3.1.2 差错检测

差错检测

设计数据链路层的原因：

- 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。
- 在原始物理传输线路上传输数据信号是有差错的。
- 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。

差错产生的原因和差错类型

- 差错控制 — 检查接收到的数据是否出现差错以及如何纠正差错；
- 通信信道的噪声分为两类：热噪声和冲击噪声；
  - 热噪声引起的差错是随机差错，或随机错；
  - 冲击噪声引起的差错是突发差错，或突发错；
- 引起突发差错的位长称为突发长度；
- 在通信过程中产生的传输差错，是由随机差错与突发差错共同构成的。
- 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。

差错控制编码的基本思想

在发送端被传输的信息位上附加一些冗余位,这些冗余位与信息位之间以某种确定的规则相互关联(约束),接收端按照既定的规则检验信息位与冗余位之间的关系。

差错控制编码的原理

- 信息位：发送端要发送的数据
- 冗余位：发送端在向信道发送信息位之前，先按照某种关系加上一定的冗余位
- 发送与接收的过程：
  - 发送时：信息位+冗余位构成码字发送；
  - 接收时：收到码字后查看信息位和冗余位，并检查它们之间的关系（校验过程），以发现传输过程中是否有差错发生。

差错控制编码分类

- 纠错码：
  - 每个传输的分组带上足够的冗余信息；接收端能发现并自动纠正传输差错。
  - 纠错码需要较多的冗余位，信道利用率不高。
- 检错码（局域网中主要使用的是检错码）
  - 分组仅包含足以使接收端发现差错的冗余信息；接收端能发现出错，但不能确定哪一比特是错的，并且自己不能纠正传输差错。

  目前在数据链路层广泛应用了循环冗余检验CRC （Cyclic Redundancy Check)的检错技术。

循环冗余检验CRC的原理

- 在计算机网络和数据通信中用得最广泛的检错码是一种漏检率低得多也便于实现的循环冗余码CRC（Cyclic Redundancy Code）
- CRC码又称为多项式码。任何一个由二进制数位串组成的代码都可以和一个只含有0和1两个系数的多项式建立一一对应的关系。

【例】 1011011对应的多项式为 $x^6+x^4+x^3+x+1$

而多项式 $x^5+x^4+x^2+x$ **对应的代码为110110**

CRC运算就是在发送数据M后面添加供差错检测用的r位冗余位，构成一个由k位信息位加上r位冗余位组成的n=k+r位码字发送出去。

循环冗余编码CRC工作原理

- 一个要发送的k位的帧对应于一个（k-1）次多项式 K(x) (k项k-1阶多项式)
- 选定一个多项式编码生成多项式G(x)为除数，G(x)为r阶
- 设待发送的k位的帧为M(x)，k>r,即发送帧M（x）比G（x）长
- 计算r位冗余位: $x^rM(x)/G(x)=Q(x)+R(x)$，其中，Q(x)为商，R(x)为余数

用二进制的模2运算进行2^r乘M的运算，相当于在M后面添加了r个0，得到（k+r）位被除数除以选定的长度为(r+1)位的除数G，得出商Q和余数R（r位，比G少1位）。

模2运算

- 在二进制运算中，加减法做模2运算，采用“加法不进位，减法不借位”的规则，即相当于异或操作，“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

- 在二进制运算中，模2乘和模2除运算举例如下算式。

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

$$
\begin{array}{r|l}
101 & 10000 \\
\cline{2-2}
101 & \\
\cline{1-1}
010 & \\
000 & \\
\cline{1-1}
100 & \\
101 & \\
\cline{1-1}
01 &
\end{array}
$$

CRC码的计算举例

- *如发送一帧信息位 M* = 101001（k=6）。
- 设 *r* = 3, 除数 P = 1101,
- 被除数是 2^rM = 101001000。
- 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^rM + *R*

