计算机网络

4. Reliability and Channel Coding

厦门大学软件学院

黄炜 助理教授

前言（不作要求）

信息论

Information Theory

信息论是运用概率论与数理统计的方法研究信息、信息熵、通信系统、数据传输、密码学、数据压缩等问题的应用数学学科。

信息论研究对象

位（bit）

传递的信息相当于抛多少次硬币

前言（不作要求）

信息熵（Entropy）

$$H\left(X\right)=-\sum_{i=1}^{n}p\left(x_i\right)\log_b p\left(x_i\right)$$

前言（不作要求）

破解密码：六位数字

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

$$= -\sum_{i=1}^{256^6} \frac{1}{256^6} \log_2 \frac{1}{256^6}$$

$$= -256^6 \frac{1}{256^6} \log_2 \frac{1}{256^6}$$

$$= 6 \log_2 256$$

$$= 48\text{bit}$$

$$= 6\text{byte}$$

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

$$= -\sum_{i=1}^{10^6} \frac{1}{10^6} \log_2 \frac{1}{10^6}$$

$$= -10^6 \frac{1}{10^6} \log_2 \frac{1}{10^6}$$

$$= 6 \log_2 10$$

$$\approx 19.93\text{bit}$$

$$\approx 2.49\text{byte}$$

前言（不作要求）

编码

Encoding

全部48bit=有效20bit+冗余28bit

冗余部分可以发挥余热

信息论

PART II Packet Transmission

Ch 8 Reliability and Channel Coding

可靠性与信道编码

分组避免了分组间错误的连坐

为什么会产生错误？

怎么发现错误？

艾滋器官移植事故 台大和成大医院各罚15万

Reactive vs. Negative

7.6 Transmission Errors 传输差错

- Lightning, power surges, and other electro-magnetic interference can introduce unwanted electrical currents in the electronic components or wires used for communication.
- Transmission errors (传输错误)
  - the problems of lost, changed, or spuriously appearing bits account for much of the complexity needed in computer networks.

突发差错

只要有电磁信号从一个点流向另一个点，它就可能受到不可预见的干扰，如热干扰、磁场干扰、或其他形式的电干扰。若信号是编码的二进制数据，则0可能变成1或1变成0。我们将差错划分为单个比特、多个比特或突发差错，取决于比特改变的数目和位置。在这三种差错中，单个比特差错是最常发生的，而突发差错是最少发生的（见图D.1）。

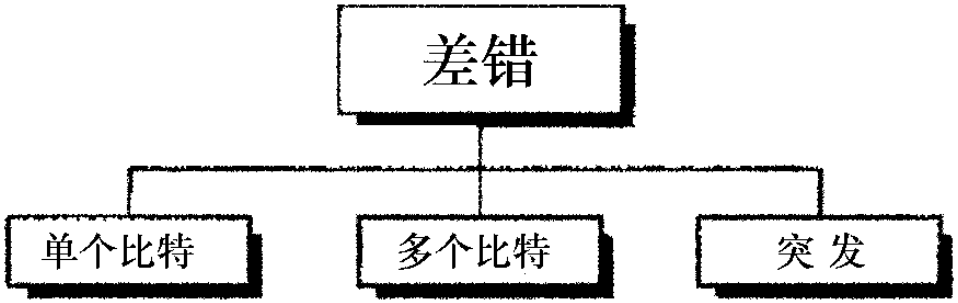


图 D.1 差错的类型

8.2 The Three Main Sources of Transmission Errors

- 所有通信系统都容易犯错
  - 原因：物理特性；未达到工业标准
  - 小错比大错更难发现
- **Interference (干扰)**
  - 元器件电子辐射；宇宙背景辐射
- **Distortion (失真)**
  - **All physical systems distort signals**
  - **As a pulse travels along an optical fiber, the pulse disperses**
  - **Wires have properties of capacitance and inductance**
- **Attenuation (衰减)**
  - **As a signal passes across a medium, signal becomes weaker**

Transmission Errors

- Shannon's Theorem
  - increase the signal-to-noise ratio：说得轻巧
    - Mechanisms like shielded wiring can help lower noise
    - Noise/interference cannot be eliminated completely
      - But many transmission errors can be detected
- Error detection adds overhead (开销)
  - In some cases, errors can be corrected automatically
- Error handling is a tradeoff (权衡)

啰嗦即能纠错

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

传输差错对数据的影响

Type Of Error	Description
Single Bit Error	A single bit in a block of bits is changed and all other bits in the block are unchanged (often results from very short-duration interference)
Burst Error	Multiple bits in a block of bits are changed (often results from longer-duration interference)
Erasure (Ambiguity)	The signal that arrives at a receiver is ambiguous (does not clearly correspond to either a logical 1 or a logical 0 (can result from distortion or interference)

Figure 8.1 The three types of data errors in a data communications system.

