第3章数据链路层之
差错检测和数据链路层协议

主要内容

- 差错编码(3.2)
- 帧同步(3.1.1 + 3.4.1)
- 数据链路层协议(3.4)

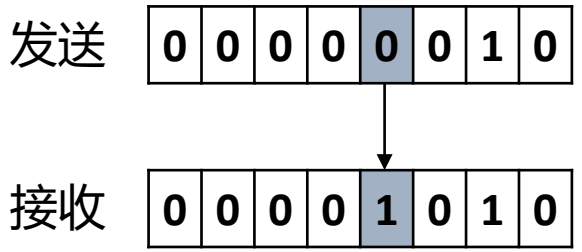
差错的特性

$$P_e = \frac{\text{发生差错的码元数}}{\text{接收的总码元数}}$$ 常常用10的负若干次方来表示

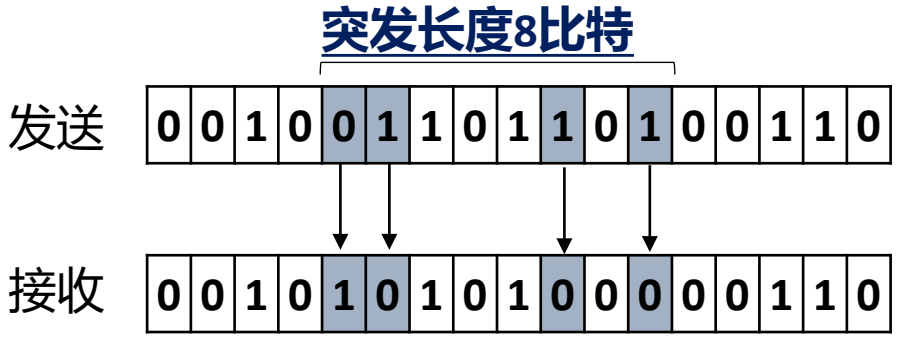
链路上有两种不同的差错：

- 随机差错：随机热噪声
  - 随机热噪声是信道固有的、持续存在的
  - 码元（比特）的差错是独立的，和前后的码元无关
  - 随机错的概率（误码率）很低，物理信道的设计保证相当大的信噪比
- 突发差错：冲击噪声
  - 外界的因素，持续时间短，突发性
  - **传输中产生差错的主要原因**
  - 突发长度：突发**差错发生的第一个码元**到**有错的最后一个码元**间所有码元的个数。比如4800bps信道上的10ms的冲击噪声的突发长度为48比特

随机差错

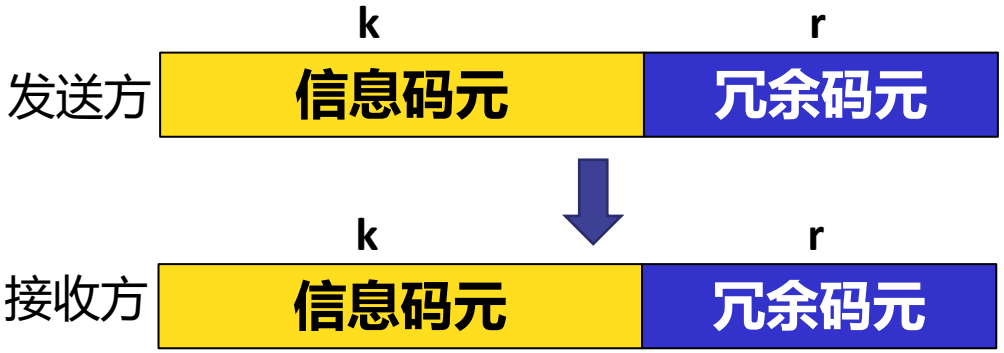


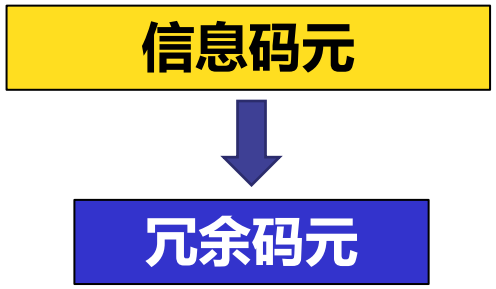
突发差错

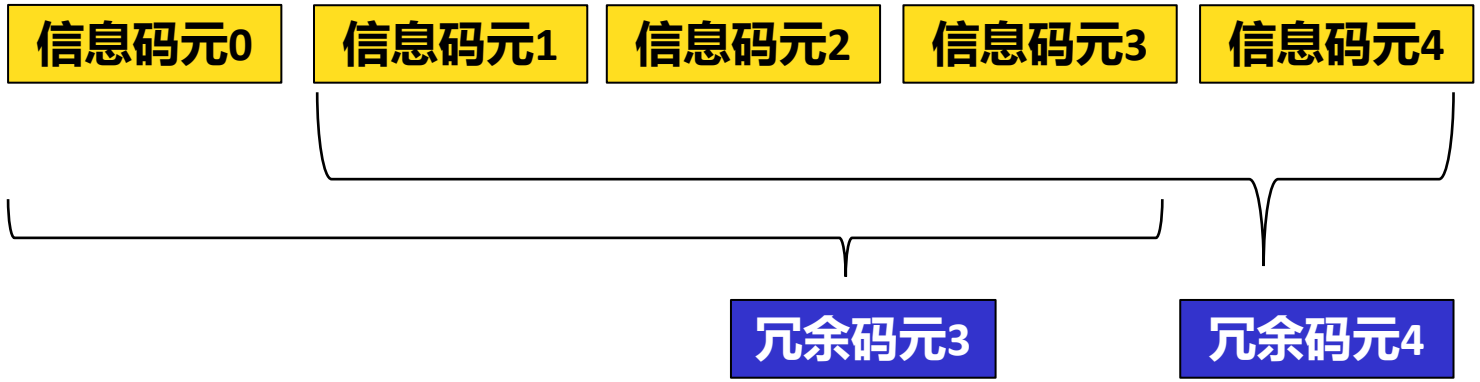


差错编码

- 对要发送的信息码元(k)附加上按照某种数学关系(监督关系式）产生的冗余（监督）码元(r)
- 编码效率R=k/(k+r)：码字中信息码元所占的比例
- 漏检率：接收者无法了解到信息码元出错的概率
- 差错编码可以分为：
  - 是否能纠错：检错码和纠错码
    - 纠正错误类型：纠正随机错误和纠正突发错误码
  - 信息码元和冗余码元之间的关系是否为线性：**线性码**和非线性码
  - 信息码元和冗余码元之间的约束方式：**分组码**和卷积码
    - 分组码：冗余码元仅仅和当前信息码元相关
    - 卷积码：冗余码元和当前以及最近的信息码元相关







