Chapter3 Data Link Layer(1)

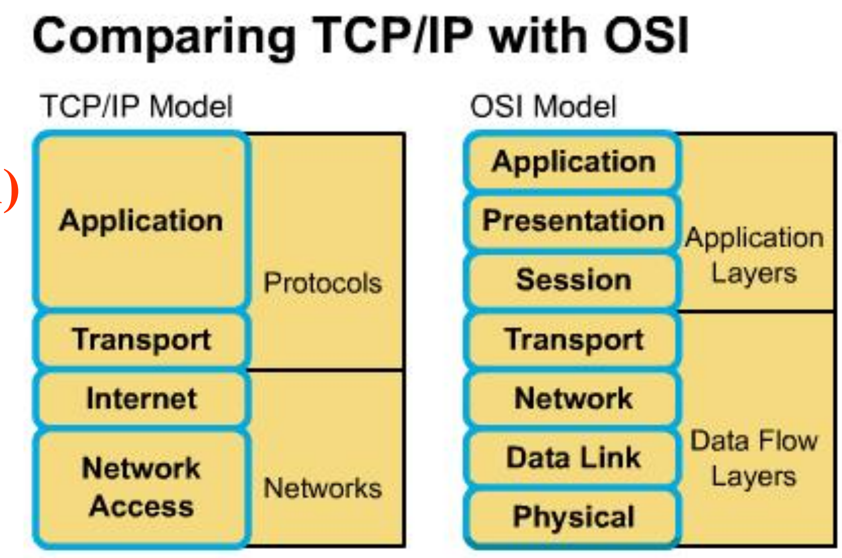
王昊翔：hxwang@scut.edu.cn

School of Computer Science & Engineering ,SCUT

Communication & Computer Network key-Lab of GD

Contents presented in Chap1&2

- Chapter 1
  - Services vs. Protocols
  - Reference Models(Encapsulation)
  - Example Networks
  - Networks Standardization
- Chapter 2
  - Theoretical Basis
  - Three transmission media
    - Guided transmission
    - Wireless transmission
    - Communication satellites
  - Three communication system
    - Public Switched Telephone Network



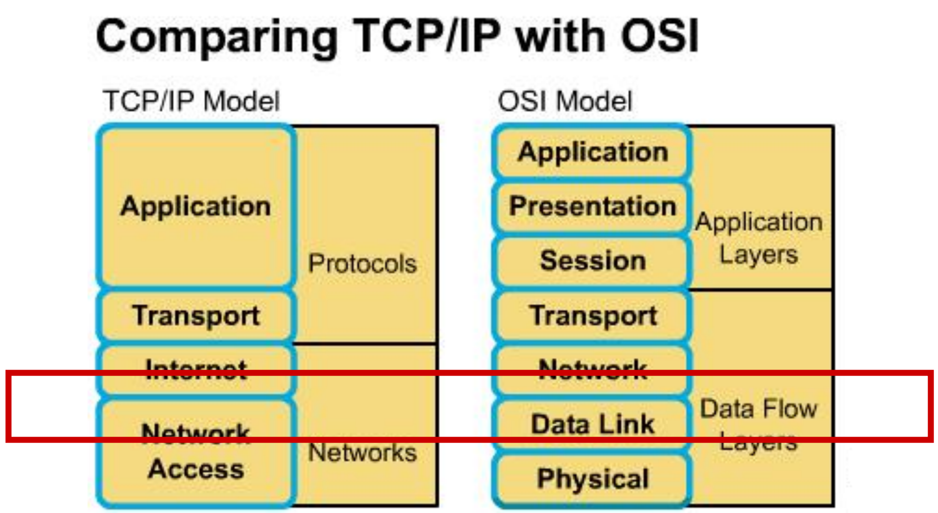
5	Application layer
4	Transport layer
3	Network layer
2	Data link layer
1	Physical layer

