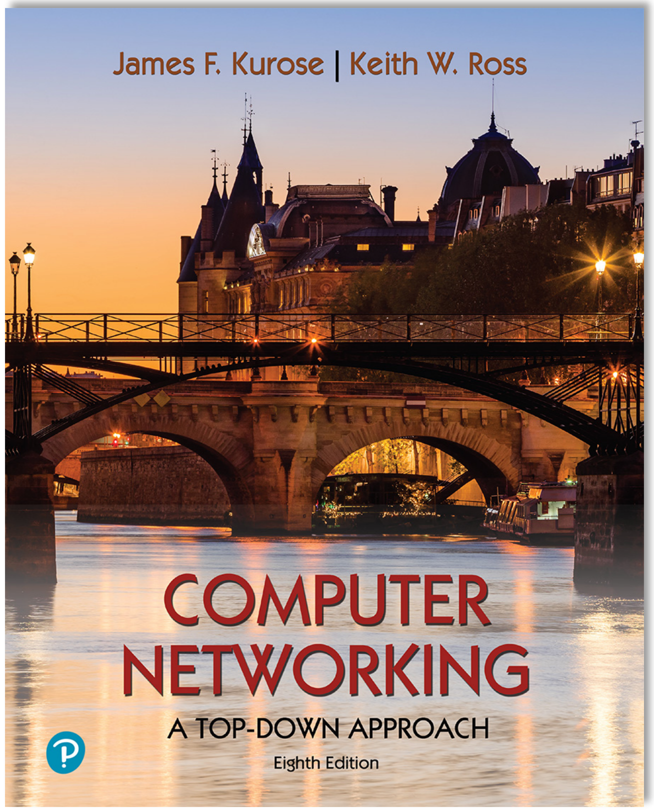
Chapter 6
The Link Layer and LANs



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

Link layer and LANs: our goals

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- datacenter networks

- instantiation, implementation of various link layer technologies

Link layer, LANs: roadmap

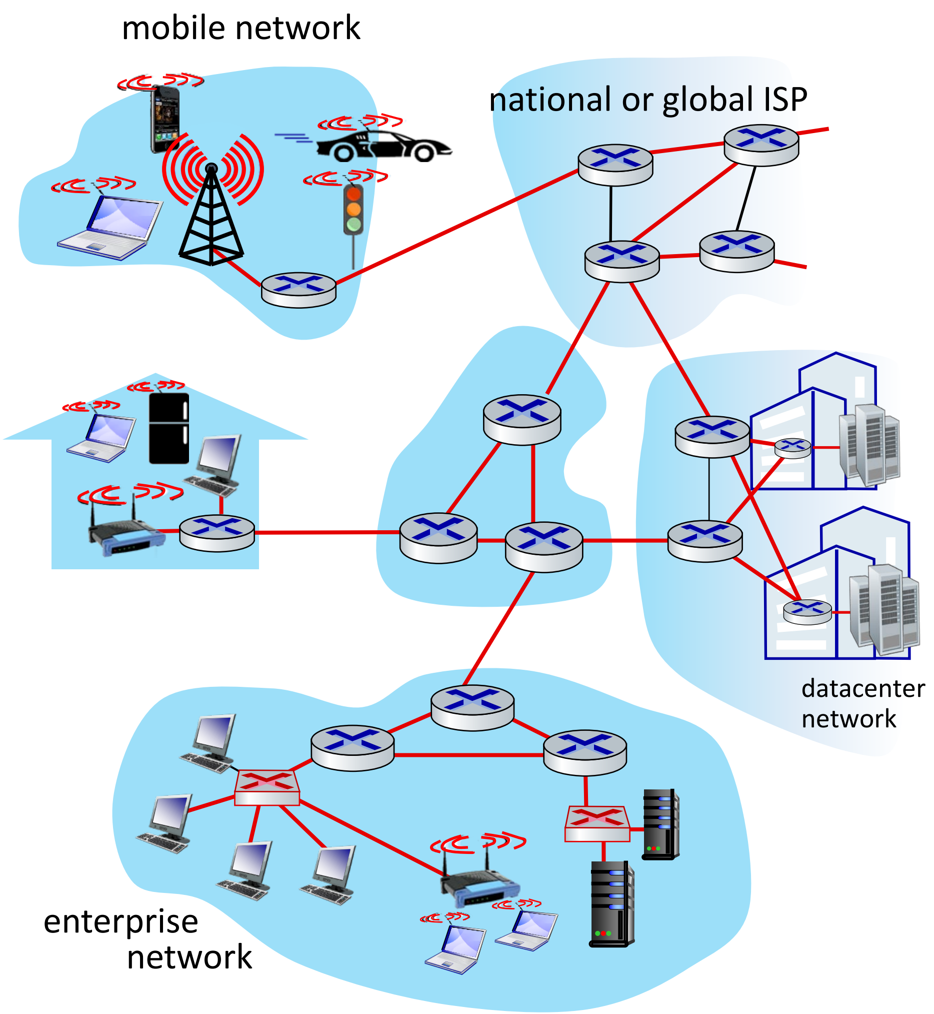
- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

