

可靠信道编码

理论课程

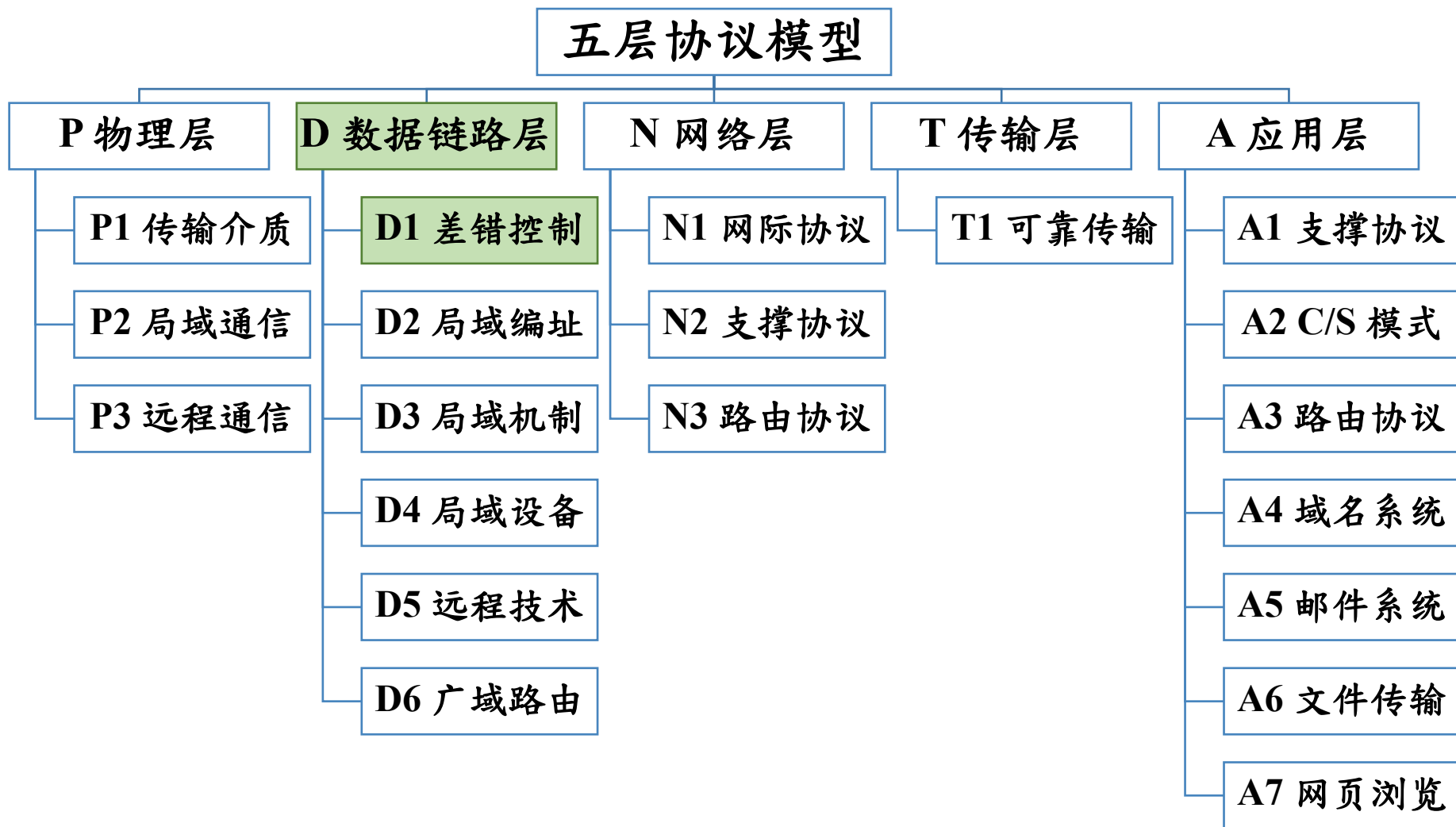


廈門大學
XIAMEN UNIVERSITY



信息学院 黄 焯
(特色化示范性软件学院) 博士, 副教授
School of Informatics Wei Huang

知识框架



主要内容

- 传输差错的原因
- 差错控制编码
 - 奇偶校验码、Internet校验和、CRC
 - 计算方法、作用与局限

对应课本章节

- **PART II Data Communication Basics**
 - **Chapter 8 Reliability And Channel Coding**