Main object of the chapter3

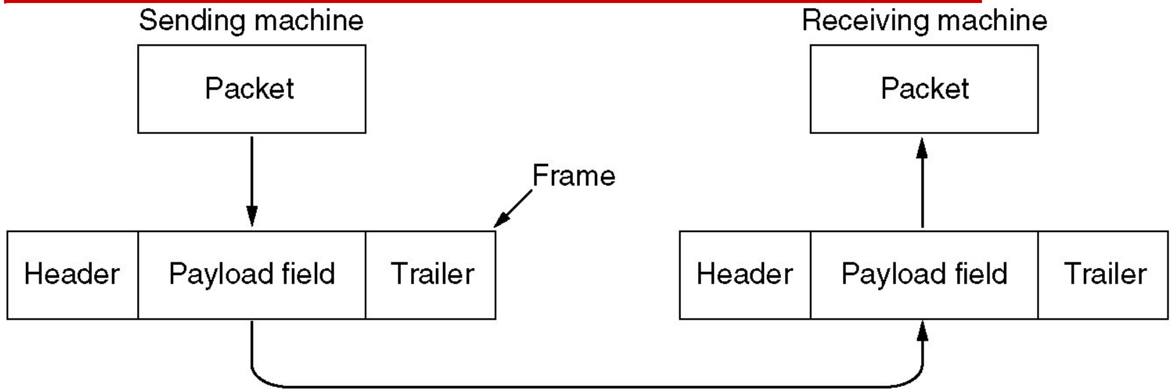
- The DLL is responsible for taking the packets of information that it receives from the Network Layer and putting them into frames for transmission.
- Each frame holds the payload plus a header and a trailer (overhead).
- It is the frames that are transmitted over the physical layer.
- Achieving reliable, efficient communication between two adjacent machines.

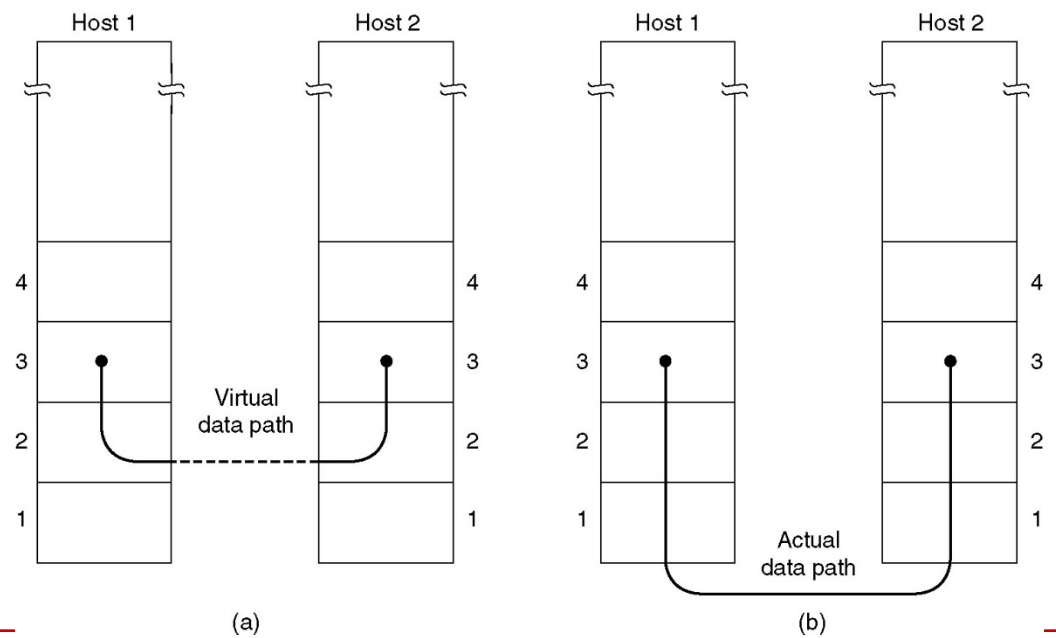
Main Functions of DLL

- Provide a well-defined service interface to the network layer.
- Deal with transmission errors.
- Regulate the flow of data , so that slow receivers are not swamped.



Relationship between packets and frames





Outline

- Data Link Layer Design Issues
- Error Detection and Correction
- Elementary Data Link Protocols
- Sliding Window Protocols
- Protocol Verification
- Example Data Link Protocols

Contents of this lecture

- Overview of data link layer
- Learn Framing methods
- Learn error-detection and error-control
  - Hamming code（海明码）
  - Cyclic Redundancy Check（循环冗余码CRC）

Service provided by DLL

- The data link layer can offer many kinds of service.
- The actual offered services can vary from system to system.
- Three common services:
  - Unacknowledged connectionless service.
    - 无确认的无连接服务
  - Acknowledged connectionless service.
    - 有确认的无连接服务
  - Acknowledged connection-oriented service.
    - 有确认的面向连接服务

Unacknowledged connectionless service

- The source machine send independent frames to the destination machine, and there is no acknowledgement from the destination machine.
- No logical connection is established beforehand or released afterward.
- The DLL will make no attempt to detect the loss of or recover a lost frame.
- This service is useful for low error rate networks and for real-time traffic where late data is worse than no data.