Link layer: introduction

terminology:

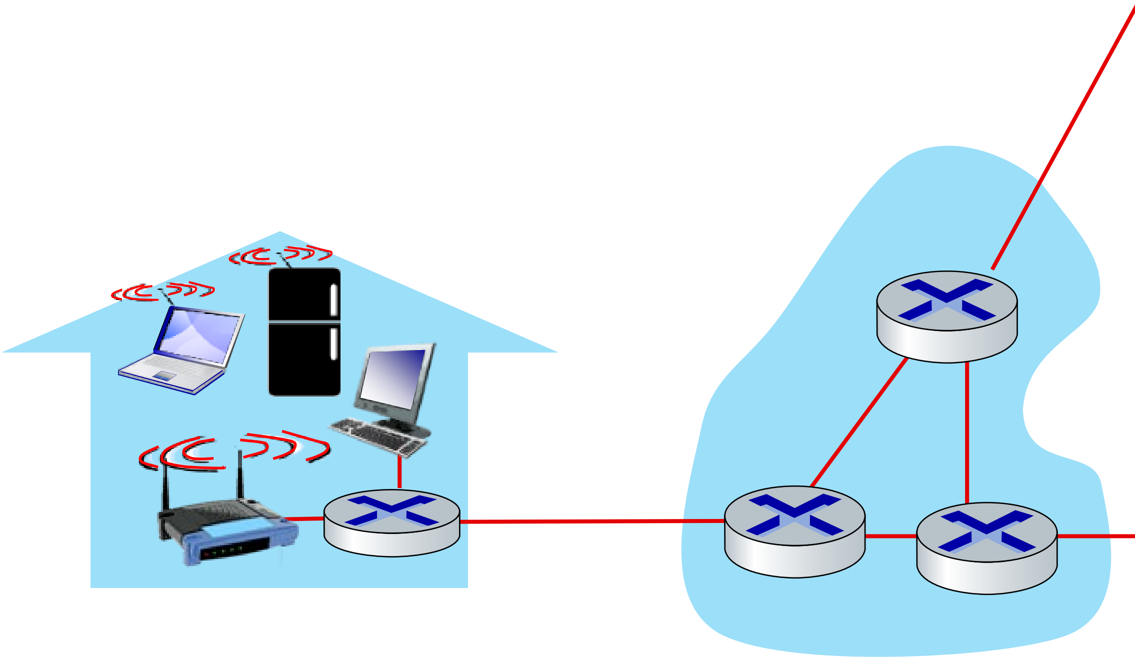
- hosts, routers: **nodes**
- communication channels that connect adjacent nodes along communication path: **links**
  - wired , wireless
  - LANs
- layer-2 packet: ***frame***, encapsulates datagram



