3.3 差错检测

3.3 差错检测

- 实际的通信链路都不是理想的，比特在传输过程中可能会产生差错：1可能会变成0，而0也可能变成1。这称为**比特差错**。

3.3 差错检测

- 实际的通信链路都不是理想的，比特在传输过程中可能会产生差错：1可能会变成0，而0也可能变成1。这称为**比特差错**。
- 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)。
- 使用**差错检测码**来检测数据在传输过程中是否产生了比特差错，是数据链路层所要解决的重要问题之一。

以太网V2的MAC帧（最大长度为1518字节）				
6字节	6字节	2字节	46 ~ 1500 字节	4字节
目的地址	源地址	类型	数 据 载 荷	FCS

PPP帧的格式						
1字节	1字节	1字节	2字节	不超过1500字节	2字节	1字节
标志	地址	控制	协议	数 据 载 荷	FCS	标志

3.3 差错检测

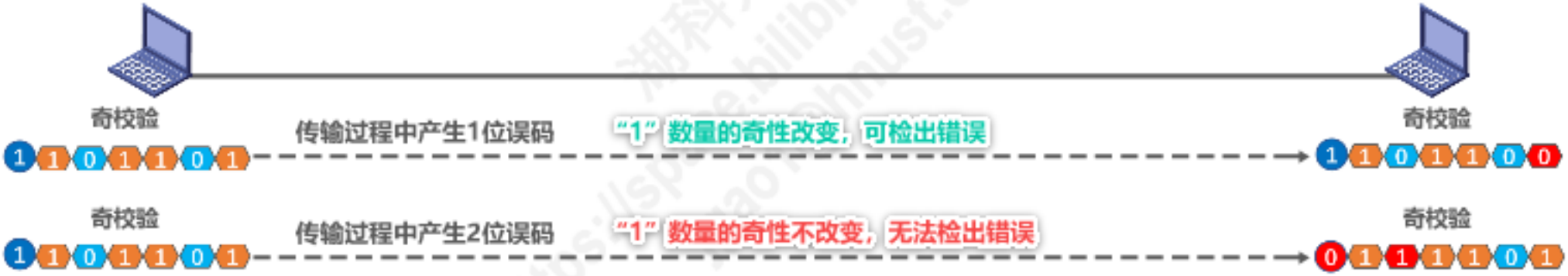
奇偶校验

- 在待发送的数据后面**添加1位奇偶校验位**，使整个数据（包括所添加的校验位在内）中**“1”的个数**为奇数（奇校验）或偶数（偶校验）。

3.3 差错检测

奇偶校验

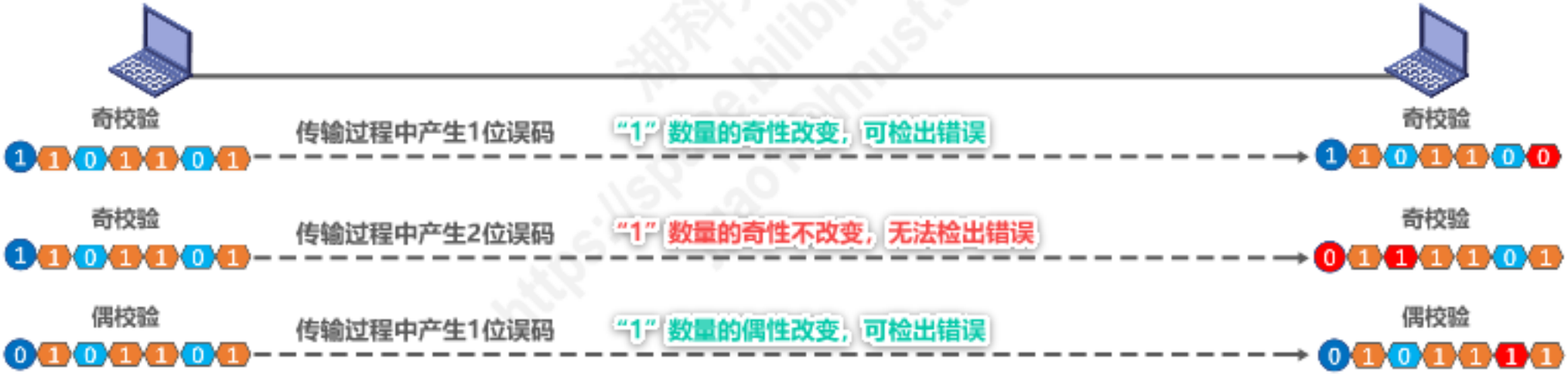
- 在待发送的数据后面**添加1位奇偶校验位**，使整个数据（包括所添加的校验位在内）中“1”的个数为奇数（奇校验）或偶数（偶校验）。