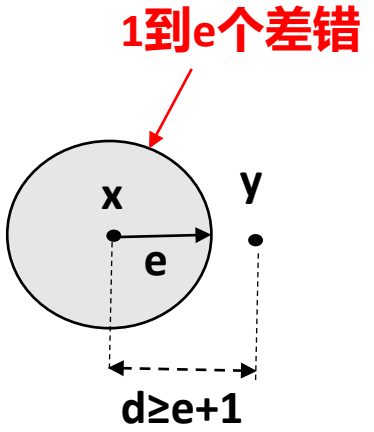
分组码：术语

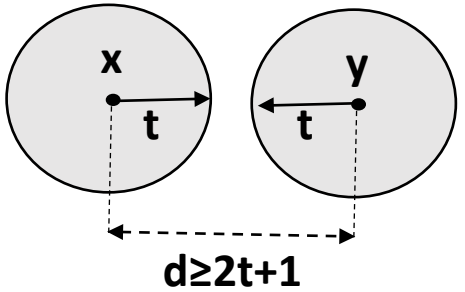
- 码元(code element)：数据通信中每一个符号的通称。二进制差错编码中一个码元对应一个比特(位)
- 码字(codeword)：若干个码元序列组成的数据单元
- 码长：码字中码元的数目
- 码距（hamming，汉明/海明距离）：两个等长码字之间对应符号不同的数目
  - 二进制码字中，两个码字进行异或(模2相加)时，对应位置如果相同则为0，如果不同则为1

  →两个码字之间异或后1的个数就是这两个码字之间的距离
  - 码字10100与11000之间的码距d = 2
- 最小码距：在码字集合中全体码字之间距离的最小数值，表示为d_{min}

汉明（海明）距离

- 一个例子：假设要传送消息包含码字为A/B
  - A: 0　　B: 1　　海明距离为1，没有检错/纠错能力
  - A: 00　　B: 11　　海明距离为2
    - 非法码字集合为{01, 10}，可以检测1比特错，没有纠错能力
  - A: 000　　B: 111　　海明距离为3
    - 检测出两位及两位以下差错
    - 纠正1比特差错
- 为**检出e个错码**，要求码集的海明距离**d≥e+1**
- 为**纠正t个错码**，要求码集的海明距离 **d≥2t+1**
- 为**检出e个错码，同时能纠正t (e>t)个错码**，则应满足　**d≥e+t+1 (e>t)**
- A：0000，B：1111，海明距离d为4
  - 能检2个错码，同时能纠1个错码：e = 2, t = 1，d≥e+t+1





- 有2个比特出错时，A和B变为 {0011, 0101, 0110, 1010, 1001, 1100}, {1100, 1010, 1001, 0110, 0101, 0011}，这些都不是合法码字，因此能检2个错码
- 有1个比特出错时，A和B变为{0001，0010，0100，1000}, {1110，1011，1101，0111}，这两个集合交集为空，从而可以知道原来传输的是什么，能纠1个错码
- 上述2种情况出现时，1个比特和2个比特出错后变化的内容不相交，能检测2个比特出错还是1个比特出错，在1个比特出错时可纠正错误

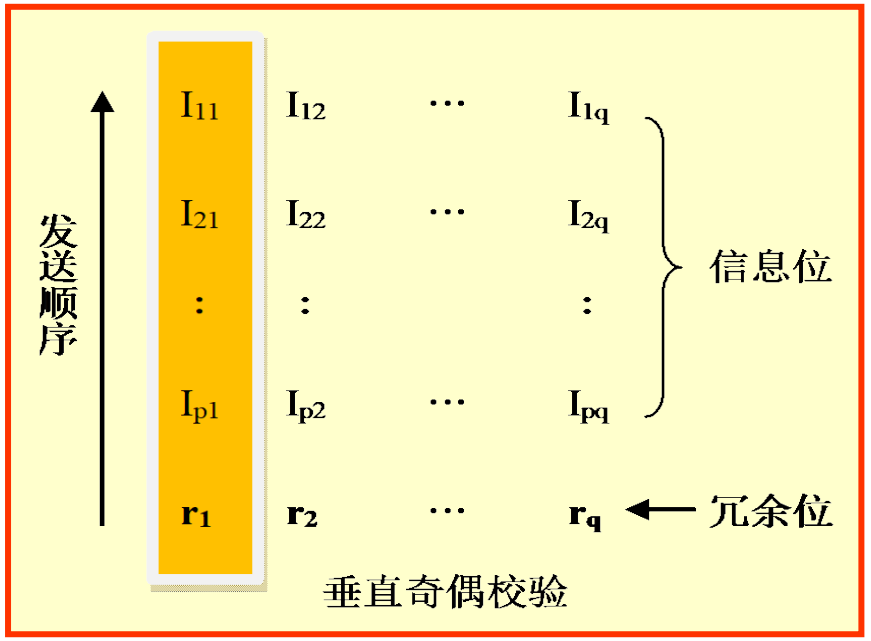
线性分组码

- 分组：将信息码元若干位为一组，该组监督码元仅与本组信息码元有关
- 线性：信息码元与监督码元之间的关系可以用一组线性方程来表示
- 线性分组码的线性特性：
  - 任意两个许用（即合法）码字的和（二进制编码一般采用异或运算）也是许用码字
  - 码组间的最小码距等于非零码的最小码重（即非0数字的数目，等于与全0码字之间的距离）
- 常用线性分组码
  - **偶校验**
  - **海明码**
  - **循环冗余码**
  - Reed-Solomon码