艾滋器官移植事故 台大和成大医院各罚15万

• Reactive vs. Negative



2011年8月，中国台湾《联合晚报》指出，一位邱姓男子23日不慎在家中坠楼、送到新竹南门医院被判定脑死。24日家属同意捐赠器官，成大医院也接获台大医院通知。24日傍晚台大器官移植小组进开刀房摘除器官，摘下的心肺肾等器官，火速移植给台大和成大医院5名患者，但25日，台大医院发现检验结果书面报告为阳性，以致误将艾滋感染者器官移植给病患。

报道指出，在中国台湾，现行艾滋病非属重大伤病加上保护感染者，因此不会特别注记在健保卡上，艾滋感染者若因艾滋病到医院就医时，需另外出示医疗卡。因此当意外坠楼的邱姓男子被送到医院急救、判定脑死时，从健保卡无从得知他是否有艾滋病。而邱姓男子家属没有联想到他是同性恋，属于艾滋病高危险群，错失第一道防线。

另外，目前医院没有向中国台湾卫生主管机关查询列管艾滋病患的程序，而是由医院自行检验。艾滋感染者是在新竹南门医院捐赠器官，血液检体送到台大医院检验，台大协调师出差到新竹，和台大医院检验部电话联系，不知是检验师口误，还是协调师听错，误将reactive（阳性）当成non-reactive（非阳性），发生严重错失。检验书面报告正确，只是移植团队在摘取器官进行移植时，没有依规再次检视，又错失最后防线。

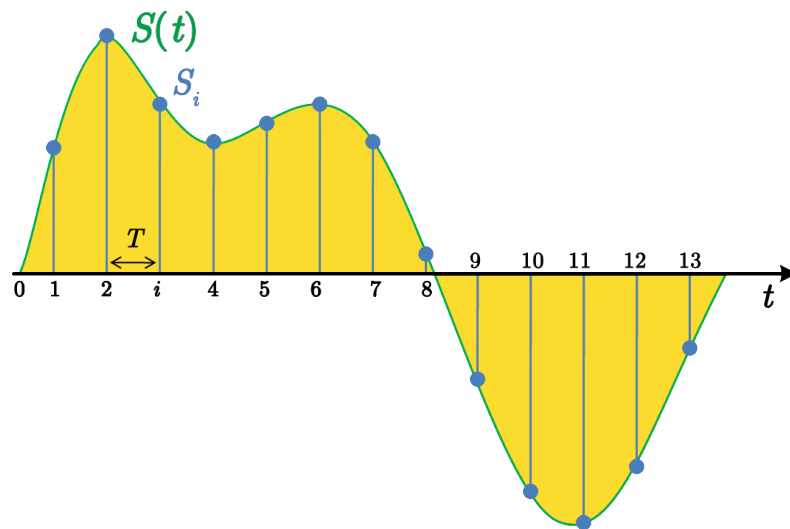
从现实社会到虚拟世界

- 现实世界：连续的

- 时间上的、数值上的，如：挥手视频，如：声波、光波
- 如果不能找到规律，则需要大量的存储能力

- 计算机世界：离散的

- 时间上：采样；数值上：编码
- 数字化
 - 1号时间、2号时间、.....
 - 1号音量、2号音量、.....



