmission) 放在一帧的末尾，表示帧的结束。



用控制字符进行帧定界的方法举例

2. 透明传输

- **问题：**如果数据中的某个字节的二进制代码恰好和 SOH 或 EOT 一样，数据链路层就会**错误**地“找到帧的边界”，导致错误。



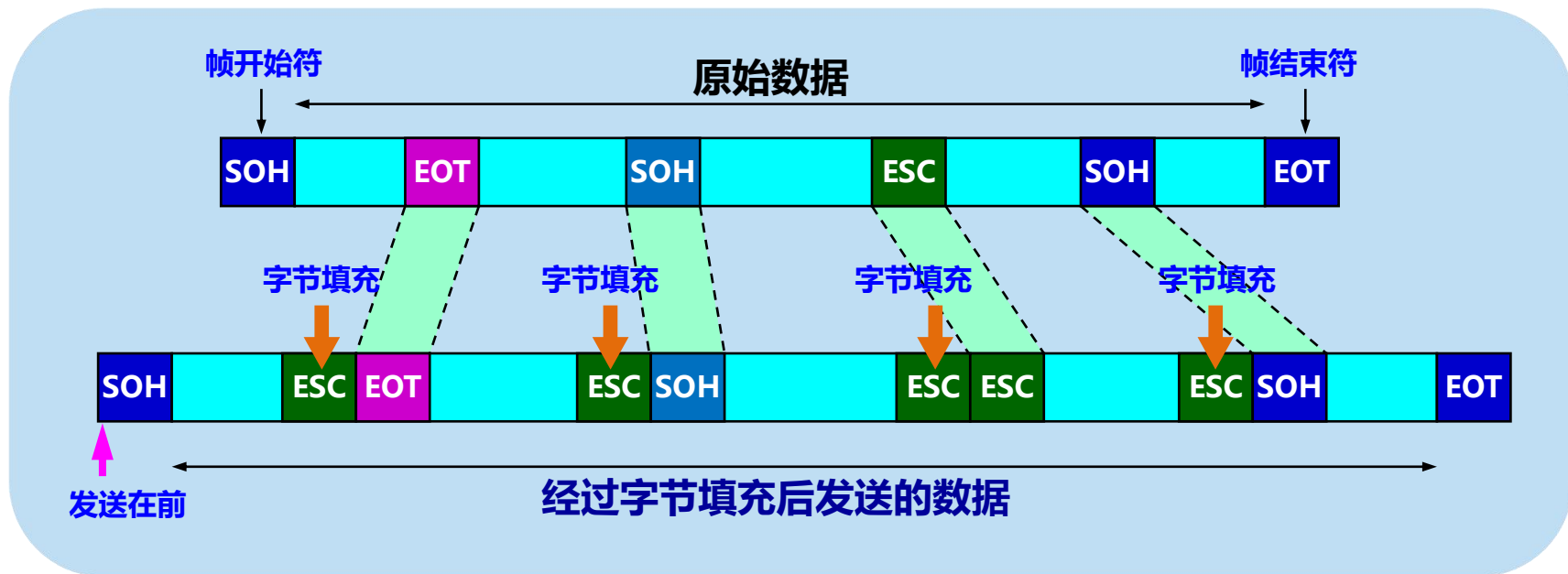
数据部分恰好出现与 EOT 一样的代码

透明

- 指某一个实际存在的事物看起来却好像不存在一样。

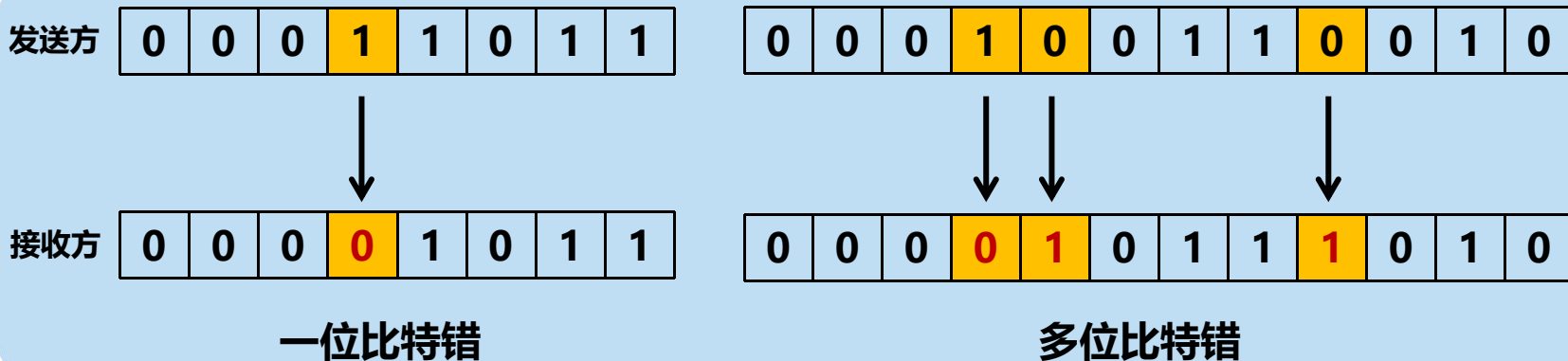
“在数据链路层透明传送数据”表示：无论发送什么样的比特组合的数据，这些数据都能够按照原样**没有差错**地通过这个数据链路层。

