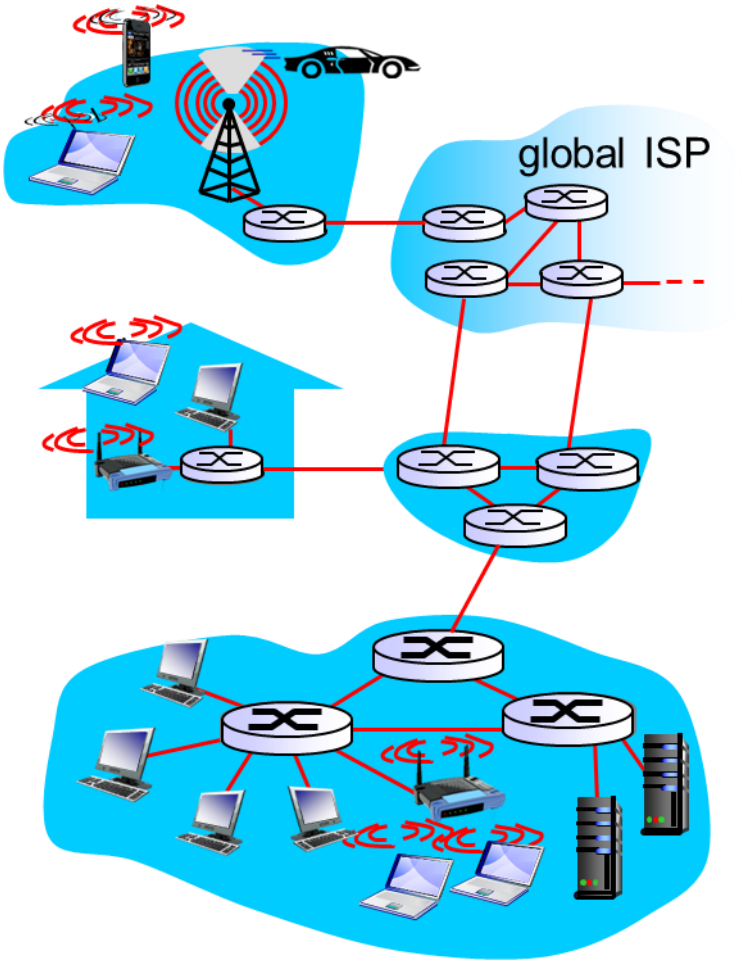
Terminology



terminology:

- hosts and routers: nodes

Error Detection and Correction are quiet distinct things

Error Detection –

Error Correction –

Internet checksum

The IP checksum is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header.

2's complement fixed point integers (8-bit)

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
1111	-1	F
1110	-2	E
1101	-3	D

Add two integers: -3 + 5 =

1's complement fixed point integers (8-bit)

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
1111	-0	F
1110	-1	E
1101	-2	D
1100	-3	C

Add the same numbers:

Simple Internet checksum example

- Pair adjacent octets to be check-summed to form 16-bit integers
- Perform 1's complement sum of these 16-bit integers

AB 00 FF DE 03 A3 9C F2 00 00 Packet

_____ Form the 16-bit words

_____ Calculate 2's complement sum

= _____ Store the sum in a 32-bit word

4B73 + 0002 = Calculate 1's complement sum

Calculate 1's complement of the 1's complement sum

~4B75 =

We send the packet including the checksum

AB 00 FF DE 03 A3 9C F2 _____

RECEIVER

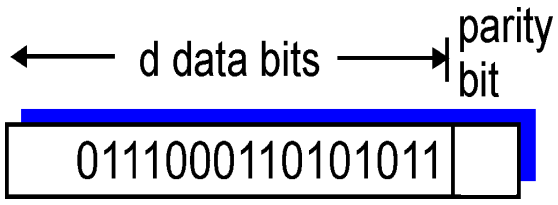
AB00+ FFDE +03A3+9CF2+_____ = _____

_____ + _____ = _____ = _____ (in 1's compliment)

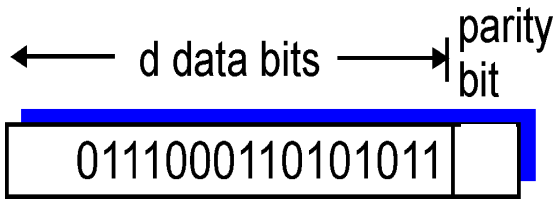
ERROR?