奇偶校验： 偶校验为线性分组码

垂直奇偶校验 发送p个信息位后附加一个冗余位

- 字符奇偶校验：7个比特+1比特冗余位
- 检测出每列（段）中所有奇数（1、3…）个错
- 突发错误的漏检率为50%!!
- 在发送和接收的过程中进行编解码

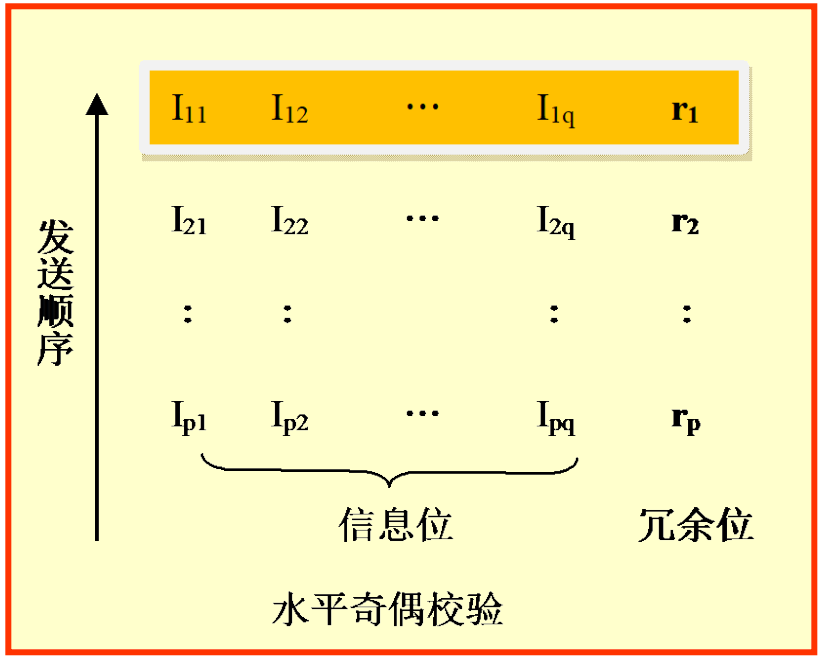


$$R=\frac{p}{p+1}$$

偶校验：$r_i = I_{1i} \oplus I_{2i} \oplus \cdots \oplus I_{pi} \quad i=1,2,...,q$

水平奇偶校验 发完p*q个信息位后附加p个冗余位

- 各段同一位上的奇数个错
- 长度小于等于p的突发差错
- 编码和检测的实现要复杂一些

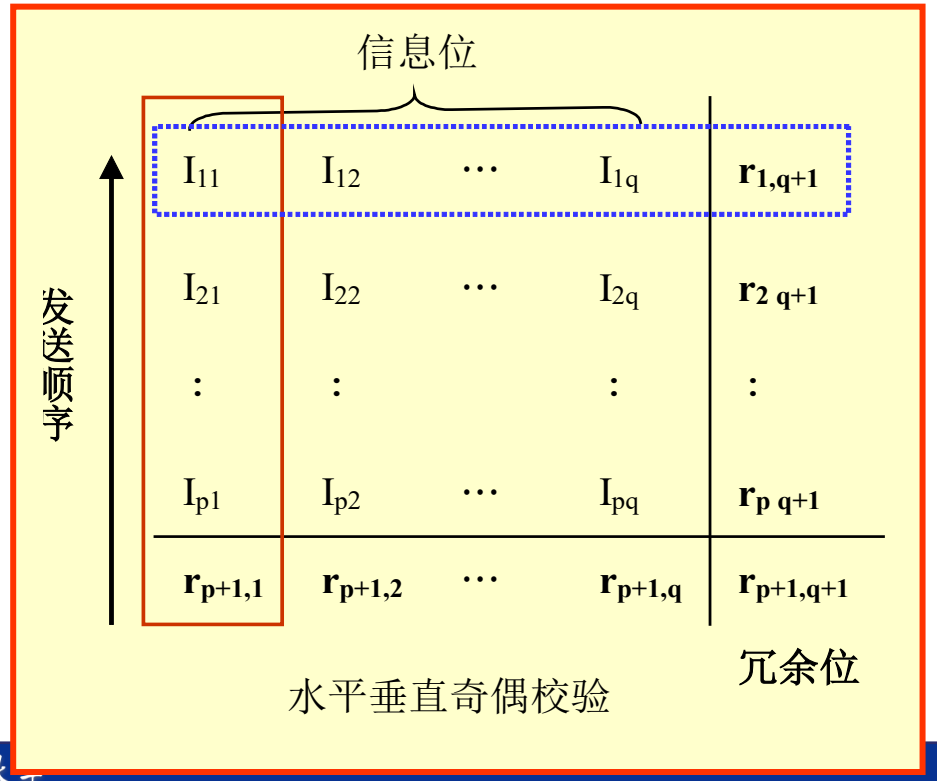


$$R=\frac{q}{q+1}$$

偶校验：$r_i = I_{i1} \oplus I_{i2} \oplus \cdots \oplus I_{iq} \quad i=1,2,...,p$

水平垂直奇偶校验

- 检测出：
  - 所有3位或3位以下的错误、奇数位错
  - 突发长度小于等于p+1的突发差错
  - 很大一部分偶数位错：差错分布以致于某一行或者某一列有奇数个差错
- 部分纠错功能：可以纠正1比特错
  - 信息块中恰好只有某一行和某一列有奇数位错时，可确定为该行和该列的交叉处



$$R = \frac{pq}{(p+1)(q+1)}$$

偶校验：

$$r_{i,q+1} = I_{i1} \oplus I_{i2} \oplus \cdots \oplus I_{iq} \qquad i = 1,2,...,p$$

$$r_{p+1,j} = I_{1j} \oplus I_{2j} \oplus \cdots \oplus I_{pj} \qquad j = 1,2,...,q$$

$$r_{p+1,q+1} = r_{p+1,1} \oplus r_{p+1,2} \oplus \cdots \oplus r_{p+1,q}$$

$$= r_{1,q+1} \oplus r_{2,q+1} \oplus \cdots \oplus r_{p,q+1}$$

循环冗余码CRC(Cyclic Redundancy Check)

