Chapter 6
The Link Layer and LANs

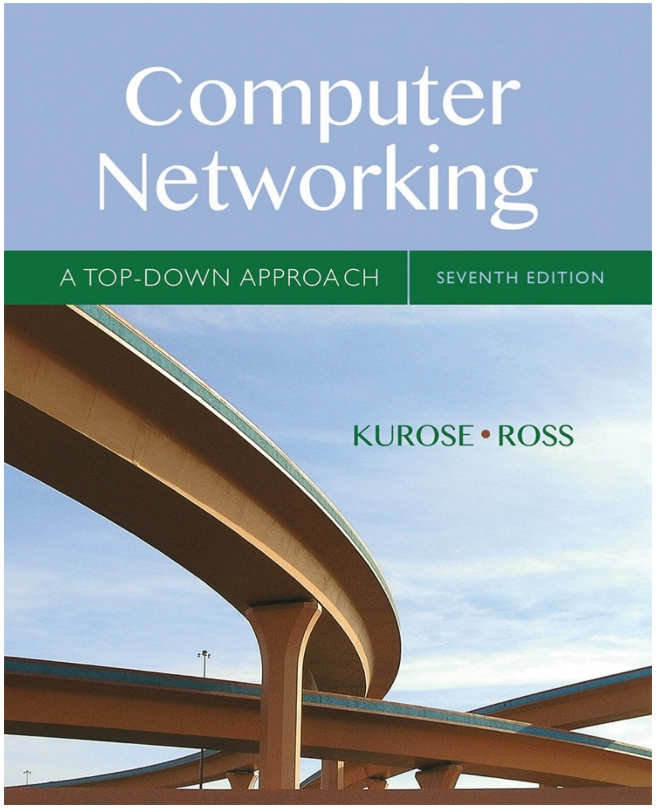
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Minor modifications made to original slides by Nathan Bowman

