

第三章 数据链路层

检错码



检错码

□ 为什么要用检错码？


- 纠错需要较多的冗余位，信道利用率不高

□ 局域网中，主要使用的是检错码

- 奇偶校验码（海明距离为2，检1位错）
- 互联网校验和
- 循环冗余校验码
- 。 。 。 。 。 。



检错码 – 奇偶校验

- 奇偶位取值等同于对数据位进行模2和运算
 - 也等同于 XOR 运算; (偶校验时)
 - 例如, 采用偶校验: 1110000  11100001
 - 接收方检查是否存在单个比特的错误



检错码 – 奇偶校验

□ 查出偶数个错误的简单方法

- 例如: 1 error, 11100**1**01; 5个1, 奇数个, 检出错
- 例如: 3 errors, 11**011**001; 5个1, 奇数个, 检出错
- 例如: 2 errors, 1110**11**01; 6个1, 偶数个, 不能检出错误, 判定为正确。

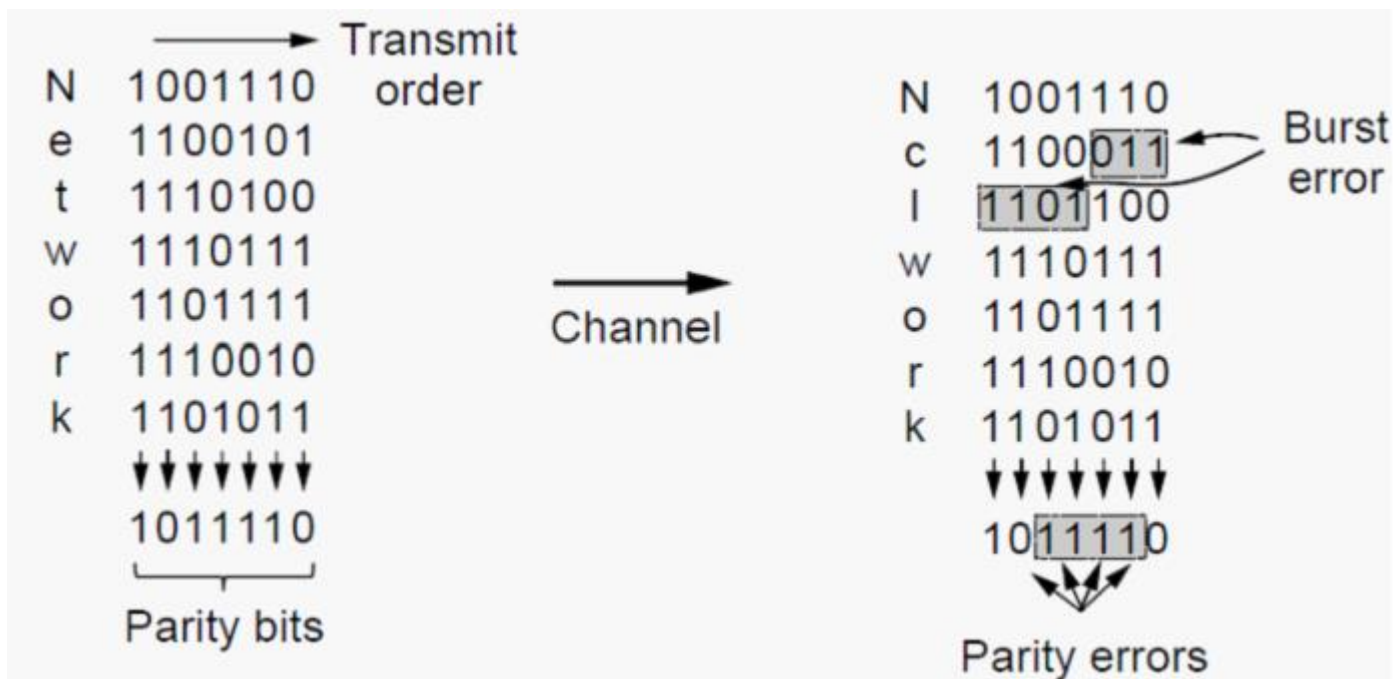
□ 出错误的概率为 $\frac{1}{2}$



检错码 – 奇偶校验

□ 交替的N位校验，可检查出最多N位的突发错误

➤ N位下的突发都可检出





检错码 – 校验和

- 校验和通常是按照N位码字来进行模2加/和运算，并将运算结果附加在数据报文尾部，作为校验位。
- 例如：16位的互联网补码校验和
- 特点：
 - 比奇偶检验更好的检错性能
 - 能检出高达N位的突发错误
 - 检错随机错误率 $1-2^N$
 - 易受系统错误干扰，比如，增加的“0”