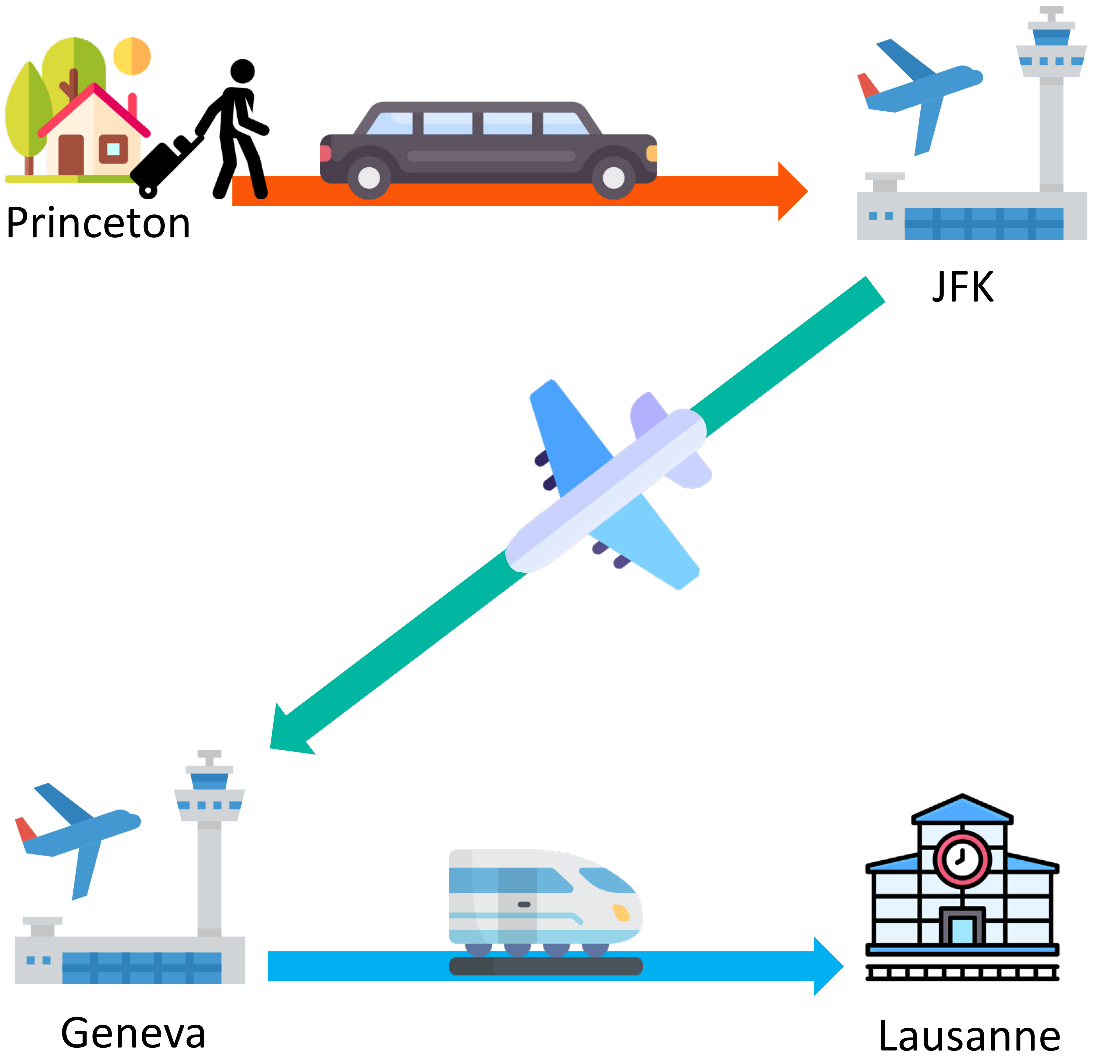
link layer has responsibility of transferring datagram from one node to physically adjacent node over a link

Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., WiFi on first link, Ethernet on next link
- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link