即：101001001，共 (*k* + *r*) 位。

CRC码的计算举例

发送帧M = 101001

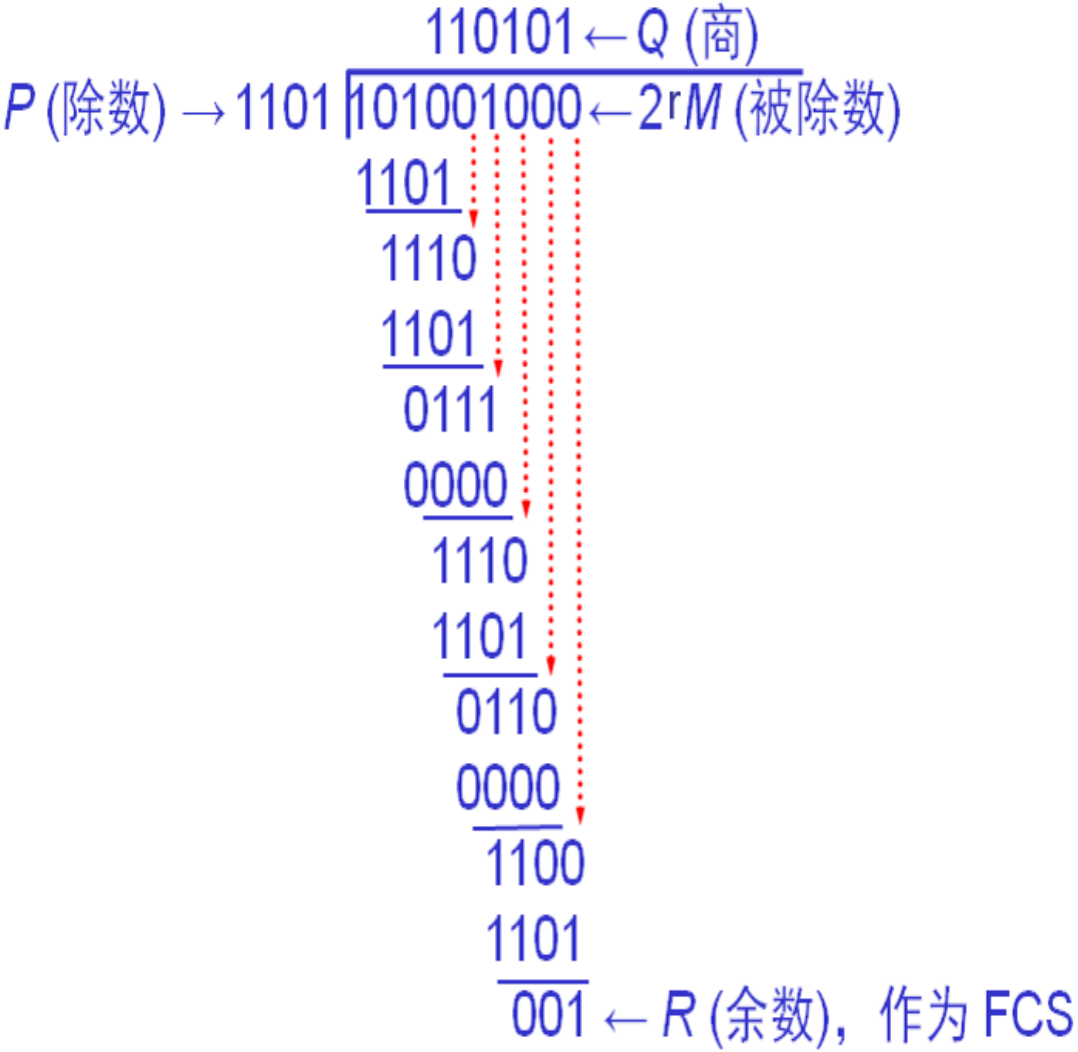
除数 P = 1101

被除数是 2^rM = 101001000

商 *Q* = 110101

余数 *R* = 001

编码后CRC码：101001001



CRC码的计算举例

因为： $x^rM(x)/G(x)=Q(x)+R(x)$

所以： $x^rM(x)+R(x)$一定能被G（x）整除，余数为0

编码后发送的CRC码：101001001

当码字到达接收方时：

- 若CRC码在接收端能被除数1101整除，即余数为0，则说明接收正确
- 若CRC码不能被除数整除，即余数不为0，则检测到出错。

广泛使用的生成多项式国际标准

- CRC-16： $G(x)=x^{16}+x^{15}+x^2+1$
- CRC-CCITT： $G(x)=x^{16}+x^{12}+x^5+1$
- CRC-32：

$$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

CRC校验码的检错能力

- CRC校验码能检查出全部单个错；
- CRC校验码能检查出全部离散的二位错；
- CRC校验码能检查出全部奇数个错；
- CRC校验码能检查出全部长度小于或等于K位的突发错；
- CRC校验码能以$[1-(1/2)^{K-1}]$的概率检查出长度为(K+1)位的突发错；
- 如果K=16，则该CRC校验码能全部检查出小于或等于16 位的所有的突发差错，并能以$1-(1/2)^{16-1}=99.997\%$的概率检查出长度为17位的突发错，漏检概率为0.003%；

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。

接收端对收到的每一帧进行 CRC 检验

- 若得出的余数 $R = 0$，则判定这个帧没有差错，就接受(accept)。
- 若余数 $R \neq 0$，则判定这个帧有差错，就丢弃。
- 但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。
- 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。

请注意：无差错接受与可靠传输的不同

- 仅用循环冗余检验 CRC 的差错检测技术只能做到“无差错接受”，即凡是接收端接受的帧（不包括丢弃的帧），都能以非常接近于 1 的概率认为这些帧在传输过程中没有产生差错，有差错的帧就丢弃而不接受）。
- “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”可能是比特差错，或是出现帧丢失、帧重复或帧失序。
- 在数据链路层使用CRC检验，只能够实现无比特差错的传输。
- 要做到“可靠传输”就必须再加上帧编号、确认和重传机制。

小结

- 目前数据链路层广泛使用循环冗余检验CRC的检错技术。
- 采用CRC校验码的系统，需要约定一个生成多项式（除数）。
- 发送方：信息位+冗余位=发送码字
- 接收方：用收到的CRC码字除以生成多项式，判定余数是否为零？
  - 为零：接收正确，接受数据帧
  - 不为零：接收有错，丢弃数据帧