Single bit parity

ODD parity



EVEN parity



Error Detection

Cannot Detect

Two dimensional bit parity

- Can detect
    - Any number of errors in a single row
    - Any number of errors in a single column
- Row and column parity bits

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

- **Correct <u>a</u>** single bit error

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

- **<u>Detect</u> any number of errors** in a single row or column (odd and even numbers)

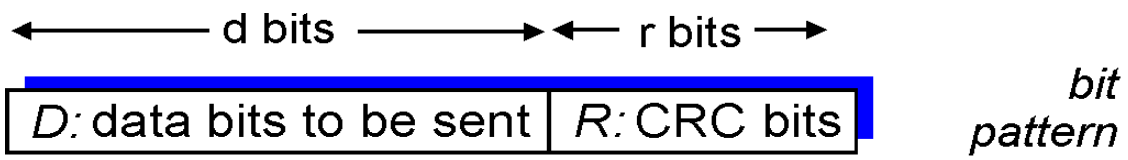
0 1 0 1 0 0 1	1
1 1 0 1 0 0 1	0
1 0 0 1 1 0 0	1
1 0 1 1 1 1 1	0
0 1 1 0 1 0 0	1
0 0 0 1 1 1 0	1
1 1 1 1 0 1 1	0

- But does it protect against everything?

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

Need something stronger. CRC codes

Works only for single ROW or Colum not MULTIPLE



$D * 2^r \quad XOR \quad R$ *mathematical formula*

Treat message as a polynomial degree (*d-1*): Message D(*x*)

Generator Polynomial G(x) is degree ***r***

Goal: choose ***r*** CRC bits, so we want ***R*** such that

- <D,R> exactly divisible by G (modulo 2)
- receiver knows G and divides <D,R> by G
- If NON-Zero reminder => Error

Choice of generator is very important as that defines the types of errors you can guard against.

- Maximize error-detecting capabilities
- Minimizing overall collision probabilities

$D \times 2^r$ XOR R = *n*G

$D \times 2^r$ = *n*G XOR R (Add mod 2 R to both sides – no carry)

$(D \times 2^r)/G$ = remainder

G(*x*) = 1101 = _____

D(*x*) = 10011010 = _____

Why is checksumming used at the transport layer and CRC used at the link layer?

Multiple access protocols (Taxonomy)

(1)

(2)

(3)

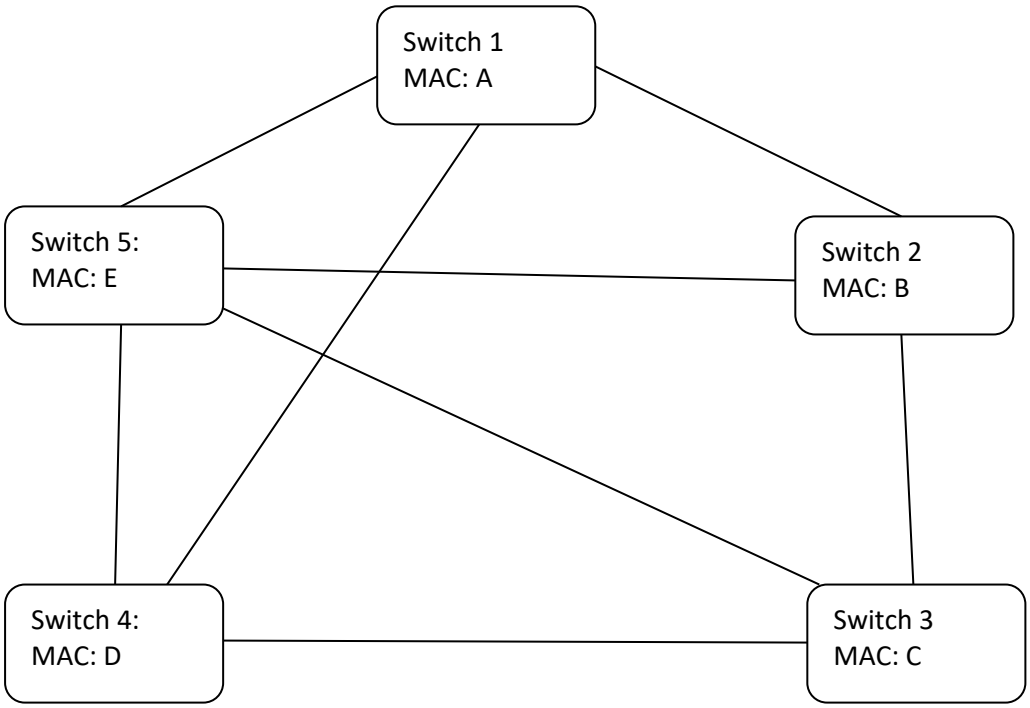
What is the problem with ALOHA and Slotted ALOHA?

Solution: Carrier Sense Multiple Access (CSMA)

(1)

(2)

Spanning Trees (link cost of 1 per link and same spanning tree priority)



Switch 1 pick the one with the smallest MAC address

Switch 1
MAC: A

Switch 5:
MAC: E

Switch 2
MAC: B

Switch 4:
MAC: D

Switch 3
MAC: C