3.3 差错检测

奇偶校验

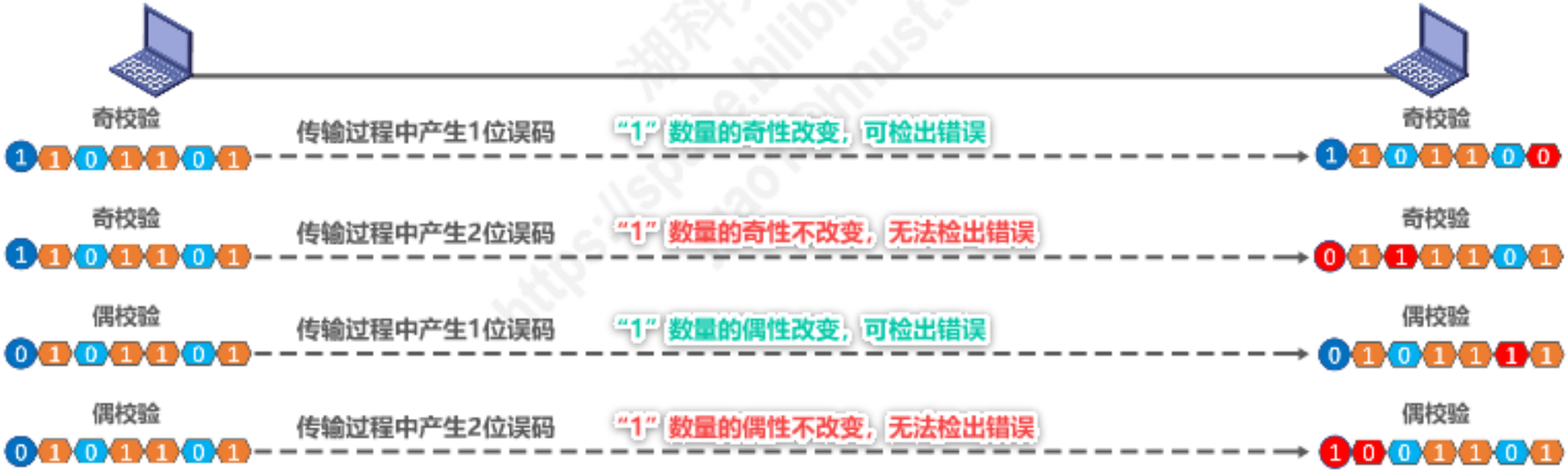
- 在待发送的数据后面**添加1位奇偶校验位**，使整个数据（包括所添加的校验位在内）中**“1”的个数**为奇数（奇校验）或偶数（偶校验）。



3.3 差错检测

奇偶校验

- 在待发送的数据后面**添加1位奇偶校验位**，使整个数据（包括所添加的校验位在内）中“1”的个数为奇数（奇校验）或偶数（偶校验）。
- 如果有**奇数个位发生误码**，则奇偶性发生变化，**可以检查出误码**；
- 如果有**偶数个位发生误码**，则奇偶性不发生变化，**不能检查出误码（漏检）**；