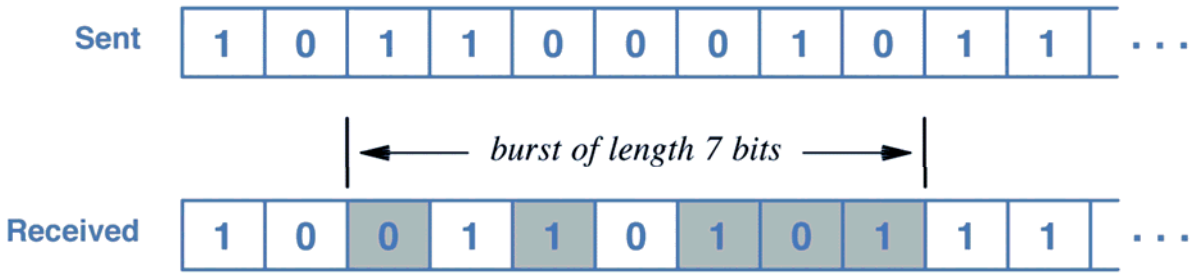


Figure 8.2 Illustration of a burst error with changed bits marked in gray.

8.4 Two Strategies for Handling Channel Errors

- Mathematical techniques have been developed
  - Overcome errors during transmission & increase reliability
  - Known collectively as channel coding (信道编码)
- divided into two broad categories:
  - Forward Error Correction (前向错误纠正, FEC) mechanisms
  - Automatic Repeat reQuest (自动重传请求, ARQ) mechanisms