用“字节填充”或“字符填充”法解决透明传输的问题



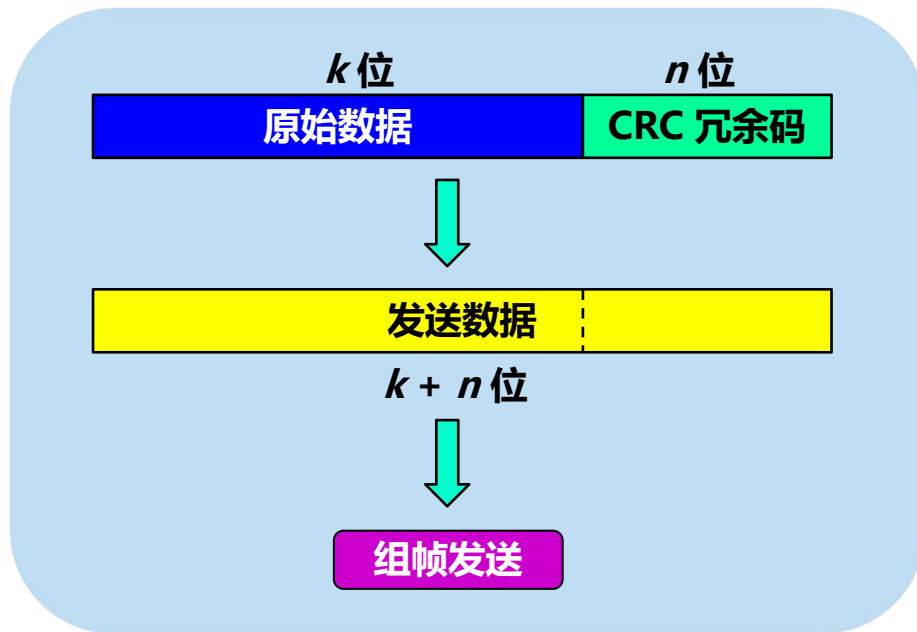
3. 差错检测

在传输过程中可能会产生**比特差错**： $1 \rightarrow 0$, $0 \rightarrow 1$ 。



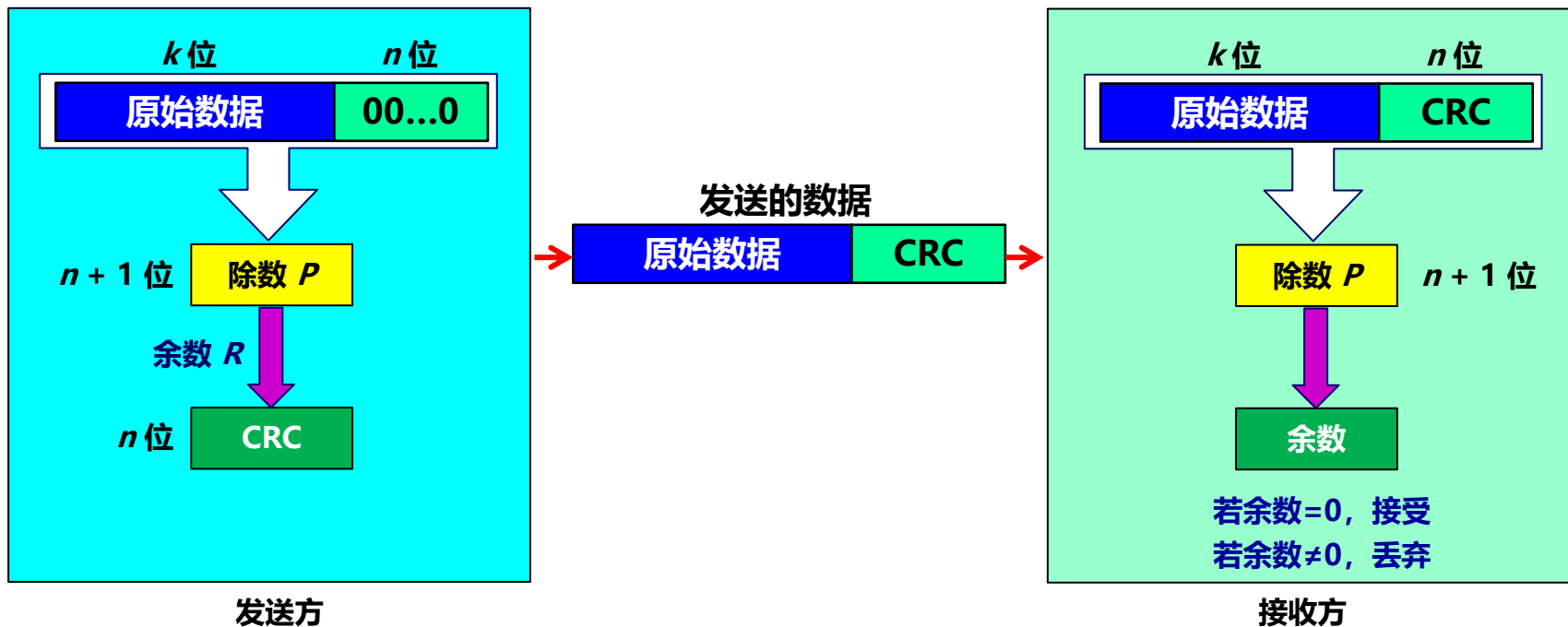
在一段时间内，传输错误的比特占所传输比特总数的比率称为**误码率 BER** (Bit Error Rate)。