3.3 差错检测

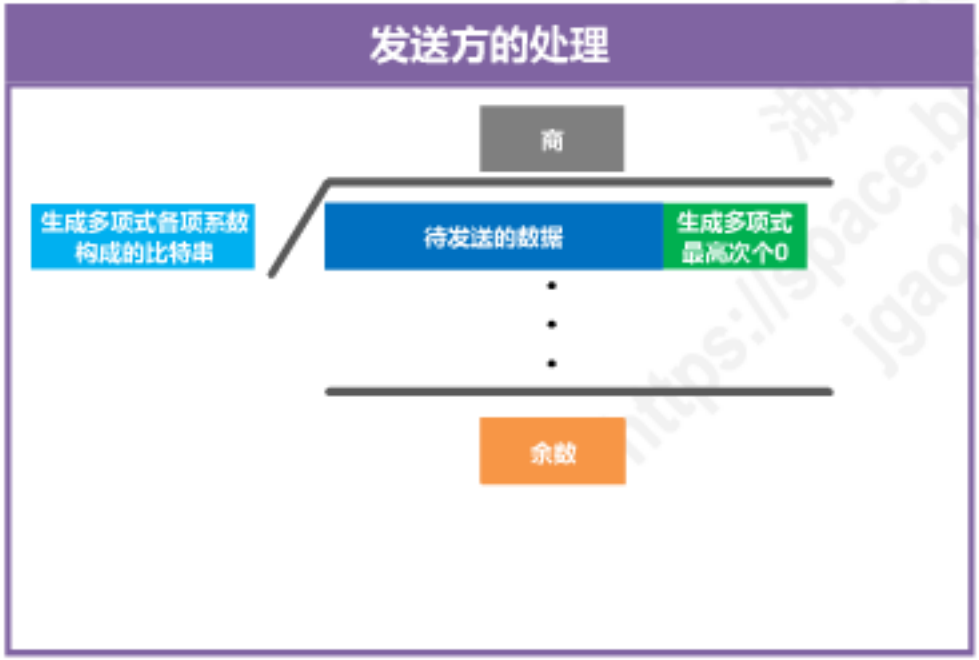
循环冗余校验CRC(Cyclic Redundancy Check)

- 收发双方约定好一个**生成多项式**G(x)；
- 发送方基于待发送的数据和生成多项式计算出差错检测码（**冗余码**），将其添加到待传输数据的后面一起传输；
- 接收方通过生成多项式来计算收到的数据是否产生了误码；

3.3 差错检测

循环冗余校验CRC(Cyclic Redundancy Check)

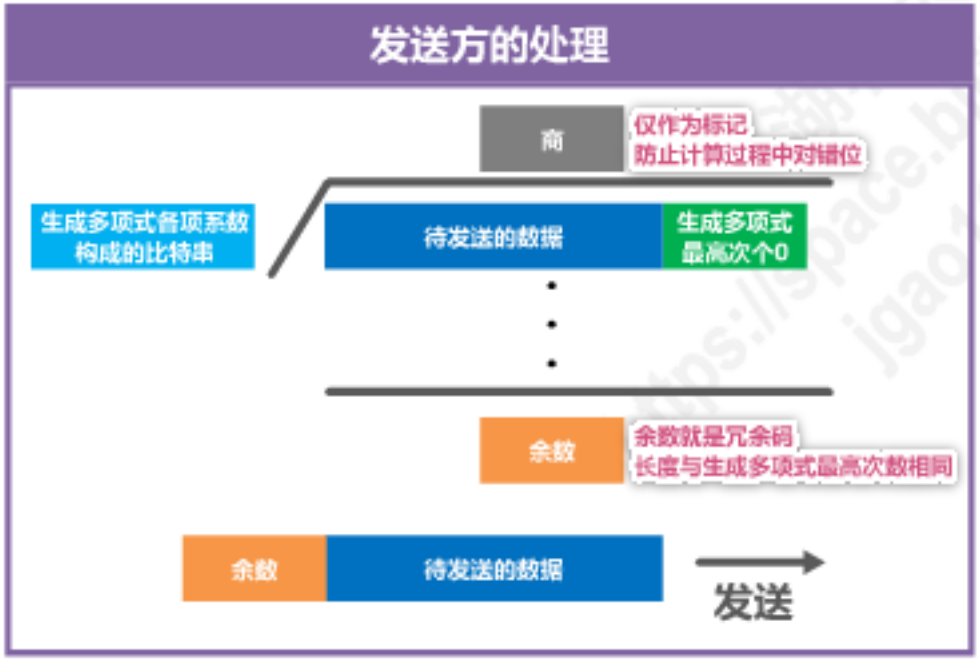
- 收发双方约定好一个**生成多项式G(x)**；
- 发送方基于待发送的数据和生成多项式计算出差错检测码（**冗余码**），将其添加到待传输数据的后面一起传输；
- 接收方通过生成多项式来计算收到的数据是否产生了误码；



