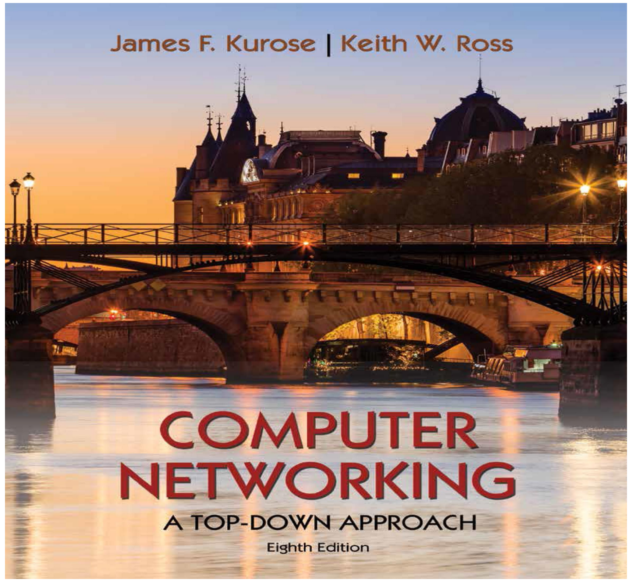
The Link Layer

Computer Networks

- Introduction to the Link Layer
- Error-detection and -correction Techniques
- Multiple Access Links and Protocols
- Switched Local Area Networks
- Link Virtualization: a Network as a Link Layer
- Data Center Networking
- Retrospective: A Day in the Life of a Web Page Request



Networks must be able to transfer data from one device to another with acceptable accuracy

Data can be corrupted during transmission

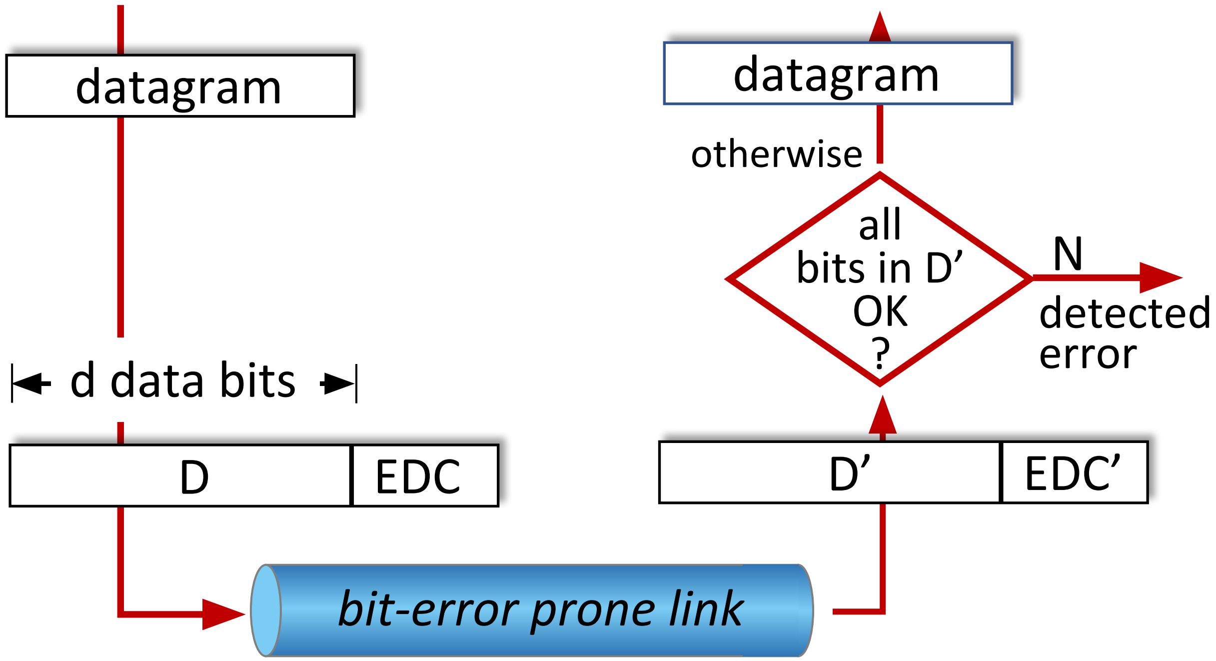
Some application can tolerate a small level of error such as random errors in audio or video transmission
But transmission of text requires very high level of accuracy

Thus, some applications require that errors be detected and corrected

In order to cope with data transmission errors
Error detection and correction bits

Error detection

EDC: error detection and correction bits (e.g., 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:

- detect single bit errors

0111000110101011	1

← *d* data bits → | parity bit

Parity bit = 1 when: Number of 1s in data is odd
Parity bit = 0 when: Number of 1s in data is even

Even parity: set parity bit so there is an even number of 1's

At receiver:

- compute parity of *d+1* received bits, if not even, then error detected
- can detect odd number of bit flips

Two-D parity:

row parity →

column parity ↓

$d_{1,1}$	...	$d_{1,j}$	$d_{1,j+1}$
$d_{2,1}$	...	$d_{2,j}$	$d_{2,j+1}$
...	...	...	...
$d_{i,1}$	...	$d_{i,j}$	$d_{i,j+1}$
$d_{i+1,1}$	...	$d_{i+1,j}$	$d_{i+1,j+1}$

- detect two-bit errors
- detect *and correct* single bit errors without retransmission!

no errors:

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

detected and correctable single-bit error:

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

parity error

parity error

Internet checksum (review, see section 3.3)

Goal: detect errors (*i.e.*, flipped bits) in transmitted segment

sender:

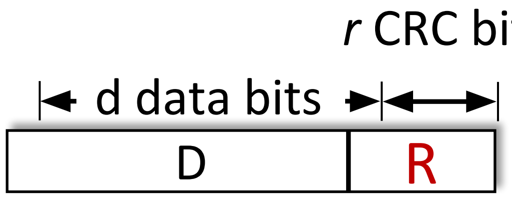
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - not equal - error detected
  - equal - no error detected. *But maybe errors nonetheless?* More later ....

Cyclic Redundancy Check (CRC)

- more powerful error-detection coding
- D: data bits (given, think of these as a binary number)
- G: bit pattern (generator), of *r+1* bits (given, specified in CRC standard)



bits to send

<D,R> = $D * 2^r$ XOR R

formula for these bits

sender: compute *r* CRC bits, R, such that <D,R> *exactly* divisible by G (mod 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)

Cyclic Redundancy Check (CRC): example

Sender wants to compute R such that:

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

... or equivalently (XOR R both sides):

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

... which says:

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

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

algorithm for computing R

G

```
                1 0 1 0 1 1
1 0 0 1 ) 1 0 1 1 1 0 0 0 0
          1 0 0 1
              1 0 1
              0 0 0
              1 0 1 0
              1 0 0 1
                  1 1 0
                  0 0 0
                  1 1 0 0
                  1 0 0 1
                    1 0 1 0
                    1 0 0 1
                      0 1 1
```

$D * 2^r$ (here, r=3)

R