8.4 Two Strategies for Handling Channel Errors

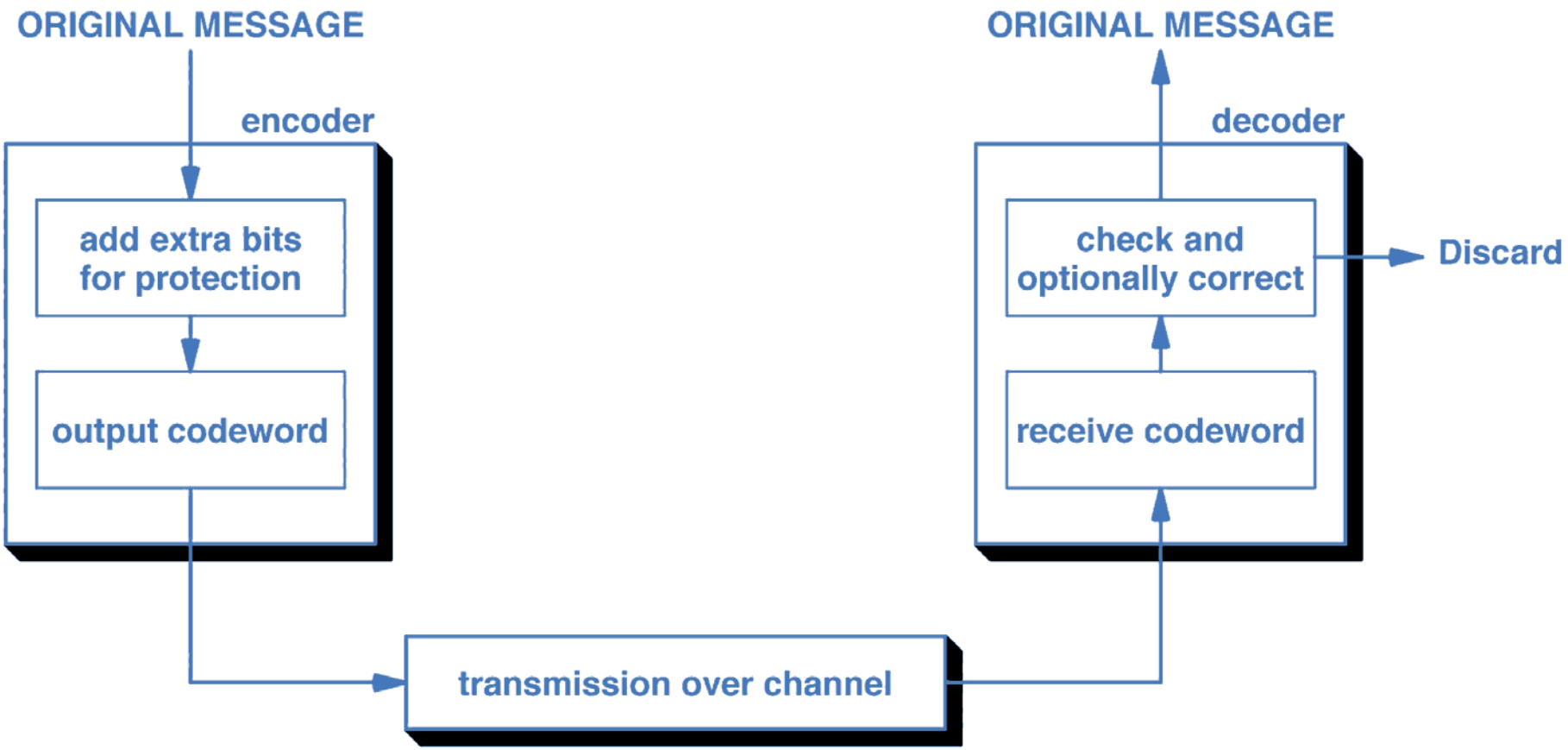


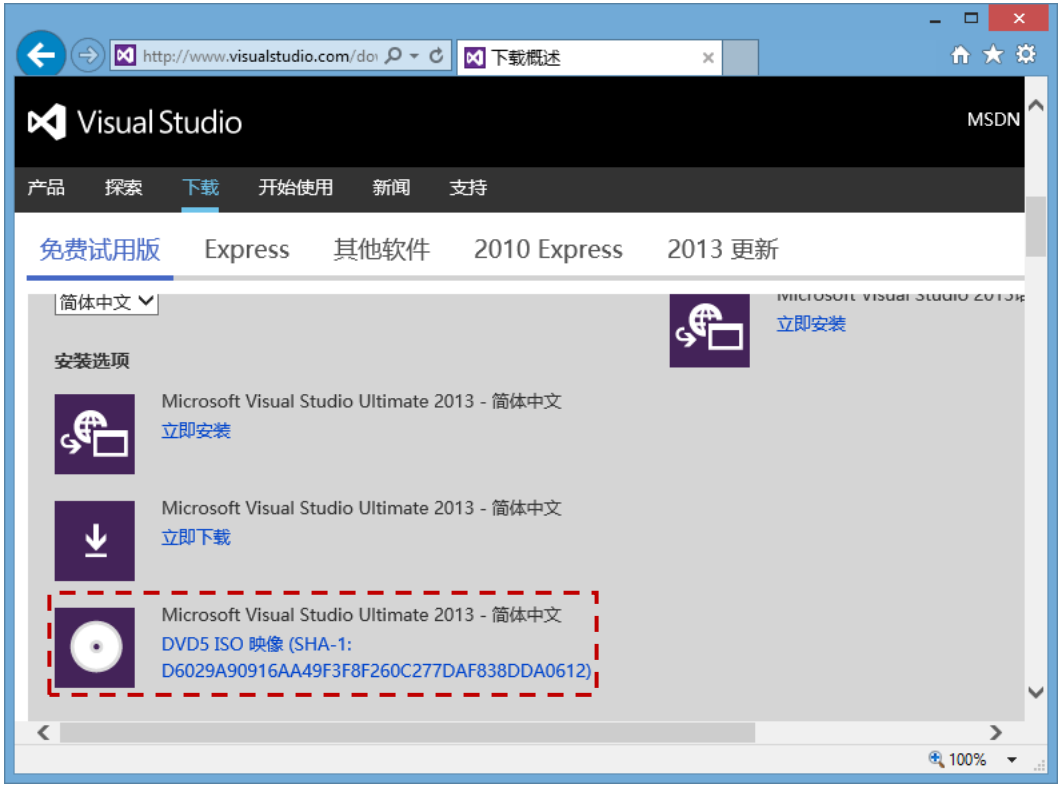
Figure 8.3 The conceptual organization of a forward error correction mechanism.

8.5 Block and Convolutional Error Codes

- **Two types of FEC techniques satisfy separate needs:**
  - **Block Error Codes (分块错误代码)**
    - 将数据分成块(block)，将冗余(redundancy)附加到每一块信息中
    - 无记忆(memoryless)：发送编码只依赖于给定块，而非之前块
  - **Convolutional Error Codes (卷积错误编码)**
    - 数据视为一系列的位，并通过一个连续的系列计算编码
    - 计算的码（一组比特）取决于当前输入和以前的一些比特流中
    - Convolutional codes (卷积码) are said to be codes with memory (记忆码)

思考题

- 日常中的校验码
  - CRC32、MD5、SHA1
- 缺点
  - 不够轻便



7.7 Parity Bits and Parity Checking

- Known as a parity check (奇偶校验), the mechanism requires the sender to compute an additional bit, called a parity bit (奇偶位), and to attach (附加) it to each character before sending.
- After all bits of a character arrive, the receiver removes the parity bit.
- There are two forms of parity: even (偶) and odd (奇).

8.6 An Example Block Error Code: Single Parity Checking

- How can additional info. be used to detect errors?
  - consider a single parity checking (SPC) mechanism
- One form of SPC defines a block to be an 8-bit unit of data (i.e., a single byte)
  - On the sending side, an encoder adds an extra bit, called a parity bit, to each byte before transmission
  - A receiver uses parity bit to check whether bits in the byte are correct
  - Before parity can be used, the sender and receiver must be configured for either even parity or odd parity

8.6 Single Parity Checking

- 多说无益，看代码吧

```
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.
```

8.6 Single Parity Checking

- A weak form of channel coding that can detect errors
  - 检测和纠正有区别：but cannot correct errors
- An even parity mechanism can only handle errors where an odd number of bits are changed
  - 未出错：对的
  - 奇数个出错的比特：认为是错的
  - 偶数个出错的比特：误认为是对的

Datawords (数据字)

the set of all possible messages

Codewords (码字)

the set of all possible encoded versions

8.7 The Mathematics of Block Error Codes and (*n*, *k*) Notation

- **Mathematically**
  - **Datawords: the set of all possible messages**
  - **Codewords: the set of all possible encoded versions**
- **(n, k) encoding scheme**
  - **a dataword contains k bits and r additional bits are added to form a codeword**
    - **where $n = k + r$**
- **The key to successful error detection lies in choosing a subset of the 2^n possible combinations that are valid codewords**
  - **The valid subset is known as a codebook**

As an example: SPC

- The set of datawords consists of any possible combination of 8-bits
  - Thus, $k = 8$ and there are 2^8 or 256 possible data words
  - The set of $n = 9$ bits, so there are 2^9 or 512 possibilities
    - However, only half of the 512 values form valid codewords
- Think of the set of all possible n-bit values and the valid subset that forms the codebook
  - If an error occurs during transmission
    - one or more of the bits in a codeword will be changed, which will either produce another valid codeword or an invalid combination
  - An ideal channel coding scheme is one where any change to bits in a valid codeword produces an invalid combination.