3.3 差错检测

循环冗余校验CRC(Cyclic Redundancy Check)

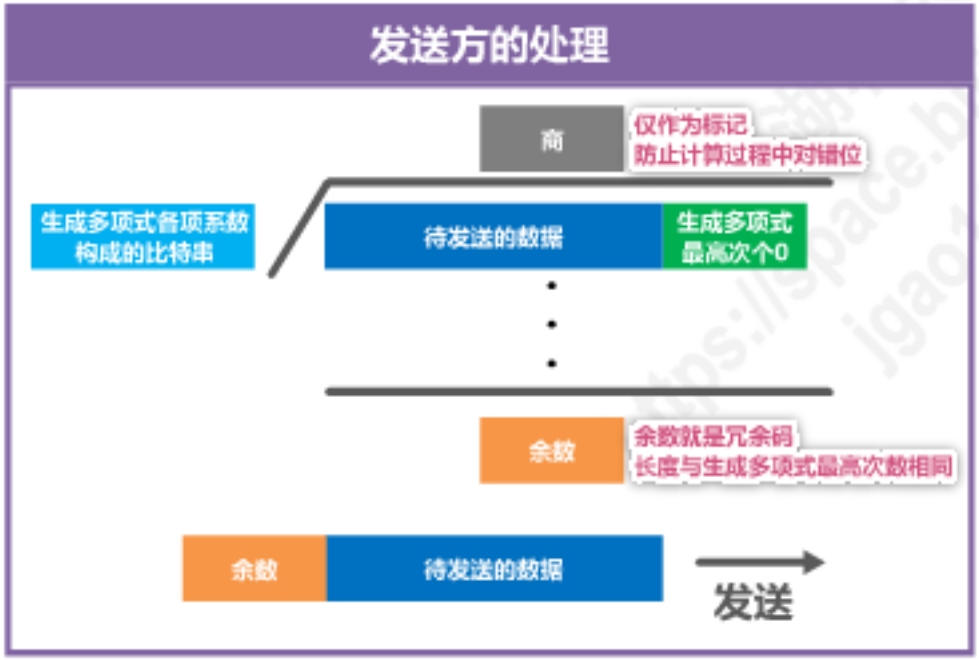
- 收发双方约定好一个**生成多项式**G(x)；
- 发送方基于待发送的数据和生成多项式计算出差错检测码（**冗余码**），将其添加到待传输数据的后面一起传输；
- 接收方通过生成多项式来计算收到的数据是否产生了误码；