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)。

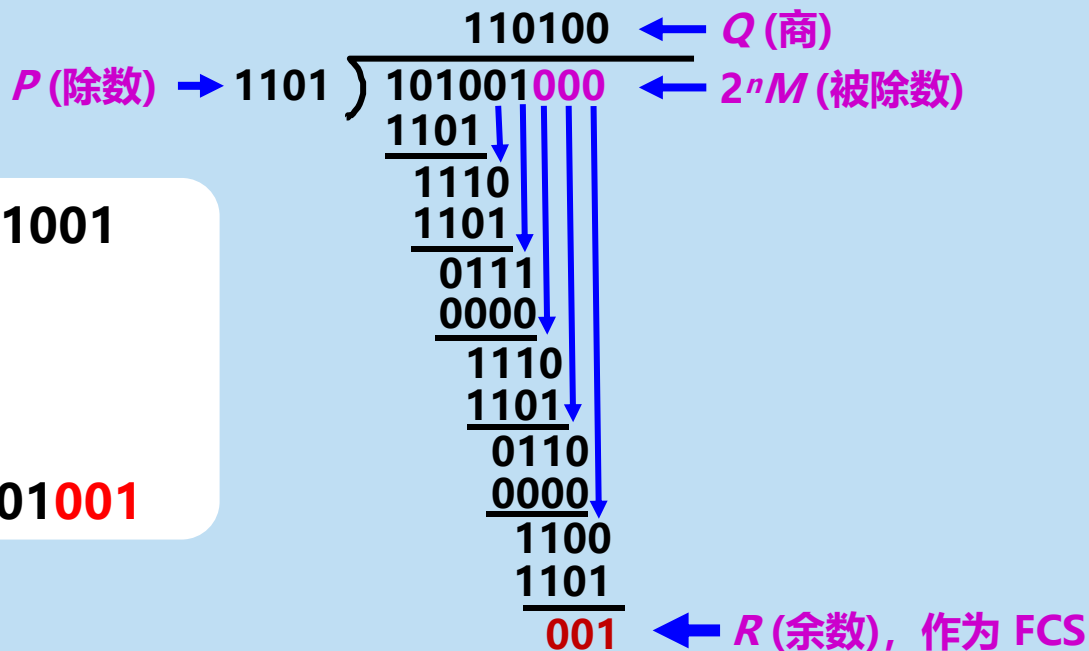
CRC 冗余码的计算举例

原始数据 $M = 101001$

除数 $P = 1101$

得到:

发送数据 = 101001001



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**。
 1. 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。
 2. 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。

广泛使用的生成多项式P(X)

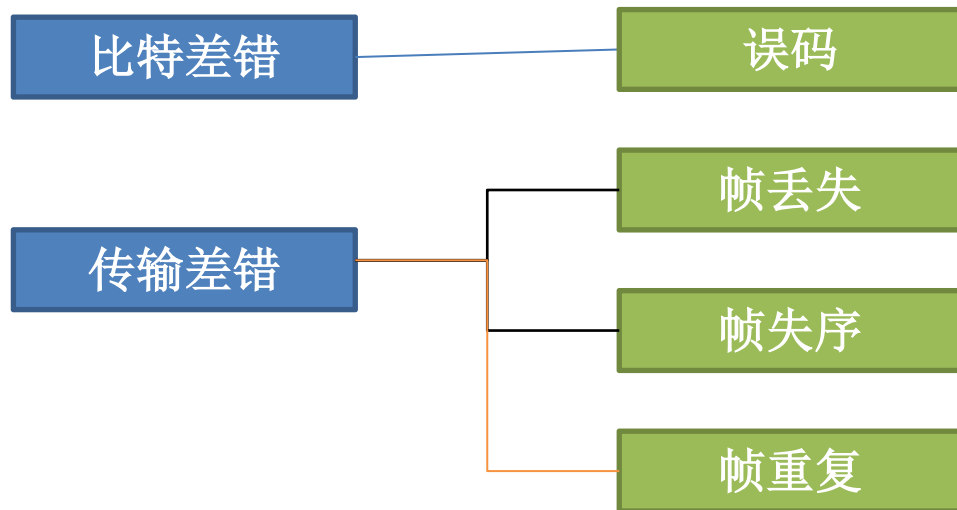
$$\text{CRC-16} = X^{16} + X^{15} + X^2 + 1$$