Transportation analogy

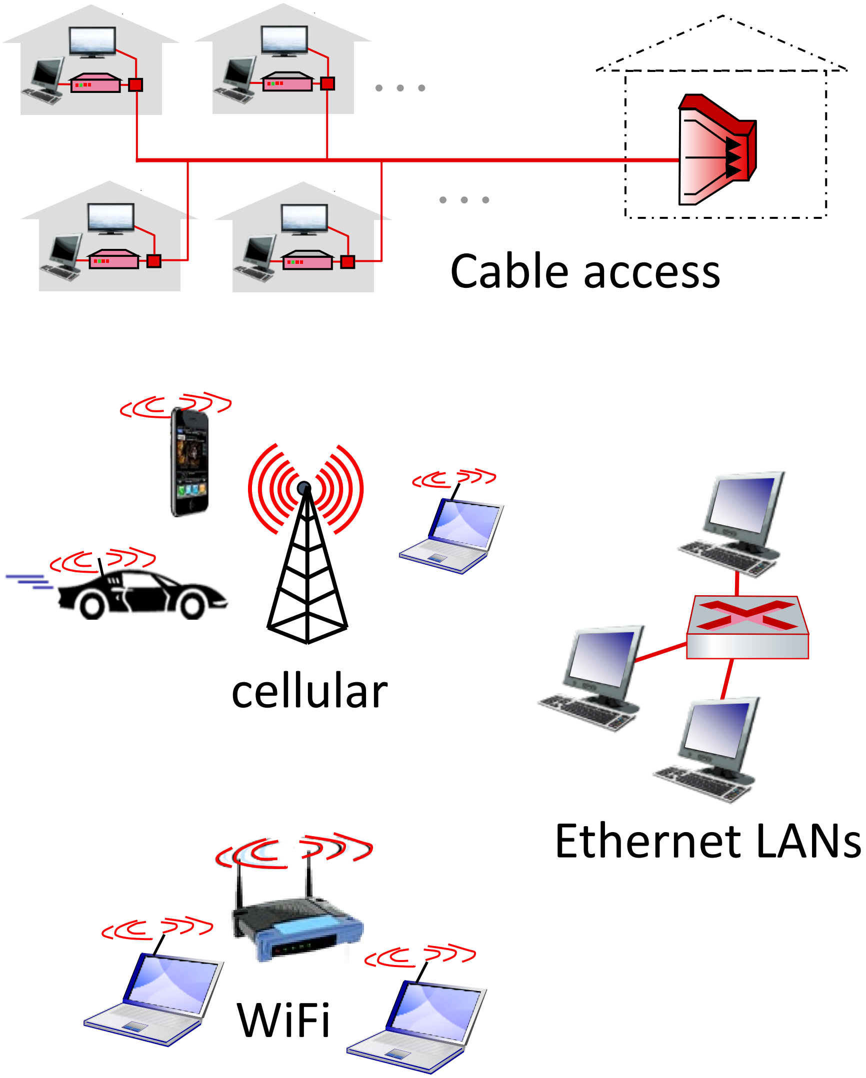


transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link-layer protocol
- travel agent = routing algorithm