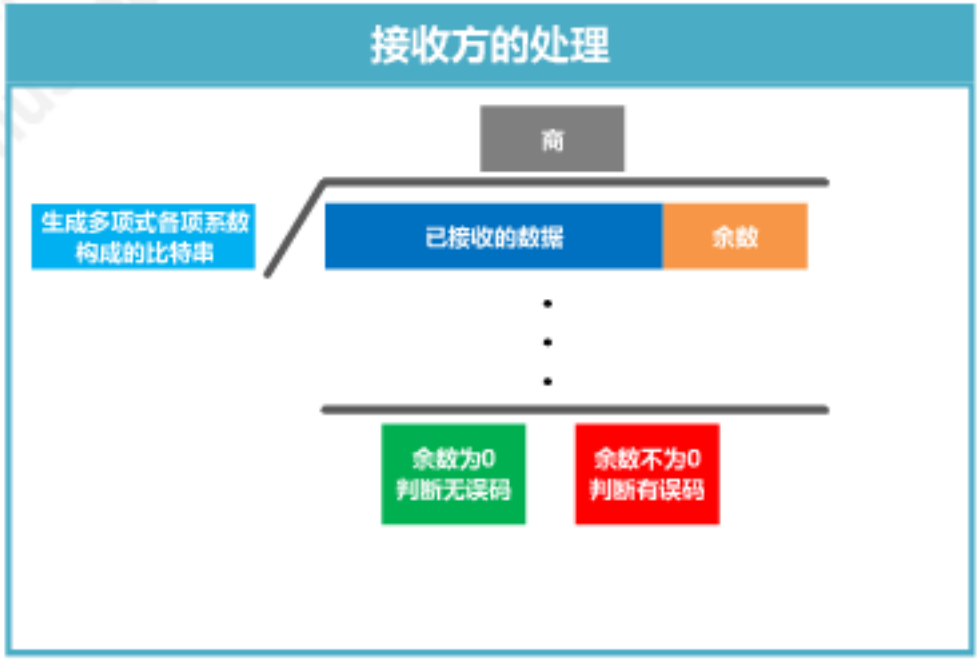


3.3 差错检测

循环冗余校验CRC(Cyclic Redundancy Check)

- 收发双方约定好一个**生成多项式**G(x)；
- 发送方基于待发送的数据和生成多项式计算出差错检测码（**冗余码**），将其添加到待传输数据的后面一起传输；
- 接收方通过生成多项式来计算收到的数据是否产生了误码；





3.3 差错检测

循环冗余校验CRC(Cyclic Redundancy Check)

- 收发双方约定好一个**生成多项式**G(x)；
- 发送方基于待发送的数据和生成多项式计算出差错检测码（**冗余码**），将其添加到待传输数据的后面一起传输；
- 接收方通过生成多项式来计算收到的数据是否产生了误码；

【生成多项式举例】

$$G(x) = x^4 + x^2 + x + 1$$
$$= \boxed{1}\cdot x^4 + \boxed{0}\cdot x^3 + \boxed{1}\cdot x^2 + \boxed{1}\cdot x^1 + \boxed{1}\cdot x^0$$

生成多项式各项系数构成的比特串：10111

3.3 差错检测

循环冗余校验CRC(Cyclic Redundancy Check)

- ☐ 收发双方约定好一个**生成多项式**G(x)；
- ☐ 发送方基于待发送的数据和生成多项式计算出差错检测码（**冗余码**），将其添加到待传输数据的后面一起传输；
- ☐ 接收方通过生成多项式来计算收到的数据是否产生了误码；

【生成多项式举例】

$$G(x) = x^4 + x^2 + x + 1$$
$$= 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

生成多项式各项系数构成的比特串：10111

【常用的生成多项式】

算法要求生成多项式必须包含最低次项

$$CRC\text{-}16 = x^{16}+x^{15}+x^2+1$$
$$CRC\text{-}CCITT = x^{16}+x^{12}+x^5+1$$
$$CRC\text{-}32 = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

3.3 差错检测

【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数

待发送信息后面添加生成多项式最高次数个1

101001000

3.3 差错检测

【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数

待发送信息后面添加生成多项式最高次数个1

2 构造除数

生成多项式各项系数构成的比特串

1101　101001000

3.3 差错检测

【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数

待发送信息后面添加生成多项式最高次数个1

2 构造除数

生成多项式各项系数构成的比特串

3 做“除法”