信息熵 (Entropy)

- 六个字符与六位数字的信息量

$$\begin{aligned} H(X) &= - \sum_{i=1}^n p(x_i) \log_b p(x_i) \\ &= - \sum_{i=1}^{256^6} \frac{1}{256^6} \log_2 \frac{1}{256^6} \\ &= -256^6 \frac{1}{256^6} \log_2 \frac{1}{256^6} \\ &= 6 \log_2 256 \\ &= 48\text{bit} \\ &= 6\text{byte} \end{aligned}$$

$$\begin{aligned} H(X) &= - \sum_{i=1}^n p(x_i) \log_b p(x_i) \\ &= - \sum_{i=1}^{10^6} \frac{1}{10^6} \log_2 \frac{1}{10^6} \\ &= -10^6 \frac{1}{10^6} \log_2 \frac{1}{10^6} \\ &= 6 \log_2 10 \\ &\approx 19.93\text{bit} \\ &\approx 2.49\text{byte} \end{aligned}$$

内容纲要

1	传输差错
2	简单的校验方法
3	循环冗余检验码

传输差错 (Transmission Errors)

- 网络研究的差错是自然产生的，不是恶意拼凑的
 - 原因：物理特性（电磁干扰会）；未达到工业标准
 - 现象：编码的二进制数据信号，0可能变成1或1变成0。
 - 小错比大错更难发现
- 差错原因分类
 - 干扰（Interference）：元器件电子辐射；宇宙背景辐射
 - 失真（Distortion）：长距离传输会受到干扰
 - 衰减（Attenuation）：通过介质的信号会变弱
- 差错类型：单个比特、突发差错、模糊差错

传输差错

- 避免差错的办法
 - 香农定理：增加信噪比（ signal-to-noise ratio ）
 - 屏蔽线等可以降低噪声，但无法完全消除
 - 错误检测（ Error detection ）增加开销（ overhead ）
 - 错误可以被检测（ detect ）到
 - 某些情况下错误的比特可以自动纠正
- 错误处理是一个权衡（ tradeoff ）

克服信道错误的两种方法

- 冗余可用于纠错
 - 压缩：去除无不确定性的部分（冗余）
 - 校验：利用冗余信息检测或纠正错误
- 信道编码是可用于克服传输错误的数学方法
 - 前向错误纠正（Forward Error Correction, FEC）
 - 在信息后附加用于纠错的码，接收方解码得到原始信息
 - 自动重传请求（Automatic Repeat reQuest, ARQ）
 - 每当消息到达，对方发回一个简短的确认（ACK）消息
 - 如果没有收到，则假设消息丢失，应重传副本
 - 接收方如果发现信息错误，则丢弃

前向错误纠正技术的两种编码

- 分块错误编码 (Block Error Codes)
 - 将数据分块(block)，将冗余(redundancy)加到每块信息中
 - 无记忆(memoryless)：编码依赖于给定块，和前面的块无关
- 卷积错误编码 (Convolutional Error Codes)
 - 数据被视为位序列，连续计算编码
 - 计算的码取决于当前输入和以前的一些位流
 - 卷积码是有记忆码 (codes with memory)

内容纲要

1	传输差错
2	简单的校验方法
3	循环冗余检验码

校验位和奇偶校验

- 奇偶校验 (parity check) 是一种机制
 - 发送方根据消息序列计算一个额外的位附加在原有消息序列后，称为奇偶位 (parity bit)
 - 接收方收到所有位后，校验并丢弃奇偶位。
- 两种机制：奇 (odd) 和偶 (even)
 - 奇校验中消息和校验位共有奇数个1；偶校验共有偶数个1
 - 发送和接收方应有同样的模式
- 只能检测部分错误
 - 奇数错：错的；偶数错：误认为是对的

单个奇偶校验位

- 奇偶校验代码

```
unsigned char val=0x5B;
unsigned char pcOdd=1;
printf("Bits: ");
for (size_t i = 0; i < 8; i++) {
    printf("%d ", val >> (7-i) & 1);
}
for (size_t i = 0; i < 8; i++) {
    pcOdd ^= val & 1;
    val >>= 1;
}
printf("\nParity bit: Even: %d; Odd: %d.\n", !pcOdd, pcOdd);
```

Bits: 0 1 0 1 1 0 1 1

Parity bit: Even: 1; Odd: 0.

