Advanced Windows Network Programming: Unit 1

Frame Parser

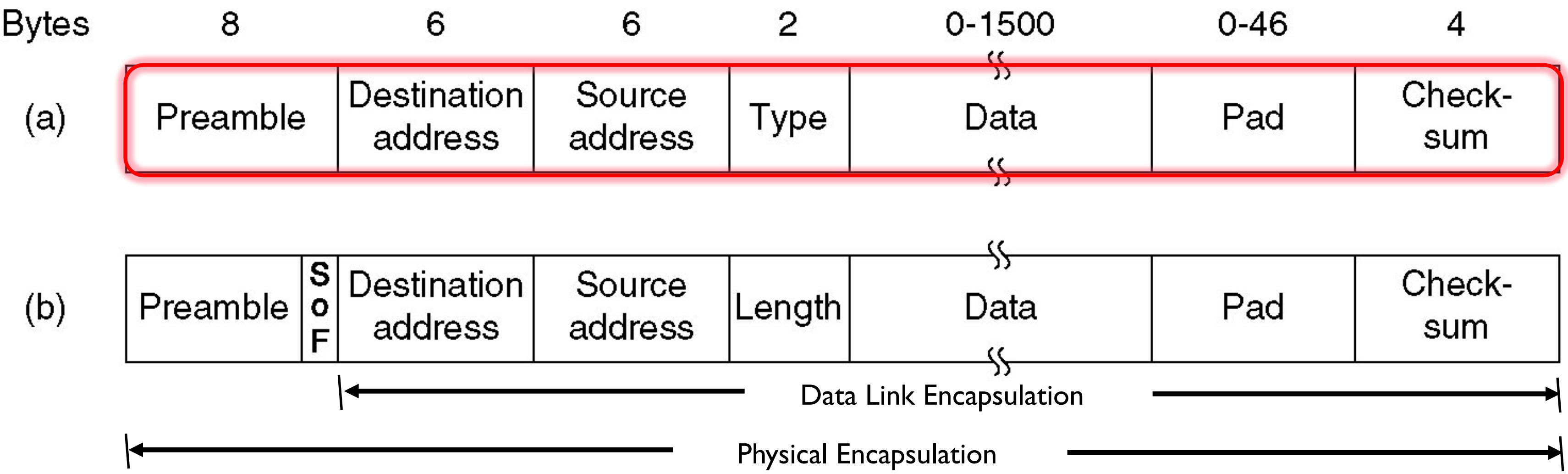
Prof. Lin Weiguo

Sept. 2019

Topics

- Ethernet Frame Format
- Error Detection and Correction

Ethernet Frame Format



Frame formats. (a) DIX Ethernet II (*RFC 894*)
(b) IEEE 802.3 (*RFC 1042*)

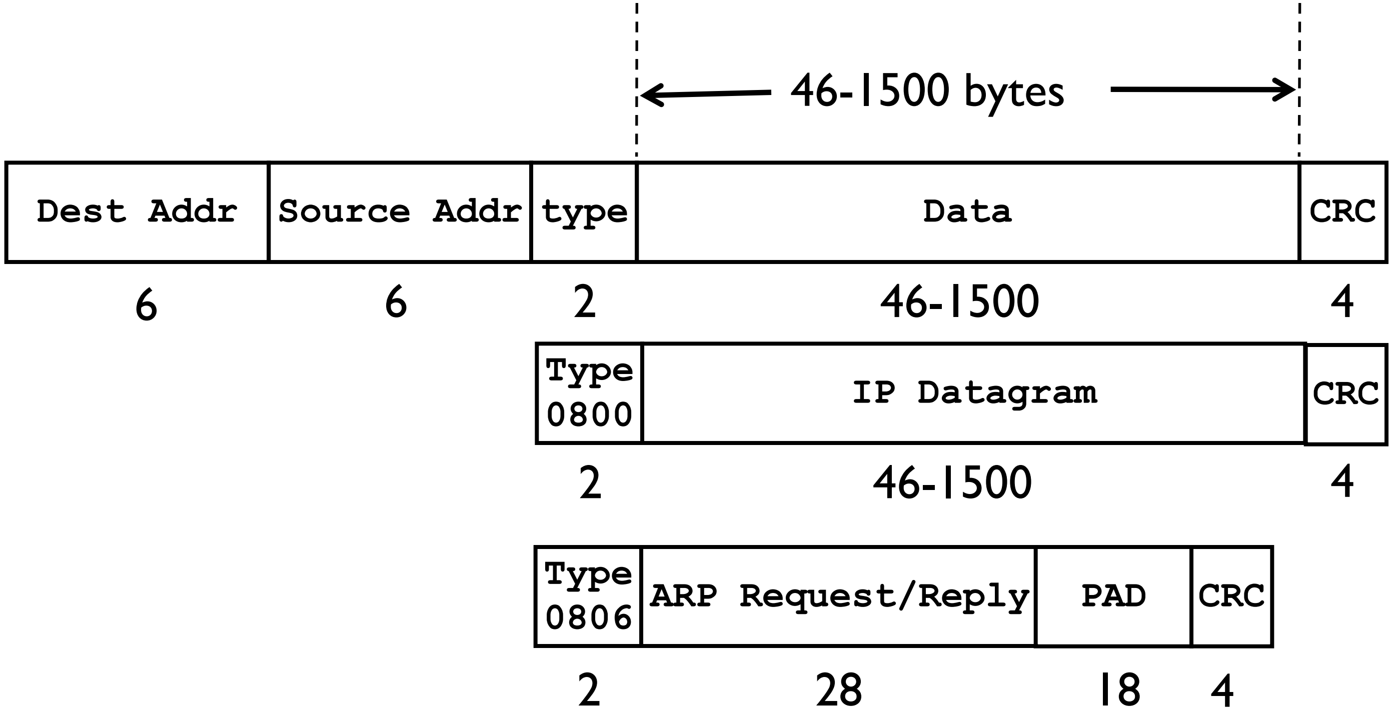
802.3 Frame Fields Definition(*RFC 1042*)

- Field Definition:

Preamble == 7 bytes of 10101010
Start == 1 byte of 10101011
Destination == 6 bytes of MAC address
multicast == sending to a group of stations.
broadcast == (destination = all 1's) to all stations on network
Source == 6 bytes of MAC address
Length == number of bytes of data **<= 1500**
Data == comes down from network layer**(<=1500 B)**
Pad == ensures 64 bytes from destination address through checksum.
checksum == 4 bytes of CRC. (FCS, frame check sequence)
the FCS is calculated based on the header and the
data field in the frame.