- 计算机网络中使用最为广泛的检错码
- 循环码（cyclic code）：任一许用码字经过循环移位后，所得到的码字仍然是许用码字
- 线性分组码：经常用(码长,信息位长度)或者(码长,信息位长度,最小码距)来描述，即(n,k)或(n,k,d)，冗余位长度r = n-k

模2运算（不带进位）：异或运算

CRC(11, 7)

n-k+1位 10111

```
                1 0 0 1 1 1 1   丢弃
          ┌─────────────────
10111     │ 1 0 1 0 0 0 1 0 0 0 0
          1 0 1 1 1
          ─────────
            1 1 0 1 0
            1 0 1 1 1
            ─────────
              1 1 0 1 0
              1 0 1 1 1
              ─────────
                1 1 0 1 0
                1 0 1 1 1
                ─────────
                  1 1 0 1   冗余码(n-k位)
```

附加n-k个0

编码：

- k个信息位之后附加n-k个0组成被除数
- 除数为n-k+1位
- 冗余位=被除数 % 除数，即被除数除以除数的余数
- 采用模二运算，即加法和减法都为不带进位的异或运算

解码：

- 判断收到的码字与编码时采用的除数相除是否能除尽，即余数为0表示没有出错

码字：10100011101

循环冗余码CRC：多项式码

- 二进制比特串与一个只有0和1两个系数的多项式一一对应
  - k位信息位对应于一个(<=k-1)次多项式K(x)：1010001 → $x^6 + x^4 + 1$
  - r位冗余位对应于一个(<=r-1)次多项式R(x)：1101 → $x^3 + x^2 + 1$
  - 生成多项式G(x)（r=n-k 次多项式，最高位的系数为1）：10111→ $x^4 + x^2 + x + 1$
- 信息位 + 冗余位（n=k+r）对应于一个(<=n-1)次多项式
  - $T(x)=x^rK(x)+R(x)$ 10100011101 → $T(x) = x^4K(x) + R(x) = x^{10} + x^8 + x^4 + x^3 + x^2 + 1$
- 编码：$x^rK(x)$（添加r个0）除以G(x)得到的余式为冗余位R(x)，可以证明T(x)/G(x)的余式为0
- 解码：接收方的码字除以G(x)，如余式为0表示无错

差错模式E(x) = 发送码字和接收码字的半加，其中1的位置对应变化了的相应比特的位置
若E(x)能被G(x)整除，则不能检测这样的错误

$x^r K(x) = G(x) \bullet Q(x) + R(x)$，记为$x^r K(x) / G(x) = R(x)$

无错时：

$T(x) = x^r K(x) + R(x) = G(x) \bullet Q(x) + R(x) + R(x) = G(x) \bullet Q(x)$，即$T(x) / G(x) = 0$

有错时：

$(T(x) + E(x))/G(x) = T(x)/G(x) + E(x)/G(x) = E(x)/G(x)$

循环冗余码CRC：一个好的生成多项式

拓展的内容

- 若r次多项式G(x)**含(x+1)的因子**，则能检测出所有**奇数位错**
  - 反证法：假设无法检测出来，可以推出E(1)=0，但是奇数位错的特征说明E(1)=1，矛盾

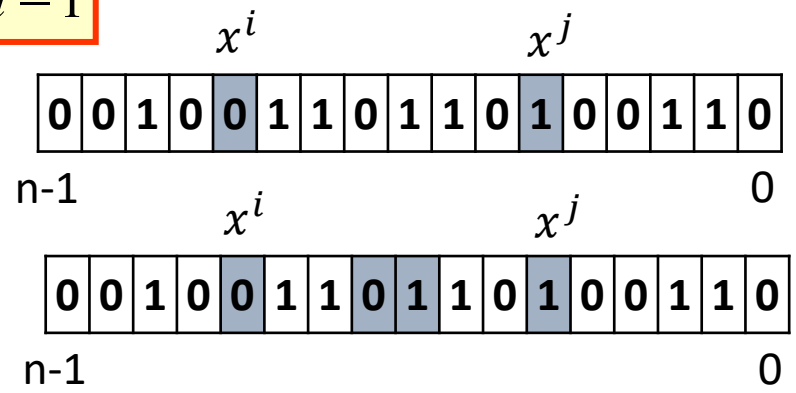
$$G(x)=(x+1)G'(x)$$
$$\text{若无法检测，}E(x)/G(x)=0\text{，则}E(x)=G(x)\bullet Q(x)=(x+1)G'(x)\bullet Q(x)$$
$$E(1)=0$$

$$E(x)\text{表示奇数位错}\rightarrow E(1)=1$$

- 若G(x)中**不含x的因子(常数项为1)**，且对任何0<e≤n-1，除不尽x^e+1，即**周期大于等于n**。则能检测出所有的**双错**: 先确定双错的差错多项式E(x)，然后判断其除以G(X)的余式不为0

$$\text{双错对应的差错多项式}E(x)=x^i+x^j=x^j(x^{i-j}+1),\ \text{其中}0<i-j\le n-1$$

$$(x^e+1)/G(x)\neq 0,\ \forall 0<e\le n-1\Rightarrow E(x)/G(x)\neq 0$$



- 若G(x)中**不含x的因子**，即G(x)中的常数项为1，则能检测出所有**突发长度≤ r的突发错**:

注意G(x)为r次多项式→ E(x)/G(x) ≠ 0

$$\text{突发长度}\le r\text{的差错多项式}E(x)=x^i+\cdots+x^j=x^j(x^{i-j}+\cdots+1),\ \text{其中}0\le i-j\le r-1$$

循环冗余码CRC：一个好的生成多项式

拓展的内容

- 若G(x)中不含x的因子，即G(x)中的常数项为1，则
  - 对突发长度为r+1的突发错误的漏检率为$2^{-(r-1)}$
  - 对突发长度b(b>r+1)的突发错误的漏检率为2^{-r}

- 首先写出相应的差错多项式
- 漏检率 = 漏检事件个数/总的事件空间个数

突发长度为r + 1的突发错误对应的差错多项式 为:

注意G(x)为r次多项式

$$E(x) = x^i + \cdots + x^j = x^j(x^{i-j} + \cdots + 1) = x^j(x^r + \cdots + 1)$$