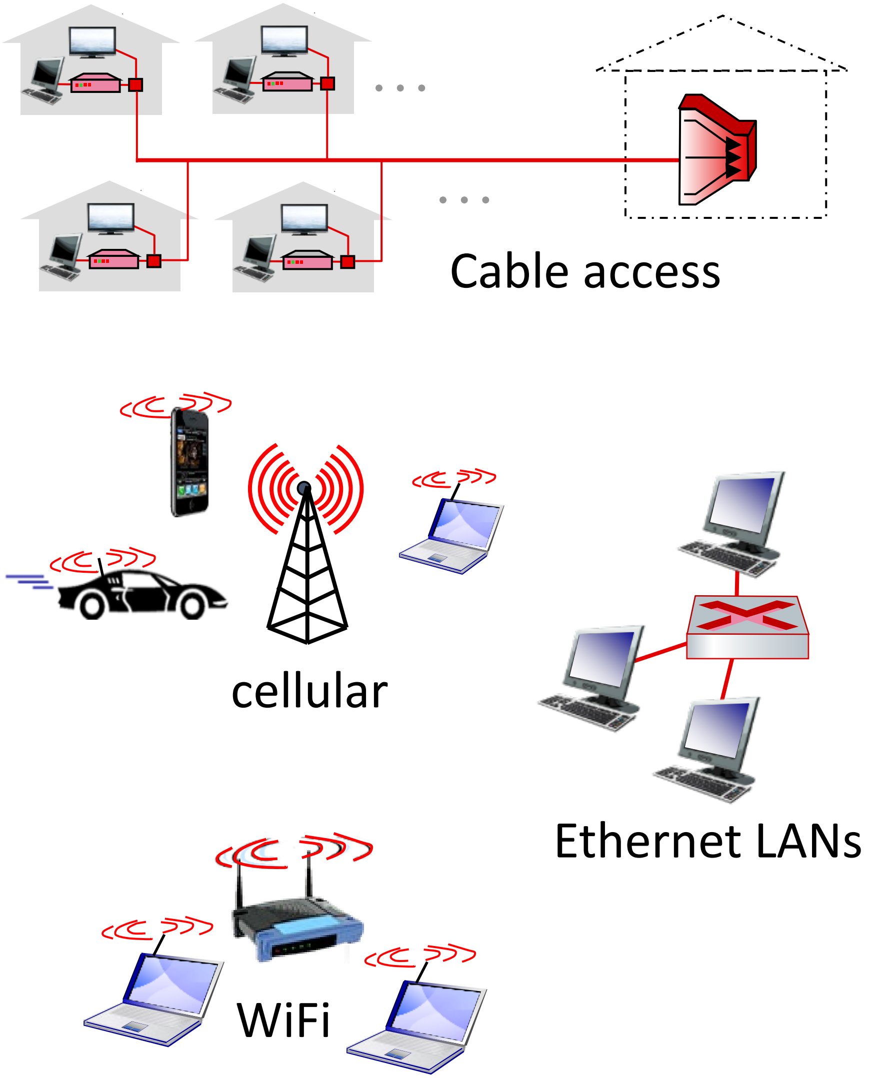
Link layer: services

- framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses in frame headers identify source, destination (different from IP address!)
- reliable delivery between adjacent nodes
  - we already know how to do this!
  - seldom used on low bit-error links
  - wireless links: high error rates
    - ***Q:*** **why both link-level and end-end reliability?**



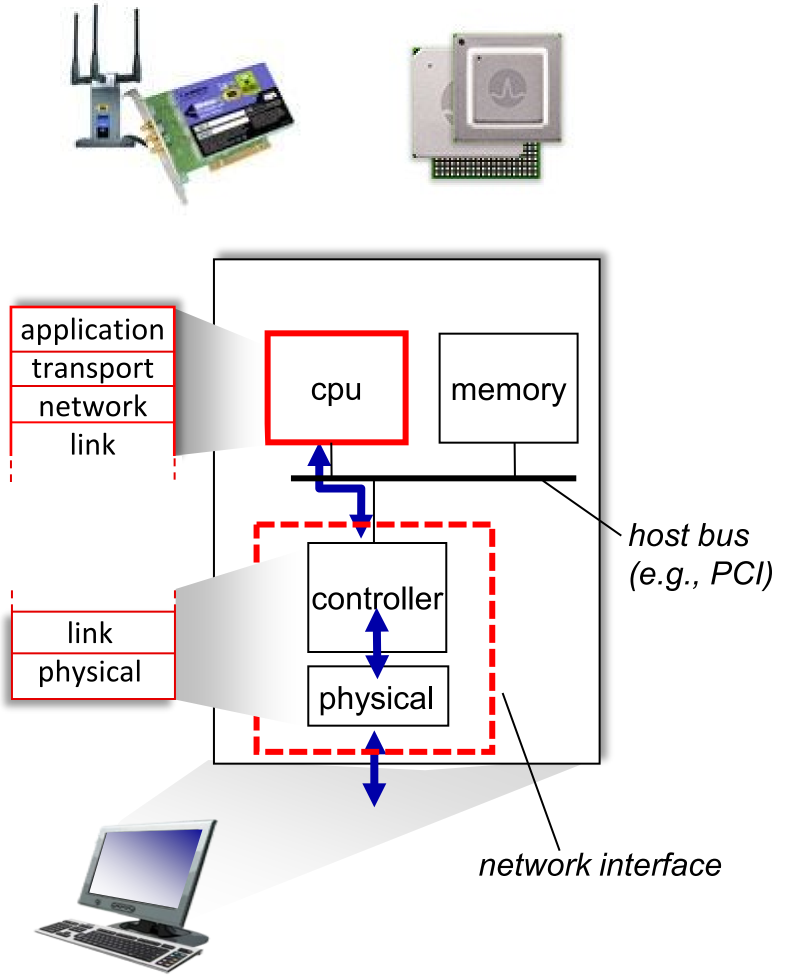
Link layer: services (more)