Type Field of Ethernet II (*RFC 894*)

- The Value of Type >1500



46-1500 bytes

Dest Addr	Source Addr	type	Data	CRC
6	6	2	46-1500	4

Type 0800	IP Datagram	CRC
2	46-1500	4

Type 0806	ARP Request/Reply	PAD	CRC
2	28	18	4

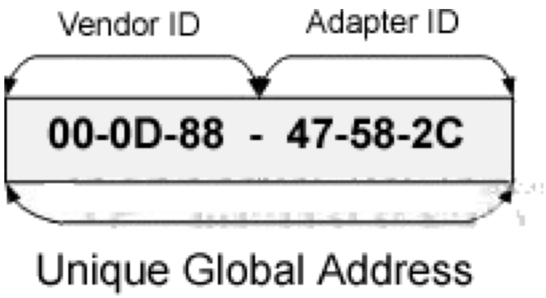
TCP/IP Illustrated, W. Richard, Stevens. Vol. 1, p23

EtherType for some common protocols

EtherType	Protocol
0x0800	Internet Protocol, Version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)
0x8035	Reverse Address Resolution Protocol (RARP)
0x809B	AppleTalk (Ethertalk)
0x80F3	AppleTalk Address Resolution Protocol (AARP)
0x8100	VLAN-tagged frame (IEEE 802.1Q)
0x8137	Novell IPX (alt)
0x8138	Novell
0x86DD	Internet Protocol, Version 6 (IPv6)
0x8808	MAC Control
0x8863	PPPoE Discovery Stage
0x8864	PPPoE Session Stage

MAC Address

- four types of MAC addresses:



Vendor ID

Adapter ID

00-0D-88 - 47-58-2C

Unique Global Address

DLink: 00-0D-88
Cisco: 00-00-0C
Intel : 00-0C-F1

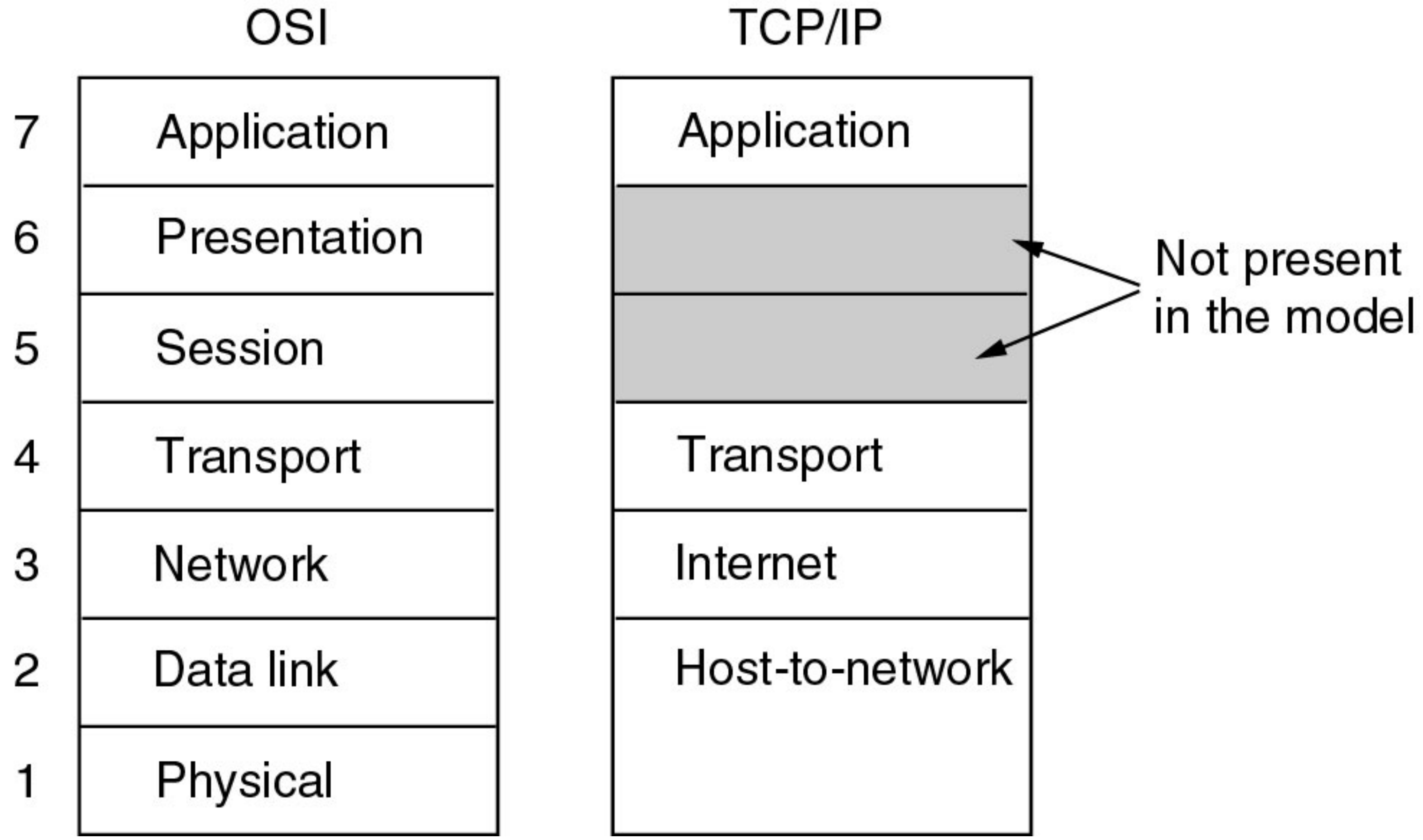
Organizationally **U**nique **I**dentifier:
http://standards.ieee.org/regauth/oui/