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

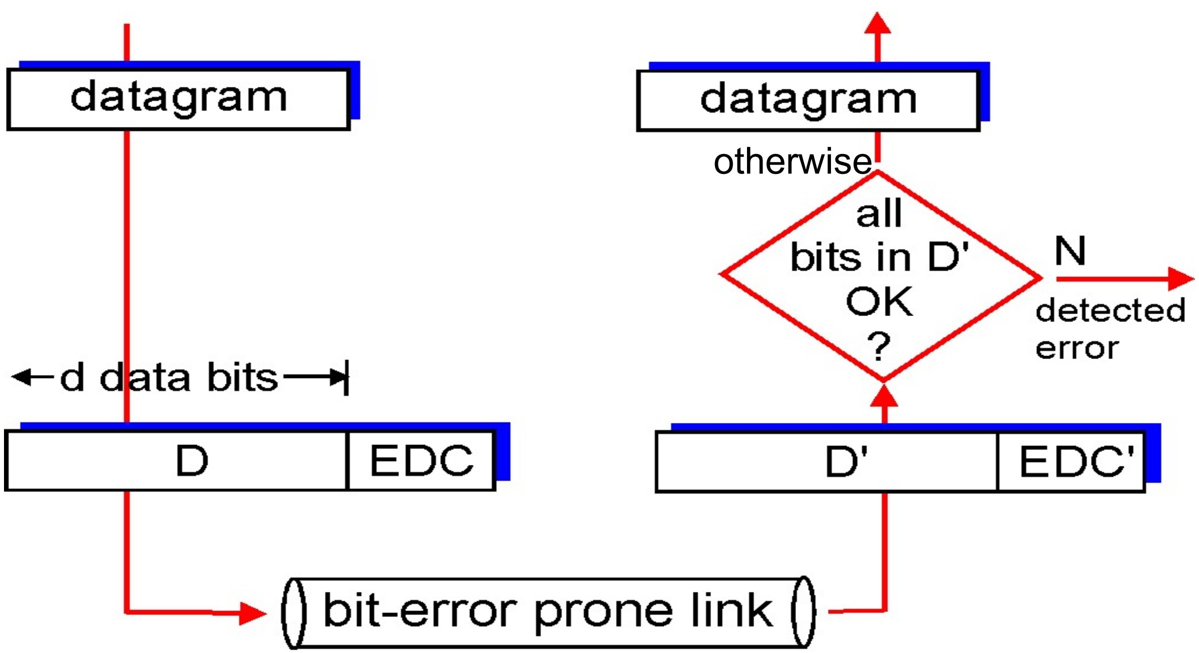
6.6 data center networking

6.7 a day in the life of a web request

Error detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

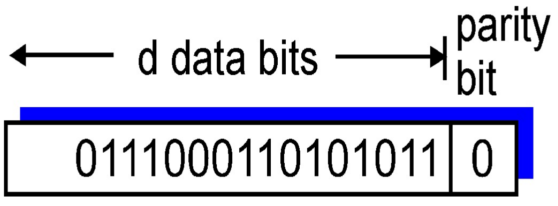
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity checking

single bit parity:

- *d*etect single bit errors



Add all bits in message modulo 2

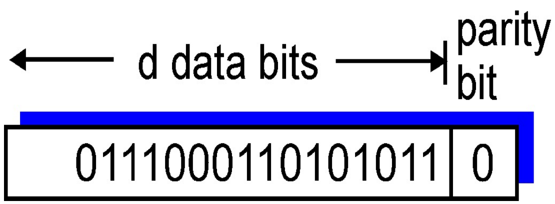
Parity bit appended to ensure sum is odd

(Could also be set to ensure sum is even – does not matter as long as sender and receiver are consistent)

Parity checking

single bit parity:

- *d*etect single bit errors



Receiver sums all bits, including parity bit, modulo 2

If sum is odd, message is accepted

Otherwise, error has occurred and message is discarded

Parity checking

Note the terminology: message is either discarded due to bit error, or it is accepted

Just because message is accepted does not necessarily mean it is correct

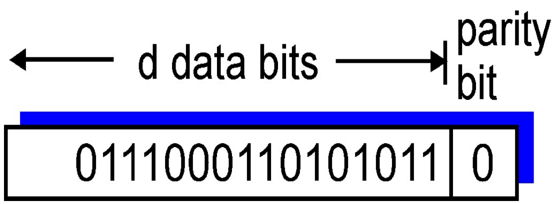
In general, schemes we examine can prove error occurred, but cannot prove error did *not* occur

Best we can do is have high degree of confidence

Parity checking

single bit parity:

- *d*etect single bit errors



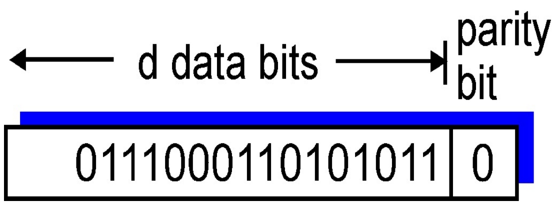
What if error occurs in parity bit itself, rather than message?

What if two bit errors occur? Three?

Parity checking

single bit parity:

- *d*etect single bit errors



Error in parity bit will cause message to be discarded

If two bit errors occur in message, they cancel one another out when taking modulo-2 sum

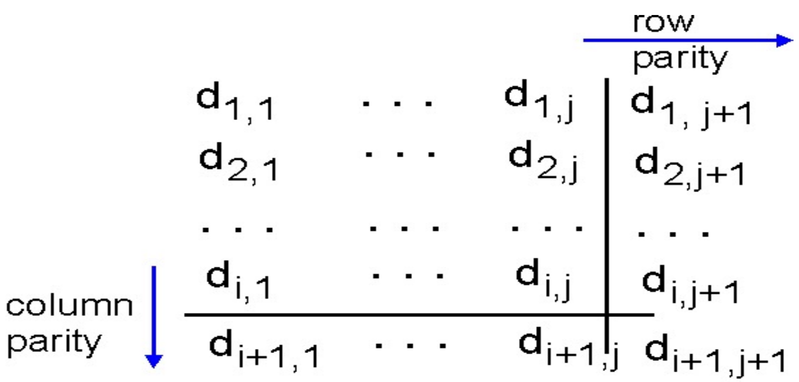
If three errors occur, it will be detected