- **flow control:**
  - pacing between adjacent sending and receiving nodes
- **error detection:**
  - errors caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- **error correction:**
  - receiver identifies *and corrects* bit error(s) without retransmission
- **half-duplex and full-duplex:**
  - with half duplex, nodes at both ends of link can transmit, but not at same time

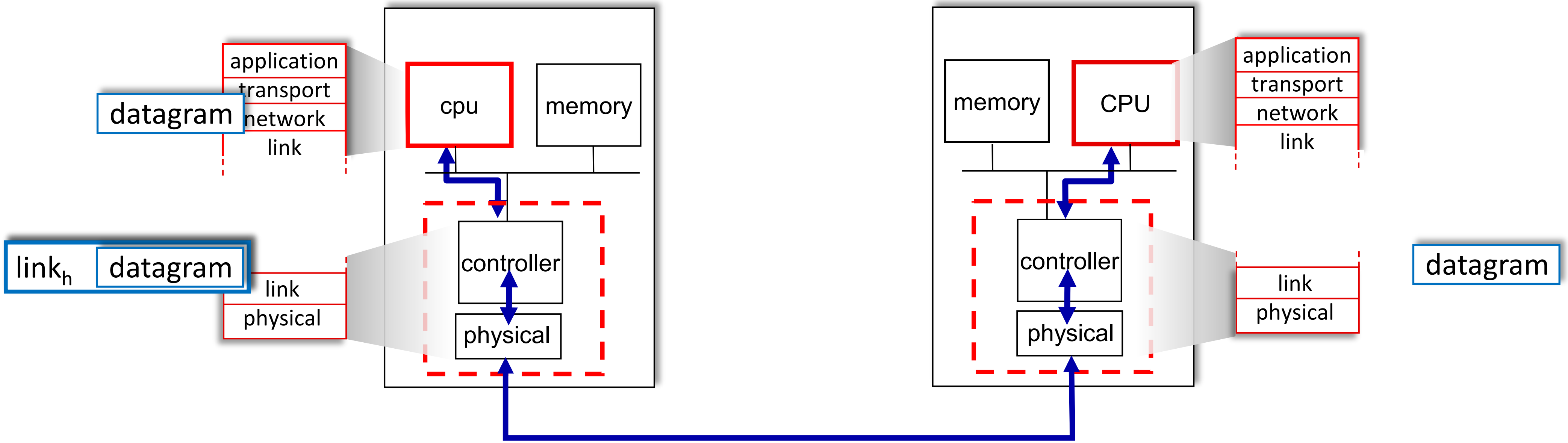


Host link-layer implementation

- in each-and-every host
- link layer implemented on-chip or in network interface card (NIC)
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Interfaces communicating



sending side:

- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

receiving side:

- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

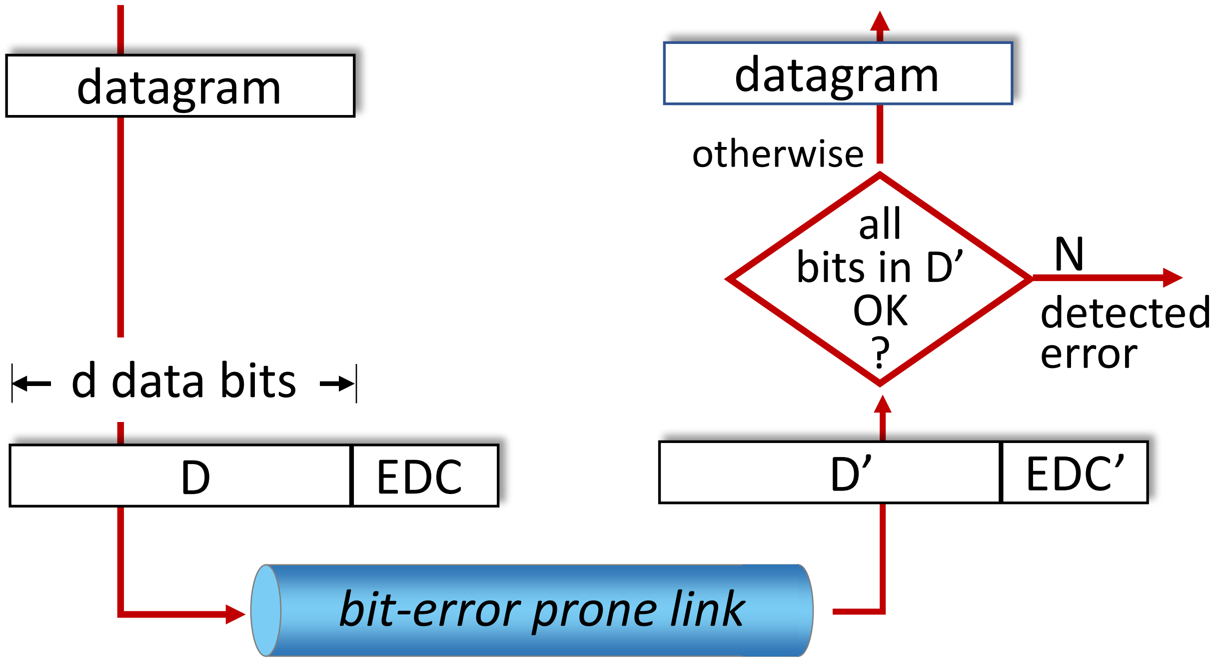
Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

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

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

At receiver:

- compute parity of *d* received bits
- compare with received parity bit – if different than error detected

Can detect *and* correct errors (without retransmission!)

- two-dimensional parity: detect *and correct* single bit errors

row parity

column parity

$$\begin{array}{cccc|c} d_{1,1} & \cdots & d_{1,j} & & d_{1,j+1} \\ d_{2,1} & \cdots & d_{2,j} & & d_{2,j+1} \\ \cdots & \cdots & \cdots & & \cdots \\ d_{i,1} & \cdots & d_{i,j} & & d_{i,j+1} \\ \hline d_{i+1,1} & \cdots & d_{i+1,j} & & d_{i+1,j+1} \end{array}$$

no errors:

$$\begin{array}{ccccc|c} 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

detected and correctable single-bit error:

$$\begin{array}{ccccc|c} 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

parity error

parity error

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

Internet checksum (review, see section 3.3)

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

sender:

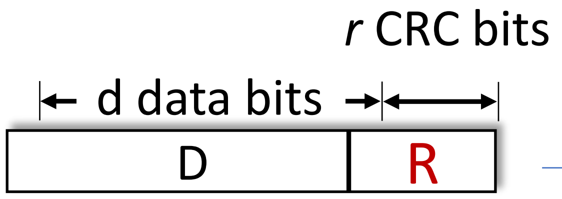
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

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

$\langle D,R \rangle = D * 2^r \text{ XOR } R$

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

```
            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
```

G = 1 0 0 1

$D * 2^r$ = 1 0 1 1 1 0 0 0 0 (here, r=3)

R = 0 1 1

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