MAC Type	Address range
Globally Unique	*0-**-**-**-**-** *4-**-**-**-**-** *8-**-**-**-**-** *C-**-**-**-**-**
Locally Administered	*2-**-**-**-**-** *6-**-**-**-**-** *A-**-**-**-**-** *E-**-**-**-**-**
Multicast	*1-**-**-**-**-** *3-**-**-**-**-** *5-**-**-**-**-** *7-**-**-**-**-** *9-**-**-**-**-** *B-**-**-**-**-** *D-**-**-**-**-** *F-**-**-**-**-** Note: except broadcast address
Broadcast	FF-FF-FF-FF-FF-FF

Brief History

- 1973, Xerox Palo Alto Research Center
  - Robert Metcalfe with David Boggs, founded 3Com
- 1979 , Xerox, DEC, Intel - DIX 1.0
- 1980, IEEE 802 project –LAN/MAN standards
- 1982, DIX 2.0: ***Ethernet II***
- 1983, Novell - ***802.3 raw***
- 1984-1985, IEEE 802.1~IEEE 802.5
  - ***802.3 SAP/802.3 SNAP***

The TCP/IP Reference Models



	OSI	TCP/IP
7	Application	Application
6	Presentation	
5	Session	
4	Transport	Transport
3	Network	Internet
2	Data link	Host-to-network
1	Physical	

Not present in the model

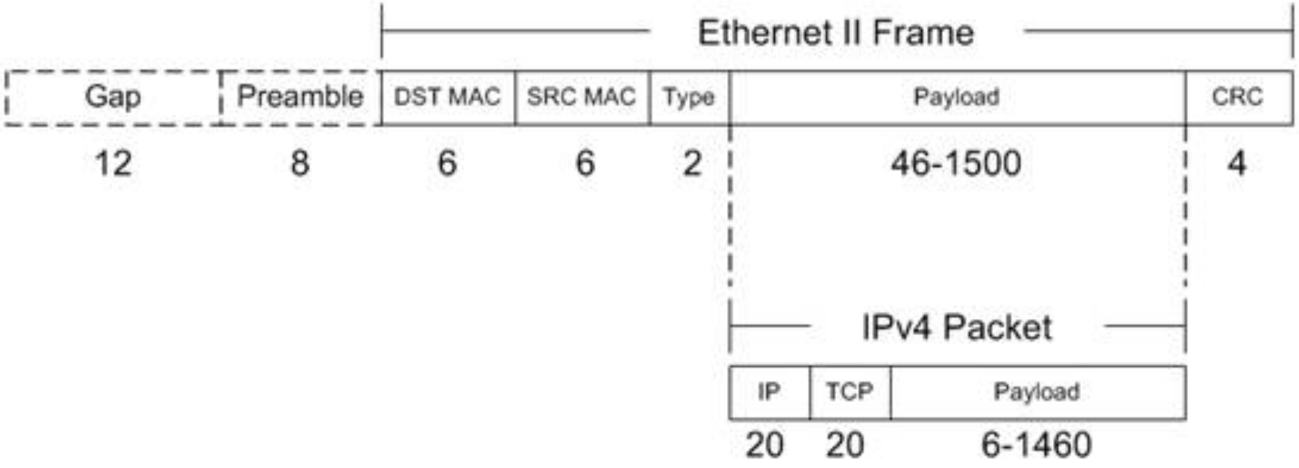
The TCP/IP reference model

Ethernet and IEEE 802 Encapsulation

(The Host Requirements RFC 1122)

- Every Internet host connected to a 10Mbps Ethernet cable:
  - **MUST** be able to send and receive packets using RFC-894 encapsulation;
  - **SHOULD** be able to receive RFC-1042 packets, intermixed with RFC-894 packets; and
  - **MAY** be able to send packets using RFC-1042 encapsulation.