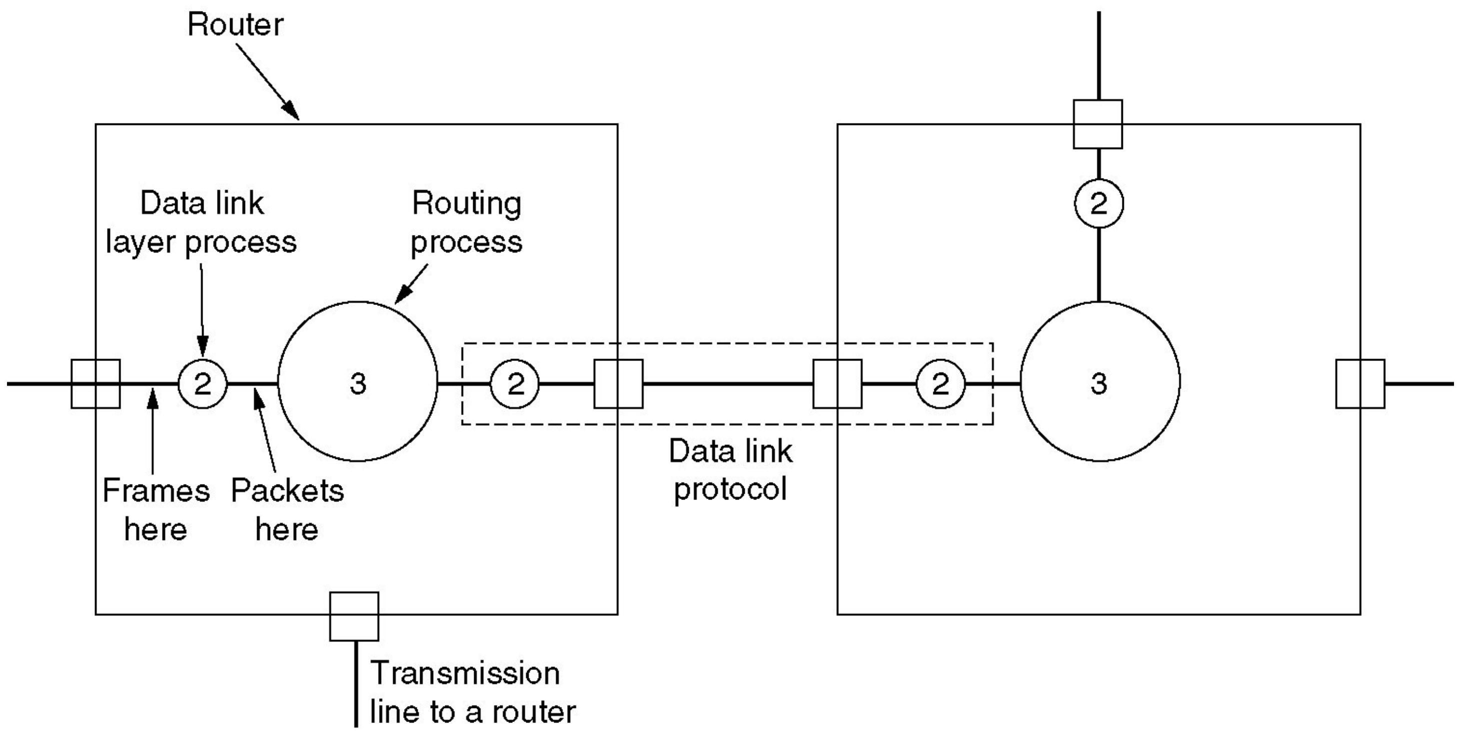
Acknowledged Connectionless Service

- The receiver acknowledges the arrival of each frame.
  - If it hasn’t arrived correctly (or within a specified time interval), it can be resent.
- This is a useful service when the connection is unreliable (such as wireless systems)
- There is no requirement for such an acknowledgement service to be implemented by the data link layer.