校验位和奇偶校验

- 奇偶校验码是可检测错误的信道编码中的一种弱形式
 - 可以检测错误，但不能纠正错误
- 行列码

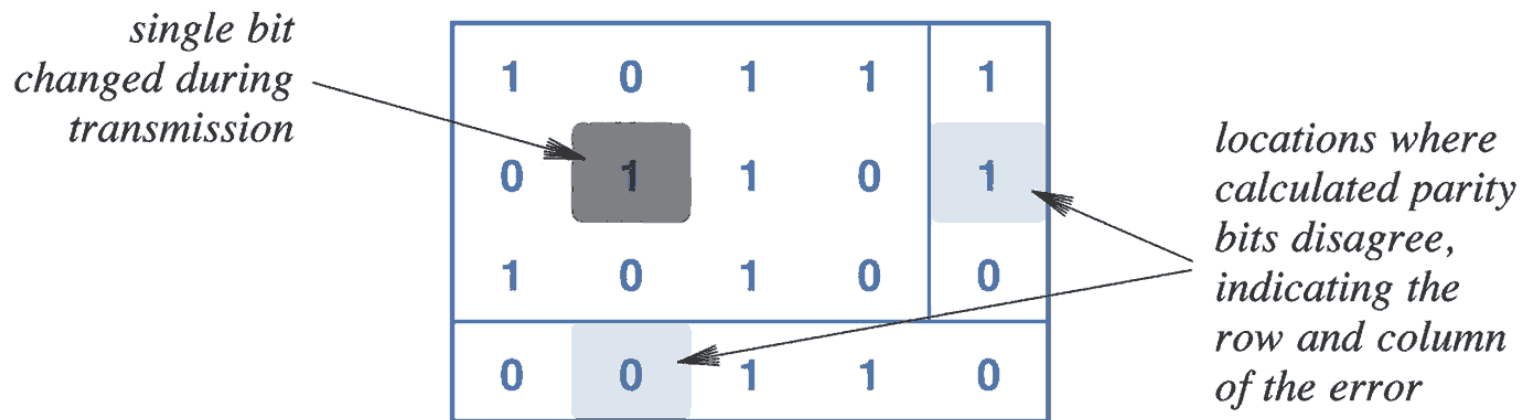


Figure 8.8 Illustration of how a single-bit error can be corrected using a row and column encoding.

块错误码和 (n, k) 符号

- 数据字 (Datawords) : 所有消息的集合
- 码字 (Codewords) : 所有正确编码的集合
- (n, k) 码
 - 数据字有 k 位, 校验码 r 位附加在数据字后形成码字
 - 度量的好坏是码率 (Code rate) $R=k/n$
- 成功的关键在于选择的是有效码字的可能组合的子集
 - 有效的子集被称为一个码本 (codebook)
 - 以奇偶校验码为例, 一半是码本

汉明距离：代码强度的量度

- 汉明距离 (Hamming distance)

- 两个字符串汉明距离是它们不同位的数量

Dataword	Codeword
0 0	0 0 1
0 1	0 1 0
1 0	1 0 0
1 1	1 1 1

(a)

$d(001, 010) = 2$	$d(010, 100) = 2$
$d(001, 100) = 2$	$d(010, 111) = 2$
$d(001, 111) = 2$	$d(100, 111) = 2$

(b)

- 码本的最小汉明距离 (minimum Hamming distance)

- 没有理想的信道编码：修改足够多位可转换至合法的码字
 - 码本中任何码字转换成另一个码字所需要改动位数的下限

Internet 校验和

- Internet校验和 (Internet checksum)
 - 数据字不需要固定大小
 - 不足16位补0
 - 计算校验和
 - 发送方在消息后添加校验和



Figure 8.9 The Internet checksum divides data into 16-bit units, appending zeroes if the data is not an exact multiple of 16 bits.