TCP/IP packet in a Ethernet II Frame



Ethernet II Frame

Gap	Preamble	DST MAC	SRC MAC	Type	Payload	CRC
12	8	6	6	2	46-1500	4

IPv4 Packet

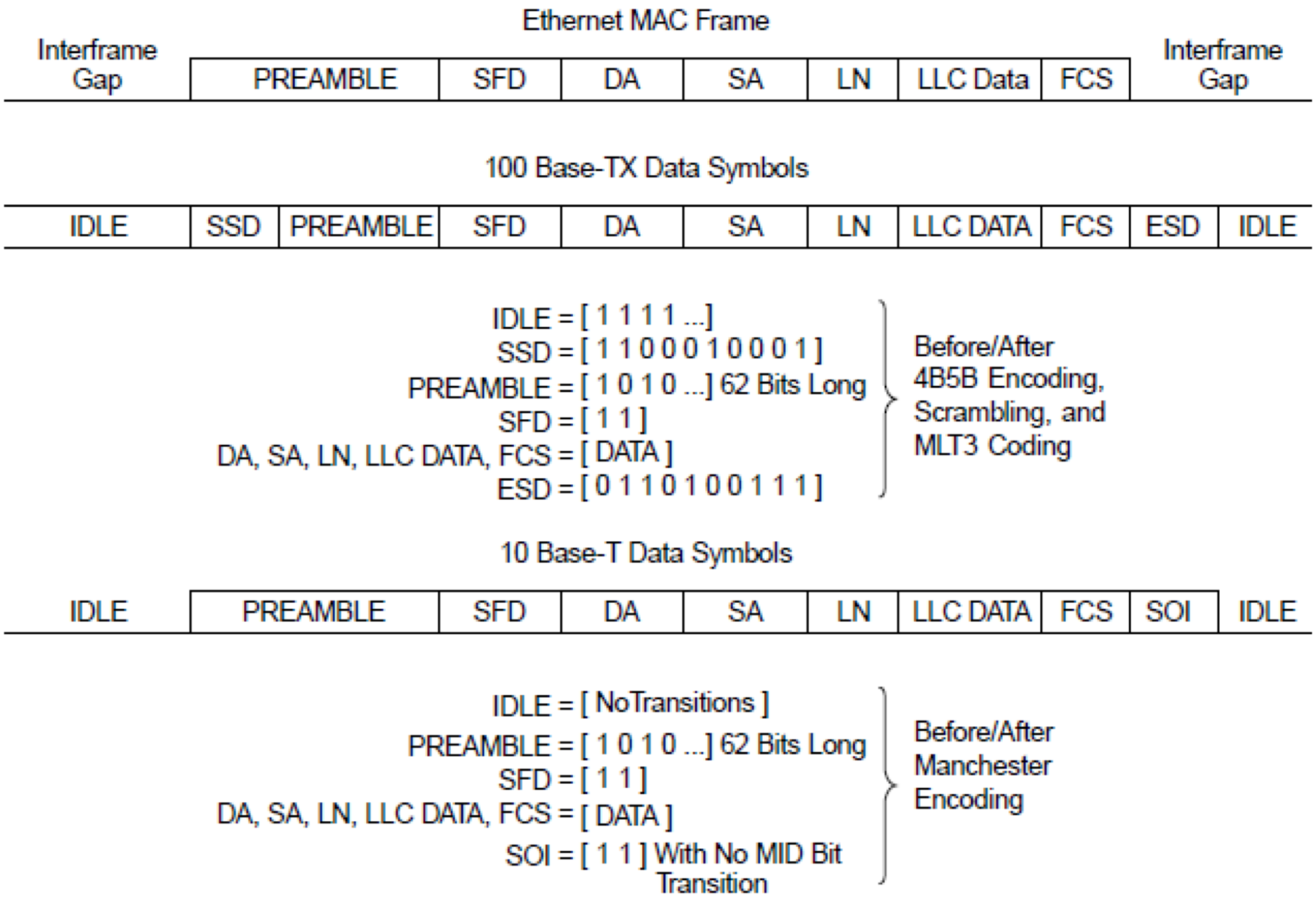
IP	TCP	Payload
20	20	6-1460

Ethernet II with the 'Preamble' and 'IFG (Inter-Frame Gap)'

ESD: End-of-Stream Delimiter

- **ESD: Within IEEE 802.3, a code-group pattern used to terminate a normal data transmission.**
  - For 10BASE-T, Start Of Idle (*SOI*) *pulse*
  - For 100BASE-T4, the ESD is indicated by the transmission of five redefined ternary code-groups named eop1-5.
  - For 100BASE-X, the ESD is indicated by the transmission of the code-group/T/R.
  - For 100BASE-T2, the ESD is indicated by two consecutive pairs of predefined PAM5×5 which are generated using unique Start-of-Stream Delimiter (SSD)/ESD coding rules.
  - For1000BASE-T, the ESD is indicated by two consecutive vectors of four quinary symbols

100BASE-TX and 10BASE-T frame format



Ethernet MAC Frame

Interframe Gap	PREAMBLE	SFD	DA	SA	LN	LLC Data	FCS	Interframe Gap

100 Base-TX Data Symbols

IDLE	SSD	PREAMBLE	SFD	DA	SA	LN	LLC DATA	FCS	ESD	IDLE

IDLE = [1 1 1 1 ...]
SSD = [1 1 0 0 0 1 0 0 0 1]
PREAMBLE = [1 0 1 0 ...] 62 Bits Long
SFD = [1 1]
DA, SA, LN, LLC DATA, FCS = [DATA]
ESD = [0 1 1 0 1 0 0 1 1 1]

Before/After
4B5B Encoding,
Scrambling, and
MLT3 Coding

10 Base-T Data Symbols

IDLE	PREAMBLE	SFD	DA	SA	LN	LLC DATA	FCS	SOI	IDLE

IDLE = [NoTransitions]
PREAMBLE = [1 0 1 0 ...] 62 Bits Long
SFD = [1 1]
DA, SA, LN, LLC DATA, FCS = [DATA]
SOI = [1 1] With No MID Bit
Transition

Before/After
Manchester
Encoding

Sample Frame

Here is a sample captured Ethernet II IP Packet decoded.
It is a Server to Client HTTP 200 OK packet.

```
========================================
Frame 1 (214 bytes on wire, 214 bytes captured)
    Arrival Time: May 13, 2004 05:17:11.266911268
    Packet Length: 214 bytes
    Capture Length: 214 bytes
    Protocols in frame: eth:ip:tcp:http:data
Ethernet II, Src: fe:ff:20:00:01:00 (fe:ff:20:00:01:00), Dst: Xerox_00:00:00 00:00:01:00:00:00)
    Destination: Xerox_00:00:00 (00:00:01:00:00:00)
    Source: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
    Type: IP (0x0800)
Internet Protocol,
Src: 216.239.59.99 (216.239.59.99), Dst: 145.254.160.237 (145.254.160.237)
Transmission Control Protocol,
Src Port: http (80), Dst Port: 3371 (3371), Seq: 0, Ack: 0, Len: 160
Hypertext Transfer Protocol
    Data (160 bytes)
========================================
```