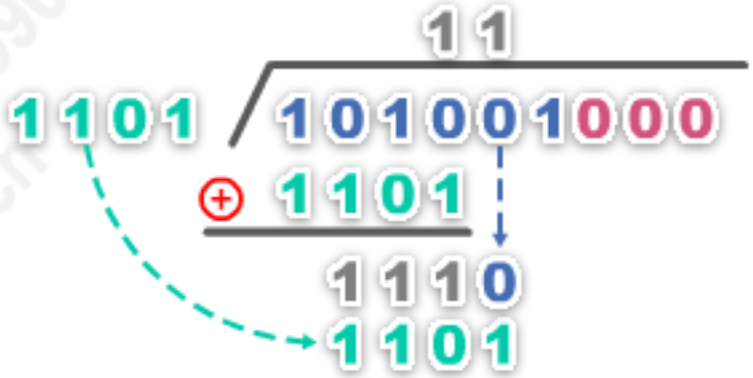
3.3 差错检测

【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数
待发送信息后面添加生成多项式最高次数个1

2 构造除数
生成多项式各项系数构成的比特串

3 做“除法”



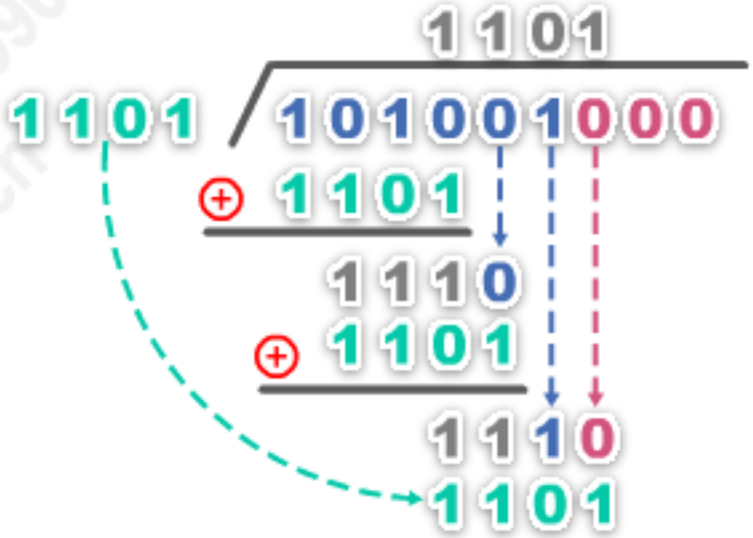
3.3 差错检测

【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数
待发送信息后面添加生成多项式最高次数个1

2 构造除数
生成多项式各项系数构成的比特串

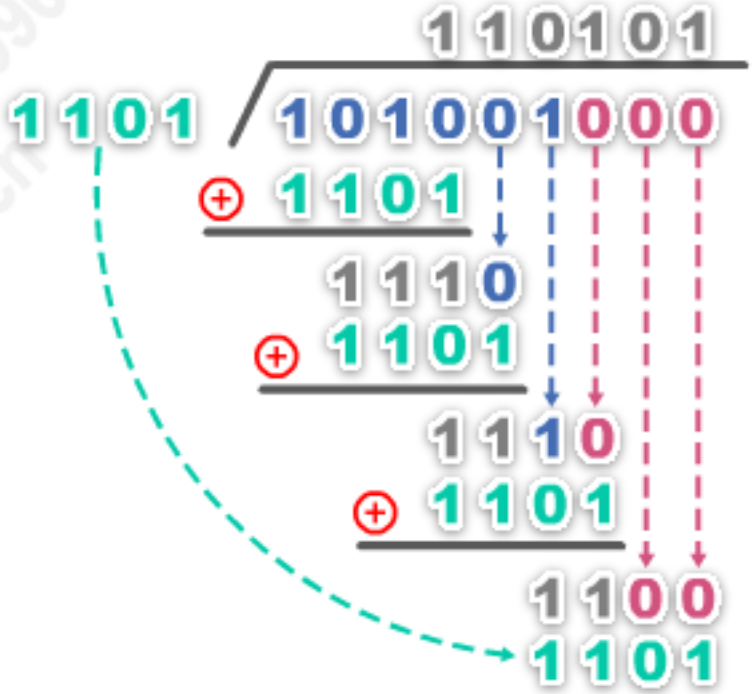
3 做“除法”