Internet 校验和

- 为了计算校验和，发送者把消息每16位作为一个整数计算总和，如果结果大于16位，则重复上述过程。
- 校验和尺寸小，额外传输开销小

H	e	l	l	o	□	w	o	r	l	d	.
48	65	6C	6C	6F	20	77	6F	72	6C	64	2E

$$4865 + 6C6C + 6F20 + 776F + 726C + 642E + \text{carry} = 71FC$$

$\underbrace{\hspace{15em}}$
2 71FA

\downarrow
2

\downarrow NOT
8E03

计算Internet Checksum

- 以下代码源于 RFC 1071 文档。

```
/* Compute Internet Checksum for "count" bytes beginning at
location "addr". */
register long sum = 0;
while (count > 1)
    sum += *(unsigned short *) addr++;  count -= 2;
if (count > 0) /* Add left-over byte, if any */
    sum += *(unsigned char *) addr;
while (sum >> 16) /* Fold 32-bit sum to 16 bits */
    sum = (sum & 0xffff) + (sum >> 16);
checksum = ~sum;
```

内容纲要

1	传输差错
2	简单的校验方法
3	循环冗余检验码

循环冗余检验码 (CRC)

- 循环冗余检验码 (Cyclic Redundancy Codes)

- 一种根据网络数据包或文件等数据产生简短固定位数校验码的散列 (Hash) 函数，主要用来检测或校验数据传输或者保存后可能出现的错误。

- 重要特征

- 任意消息字长

- 出色的错误检测

- 快速硬件实现 (位移寄存器，异或门)

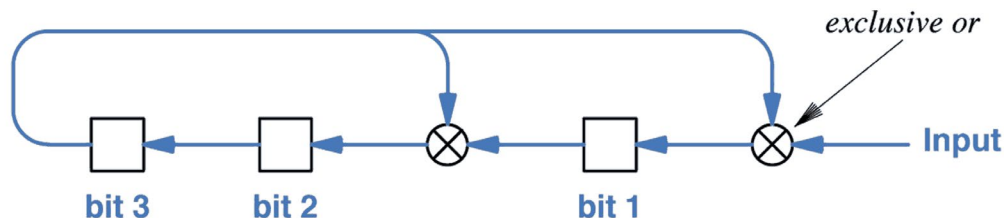


Figure 8.13 A hardware unit to compute a 3-bit CRC for $x^3 + x^1 + 1$.

循环冗余检验码 (CRC)

- 可以视为不带借位的二进制数除法
 - 每个位的二进制数作为一个多项式的系数
- 除数多项式称为生成多项式 (generator polynomial)
 - 一个理想的多项式是不可约的 (只能被自己和1整除)
 - 有一个以上的非零系数多项式可检测所有的单比特错误

除数1011可视为 $x^3 + x + 1$

被除数1010000可视为 $x^6 + x^4$

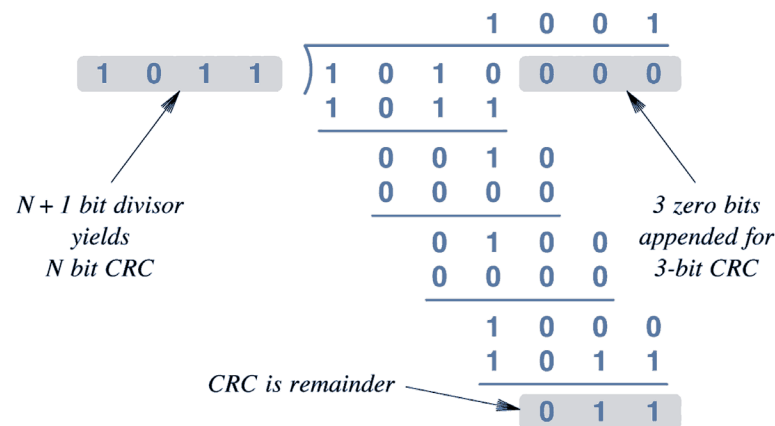


Figure 8.12 Illustration of a CRC computation viewed as the remainder of a binary division with no carries.