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

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

01

可靠传输的相关基本概念

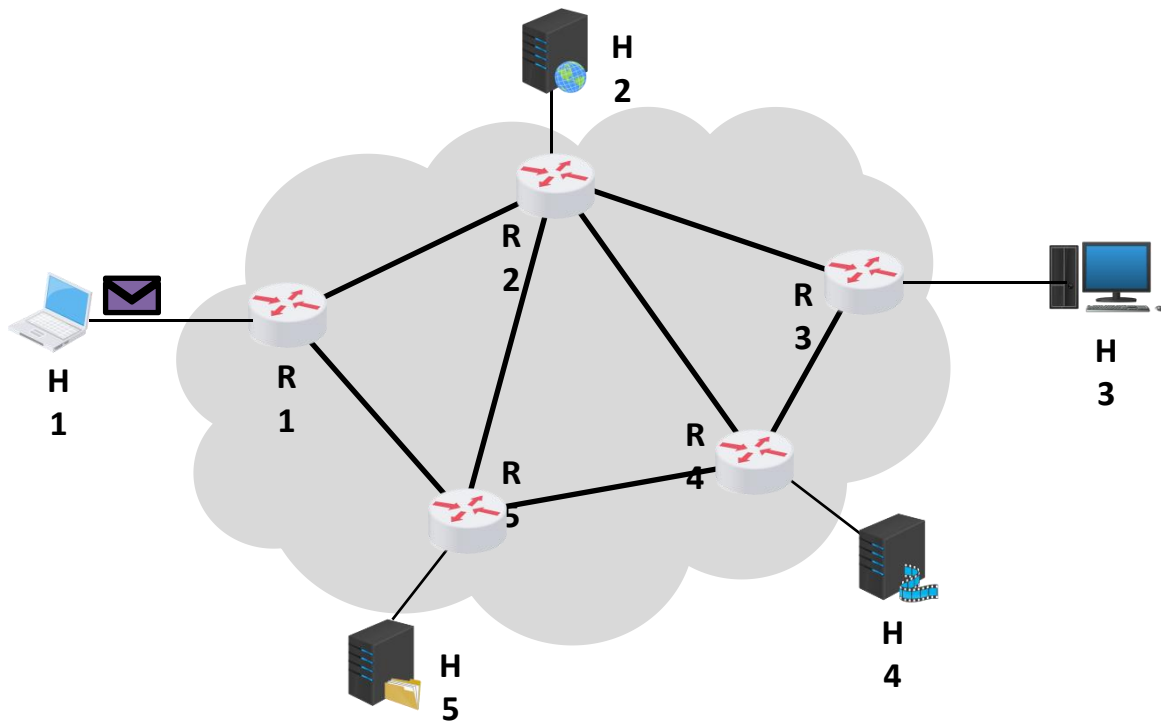


传输差错

帧丢失

帧失序

帧重复

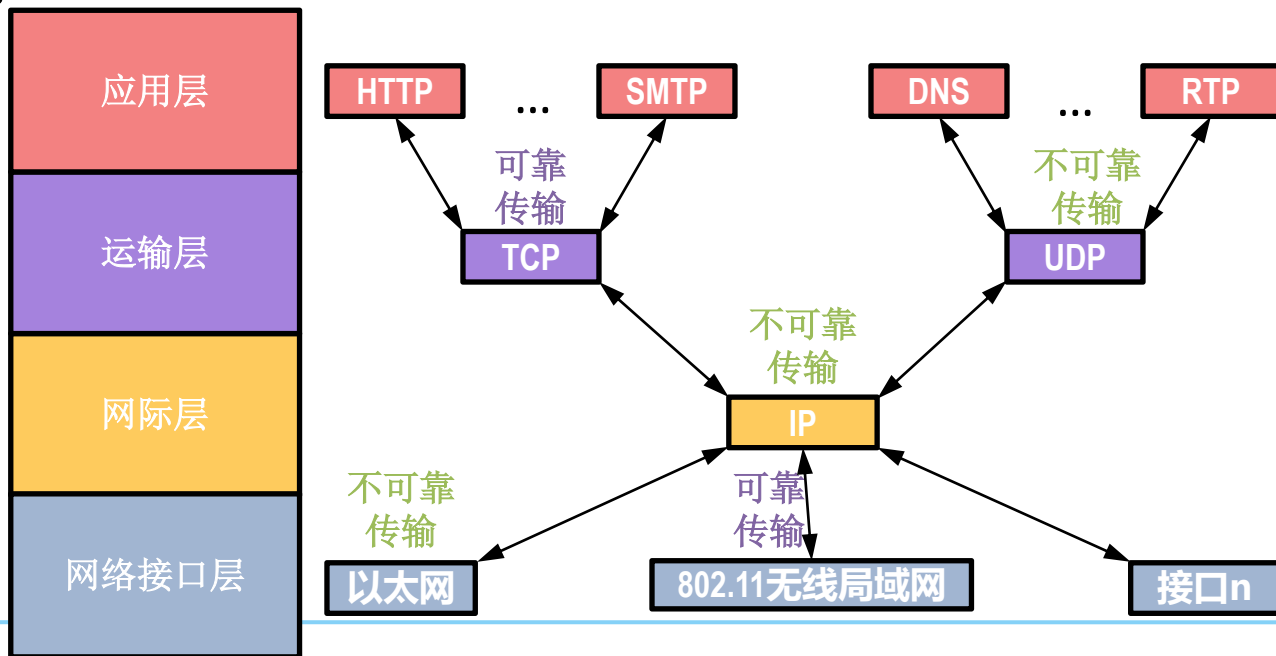


注意：“无比特差错”与“无传输差错”是不同的

- **可靠传输：**数据链路层的发送端发送什么，在接收端就收到什么。
- **传输差错**可分为两大类：
 - ◆ 比特差错；
 - ◆ 传输差错：帧丢失、帧重复或帧失序等。
- 在数据链路层使用 CRC 检验，能够实现无比特差错的传输，但这还不是可靠传输。
- 要做到可靠传输，还必须再加上帧编号、确认和重传等机制。