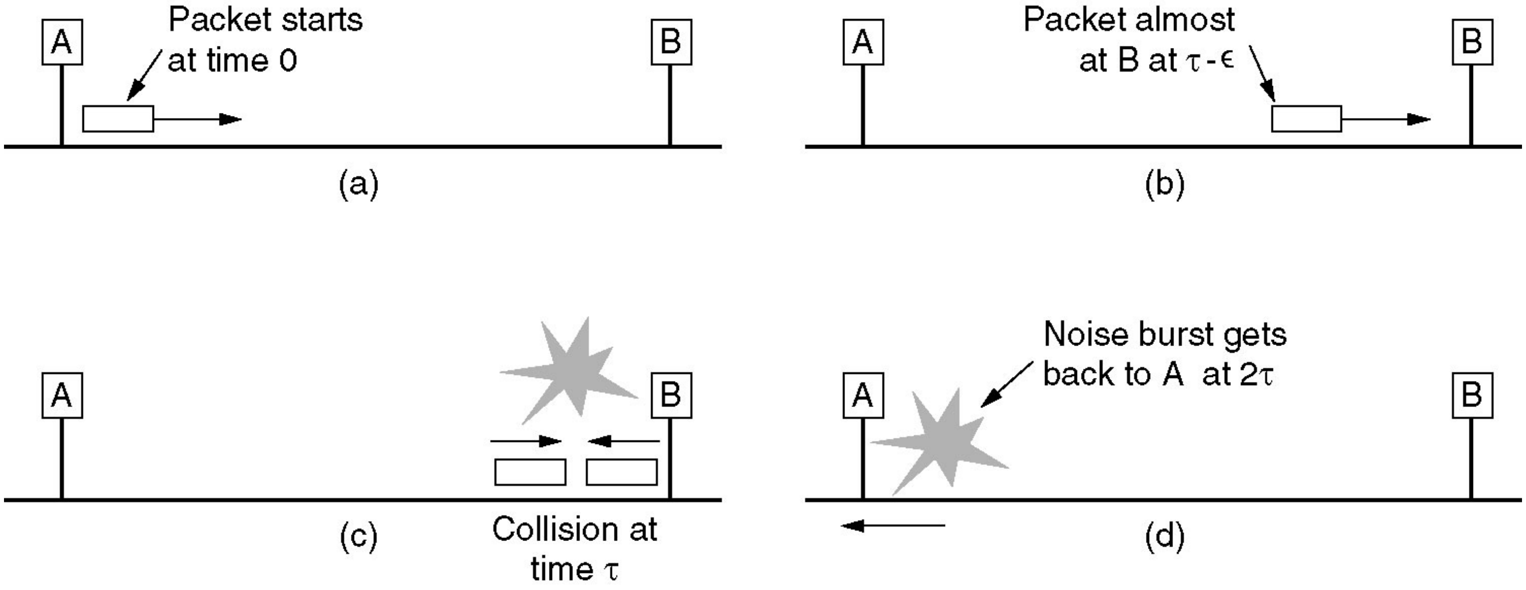
Ethernet Framing FLAG

- Only 1 Start Flag(8 bytes Preamble) for a Ethernet Frame
  - Compare with HDLC (Tanenbaum. Sec 3.6 P235)
  - A Point-to-Point WAN is different from a broadcasting LAN
  - Ethernet devices must allow a minimum idle period between transmission of Ethernet frames known as the interframe gap (IFG) or interpacket gap (IPG).
  - Receiver determines the end of frame by the gap between two frames

The PAD

- When a transceiver detects a collision, it truncates the current frame, which means that stray bits appear on the cable all the time. To make it easier to distinguish valid frames from garbage, Ethernet requires that there being a min frame length
- **The pad ensures transmission takes enough time so it's still being sent when the first bit reaches the destination. The frame needs to still be going out when the noise burst from another stations collision detection gets back to the sender.**

Collision detection in worst case



Collision detection can take as long as 2τ.

Length of The PAD

- For a 10Mbps LAN with a maximum length of 2500 meters and four repeaters, the round-trip time is nearly 50us in the worst case. At 10 Mbps, a bit takes 100ns, so 500 bits is the smallest frame that is guaranteed to work, for safety, this number was rounded to 512 bits or 64 bytes.

References

- IEEE CSMA/CD Std 802.3-2008
  - *http ://standards.**ieee**.org/get**ieee**802/**802.3**.html*
- IEEE list of EtherType values
  - http://en.wikipedia.org/wiki/EtherType
  - *http://standards.ieee.org/regauth/ethertype/eth.txt*
- RFC 894 - A Standard for the Transmission of IP Datagrams over Ethernet Networks
- RFC826 – Ethernet Address Resolution Protocol
- RFC 1042 - Standard for the transmission of IP datagrams over IEEE 802 networks

Error Detection and Correction

- Error-Correcting Codes
- Error-Detecting Codes

Putting in enough redundancy along with the data to be able to detect (and correct) data errors.

Error

- In data communication, line noise is a fact of life (e.g., signal attenuation, natural phenomenon such as lightning, and the telephone worker). Moreover, noise usually occurs as bursts rather than independent, single bit errors. For example, a burst of lightning will affect a set of bits for a short time after the lightning strike.

Error Correcting Codes

- Detecting and correcting errors requires redundancy - sending additional information along with the data.
- There are two types of attacks against errors
  - **Error Detecting Codes:** Include enough redundancy bits to detect errors and use ACKs and retransmissions to recover from the errors.
  - **Error Correcting Codes:** Include enough redundancy to detect and correct errors.

Hamming Distance

- Given any two codewords, we can determine how many of the bits differ by exclusive or (XOR) of the two words. The number of 1 bits in the result is the **Hamming Distance**.