Hamming距离

两个码字之间的差异

8.8 Hamming Distance: A Measure of a Code's Strength

- **No channel coding scheme is ideal!**
  - 修改足够的比特可以转换至一个合法的码字
- 一个码字转换成另一个码字的最小比特数
- **Given two strings of n bits each, the Hamming distance is defined as the number of differences**

d(000,001) = 1	d(000,101) = 2
d(101,100) = 1	d(001,010) = 2
d(110,001) = 3	d(111,000) = 3

Figure 8.5 Examples of Hamming distance for various pairs of 3-bit strings.

Compute Hamming distance

- One way to compute the Hamming distance consists of
  - taking the exclusive or (xor) between two strings
  - and counting the number of 1 bits in the answer
- E.g. consider the Hamming distance between strings 110 and 011
  - The xor of the two strings is: (contains two 1 bits)

$$110 \oplus 011 = 101$$

  - Therefore, the Hamming distance between 011 and 101 is 2

8.9 The Hamming Distance Among Strings in a Codebook

- **Errors can transform a valid codeword into another valid codeword**
  - **To measure such transformations, we compute the Hamming distance between all pairs of codewords in a given codebook**
- **Consider odd parity applied to 2-bit datawords**
  - **Figure 8.6 lists the four (4) possible datawords**

8.9 The Hamming Distance Among Strings in a Codebook

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)

Figure 8.6 (a) The datawords and codewords for a single parity encoding of 2-bit data strings, and (b) the Hamming distance for all pairs of codewords.

8.9 The Hamming Distance Among Strings in a Codebook

- We use to denote the **minimum Hamming distance**, $d_{\min}$ among pairs in a codebook
  - How many bit errors can cause a transformation from one valid codeword into another valid codeword?
- In the SPC example of Fig. 8.6, the set consists of the Hamming dist. between each pair of codewords $d_{\min}=2$
  - The definition means that there is at least one valid codeword that can be transformed into another valid codeword, if **2-bit** errors occur during transmission

8.10 The Tradeoff Between Error Detection and Overhead

- **A large value of $d_{\min}$ is desirable**
  - **because the code is immune to more bit errors, if fewer than $d_{\min}$ bits are changed, the code can detect $e = d_{\min} - 1$ errors**
    - **the relationship between $d_{\min}$ and e, the maximum number of bit errors that can be detected:**
- **A code with a higher value of $d_{\min}$ sends more redundant (冗余) information than an error code with a lower value of $d_{\min}$**
- **Code rate (码率) that gives the ratio of a dataword size to the codeword size as shown in Eq. 8.2**

$$R = \frac{k}{n}$$

8.11 Error Correction with Row and Column (RAC) Parity

- **Imagine an array of 3-rows and 4-columns, with a parity bit added for each row and for each column**
  - **Figure 8.7 illustrates the arrangement, which is known as a Row and Column (RAC) code**
    - **Example RAC has n= 20, which means that it is a (20, 12) code**

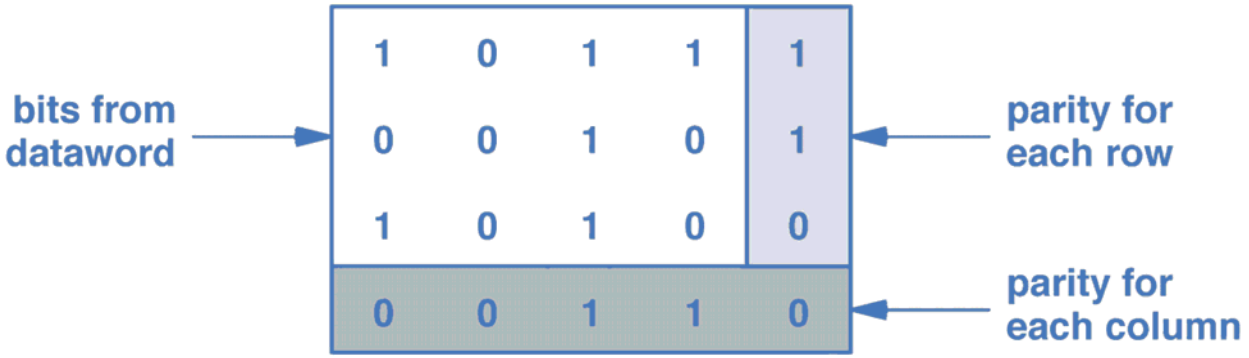


Figure 8.7 An example of row and column encoding with data bits arranged in a 3 × 4 array and an even parity bit added for each row and each column.