In general, even numbers of errors are not detected

Parity checking

two-dimensional bit parity:

- detect and correct single bit errors



Arrange message in i x j grid of bits

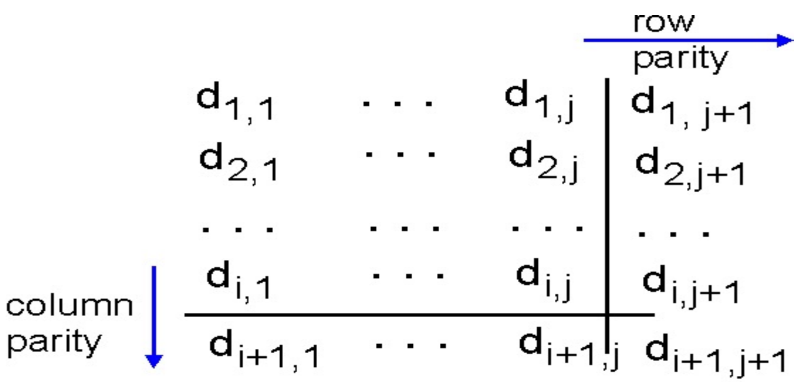
Compute parity of each row and column separately, and one additional parity for new row and column

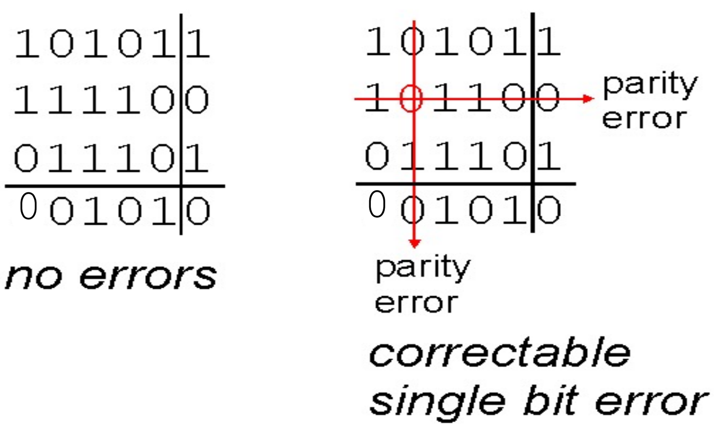
Requires more overhead:
i + j + 1 additional bits

Parity checking

two-dimensional bit parity:

- detect and correct single bit errors





Allows for *correction* of single bit errors

Can detect, but not correct, errors of two bits

Internet checksum (review)

goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

sender:

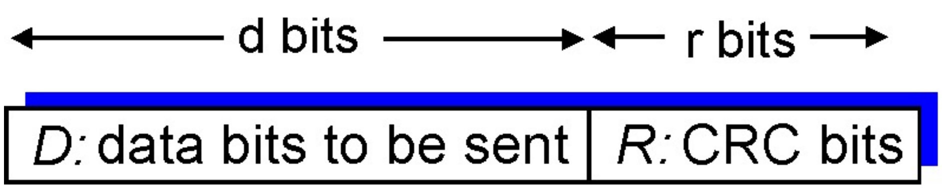
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless*

Cyclic redundancy check

- more powerful error-detection coding
- 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 (Ethernet, 802.11 WiFi, ATM)



bit pattern

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

CRC example

want:

$D \cdot 2^r$ XOR $R = nG$

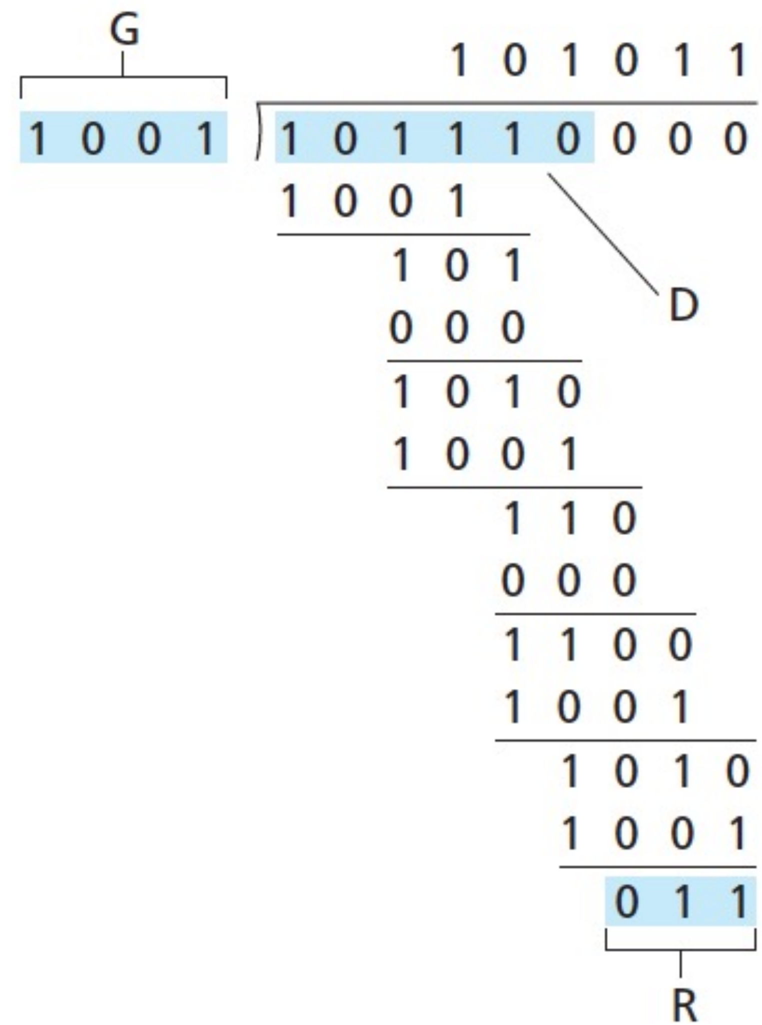
equivalently:

$D \cdot 2^r = nG$ XOR R

equivalently:

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

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

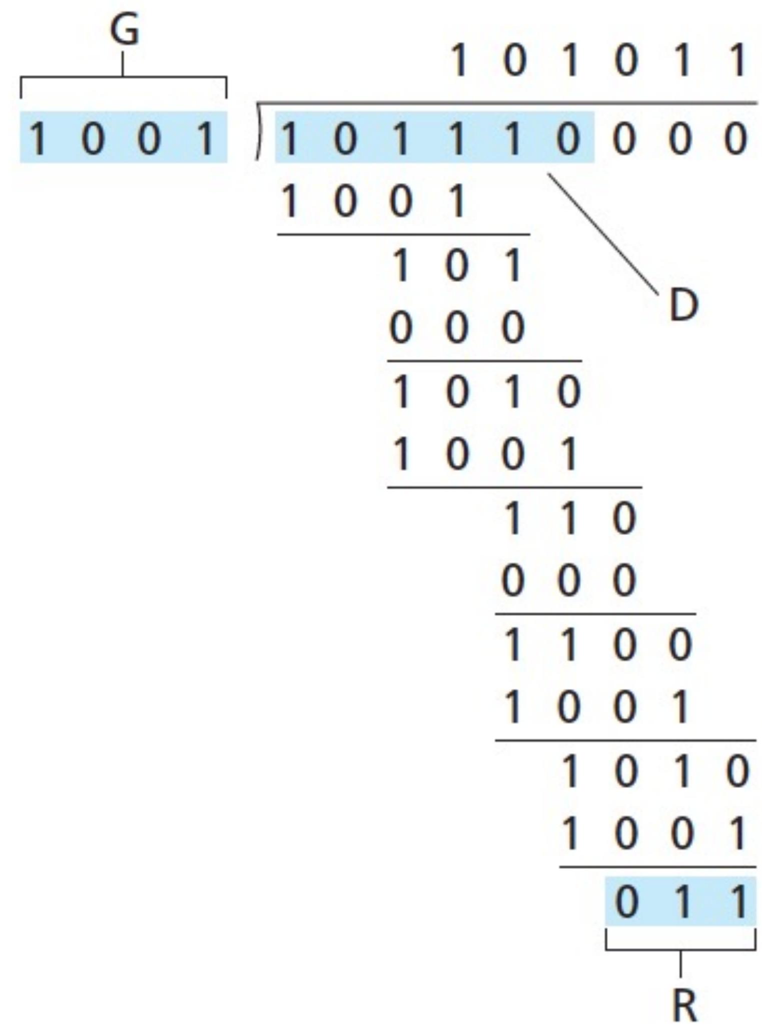
CRC example

Computation was performed at sender to determine R

Sender appends R to D:

101110011

Dividing 1001 into 101110011 produces remainder of 0 (by design)

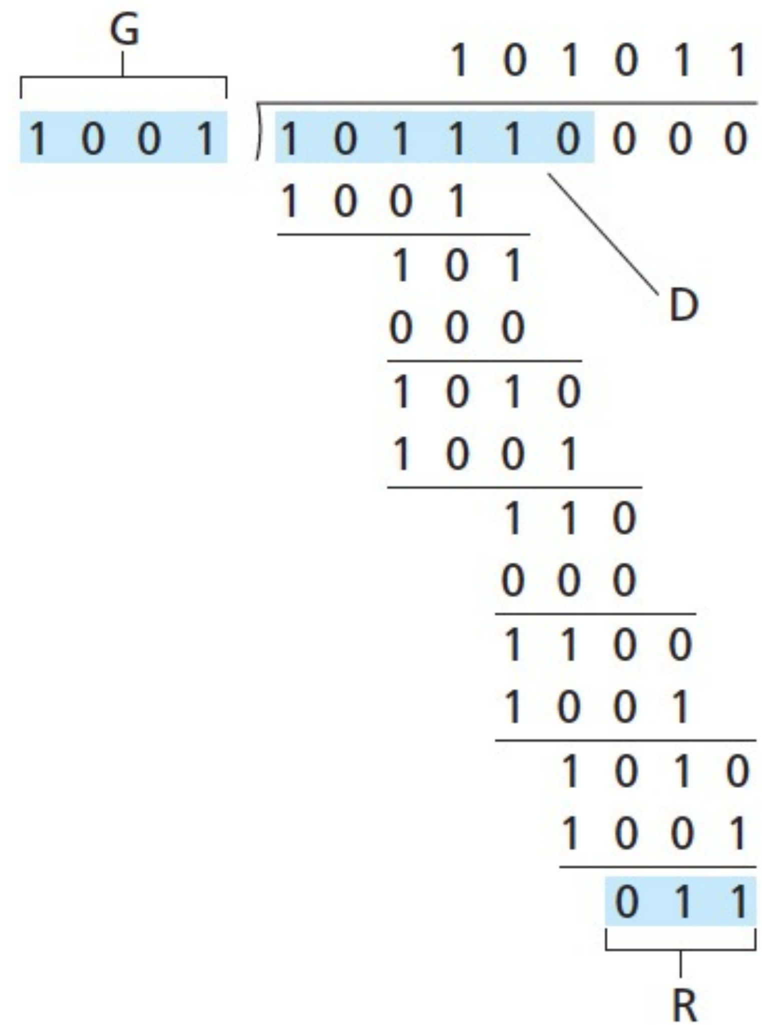


* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

CRC example

Receiver uses same G to verify D' using same process of division

If remainder is nonzero, error has occured



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/