Acknowledged Connection-Oriented Service

- A connection is established between the two machines, and the frames are then transmitted.
- Each frame sent over the connection is numbered and each frame is acknowledged.
- The frames are guaranteed to arrive only once and in order.
- The connection is released once the communication is complete.
- This is the same as a “reliable” bit stream.

Data Flow Over Two Routers



Framing

- The data link layer must use the service provided by the physical layer in order to provide service to the network layer.
- The Physical Layer is only able to put a raw bit stream on the transmission media.
- Bit stream is not guaranteed to be error free.
- It is up to the data link layer to detect and, if necessary, correct errors.

Framing(cont'd)

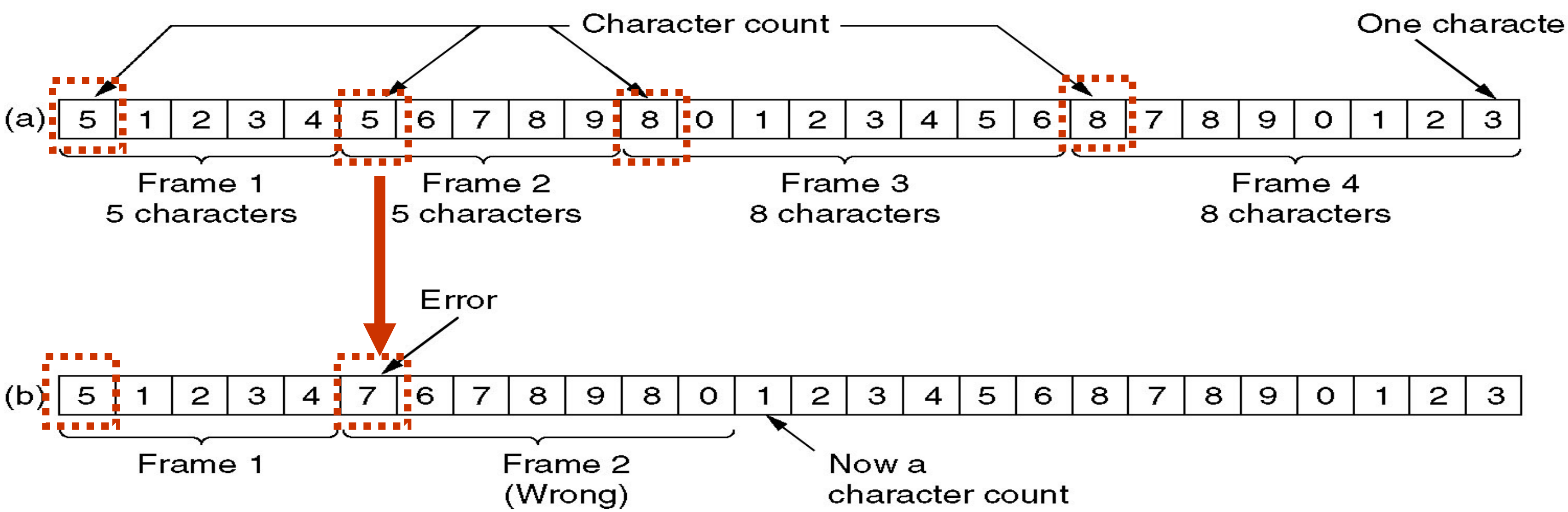
- The DDL can be able to break up the bit stream into discrete frames.
- Compute the checksum for each frame and the checksum is recomputed when a frame arrives at the destination.
- Breaking the bit stream up into frames is somewhat difficult.
  - Time gaps
- We need to look at other methods of denoting the start and finish of a frame.