- If two codewords are **d** bits apart, **d** errors are required to convert one to the other.

Hamming Distance

- In general, all **2^m** possible data words are legal. However, by choosing check bits carefully, the resulting codeword will have a large Hamming Distance. The larger the Hamming distance, the better the codes are able to detect errors.
- To detect ***d 1-bit*** errors requires having a Hamming Distance of at least ***d + 1*** bits.
- To correct ***t*** errors requires ***2t + 1*** bits. Intuitively, after ***t*** errors, the garbled messages is still closer to the original message than any other legal codeword.

Parity Bits

- A single parity bit is appended to each data block (e.g. each character in ASCII systems) so that the number of 1 bits always adds up to an even (odd) number.

1000000(1)   1111101(0)

- The Hamming Distance for parity is 2, and it cannot correct even single-bit errors (but can detect single-bit errors).

Redundant bits for Error Correction

- Normally, 2 bits needed to represent 4 possible values:

"00" "01" "10" "11"

- As an example, consider a 10-bit code used to represent 4 possible values:

"00000 00000", "00000 11111", "11111 00000", and "11111 11111".

Its Hamming distance is *5*, and we can correct *2* single-bit errors.

Catching Errors

- Example:

  "10111 00010" becomes
  " 11111 00000" by changing only two bits.

- However, if the sender transmits

  "11111 00000" and the receiver sees
  "00011 00000", the receiver will not correct the error properly.

- Finally, in this example we are guaranteed to catch all 2-bit errors, but we might do better:

  if "00111 00011" contains 4 single-bit errors, we will reconstruct the block correctly.

Error Correction

- **What's the fewest number of bits needed to correct single bit errors?**

  Let us design a code containing $n = m + r$ bits that corrects all single-bit errors (m is the number of message (data) bits and r is number of redundant (check) bits):

- There are 2^m legal messages (e.g., legal bit patterns).

- Each of the ***m*** messages has ***n*** illegal codewords a distance of ***1*** from it. That is, if we systematically invert each bit in the corresponding ***n***-bit codeword, we get illegal codewords a distance of ***1*** from the original. Thus, each message requires ***n + 1*** bits dedicated to it (***n*** that are one bit away and ***1*** that is the message).

Error Correction

- The total number of bit patterns is $(n + 1) * 2^m \le 2^n$. That is, all $(n+1) * 2^m$ encoded messages should be unique, and there can't be fewer messages than the 2^n possible code-words.

- Since $n = m + r$, we get:

  $(m + r + 1) * 2^m \le 2^{m+r}$ or $(m + r + 1) \le 2^r$

- For instance, $m = 10$, $r \ge 4$.

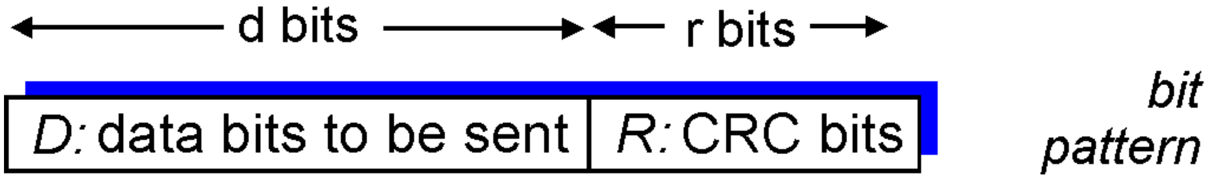
- This formula gives the absolute lower limit on the number of bits required to detect (and correct) 1-bit errors.

Error Detection

- Error correction is relatively expensive (computationally and in bandwidth.)
  - For example, 10 redundancy bits are required to correct 1 single-bit error in a 1000-bit message. In contrast, detecting a single bit error requires only a single-bit, no matter how large the message.
- The most popular error detection codes are based on polynomial codes or cyclic redundancy codes (CRCs).
- Allows us to acknowledge correctly received frames and to discard incorrect ones.

Cyclic Redundancy Check

- View data bits, D, as a binary number
- Choose **r+1** bit pattern (generator), G
- Goal: choose **r** CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- Widely used in practice



$D * 2^r \text{ XOR } R$

mathematical formula

Cyclic Redundancy Check

- The CRC is "redundant" because it adds no information, but more complicated than a checksum.
- A single corrupted bit in the data will result in a one bit change in the calculated CRC but multiple corrupted bits may cancel each other out.
- CRCs treat blocks of input bits as coefficient-sets for polynomials. E.g., binary 10100000 implies the polynomial:

$$\boxed{1\text{x } 2^7 + 0\text{x } 2^6 + 1\text{x } 2^5 + 0\text{x } 2^4 + 0\text{x } 2^3 + 0\text{x } 2^2 + 0\text{x } 2^1 + 0\text{x } 2^0}$$

- A second polynomial, with constant coefficients, is called the "**generator polynomial**". This is divided into the message polynomial, giving a **quotient** and **remainder**. The **coefficients** of the remainder form the bits of the final CRC.