$E(x)/G(x) = 0$的唯一可能是$G(x) = (x^r + \cdots + 1)$,

r次多项式$(x^r + \cdots + 1)$有2^{r-1}个，其中只有一个检测不出，所以漏检率 $= 1/2^{r-1}$

突发长度为$b >$ r + 1的突发错误对应的差错多项式 为: b - 1 = i - j

$$E(x) = x^i + \cdots + x^j = x^j(x^{i-j} + \cdots + 1) = x^j(x^{b-1} + \cdots + 1)$$

$E(x)/G(x) = 0$，则$G(x) \bullet Q(x) = x^{b-1} + \cdots + 1$

$Q(x)$为$b - 1 - r$次多项式，即

$$Q(x) = x^{b-r-1} + \cdots + 1$$

漏检率 $= 2^{b-r-2}/2^{b-2} = 2^{-r}$

循环冗余码CRC：生成多项式

- 选取r次多项式G(x)，满足：
  - 含有x+1因子，可检测出奇数位错
  - 常数项不为0：不含x的因子，突发长度小于等于r的突发错
  - 周期大于等于n，（且常数项不为0），检测出所有双错
- 检测出双错、奇数位错、**突发长度小于等于r的突发错**
- **突发长度为r+1的突发错误的漏检率为$2^{-(r-1)}$**，对**突发长度b(b>r+1)的突发错误的漏检率为2^{-r}**
- 常用的CRC多项式：　　r=16时，$1-2^{-15}$=99.997%　　$1-2^{-16}$=99.998%

$$CRC-1=x+1$$

$$CRC-12=x^{12}+x^{11}+x^3+x^2+x+1$$

$$CRC-16=x^{16}+x^{15}+x^2+1$$

$$CRC-16-CCITT=x^{16}+x^{12}+x^5+1$$

$$CRC-32-IEEE802.3=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

纠正一比特错的线性分组码：汉明（海明）码

回顾奇偶校验：

- 长度为k=n-1的信息位$a_{n-1}a_{n-2}\ldots a_1$加上一个偶校验位a_0（$a_{n-1}a_{n-2}\ldots a_1a_0$）
- 接收端：利用监督关系式计算校正因子S（=0和1），分别区分无错和有错的情况

$$S = a_{n-1} \oplus a_{n-2} \oplus \cdots \oplus a_1 \oplus a_0$$

要构造纠正**一比特差错**的编码：

- k位信息位后增加r个冗余位构成n(=k+r)位的码字
- 每个冗余位是通过信息位中的某些位半加后的结果
- 接收方通过r个监督关系式产生r个校正因子来区分无错和n位的码字中某一位错

$$2^r \geq k + r + 1$$

等式满足时称为完备(perfect)码

纠正一比特错的线性分组码：示例

假设信息位为k=4，则r≥3。取r=3，n=7，即$a_6a_5a_4a_3+a_2a_1a_0$

- 冗余位a_2、a_1和a_0是信息位中的某几位半加得到
- 三个监督关系式和校正因子$S_2S_1S_0$
  - 某个冗余位$a_2a_1a_0$与编码时采用的信息位半加
  - $S_2S_1S_0$区分无错和7位码字中某一位有错的情况

编码效率为4/7

线性分组码：无错时对应的校正因子应该为000

$S_2S_1S_0$	000	001	010	100	011	101	110	111
错码位置	无错	a_0	a_1	a_2	a_3	a_4	a_5	a_6

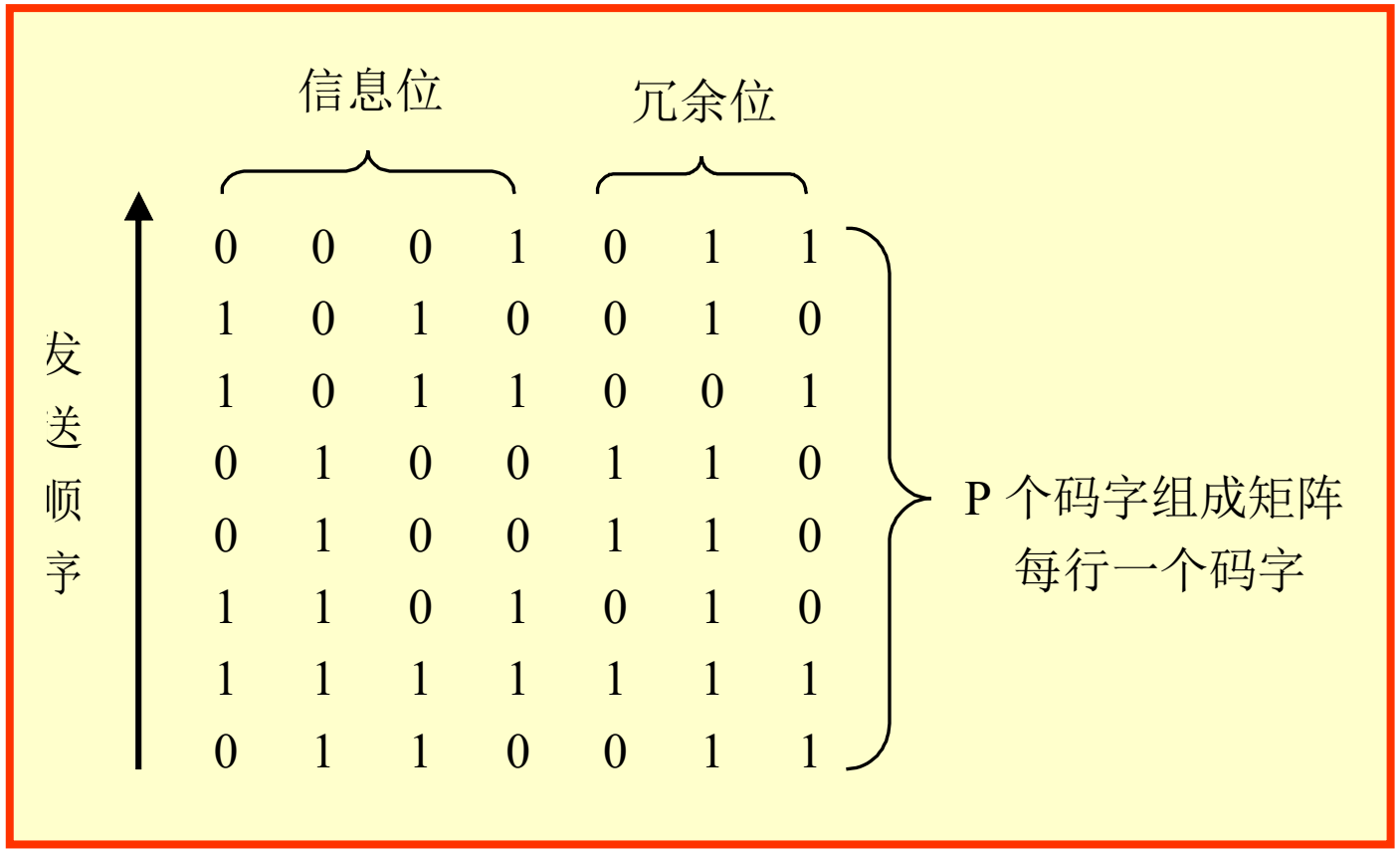
$$S_2 = a_2 \oplus a_4 \oplus a_5 \oplus a_6$$
$$S_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_6$$
$$S_0 = a_0 \oplus a_3 \oplus a_4 \oplus a_6$$