Four framing method

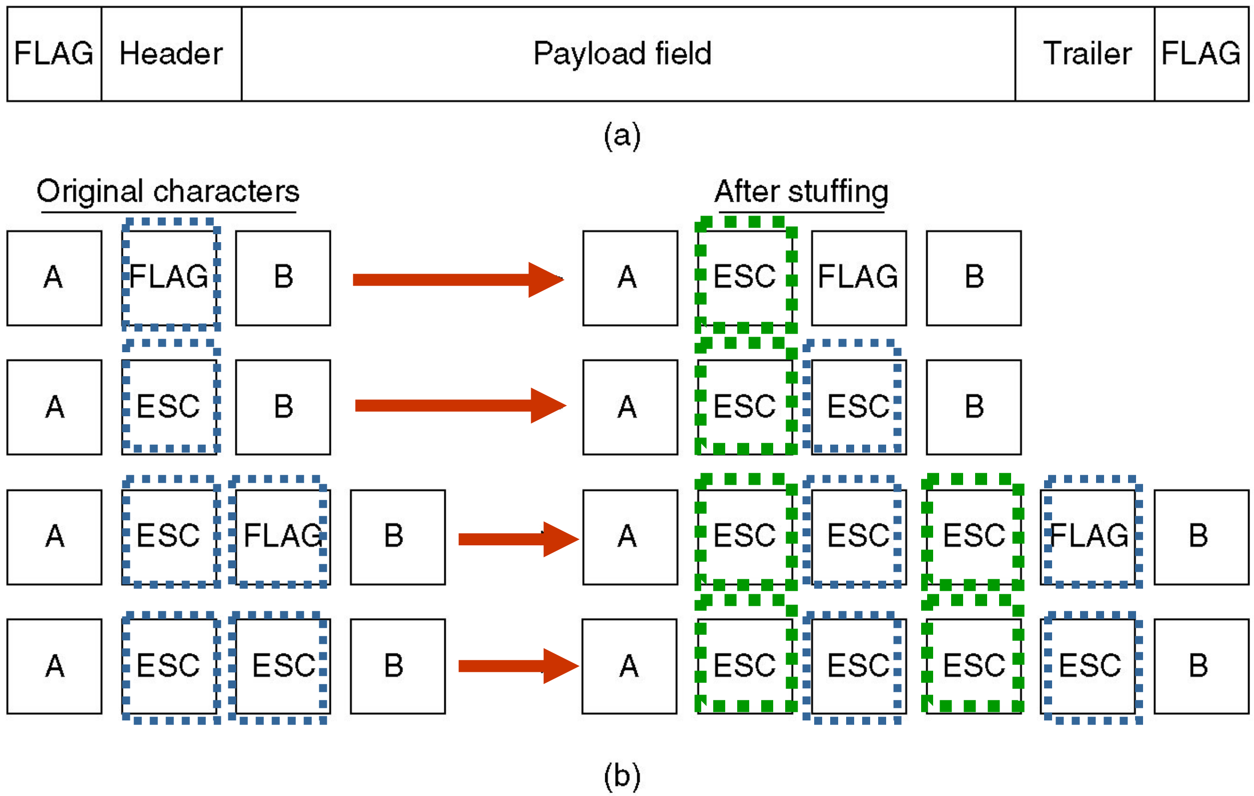
- Character count（字符计数法）
- Flag bytes with byte stuffing（带字节/字符填充的分界符法）
- Starting and ending flags, with bit stuffing（带位填充的分界标志法）
- Physical layer coding violations（物理层编码违例法）

Character count

- Insert time gaps between frames,impossible
- Uses a field in the header to specify the number of characters in the frame
- Problem



Flag Bytes with byte stuffing



Flag Bytes with bit stuffing

- Allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character.
- Each frame begins and ends with a special bit pattern, 01111110 (in fact, a flag byte).
- Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream.
- With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern.

Example

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Stuffed bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Physical layer coding violations

- 物理层编码违例法
- Only applicable to networks in which the encoding on the physical medium contains some redundancy.
- For example, some LANs encode 1 bit of data by using 2 physical bits. Normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair.
- Most DLL protocols use a combination of character count with another method for extra safety. This increases the chances of catching an error.

Error Control

- We use byte stuffing, bit stuffing and checksum as a method for detecting and determining errors in the data that we send.
- We also have to deal with making sure that the frames make it to their destination.
- The receiver sends back a control frame acknowledging the received frame and the condition of the frame.
- A timeout can occur if the acknowledgement doesn't arrive, resulting in the frame being resent.
  - Timeout interval

Error Control (cont'd)

- Resending the frame can also cause problems – what happens when the same frame is received twice or more times?
- We can also sequentially number the frames to prevent this problem.
- There are many different ways to do this type of error control (and it can be done at different levels as well).
- Managing the timers and sequence numbers are important parts of the data link layer's duties.P192

Flow Control

- We must deal with the issue where the sender is sending data at a higher rate than the receiver can receive the data.
- There are two approaches to this problem:
  - feedback-based flow control
    - feedback is used to tell the sender how the receiver is doing or to send another frame
  - rate-based flow control
    - the transfer rate is fixed by the sender
    - this is never used in the DLL

Why do we need Error Detection And Correction?