Cyclic Redundancy Check

Want:

$D \cdot 2^r \text{ XOR } R = nG$

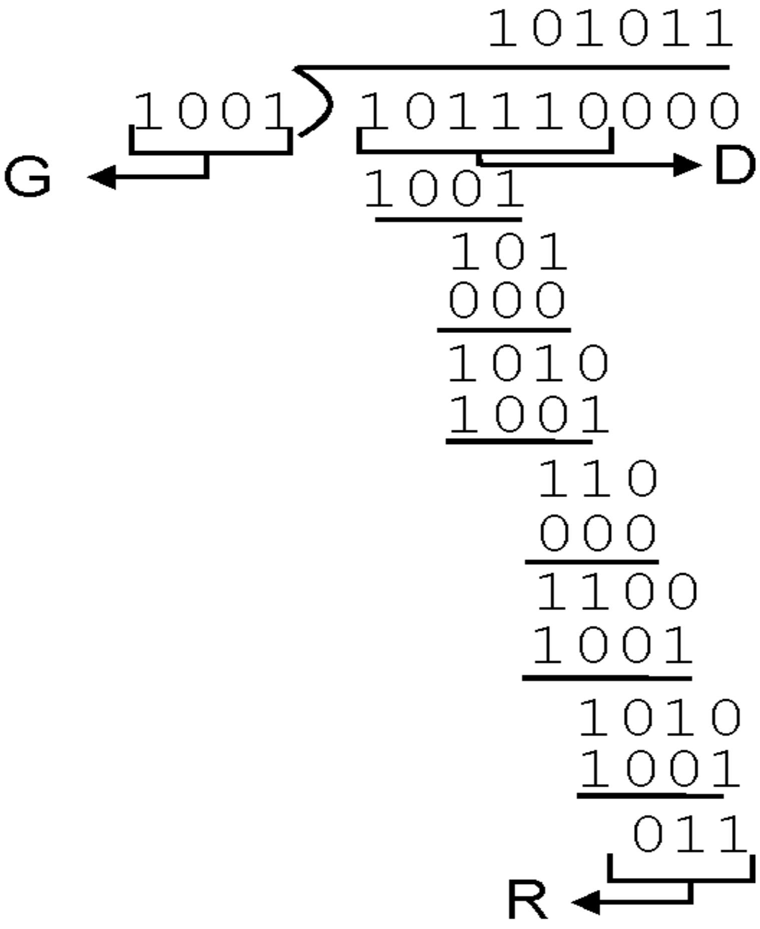
equivalently:

$D \cdot 2^r = nG \text{ XOR } R$

equivalently:

if we divide $D \cdot 2^r$ by G, want remainder R

$R = \text{remainder}[\overline{D \cdot 2^r} / G]$

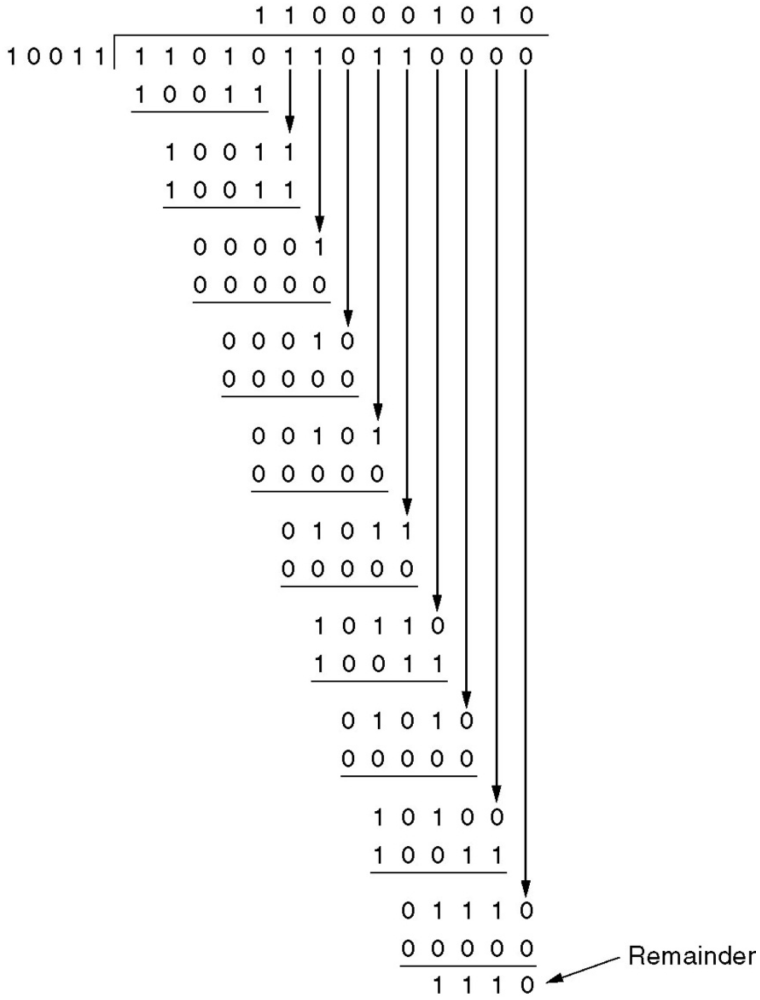


Cyclic Redundancy Check

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



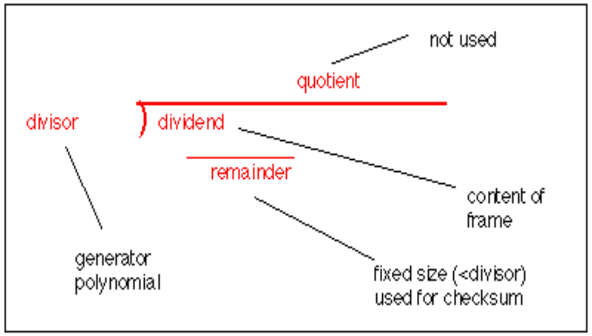
Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Calculation of the polynomial code checksum.