8.11 Error Correction with RAC Parity

- **How error correction works? Assume that one of the bits in Fig. 8.7 is changed during transmission:**
  - **When the receiver arranges the bits into an array and parity bits are recalculated**
    - **two of the calculations will disagree with the parity bits received**
    - **a single bit error will cause two calculated parity bits to disagree with the parity bit received**

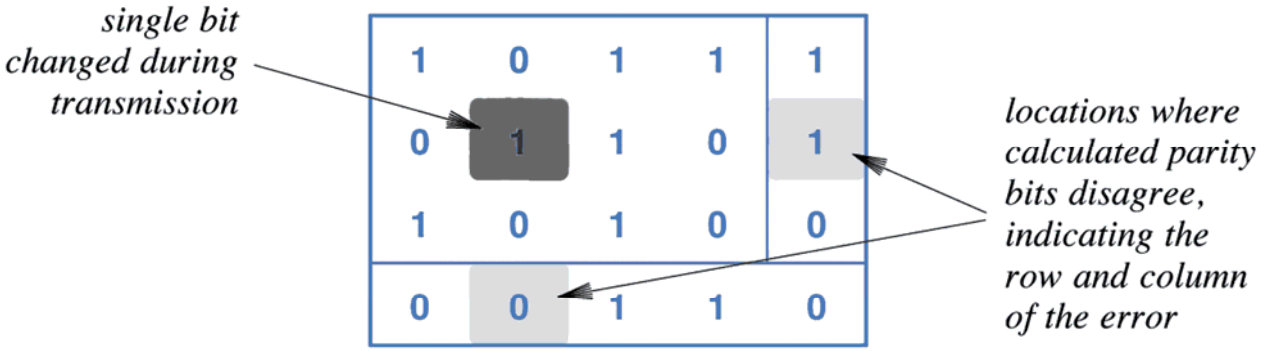


Figure 8.8 Illustration of how a single-bit error can be corrected using a row and column encoding.

8.11 Error Correction with RAC Parity

- The two disagreements correspond to the row and column position of the error
- A receiver uses the calculated parity bits to determine exactly which bit is in error, and then corrects the bit
  - an RAC can correct any error that changes a single data bit
- What happens to an RAC code if an error changes more than one bit in a given block?
- RAC can only correct single-bit errors
- In cases where two or three bits are changed
  - an RAC encoding is able to detect an odd number of errors

Internet Checksum

8.12 The 16-Bit Checksum Used in the Internet

- **Internet checksum is a particular coding scheme plays a key role in the Internet**
  - **the code consists of a 16-bit 1s complement checksum**
  - **does not impose a fixed size on a dataword**
    - **the algorithm allows a message to be arbitrarily long**
    - **and computes a checksum over the entire message**
- **The Internet checksum treats data in a message as a series of 16-bit integers, as Fig. 8.9 below illustrates**

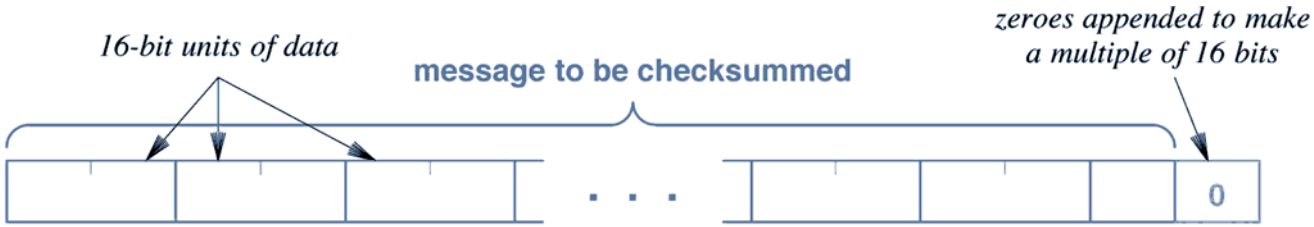


Figure 8.9 The Internet checksum divides data into 16-bit units, appending zeroes if the data is not an exact multiple of 16 bits.

8.12 The 16-Bit Checksum Used in the Internet

Given:

A message, *M*, of arbitrary length

Compute:

A 16-bit 1s complement checksum, *C*, using 32-bit arithmetic

Method:

Pad *M* with zero bits to make an exact multiple of 16 bits
Set a 32-bit checksum integer, *C*, to 0;
for (each 16-bit group in *M*) {
    Treat the 16 bits as an integer and add to *C*;
}
Extract the high-order 16 bits of *C* and add them to *C*;
The inverse of the low-order 16 bits of *C* is the checksum;
If the checksum is zero, substitute the all 1s form of zero.

8.12 The 16-Bit Checksum Used in the Internet

- To compute a checksum, a sender
  - adds the numeric values of the 16-bit integers
  - and it transmits the result
- To validate (验证) the message, a receiver performs the same computation
  - Why is a checksum computed as the arithmetic inverse (运算的逆) of the sum instead of the sum?

勘误

此处中文课本有误

把校验和与和值相加的结果将为 0

“全1”表示0

计算Internet Checksum的疑问

- 最可靠的资料应属 RFC 1071 文档。
  - 取反后不需要加 1

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

7.9 Detecting Errors with Checksums

- Many computer network systems send a checksum (校验和) along with each packet to help the receiver detect errors.
- To compute a checksum, the sender treats each pair of characters as a 16-bits integer and computes the sum. If the sum grows larger than 16 bits, the carry bits (进位) are added into the final sum.