- The local loops are still analog twisted copper pairs and errors are still common.
- Wireless communication is becoming more common, and the error rates are orders of magnitude worse than on the interoffice fiber trunks.
- Transmission errors are going to be with us for many years to come.

Types of Error

- Errors come in bursts
- Independent single-bit errors
- Example
  - block size is 1000 bits
  - error rate is 0.001 per bit
- Burst errors are much harder to correct than isolated errors

Error Processing

- **Error-correcting codes**
  - **Include enough redundant information along with each block of data sent.**
  - **The receiver can deduce the transmitted data**
  - **The use of error-correcting codes is often referred to as forward error correction(前向纠错).**
- **Error-detecting codes**
  - **Include only enough redundancy**
  - **Allow the receiver to deduce that an error occurred, but not correct error, and just have it request a retransmission.**

Error Processing (cont'd)

- Each of the two techniques is applicable to different circumstances.
- Error-correcting code
  - The overhead is high but it reduces the need to resend frames.
  - best suited for networks with high error (wireless).
- Error-detecting code
  - suite for highly reliable channel, such as fiber
  - just retransmit when errors are found

Codeword（码字）

- A frame consists of m data (i.e., message) bits and r redundant, or check bits.
- The total length be n (i.e., n = m + r)
- An n-bit unit containing data and check bits is often referred to as an **n-bit codeword**.
  - N位码字

Hamming Distance（海明距离）

- **Given two** codewords**, we can determine the "difference" between the two codewords based on the number of bit difference.**
- **The number of bit positions in which two codewords differ is called the Hamming distance.**
- **If two codewords are a Hamming distance d apart, it will require d single-bit errors to convert one into the other.**
- **We can just XOR （异或） the two codewords and then count the number of 1's in the result.**

Hamming Distance Example

- 1100110000011001
- 0101110010011001
- ---------------------XOR
- 1001000001000000
- The Hamming Distance is 3.
- This means 3 bits would have to "flip" before we would be unable to detect an error.

Codewords

- we can construct a complete list of the legal codewords and find two codewords whose Hamming distance is minimum, this distance is the Hamming distance of the complete codes.
- The error-detecting and error-correcting properties of a code depend on its Hamming distance.
  - In order to *detect* **d** errors, we need a distance of **d+1**.
  - In order to *correct* **d** errors, we need a distance of **2d+1**.

The Parity Bit(奇偶位)

- A simple example of an error-detecting code
- A single parity bit is appended to the data.
- The parity bit is chosen so that the number of 1 bits in the codeword is even (or odd).
  - Data: 1011010
  - Even: 1011010 0 （偶校验）
  - Odd: 1011010 1 （奇校验）
- The Hamming distance equals 2.
  - if one bit has flipped , it is able to identify errors.
  - If two bits have flipped, then the parity bit will report the correct parity for the data.

Simple Example Of An Error-correcting Code

- Consider a code with only four valid codewords:
  - 0000000000, 0000011111, 1111100000, and 1111111111
- The hamming distance equals 5, so it can correct double errors.
  - Send 0000011111
  - Receive 0000000111
  - Deduce the data sent: 0000011111
  - Send 0000000000
  - Receive 0000000111
  - Deduce the data sent: 0000011111

?

Correcting Larger Errors

- As the Hamming distance increases, so does the ability to correct errors in the data.
- With a hamming distance of 5, it is possible to correct codewords that have 2 errors in them.
- As the Hamming distance of a set increases, the number of valid codewords decreases.