互联网校验和计算文档

□ RFC1071: computing the internet checksum

- (1) 待校验的相邻字节成对组成16比特的整数一行，按列从低位开始计算其模2和；并将结果按位取反码，作为校验和取值。
 - (2) 检查校验和时，将所有字节，包括校验和，进行相加并求二进制反码。接收方：如果结果为全1，无错误
- 注意：如果某列的模2和有溢出，向高位进位，如果高位产生进位，循环向低位进位。



检错码—循环冗余检错码CRC

- 任何一个 k 位的帧看成为一个 $k-1$ 次的多项式
 - 如：1011001看成 $x^6+x^4+x^3+x^0$ (k 项 $k-1$ 阶多项式)
- 设定一个多项式编码生成多项式 $G(x)$ ， $G(x)$ 为 r 阶
- 设一个 m 位的帧的多项式为 $M(x)$ ， $m > r$ ，即 $M(x)$ 比 $G(x)$ 长
- 如 $x^r M(x)/G(x) = Q(x) + R(x)$ 其中 $Q(x)$ 为商、 $R(x)$ 为余数，则
($x^r M(x) - R(x)$)一定能被 $G(x)$ 整除，即余数为0
- 在二进制运算中，减法和加法都做异或运算，即相同得0，相异得1，比如： $0+1=1$ ， $1+1=0$ ， $0-1=1$



一个十进制的仿真例子

- 收发双方约定，被“3”整除
- 发送方发送数字“23”， $23/3=7+2$ ，所以，发送 $23-2=21$
- 收方收到了“21”，用“3”除，刚好除尽，没出错。如果收到“22”，不能被“3”除，错了。
 - 但是，如果，收到了“24”？会怎样呢？

什么是模2运算？

模2加 以及 模2减 等同于异或运算，即相同得0，相异得1。

$$0 \oplus 0 = 0; 0 \oplus 1 = 1;$$

$$1 \oplus 0 = 1; 1 \oplus 1 = 0.$$

相同得0，不同得1

– Modulo 2 multiplication:

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

-- Modulo 2 division:

$$\begin{array}{r} 101 \overline{) 10000} \\ \underline{101} \\ 010 \\ \underline{000} \\ 100 \\ \underline{101} \\ 01 \end{array}$$

例：CRC码计算

□ 如一帧为1101011011 ($m=10$)

$$\text{即 } M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x + 1$$

□ $G(x) = x^4 + x + 1$ ($r=4$ 阶)

□ $T(x) = x^4 M(x)$ (相当于在原码字后加 r 个0)