7.9 Detecting Errors With Checksums

- **The small size of the checksum means the cost of transmitting the checksum is usually much smaller than the cost of transmitting the data.**

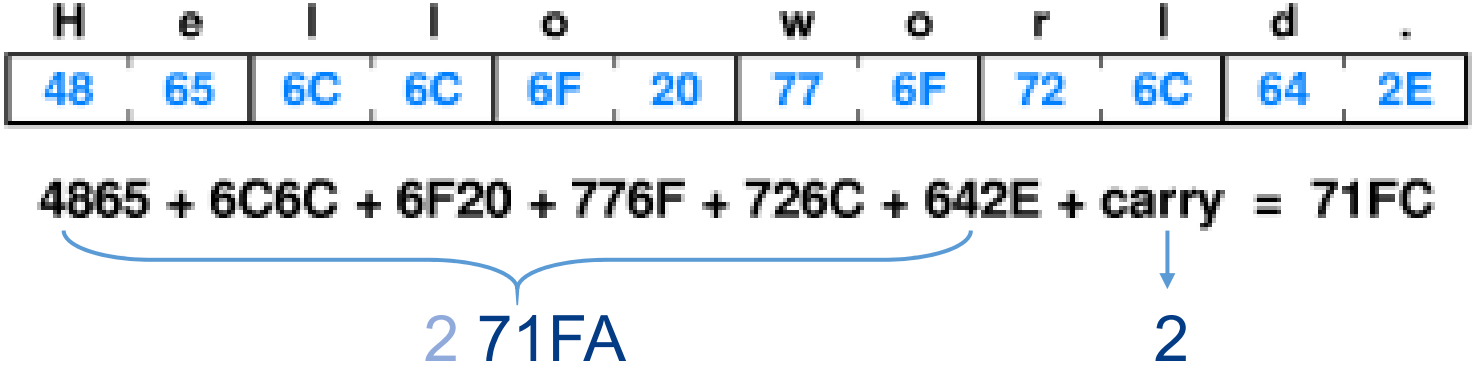


Figure 7.6 An example 16-bit checksum computation for a string of 12 ASCII characters. Characters are grouped into 16-bit quantities, added together using 16-bit arithmetic, and the carry bits are added to the result.

Disadvantage of Checksums

- **Checksums (校验) have the disadvantage of not detecting all common errors.**

Data Item In Binary	Checksum Value	Data Item In Binary	Checksum Value
0001	1	0011	3
0010	2	0000	0
0011	3	0001	1
0001	1	0011	3
totals	7		7

Figure 7.7 Illustration of how a checksum can fail to detect transmission errors. Reversing the value of the second bit in each data item produces the same checksum.

Cyclic Redundancy Codes

(循环冗余检验码，CRC)

8.13 Cyclic Redundancy Codes (CRC)

- 由W. Wesley Peterson于1961年发表
- 一种根据网络数据包或电脑文件等数据产生简短固定位数校验码的一种Hash (散列)函數，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。
- Key properties of CRC are summarized below
  - Arbitrary Length Message (任意字长)
  - Excellent Error Detection (出色的错误检测)
  - Fast Hardware Implementation (快速硬件实现)

8.13 Cyclic Redundancy Codes (CRC)

- **Cyclic is derived from a property of the codewords:**
  - **A circular shift (移位) of the bits of any codeword produces another one**

Dataword	Codeword
0000	0000 000
0001	0001 011
0010	0010 110
0011	0011 101
0100	0100 111
0101	0101 100
0110	0110 001
0111	0111 010

Dataword	Codeword
1000	1000 101
1001	1001 110
1010	1010 011
1011	1011 000
1100	1100 010
1101	1101 001
1110	1110 100
1111	1111 111

Figure 8.11 An example (7,4) cyclic redundancy code.