Cyclic Redundancy Check

A software implementation: input byte

$G(x)=x^{16}+x^{12}+x^{5}+1$



```
unsigned short crc = 0xFFFF;
unsigned short temp;
unsigned short index;

for (index = 0; index <= 7; index++)
{
    temp = (crc >> 15) ^ (byte >> 7);
    crc <<= 1;
    if(temp)
    {
        crc ^= 0x1021;
    }
    byte <<= 1;
}
```

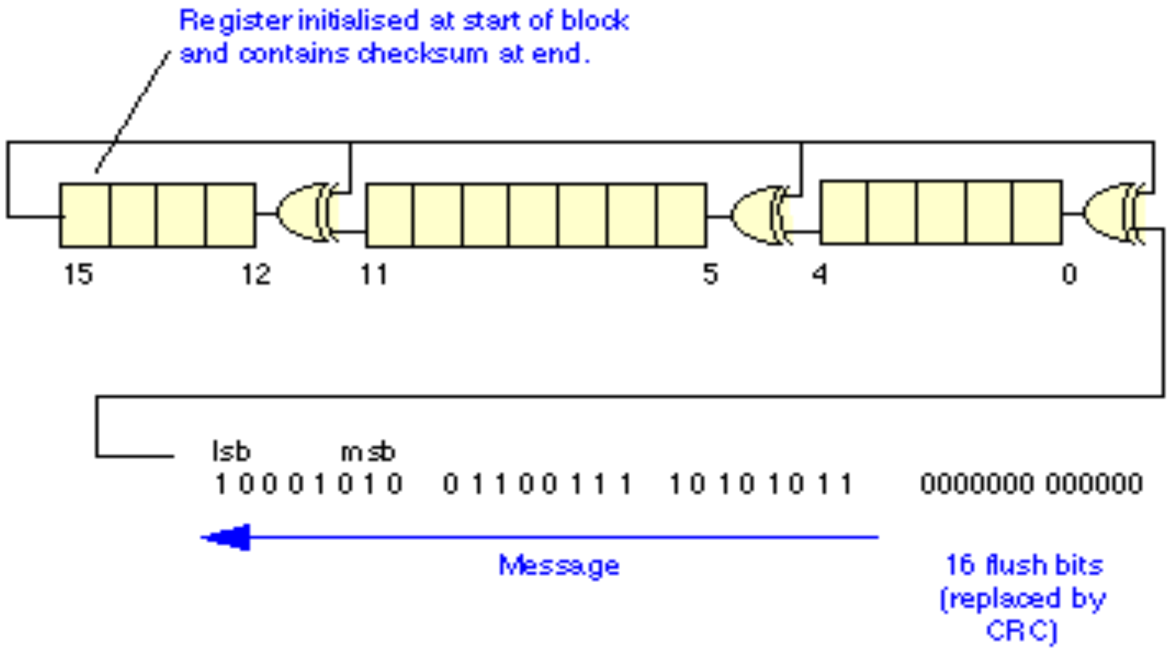
CRC Properties

- Properties
  - Characterize error as E(x), Divisor polynomial as C(x)
  - Error detected unless C(x) divides E(x)
    - (*i.e.*, E(x) is a multiple of C(x))
- What errors can we detect?
  - All single-bit errors, if x^k and x^0 have non-zero coefficients
  - All double-bit errors, if C(x) has at least three terms
  - All odd bit errors, if C(x) contains the factor (x + 1)
  - Any bursts of length < k, if C(x) includes a constant term
  - Most bursts of length ≥ k

Polynomials commonly used

- Three polynomials are in common use they are:
  - CRC-16 = x16+x15+x2+1 (used in HDLC)
  - CRC-CCITT = x16+x12+x5+1
  - CRC-32 = x32+x26+x23+ x22+x16+x12+ x11+x10+x8+ x7+x5+x4+ x2+x1+1 (used in Ethernet)

CRC hardware implementation

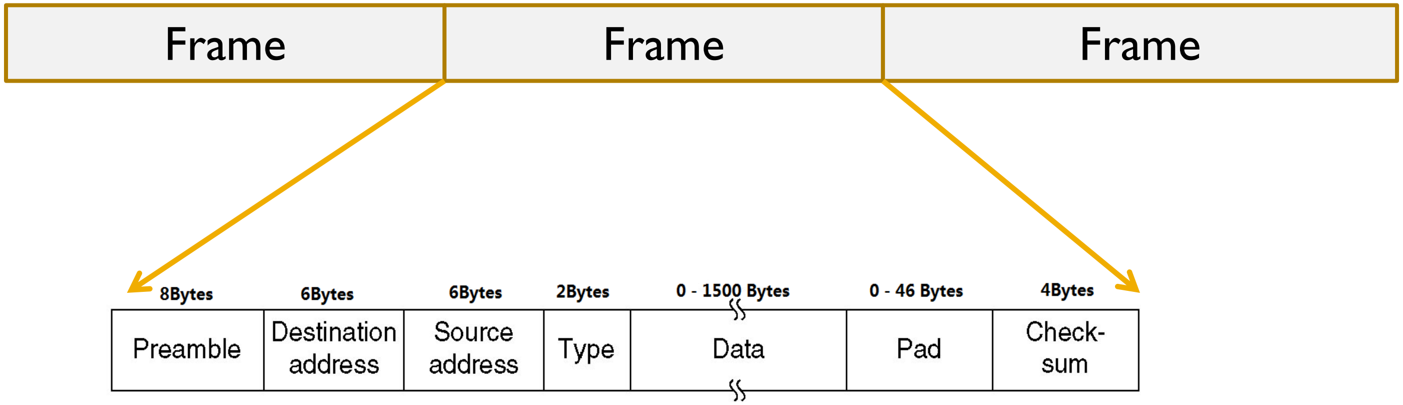

Register initialised at start of block and contains checksum at end.

A hardware implementation for a 16-bit CRC
(*Shift Register* Sequences)

Reference

- Wiki: Cyclic redundancy check
  - http://en.wikipedia.org/wiki/Cyclic_redundancy_check
  - http://en.wikipedia.org/wiki/Computation_of_CRC
- On-line CRC calculation and free library
  - http://www.lammertbies.nl/comm/info/crc-calculation.html
- A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS (CRC Look-up Table)
  - http://www.ross.net/crc/crcpaper.html

structure of the Demo Frame File



FrameParser.rar @ tlc.cuc.edu.cn