$$a_2 = a_4 \oplus a_5 \oplus a_6$$
$$a_1 = a_3 \oplus a_5 \oplus a_6$$
$$a_0 = a_3 \oplus a_4 \oplus a_6$$

纠正一比特错的线性分组码

纠正突发错误：

- 连续P个码字排成一个矩阵(pxn)，每行一个码字，从而可以纠正突发长度≤P的突发错



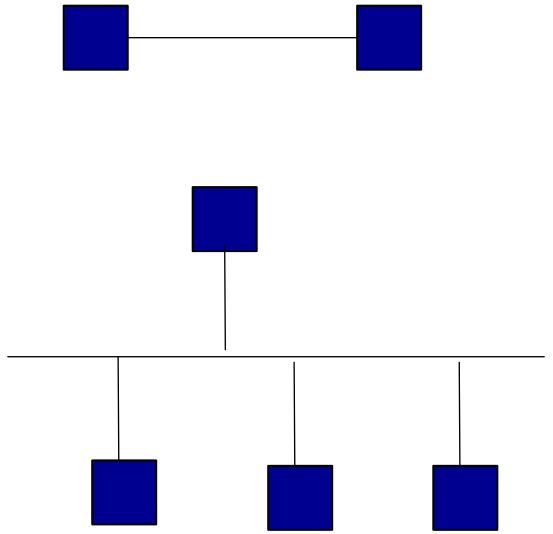
差错控制方式

- 自动请求重发ARQ：Automatic Request for Repeat
  - 接收方检测错误，通知发送方重传
  - 双向信道，发送方缓存发送的数据
- 前向纠错FEC： Forward Error Correction
  - 接收方不仅可以检测错误，而且知道错误的位置
  - 采用纠错码，无需反向信道，无需重发
- 混合纠错HEC：ARQ和FEC结合（误码率较高时采用ARQ，低时采用FEC）

数据链路层的功能

在物理层提供的比特流传输服务基础上把高（网络）层来的数据沿链路传递给相邻的节点

- 帧同步：帧(Frame)的起始和结束定位
  - 传输过程中可能有差错（只需要重传那些出错的帧）；更有效地利用链路
- 差错控制
  - 差错检测: 差错编码
  - 反馈重发：确认和重传
  - 超时和序号：一帧可能完全消失
- 流量控制: 滑动窗口协议
  - 接收方的能力有限制，控制发放方的发送速率
- 链路管理
  - 点到点的链路相对来说要简单些：
    - 如果采用面向连接的方式：通知对方已准备好，序号初始化等，通信过程中维持链路，最后释放链路
  - 多路访问：多个节点共享一条广播链路

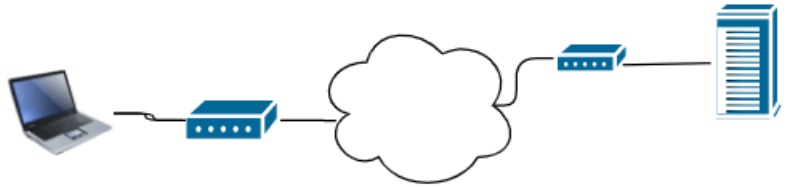


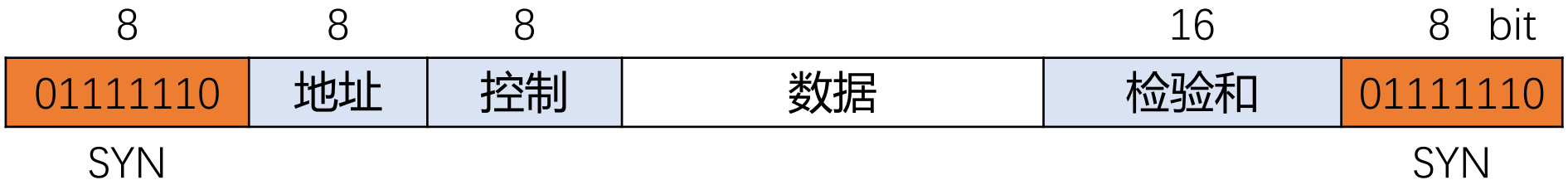
帧同步机制： 3.4协议举例

- 异步协议，**以字符为单位**进行同步，比如RS232接口
  - 传输字符，在字符起始处进行同步
  - 发送方和接收方采用近似同一频率的时钟,短时间内时钟的偏移是可以忍受的
- 同步协议，**以帧为单位**进行同步
  - 在**帧（比较长的数据单元）的起始处**同步
  - 在帧传输过程中，维持固定的时钟，或者采用某种方法（比如物理层介绍的4B/5B编码等）将时钟信号编码进数据中
  - 如何决定帧的开始和结束?
    - 面向字符的同步协议：特殊字符表示帧的开始和结束
    - 面向比特的同步协议：比特模式表示帧的开始和结束
    - 字节计数的同步协议：特殊字符表示开始，字节数表示结束
    - 违例编码法：采用不在正常数据部分出现的物理信号模式