可靠传输的相关基本概念

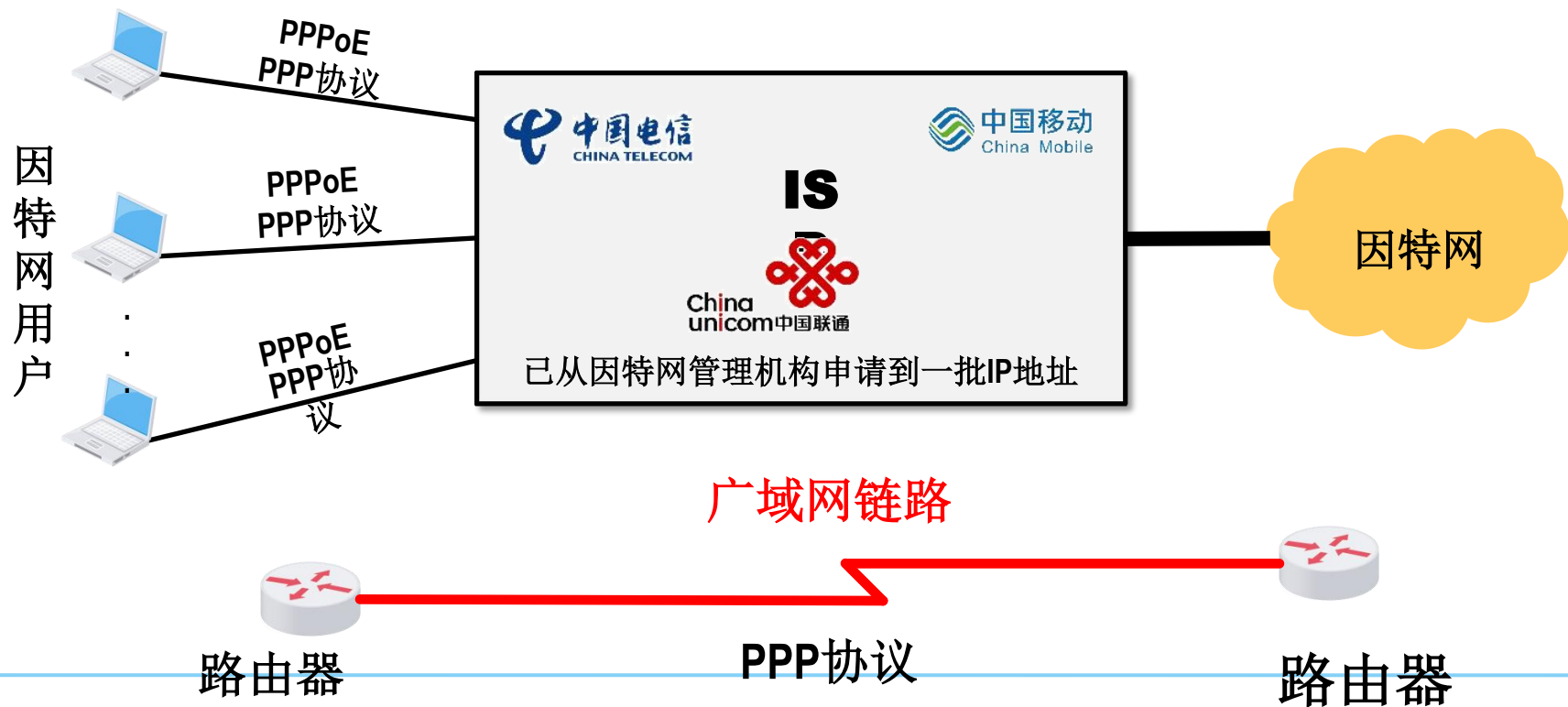
- 可靠传输服务并不局限于数据链路层，其他各层均可选择实现可靠传输。
- 可靠传输的实现比较复杂，开销比较大，是否使用可靠传输取决于应用需求。