生成多项式的国际标准

• 生成多项式 $G(x)$ 的国际标准

标准	生成多项式
CRC-8	$x^8 + x^2 + x + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

帧格式和错误检测机制

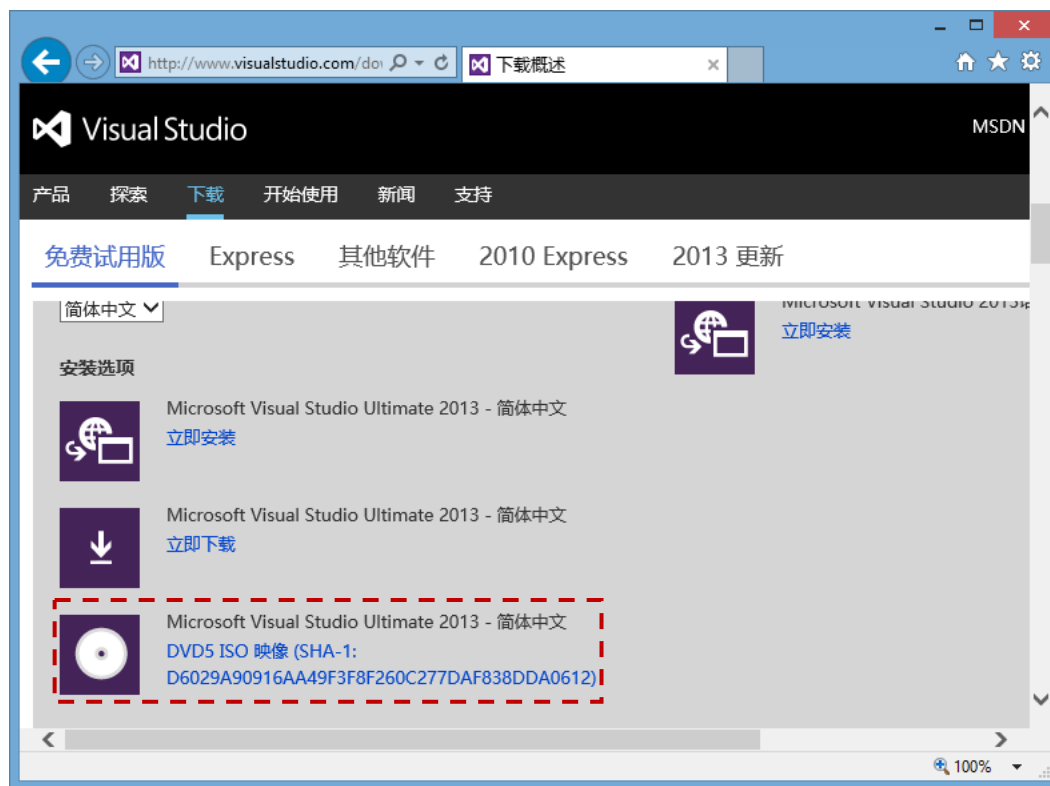
- 错误检测信息作为帧的一部分传输

帧首定界符	包含字节填充的数据块	帧尾定界符	CRC
-------	------------	-------	-----

- 发送方计算信息的校验，随着帧数据发送附加信息。
- 接收方用相同的方法计算校验值
 - 一般将校验值一起计算，而不是计算校验值再与发来的校验值比较。
- 一般构造复杂的发送方，简单的接收方。

更复杂的校验码

- 错误检测方案都不完美，因为传输错误可以影响更多数据。
- 信息安全用的校验和
 - MD5、SHA系列



谢谢观看



廈門大學
XIAMEN UNIVERSITY



信息学院 黄 焯
(特色化示范性软件学院) 博士, 副教授
School of Informatics Wei Huang