$$= x^4 (x^9 + x^8 + x^6 + x^4 + x^3 + x + 1)$$

$$= x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4$$

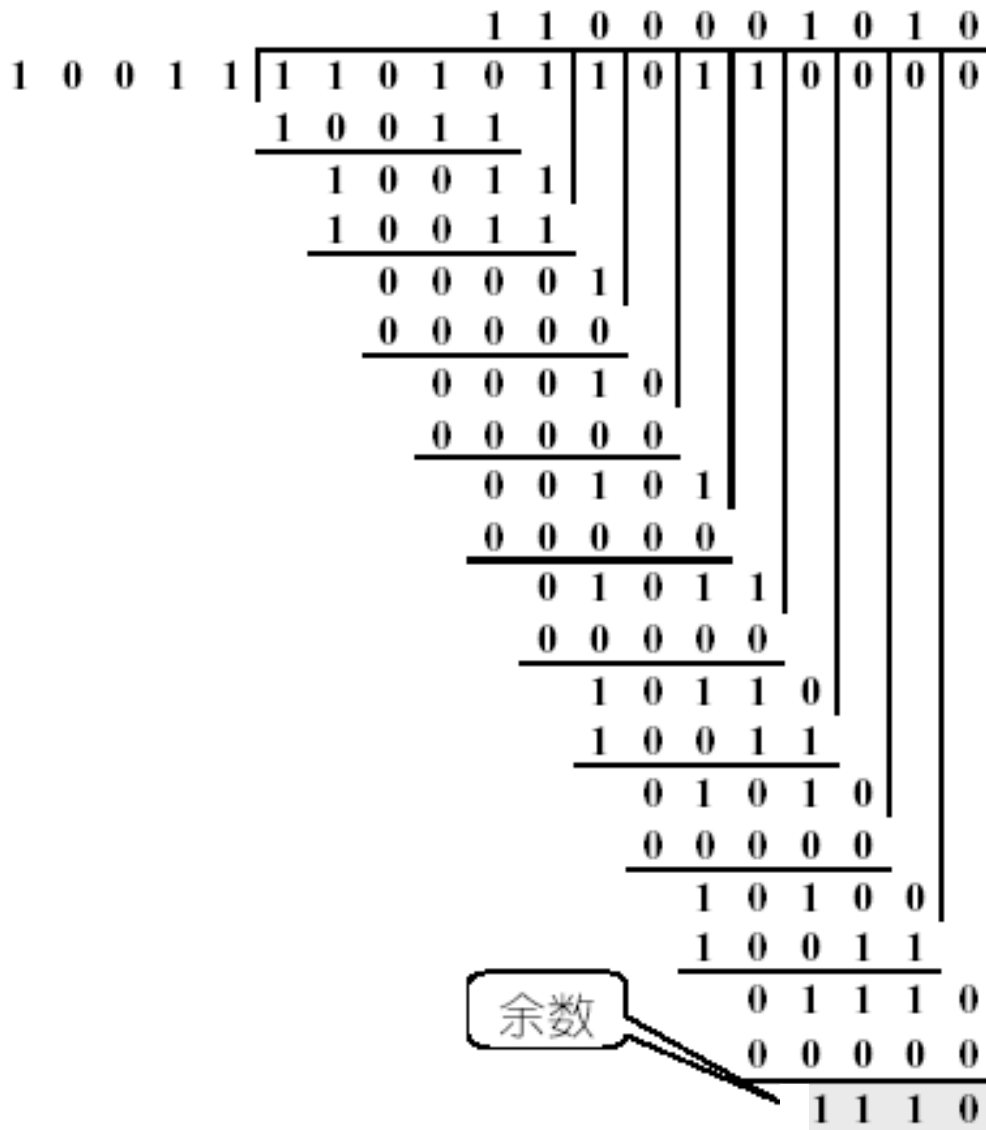
例：CRC码计算

帧: 1101011011

除数: 10011

传输帧: 1101011011110

帧数据 余数



例：CRC码计算

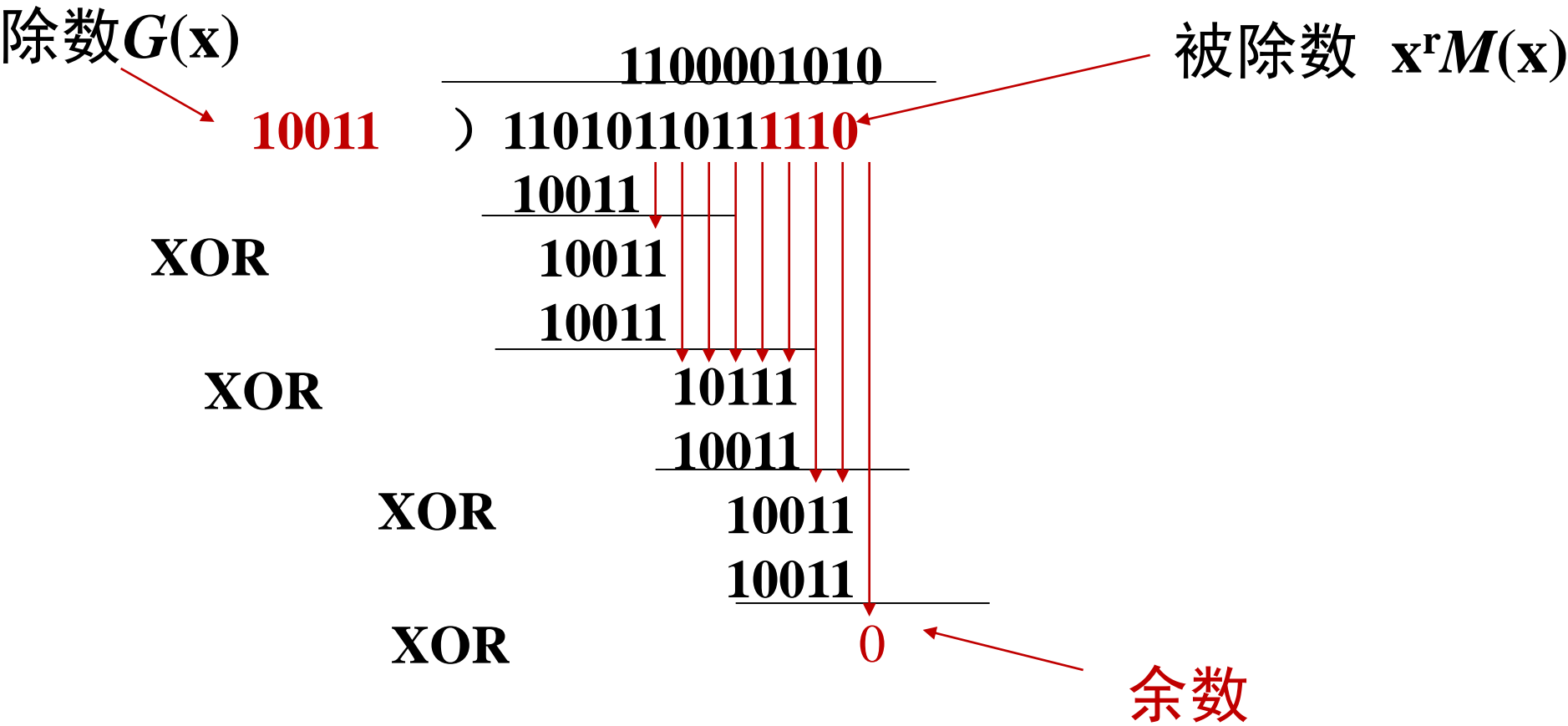
计算 $11010110110000/10011$ 得余数1110， $11010110110000-1110=11010110111110$ ，所以：