1 bit Hamming Error-correcting codes

- A n-bit code with m message bits and r check bits that will allow all single errors to be corrected.
- Relations among n, m and r
- 纠正单个错误需要的校验位的下界满足:

m	r	n（码字的总位数）
1	2	3
2~4	3	5~7
5~11	4	9~15
12~26	5	17~31
27~57	6	33~63
58~120	7	65~127

$$(n+1)2^m \le 2^n$$

$$n = m + r$$

$$\Downarrow$$

$$(m + r + 1) \le 2^r$$

1 bit Hamming Error-correcting codes(cont'd)

- The bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and so on. （从左到右编号，从1开始）
- The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits. （编号为2的乘幂的位是校验位）
- The rest (3, 5, 6, 7, 9, etc.) are filled up with the m data bits. （其余位是数据位）
- Each check bit forces the parity of some collection of bits, including itself, to be even (or odd).
- A bit may be included in several parity computations.
  - 11 = 1 + 2 + 8

How to decide check-bit？

- **Check bit 1: 1,3,5,7,9,11…**
- **Check bit 2: 2、3、6、7、10、11、…**
- **Check bit 4：4、5、6、7、……**
- **Check bit 8：8、9、10、11、……**

Example （sender）

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

7位数据位

如何确定校验位？

Conputing of Check bits(m=7,r=4)

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7
$1=2^0$	√		√		√		√		√		√
$2=2^1$		√	√			√	√			√	√
$4=2^2$				√	√	√	√				
$8=2^3$								√	√	√	√

Example

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7
	-	-	1	-	0	0	1	-	0	0	0
	0	0	-	1	-	-	-	0	-	-	-
	0	0	1	1	0	0	1	0	0	0	0

使用偶校验（even），一个校验集合里的1的个数是偶数

Do exercise

Raw code：1100001

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7
	-	-	1	-	1	0	0	-	0	0	1
	?	?	-	?	-	-	-	?	-	-	-
	1	0	1	1	1	0	0	1	0	0	1

Hamming correct-error（reciever）

- initializes a counter to zero
- examines each check bit, k (k = 1, 2, 4, 8, ..., 2^k), to see if it has the correct parity
- If incorrect parity, adds the number of check bit k to the counter
  - counter = 0, valid code
  - counter ≠0, invalid code
    - The value of the counter denotes the number of the incorrect bit.

An example

- If receiving a codeword 00111000100，question is : the code is correct or not ,the correct one is ?
- Solution: (m=7,r=4)
  - The first：00111000100，three 1，wrong
  - The second：00111000100，one 1，wrong
  - The forth：00111000100，two 1，correct
  - The eigth：00111000100，one 1，wrong
- Couter：1+2+8=11
- The 11th bit is wrong，change the bit from 0 to 1, so the corrected codeword is: 00111000101

Correct burst errors

- Hamming codes can only correct a single error.
- To correct burst errors, a sequence of k consecutive codewords are arranged as a matrix, one codeword per row. And the data should be transmitted one column at a time, starting with the leftmost column.
- Uses kr check bits to make blocks of km data bits immune to a single burst error of length k or less.

Correct burst errors fig.

m+r列

K行

P1,P2,A1,P3,A2,A3,A4…… Am,Pr

P1,P2,B1,P3,B2,B3,B4…… Bm,Pr

P1,P2,C1,P3,A2,C3,C4…… Cm,Pr

……

P1,P2,K1,P3,K2,K3,K4…… Km,Pr

Do exercise（1/2）

- **Original data:** 10101111,even**-parity Hamming code, if hope to correct one single error, What is the Hamming code for it?**
- Solution： m=8， According $(m+r+1) \leq 2^r$:

r=4， ??1?010?1111

P1=B1⊕B3⊕B5⊕B7⊕B9⊕B11 =∑(0,1,0,0,1,1)=1

P2=B2⊕B3⊕B6⊕B7⊕B10⊕B11=∑(0,1,1,0,1,1)=0

P3=B4⊕B5⊕B6⊕B7 ⊕B12 =∑(0,0,1,0,1)=0

P4=B8⊕B9⊕B10⊕B11⊕B12 =∑(0,1,1,1,1)=0

So, Hamming code is: 101001001111

Do exercise（2/2）

- All conditions is as above, if receiver has received a codeword like： 1 0 0 1 1 0 0 0 1 1 0 0 (m=8,r=4），Question: Is the codeword is correct or not? What is the corresponding correct one if wrong?