面向字符的同步协议：异步PPP

- 利用特殊定义的字符SYN(0x7E=01111110)来标识帧的起始和结束位置
- 字符填充：特殊字符可能在数据部分出现
  - 利用转义字符DLE(0x7D=01111101)来实现数据透明
  - 除了SYN/DLE等外，可能还有其他控制字符也需要转义
  - **补充：ACCM异步控制字符映射表(32比特的整数)**
    - 在实际的链路前还可能包括一个Modem，其会解释一些控制字符
    - ACCM给出了除了SYN和DLE外还有**哪些控制字符(0~31)需要转义**，如XON/XOFF
    - 在PPP链路建立时发送ACCM，协商需要转义的控制字符
    - ACCM中的对应的比特n为1时表示控制字符n需要转义





特殊字符随采用的字符编码集而不同：ASCII/EBCDIC

面向字符的同步协议：异步PPP

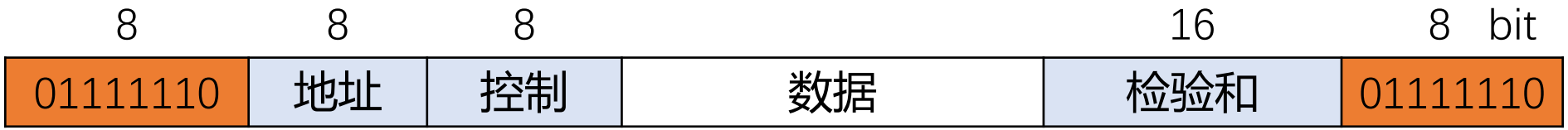
如何实现数据透明?

- 需要转义的字符：SYN、DLE，以及那些可能被modem解释从而通过ACCM机制描述的那些字符(<0x20)
- 发送方：
  - 如果需要转义的特殊字符(如SYN, 0x7e)在数据部分出现，在其前面添加DLE，同时该字符半加0x20（保证其大于等于0x20，即32）：如0x7e → 0x7d5e
  - 如果DLE本身出现在数据部分，同样添加DLE，即0x7d→ 0x7d5d
- 接收方：
  - 看到DLE(0x7D)，去掉DLE，后面的字节半加0x20来恢复原来的数据

数据部分	7E		7C	7D		7D		11	
转义后	7D	5E	7C	7D	5D	7D	5D	7D	31

面向比特的同步规程

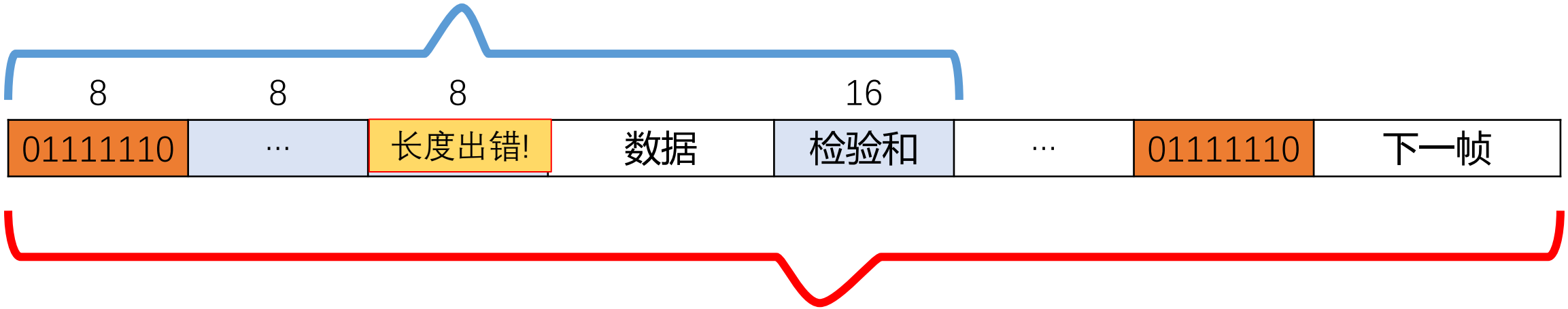
- 通过一个特定的比特模式“01111110”来标识帧的起始和结束位置
- 比特填充：保证数据部分不会出现连续5个1
  - 帧中的其他字段中如果出现连续5个1，则之后插入一个0
  - 当接收时，如果出现连续5个1后跟一个0，则删除0



数据部分	01000001 11111000 01111110 11111111 11100010
转义后	01000001 111101000 011111010 111110111 110100010

字节计数的同步规程

- 同步字符进行帧同步，标志帧开始
- 字节计数来确定帧的结束边界位置
- 一旦字节计数出错带来灾难性后果，在数据链路层较少使用
- 可用于更高层协议，比如基于TCP的应用层协议中



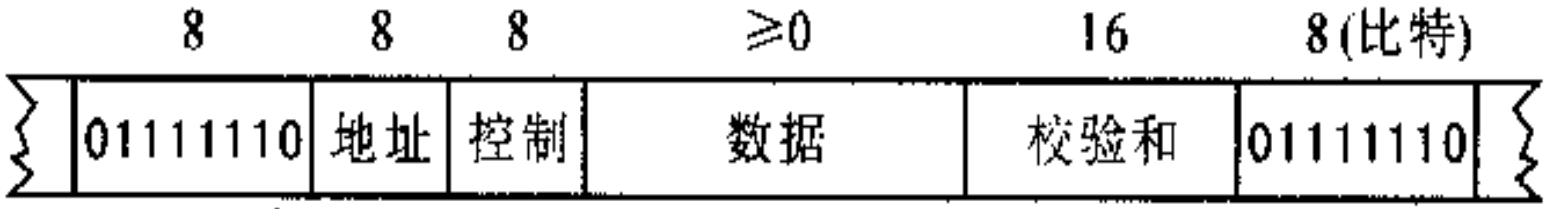
违例编码法

- 通过不会在数据部分出现的编码信号来进行同步
  - 比如曼彻斯特编码中的高-高/低-低信号
- 以太网帧：面向比特的同步规程 + 违例编码
  - 7个字节的前导，“唤醒”接收者，接收者利用它进行时钟同步
  - 1个字节的帧开始，“提醒”接收者帧的开始
  - 通过**检测到链路的空闲**确定帧的结束
    - 早期通过检测是否有载波来实现
    - 现在采用检测到**空闲信号模式(违例编码）**来实现