3.3 差错检测

【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1. **构造被除数**
   待发送信息后面添加生成多项式最高次数个1
2. **构造除数**
   生成多项式各项系数构成的比特串
3. **做“除法”**



3.3 差错检测

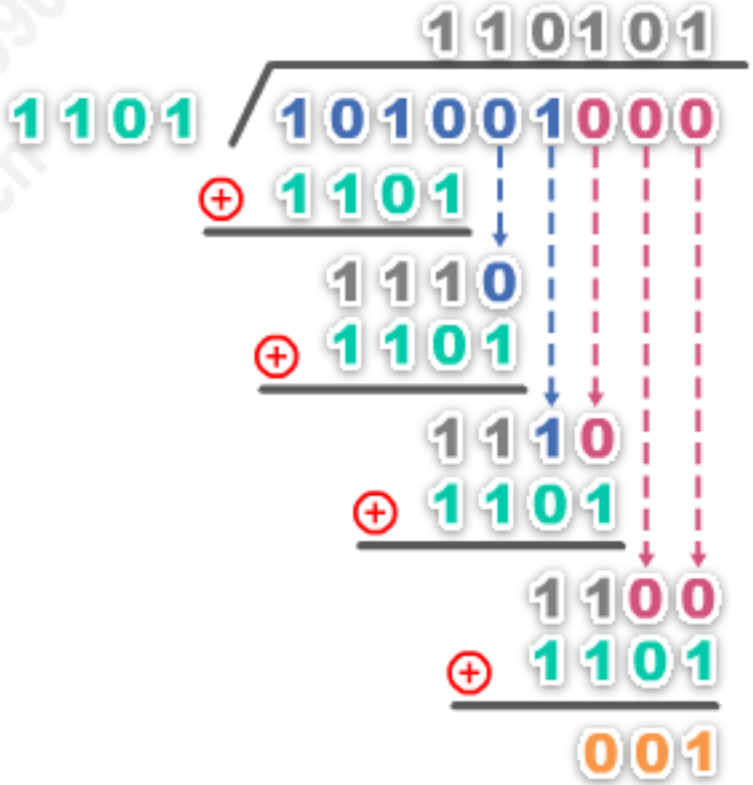
【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数
待发送信息后面添加生成多项式最高次数个1

2 构造除数
生成多项式各项系数构成的比特串

3 做“除法”