8.13 Cyclic Redundancy Codes (CRC)

- **Consider the division of binary numbers under the assumption of no carries**

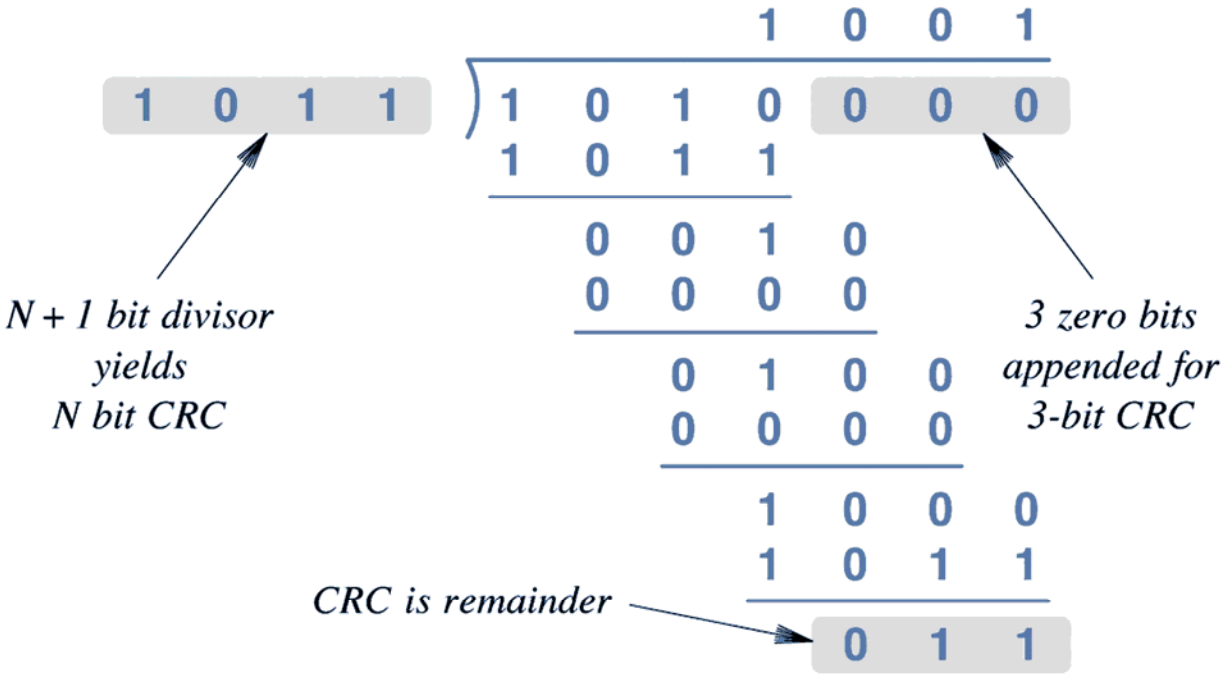


Figure 8.12 Illustration of a CRC computation viewed as the remainder of a binary division with no carries.

8.13 Cyclic Redundancy Codes (CRC)

- **How can mathematicians view the above process as a polynomial division?**
  - **think of each bit in a binary number as the coefficient of a term in a polynomial**
- **For example, we can think of the divisor in Figure 8.12, 1011, as coefficients in the following polynomial:**

$$1 \times x^3 + 0 \times x^2 + 1 \times x^1 + 1 \times x^0 = x^3 + x + 1$$

  - **Similarly, the dividend in Figure 8.12, 1010000, represents the polynomial:** $x^6 + x^4$

8.13 Cyclic Redundancy Codes (CRC)

- Term generator polynomial (生成多项式) to describe a polynomial that corresponds to a divisor
  - The selection of a generator polynomial is key to creating a CRC with good error detection properties
- An ideal polynomial is irreducible (i.e., can only be divided evenly by itself and 1)
- A polynomial with more than one non-zero coefficient can detect all single-bit errors

8.14 An Efficient Hardware Implementation of CRC

- CRC hardware is arranged as a shift register with exclusive or (xor) gates between some of the bits
- Figure 8.13 illustrates the hardware needed for the 3-bit CRC computation from Figure 8.12

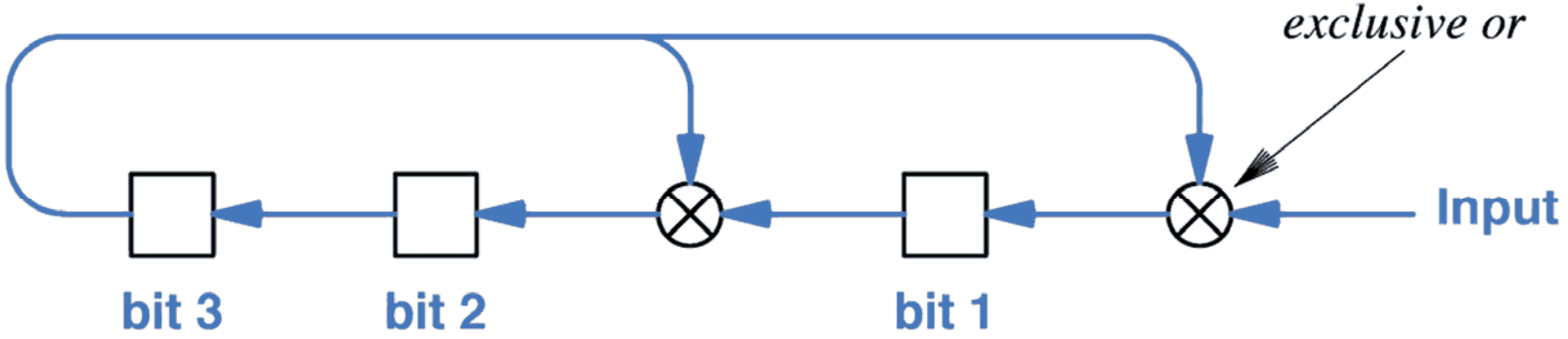


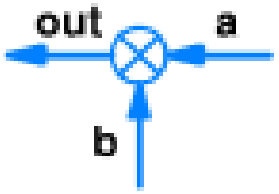
Figure 8.13 A hardware unit to compute a 3-bit CRC for $x^3 + x^1 + 1$.

生成多项式 G(x) 的国际标准

- 生成多项式 G(x) 的国际标准
  - **CRC-8** $x^8 + x^2 + x + 1$
  - **CRC-10** $x^{10} + x^9 + x^5 + x^4 + x^2 + 1$
  - **CRC-12** $x^{12} + x^{11} + x^3 + x^2 + x + 1$
  - **CRC-16** $x^{16} + x^{15} + x^2 + 1$
  - **CRC-CCITT** $x^{16} + x^{12} + x^5 + 1$
  - **CRC-32**

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

Hardware Implementation of CRC



a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

(a) *(b)*

Figure 7.8 (a) A diagram of hardware that computes an *exclusive or*, and (b) the output value for each of the four combinations of input values. Such hardware units are used to calculate a CRC.