7	1	6	6	2	46-1500	4
前导	帧开始	目的MAC地址	源MAC地址	帧类型	数据+填充	CRC
10101010	10101011					

数据链路层协议：HDLC（High-Level Data Link Control）

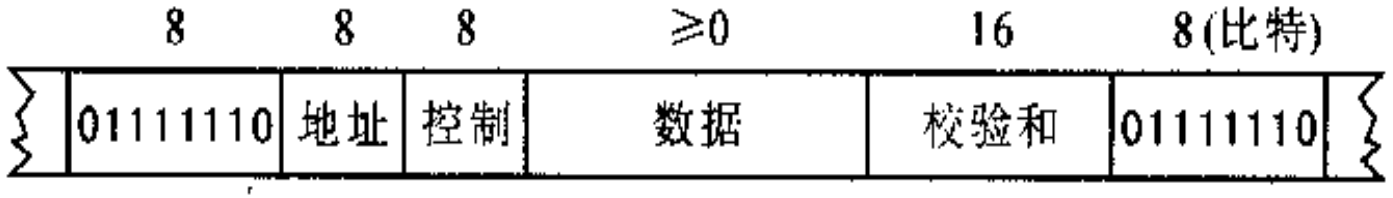
- 用于点到点和点到多点链路，两端可以充当主站、从站或者主从站的角色
- **面向比特的同步规程**：01111110表示帧的开始和结束
- 地址：标识目的节点（主站）或者源节点（从站），最低(E)为扩展位
- 控制：指出属于那种帧
  - P(oll)/F(inal)：主站发出时表示P，如果为1要求对方响应；从站发出时表示F，为1表示响应结束



注意在描述帧格式时采用最低位在前的顺序

数据链路层协议：HDLC

- 信息I帧：携带用户数据，帧的序号在N(S)中。并且支持捎带确认：N(R)表示之前的所有帧都正确
- 监控S帧：单独确认机制，没有数据部分
  - RR(00,Receive ready)：准备接收I帧，单独确认+XON流量控制
  - RNR(10,Receive not ready)：尚未准备好接收，单独确认+XOFF流量控制
  - REJ (01, Reject)：要求发送方重传N(R)开始的I帧，回退N
  - SREJ(11,Selective reject)：要求发送方重传编号为N(R)的I帧，选择重传
- 无编号U帧：其它链路控制功能
  - M=00000：UI（无编号信息）帧 ，控制字段取值0X03



注意在描述帧格式时采用最低位在前的顺序

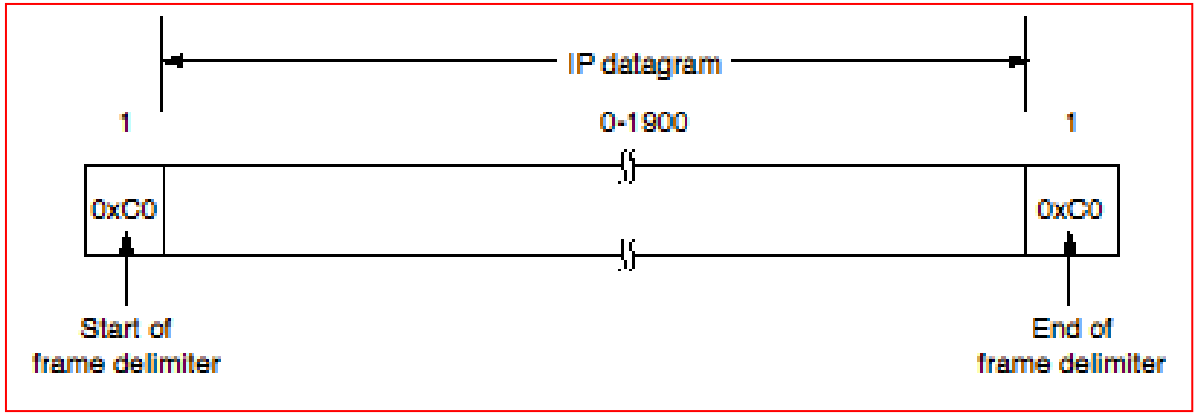
	3			3
I帧	0	N(S)	P/F	N(R)

		2		3
S帧	1	0	类型	P/F	N(R)

U帧	1	1	M	P/F	M

数据链路层协议：PPP（Point-to-Point Protocol）

- SLIP是一种面向字符的同步协议，仅仅包含封装功能
- PPP的设计目标
  - 用于任何类型的点到点链路
  - 可以支持多个网络层协议
  - 提供差错检测功能
  - 不提供流量控制和差错控制



- PPP协议包括：
  - 链路控制协议LCP（0xc021）：链路维护、认证和选项协商
    - 认证机制支持CHAP（共享密钥）和PAP（明文）
  - 网络层协议(0x0021:IP): 支持IP等协议
  - 网络控制协议NCP(0x8021:IPCP)：
    - 网络层协议参数的协商
    - IP地址, 缺省路由器, DNS服务器等

1	1	1	1 或 2	可变长	2 或 4	1
01111110	11111111	00000011	协议	数据	FCS	01111110
标志	地址	控制				标志

- 异步PPP：面向字符的同步协议，常用软件实现
- 同步PPP：面向比特的同步协议，常用硬件实现

PPP通信阶段状态转换图

- 最初为dead状态，物理链路建立进入链路建立状态
- 链路建立状态：
  - 利用LCP协议协商选项，最大帧（Maximum Receive Unit）大小、认证机制等
  - LCP协议监测链路的质量

- 协商、认证完成后进入网络层配置状态
  - IPCP协议配置IP地址、是否压缩等
- 进入打开状态：开始传输网络层数据
- 一方发送LCP终止请求进入中止中状态
- 中止确认后进入Dead状态