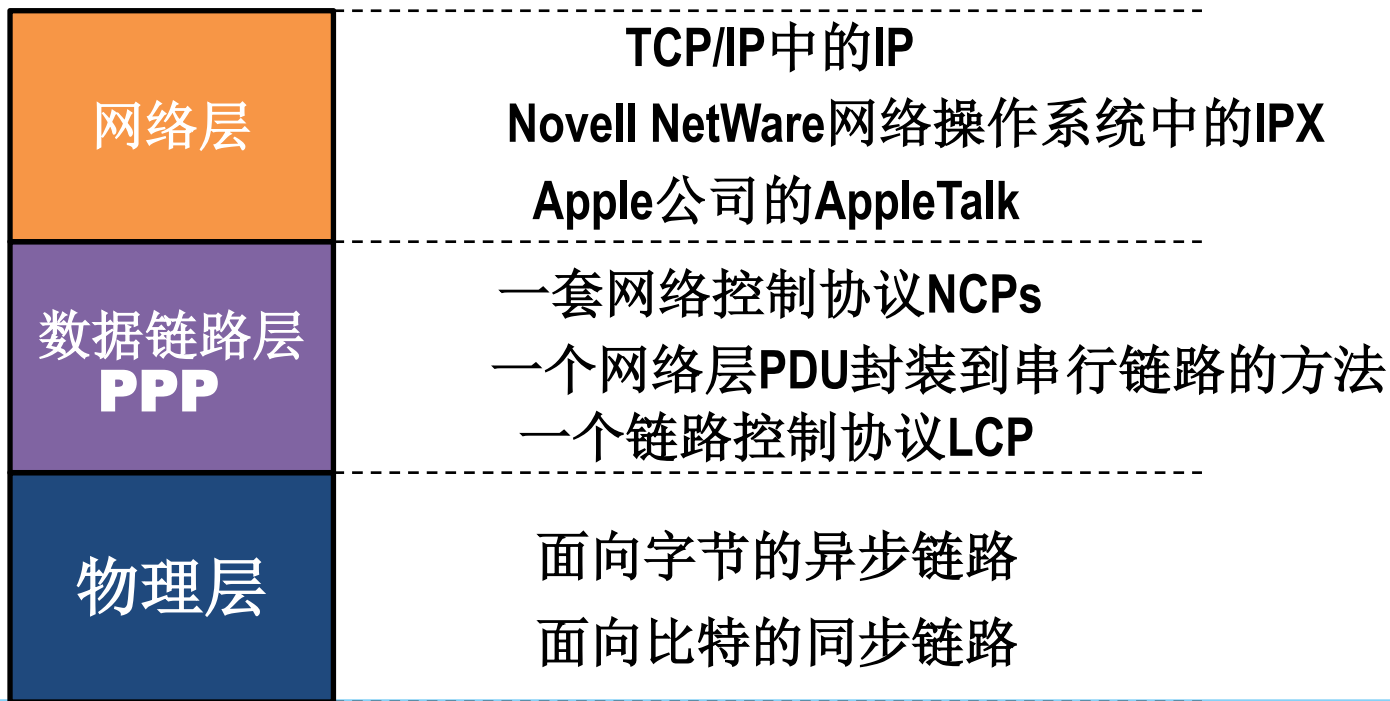
- 对于点对点的链路，目前使用得最广泛的数据链路层协议是**点对点协议 PPP** (Point-to-Point Protocol)。
- PPP 协议在 1994 年就已成为互联网的正式标准 [RFC 1661, STD51]。



■ 点对点协议PPP主要有两种应用：



■ 从网络体系结构的角度看点对点协议PPP的组成



PPP的帧格式



PPP的帧格式



标志（Flag） 字段：PPP帧的定界符，取值为0x7E。

地址（Address） 字段：取值为0xFF，预留（目前没有什么作用）。

控制（Control） 字段：取值为0x03，预留（目前没有什么作用）。

协议（Protocol） 字段：其值用来指明帧的数据载荷应向上交付给哪个协议处理。

PPP的帧格式

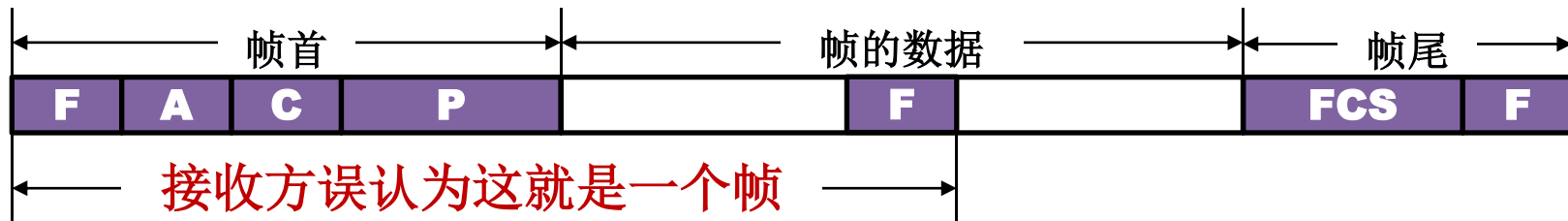


协议（Protocol） 字段：其值用来指明帧的数据载荷应向上交付给哪个协议处理。

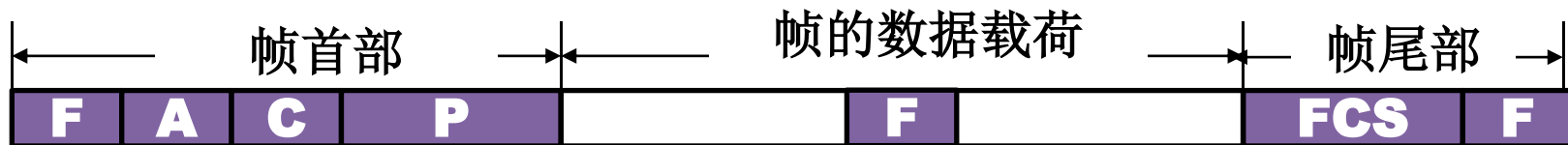
7E	FF	03	0021	IP数据报	FCS	7E
7E	FF	03	C021	LCP分组	FCS	7E
7E	FF	03	8021	NCP分组	FCS	7E