- **Solution：**

$$P1=B1\oplus B3\oplus B5\oplus B7\oplus B9\oplus B11 =\sum(1,0,1,0,1,0)=1$$
$$P2=B2\oplus B3\oplus B6\oplus B7\oplus B10\oplus B11=\sum(0,0,0,0,1,0)=1$$
$$P3=B4\oplus B5\oplus B6\oplus B7 \oplus B12 =\sum(1,1,0,0,0)=0$$
$$P4=B8\oplus B9\oplus B10\oplus B11\oplus B12 =\sum(0,1,1,0,0)=0$$

So, Counter=1+2=3， the 3rd bit is wrong, correct one is：

1 0 1 1 1 0 0 0 1 1 0 0

Error Detection

- Error-detecting codes **only include enough data to let the receiver determine whether the data is faulty.**
- **If the error rate of physical link is much lower, error detection and retransmission is usually more efficient.**
  - **copper wire or fiber**

How efficient, an example

- For comparison, consider a channel with error rate of 10^{-6} per bit. Let block size be 1000 bits.
  - To correct a single error (by Hamming code), 10 check bits per block are needed. To transmit 1000 blocks, 10,000 check bits (overhead) are required.
  - To detect a single error, a single parity bit per block will suffice. To transmit 1000 blocks, only one extra block (due to the error rate of 10^{-6} per bit) will have to be retransmitted, giving the overhead（开销） of only 2001 (= 1000*1 + 1001) bits.

Polynomial Code(多项式编码)

- Also known as a CRC (Cyclic Redundancy Check，循环冗余校验码).
- Based upon treating bit strings as representations of polynomials with coefficients of 0 and 1
  - Example: 110001
  - five-degree six-term polynomial（6项5阶多项式）
  - $1*x^5 + 1*x^4+0*x^3+0*x^2+0*x^1+1*x^0=x^5+x^4+1$
- Polynomial arithmetic is done modulo 2. Both addition and subtraction are identical to EXCLUSIVE OR（等同于异或）:
  - 10011011　　01010101
  - \+ 11001010　　- 10101111
  - \---------------　　---------------
  - 01010001　　11111010

What is modulo 2?

- **Modulo 2 addition & substraction: XOR logic**

  $$0\oplus0=0;\ 0\oplus1=1;$$

  $$1\oplus0=1;\ 1\oplus1=0.$$

– **Modulo 2 mltiplication:**

```
      1 0 1 0
×       1 0 1
      -------
      1 0 1 0
    0 0 0 0
  1 0 1 0
  -----------
  1 0 0 0 1 0
```

-- **Modulo 2 division:**

```
             1 0 1
        ___________
1 0 1 / 1 0 0 0 0
        1 0 1
        -----
          0 1 0
          0 0 0
          -----
            1 0 0
            1 0 1
            -----
              0 1
```

Polynomial Code (cont'd)

- The basic idea of the CRC method:
  - The sender and receiver agree upon a **generator polynomial（生成多项式）**, G(x), in advance.
  - The sender appends a **checksum** to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by G(x).
  - When the receiver gets the frame, it tries dividing it by the same G(x). If there is a remainder, there must have been an error and a retransmission will be requested.

Algorithm For Computing Checksum

1. Let r be the degree of G(x). Append r zero bits to the low-order end of the frame so it now contains m + r bits and corresponds to the polynomial $x^r M(x)$.
2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^r M(x)$, using modulo 2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial T(x).

An example of CRC

- **Frame: 1101011011（m=10）**

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

- **$G(x)=x^4+x+1$ （r=4阶）**
- **$x^4M(x)$** （相当于在原码字后加r个0）

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

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

- **$x^4M(x)$- reminder($x^4M(x)$/ $G(x)$)=?**

An example of CRC (cont'd)

Frame : 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

```
                1 1 0 0 0 0 1 0 1 0
         ____________________________
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
            1 0 0 1 1
            ---------
              1 0 0 1 1
              1 0 0 1 1
              ---------
                0 0 0 0 1
                0 0 0 0 0
                ---------
                  0 0 0 1 0
                  0 0 0 0 0
                  ---------
                    0 0 1 0 1
                    0 0 0 0 0
                    ---------
                      0 1 0 1 1
                      0 0 0 0 0
                      ---------
                        1 0 1 1 0
                        1 0 0 1 1
                        ---------
                          0 1 0 1 0
                          0 0 0 0 0
                          ---------
                            1 0 1 0 0
                            1 0 0 1 1
                            ---------
                              0 1 1 1 0
                              0 0 0 0 0
                              ---------
                                1 1 1 0   <--- Remainder
```

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Summary of this lecture

- Learn functions of DLL
- Learn and Master framing method
- Master error detection and correction methods
  - Hamming code海明码
  - Polynomial code(CRC)