编码后得CRC码为：11010110111110

当这个码字达到接收方时：

- 如CRC码在接收端能被10011整除则说明接收正确。
- 如发送方发送的 $T(x)$ ，接收方收到的是 $T(x)+E(x)$ ，如果不能被整除，则被检测到已出错。

接收端的检查





生成多项式国际标准

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

用于字符长度为6位

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

用于字符长度为8位

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

用于字符长度为8

□ CRC32:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x + 1$$



CRC计算算法

- 在数据帧的低端加上 r 个零，对应多项式为 $X^rM(x)$
- 采用模2除法，用 $G(x)$ 去除 $X^rM(x)$ ，得余数
- 采用模2减法，用 $X^rM(x)$ 减去余数，得到带CRC校验和的帧



CRC小结

□ Sender

- 在数据帧的低端加上 r 个零，对应多项式为 $X^rM(x)$
- 采用模2除法，用 $G(x)$ 去除 $X^rM(x)$ ，得余数
- 采用模2减法，用 $X^rM(x)$ 减去余数，得到带CRC校验和的帧

□ Receiver

- 用收到的帧去除以 $G(x)$
- 为零：无错误产生
- 非零：发生了错误，重传



CRC课堂练习

如果生成多项式是 $G(x) = x^3 + x^2 + 1$ ，待传送的原始码字分别是 1111 和 1100，请计算采用CRC编码后的码字分别是多少？

参考答案

1111 111

1100 101



解题过程

解答（1）：生成多项式是3阶的，所以 $r=3$ ，生成多项式对应的位（除数）是：1101

待传输的1111，移位后变为：1111000（被除数），得到余数111，
用1111000-111，得到编码后的码字为：1111**111**

$$\begin{array}{r} 1011 \\ 1101 \overline{) 1111000} \\ \underline{1101} \\ 1000 \\ \underline{1101} \\ 1010 \\ \underline{1101} \\ 111 \end{array}$$



解题过程

解答（2）：生成多项式是3阶的，所以 $r=3$ ，生成多项式对应的位（除数）是：1101

待传输的1100，移位后变为：1100000（被除数），得到余数101，用1100000-101，得到编码后的码字为：1100**101**

$$\begin{array}{r} 1001 \\ 1101 \overline{) 1100000} \\ \underline{1101} \\ 1000 \\ \underline{1101} \\ 101 \end{array}$$



小结

- 检错码有很多，常见的有奇偶校验、互联网校验和、循环冗余校验等。
- 采用循环冗余校验码的系统，需要约定一个生成多项式（除数）。
- 发送方：码字就是被除数减去模2除法的余数。
- 接收方：判定余数是否为零？
 - 为零：无错
 - 不为零：有错

思考题

- 什么是检错码？
- 采用循环冗余校验码，发送方怎么做？
- 采用循环冗余校验码，接收方怎么做？
- 循环冗余校验码，能够检查出多少位错误？

谢谢观看

致谢

本课程课件中的部分素材来自于：（1）清华大学出版社出版的翻译教材《计算机网络》（原著作者：Andrew S. Tanenbaum, David J. Wetherall）；（2）思科网络技术学院教程；（3）网络上搜到的其他资料。在此，对清华大学出版社、思科网络技术学院、人民邮电出版社、以及其它提供本课程引用资料的个人表示衷心的感谢！

对于本课程引用的素材，仅用于课程学习，如有任何问题，请与我们联系！