4 检查余数
余数的位数应与生成多项式最高次数相同，如果位数不够，则在余数前补0来凑足位数。



3.3 差错检测

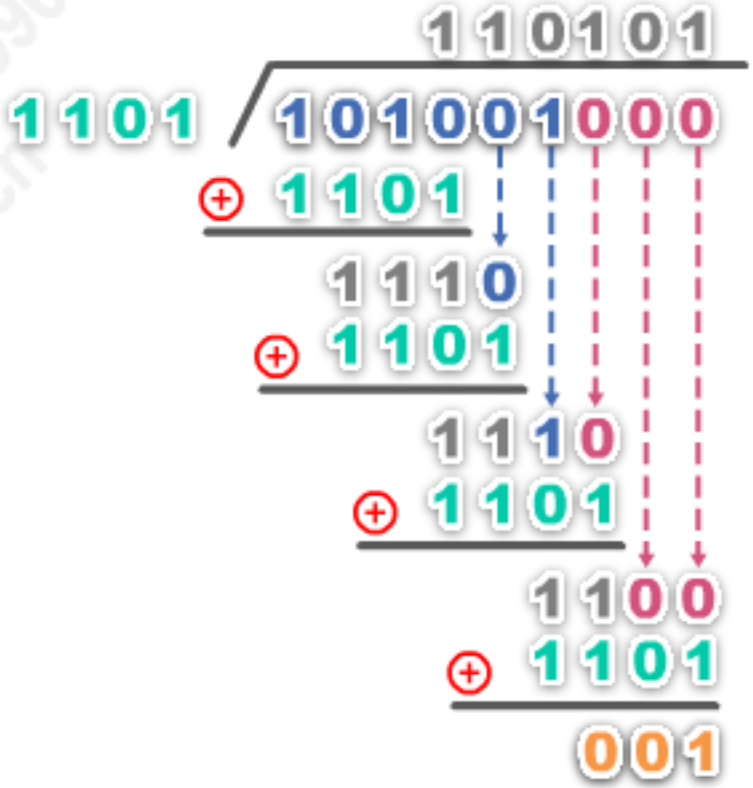
【循环冗余校验CRC举例】待发送的信息为101001，生成多项式为$G(x) = x^3 + x^2 + 1$，计算余数。

1 构造被除数
待发送信息后面添加生成多项式最高次数个1

2 构造除数
生成多项式各项系数构成的比特串

3 做“除法”

4 检查余数
余数的位数应与生成多项式最高次数相同，如果位数不够，则在余数前补0来凑足位数。



发送

3.3 差错检测

【循环冗余校验CRC举例】接收到的信息为101101001，生成多项式为 $G(x) = x^3 + x^2 + 1$，判断传输是否误码？

1 构造被除数
接收到的信息就是被除数

2 构造除数
生成多项式各项系数构成的比特串

3 做“除法”

4 检查余数
余数为0，可认为传输过程无误码；
余数不为0，可认为传输过程产生误码。

```
          110010
      ┌─────────
1101  │ 101101001
      ⊕ 1101
        ──────
        1100
      ⊕ 1101
        ──────
        1100
      ⊕ 1101
        ──────
          11
```

余数不为0，表明传输过程产生误码！

3.3 差错检测

- **检错码**只能检测出帧在传输过程中出现了差错，但并不能定位错误，因此**无法纠正错误**。
- 要想纠正传输中的差错，可以使用冗余信息更多的**纠错码**进行**前向纠错**。但纠错码的开销比较大，在**计算机网络中较少使用**。

3.3 差错检测

- **检错码**只能检测出帧在传输过程中出现了差错，但并不能定位错误，因此**无法纠正错误**。
- 要想纠正传输中的差错，可以使用冗余信息更多的**纠错码**进行**前向纠错**。但纠错码的开销比较大，在**计算机网络中较少使用**。
- 循环冗余校验**CRC**有很好的检错能力（**漏检率非常低**），虽然计算比较复杂，但非常**易于用硬件实现**，因此被**广泛应用于数据链路层**。
- 在计算机网络中通常采用我们后续课程中将要讨论的**检错重传方式来纠正传输中的差错，或者仅仅是丢弃检测到差错的帧**，这取决于数据链路层向其上层提供的是可靠传输服务还是不可靠传输服务。

3.3 差错检测

- **检错码**只能检测出帧在传输过程中出现了差错，但并不能定位错误，因此**无法纠正错误**。
- 要想纠正传输中的差错，可以使用冗余信息更多的**纠错码**进行**前向纠错**。但纠错码的开销比较大，在**计算机网络中较少使用**。
- 循环冗余校验**CRC**有很好的检错能力（**漏检率非常低**），虽然计算比较复杂，但非常**易于用硬件实现**，因此被**广泛应用于数据链路层**。
- 在计算机网络中通常采用我们后续课程中将要讨论的**检错重传方式来纠正传输中的差错**，**或者仅仅是丢弃检测到差错的帧**，这取决于数据链路层向其上层提供的是可靠传输服务还是不可靠传输服务。