帧检验序列（Frame Check Sequence, FCS） 字段：其值是使用循环冗余校验CRC计算出的检错码。

PPP帧的透明传输



PPP帧的透明传输

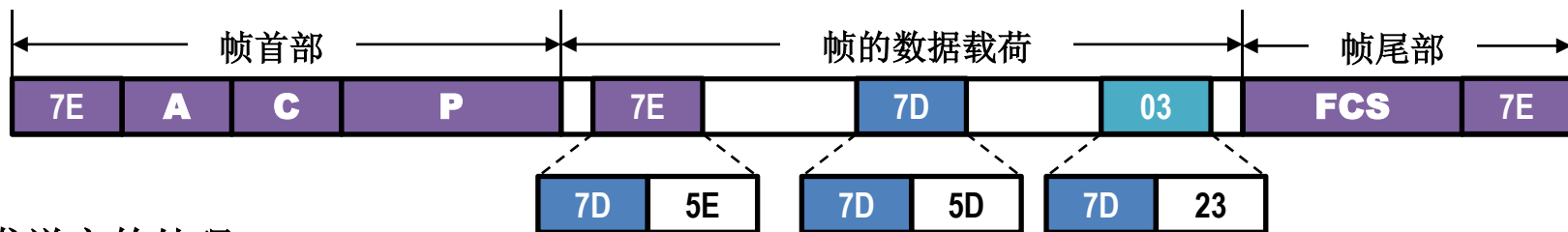


标志（Flag字段：是PPP帧的定界符，取值为：

从字节角度看：取值为**0x7E**。

从比特角度看：取值为**01111110**。

面向字节的异步链路使用字节填充来实现透明传输[RFC1662]

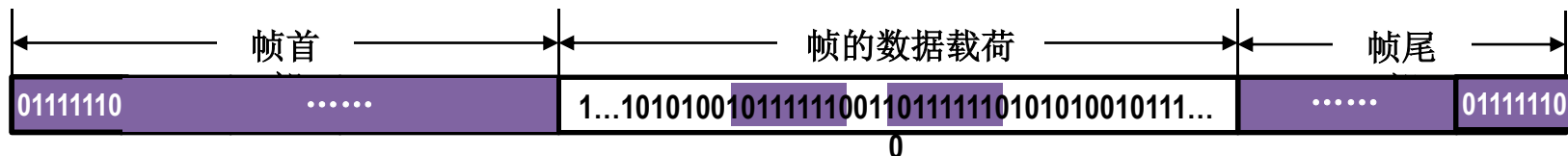


发送方的处理:

- (1) 将数据载荷中出现的每一个**0x7E**减去**0x20**（相当于异或**0x20**），然后在其前面插入**转义字符0x7D**。
- (2) 若数据载荷中原来就含有**0x7D**，则把每一个**0x7D**减去**0x20**，然后在其前面插入**转义字符0x7D**。
- (3) 将数据载荷中出现的每一个**ASCII码控制字符**（即数值小于**0x20**的字符），加上**0x20**（相当于异或**0x20**，将其转换成非控制字符），然后在其前面插入**转义字符0x7D**。