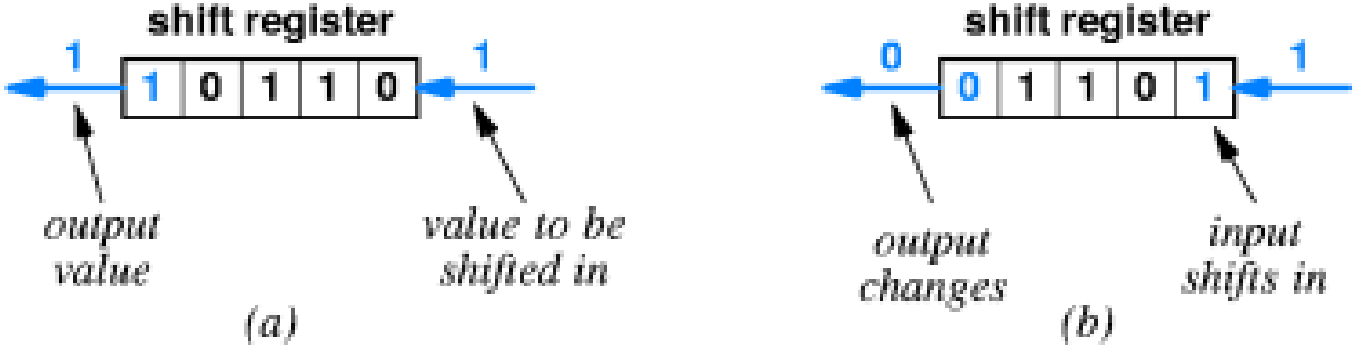


Figure 7.9 A shift register (a) before and (b) after a shift operation. During a shift, each bit moves left one position, and the output becomes equal to the leftmost bit.

7.11 Combining Building Blocks

- **Three shift registers and three exclusive or units can be combined to compute a 16_bit CRC.**

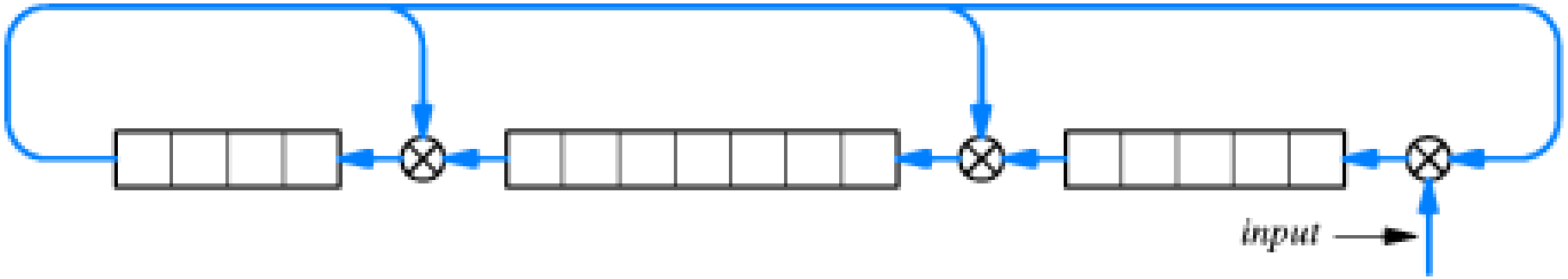


Figure 7.10 A diagram of the hardware used to compute a CRC. After bits of a message have been shifted into the unit, the shift registers contain the *16*-bit CRC for the message.

7.12 Burst Errors突发错误

- Two categories of common errors make CRCs especially useful.
  - First, hardware failures sometimes cause a specific set of bits to be damaged. E.g., vertical errors (垂直错误)
  - Second, CRCs are especially useful for detecting errors that involve changes to a small set of bits near a single location. Such errors are called burst errors (突发错误).

7.13 Frame Format and Error Detection Mechanisms

- Network usually associate error detection information with each frame. The sender calculates information such as a checksum or CRC, and transmits the additional information along with the data in the frame.
- The receiver calculates the same value and compares it to the additional information that arrives in the frame.

For example:

- **No error detection scheme is perfect because transmission errors can affect the additional information as well as the data.**

Figure 7.11 A modification of the frame format from Figure 7.3 that includes a *16*-bit CRC.

Automatic Repeat reQuest

自动重传请求 (ARQ)

8.15 Automatic Repeat reQuest (ARQ) Mechanisms

- Whenever one side sends a message to another, the other side sends a short acknowledgement (ACK) message back
  - For example, if A sends a message to B, B sends an ACK back to A
  - Once it receives an ACK, A knows that the message arrived correctly
  - If no ACK is received after T time units, A assumes the message was lost and retransmits a copy

8.15 ARQ Mechanisms

- ARQ is especially useful in cases of dealing with detecting errors
  - but not in cases for error correction
  - many computer networks use a CRC to detect transmission errors
- An ARQ scheme can be added to guarantee delivery if a transmission error occurs
  - the receiver discards the message if an error occurs
  - and the sender retransmits another copy

4. THANK YOU.

厦门大学软件学院

黄炜 助理教授