接收方的处理: 进行与发送方相反的变换。

面向比特的同步链路使用零比特填充来实现透明传输





PPP帧的透明传输

面向比特的同步链路使用零比特填充来实现透明传输



信息字段中**出现**了和标志字段 F 完全一样的 8 比特组合 0x7E

0 1 0 **0 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
填入比特 0 再发送出去

0 1 0 **0 1 1 1 1 1 0** 1 0 0 0 1 0 1 0

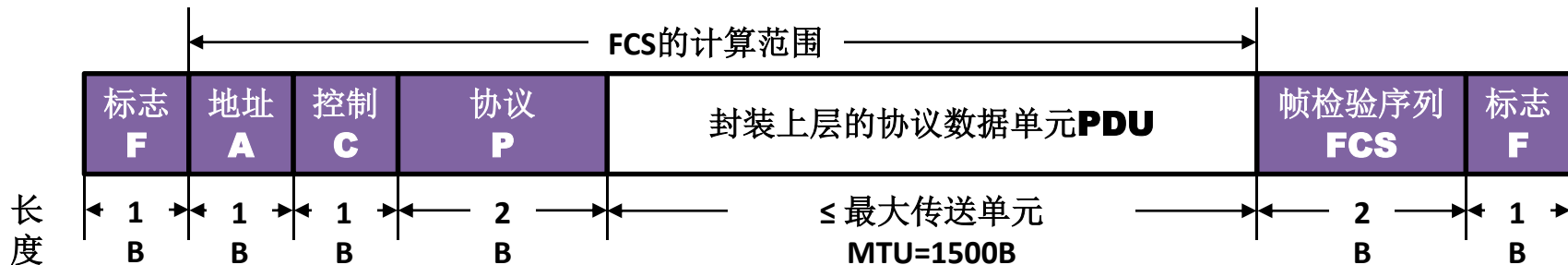
发送端填入 0 比特

接收端把 5 个连 1
之后的比特 0 删除

0 1 0 **0 1 1 1 1 1** 1 0 0 0 1 0 1 0

接收端删除填入的 0 比特

04 PPP帧的差错检测



帧检验序列FCS字段：其值是使用循环冗余校验CRC计算出的检错码。

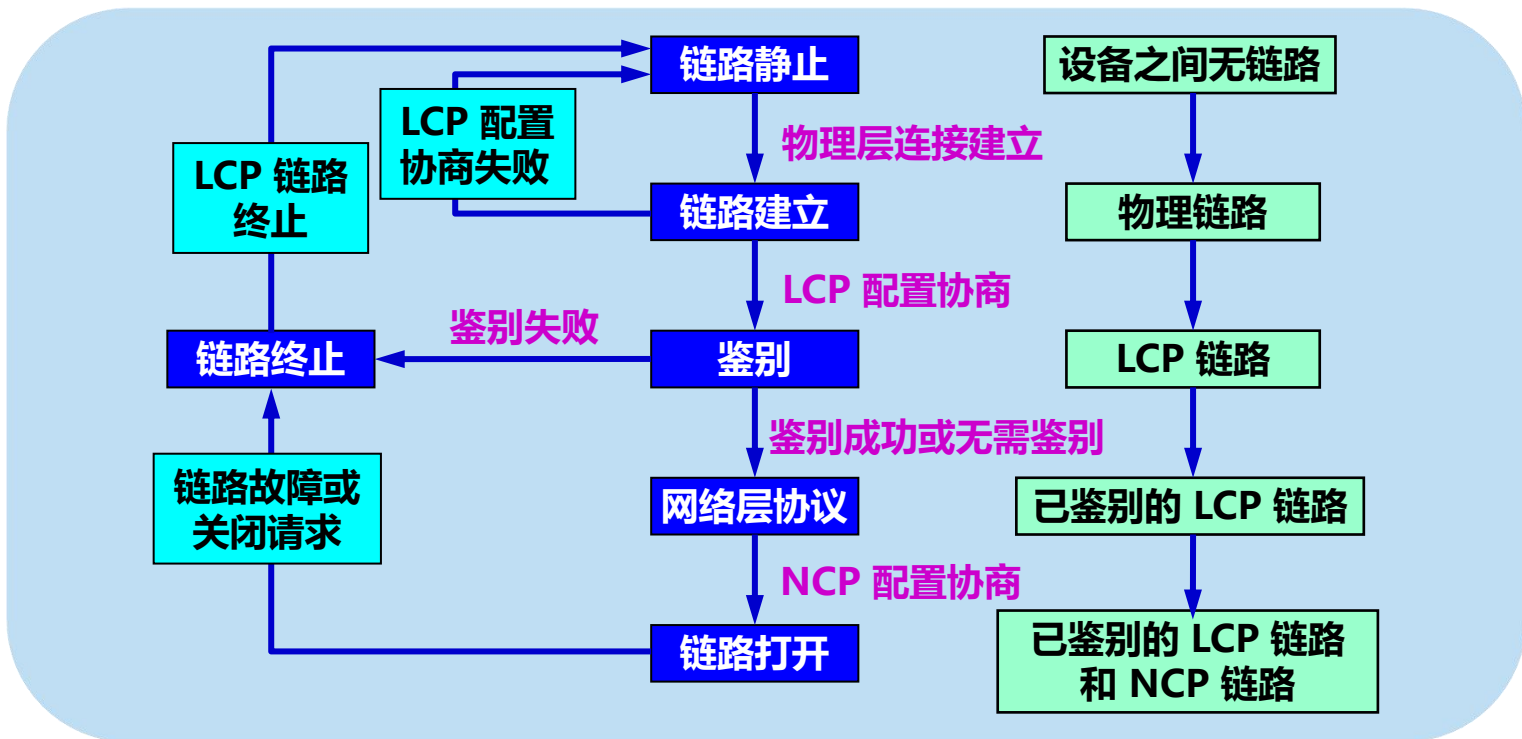
CRC采用的生成多项式为 $CRC - CCITT = X^{16} + X^{12} + X^5 + 1$

3.2.3 PPP 协议的工作状态

PPP 链路初始化过程:

- 用户拨号接入 ISP 后, 就建立了一条从用户个人电脑到 ISP 的物理连接。
- 用户个人电脑向 ISP 发送一系列的**链路控制协议 LCP** 分组 (封装成多个 PPP 帧), 以便建立LCP连接。
- 之后进行网络层配置。**网络控制协议 NCP** 给新接入的用户个人电脑分配一个临时的 IP 地址。
- 当用户通信完毕时, NCP **释放**网络层连接, 收回原来分配出去的IP地址。
LCP **释放**数据链路层连接。最后**释放**的是物理层的连接。

PPP 协议的状态图



本讲小结

- 数据链路层的三个基本问题（封装成帧、透明传输、差错检测）

掌握：

- 封装成帧：**帧**头和帧尾的特殊标记（0X7E、01111110）
- 透明传输：字节填充、比特填充。
- 差错检测：CRC循环冗余的原理及实现。

了解：PPP协议字段及其工作过程。