Lecture 2: Layering & End-to-End

CSE 222A: Computer Communication Networks

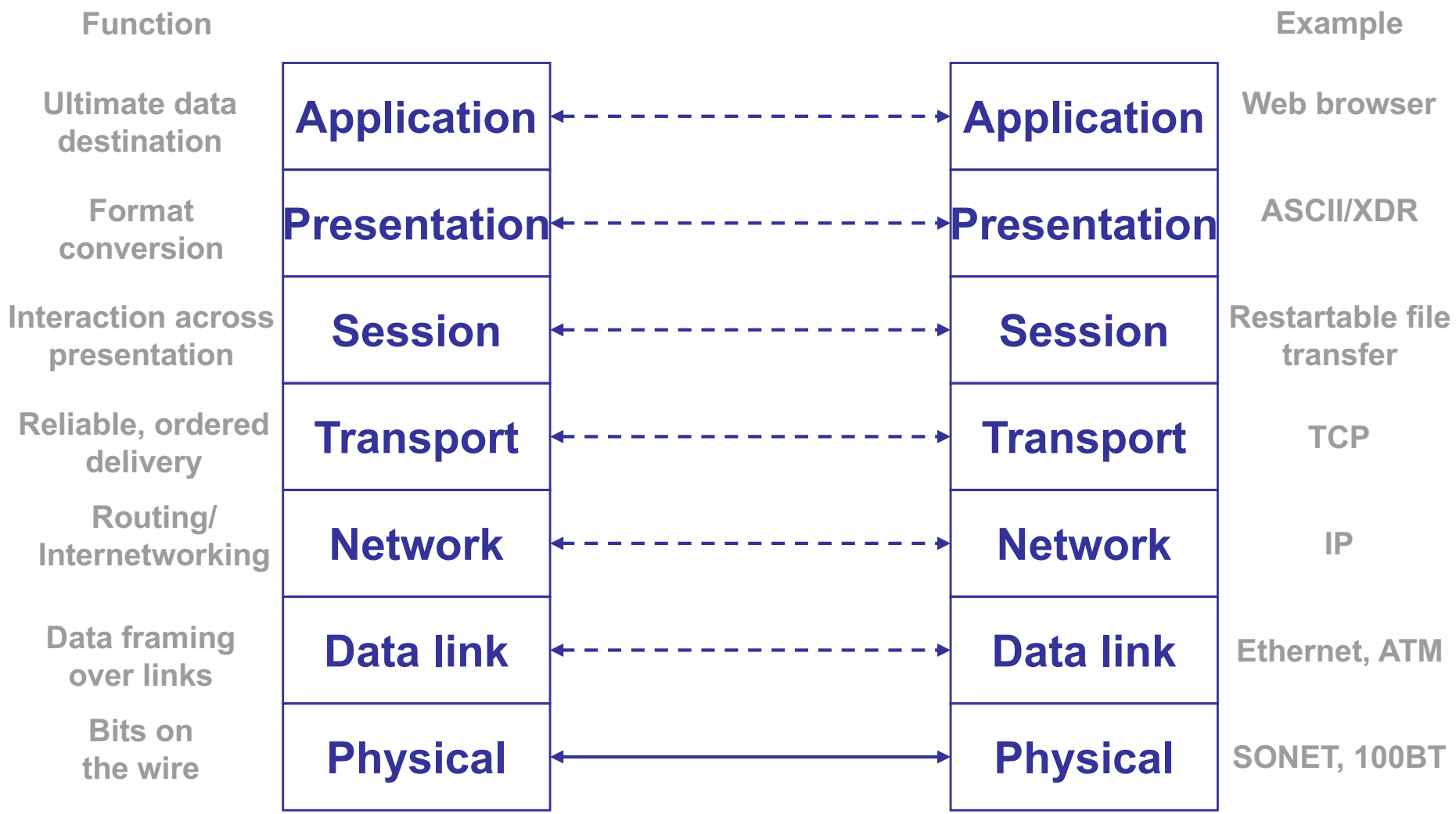
Alex C. Snoeren

Thanks: Mike Freedman & Amin Vahdat

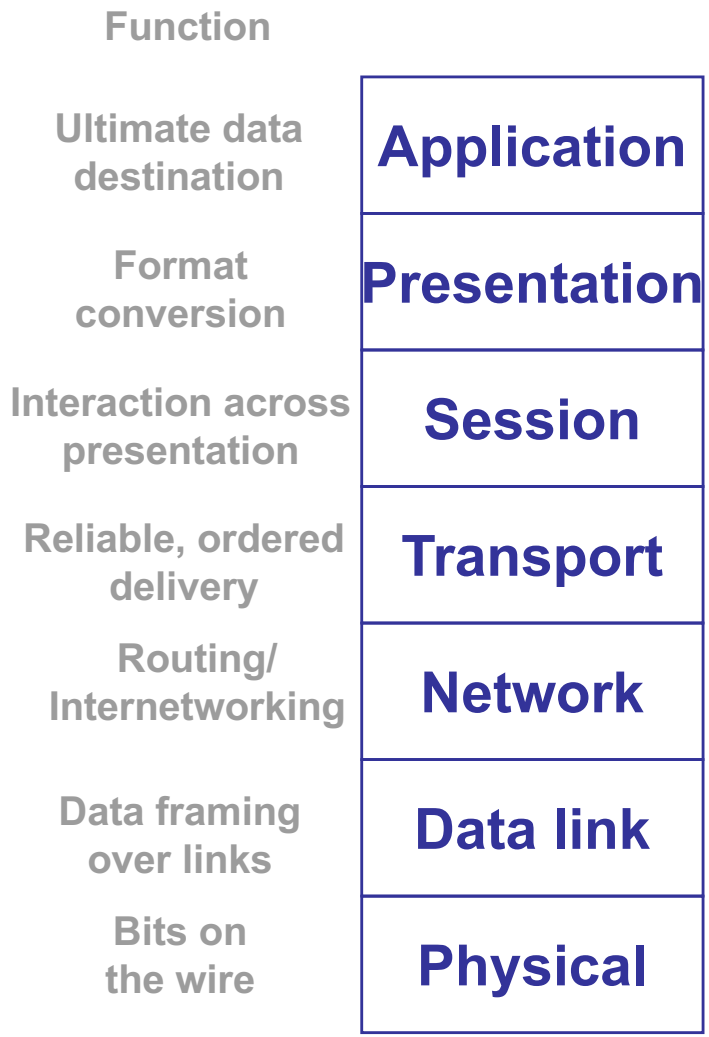
Lecture 2 Overview

- Layering
  - Application interface
  - Transport services
- Discussion of End-to-End principle

OSI Model


Function
Example
Ultimate data destination
Application
Application
Web browser
Format conversion
Presentation
Presentation
ASCII/XDR
Interaction across presentation
Session
Session
Restartable file transfer
Reliable, ordered delivery
Transport
Transport
TCP
Routing/ Internetworking
Network
Network
IP
Data framing over links
Data link
Data link
Ethernet, ATM
Bits on the wire
Physical
Physical
SONET, 100BT

OSI Model



Where does security go?

What about reliability?

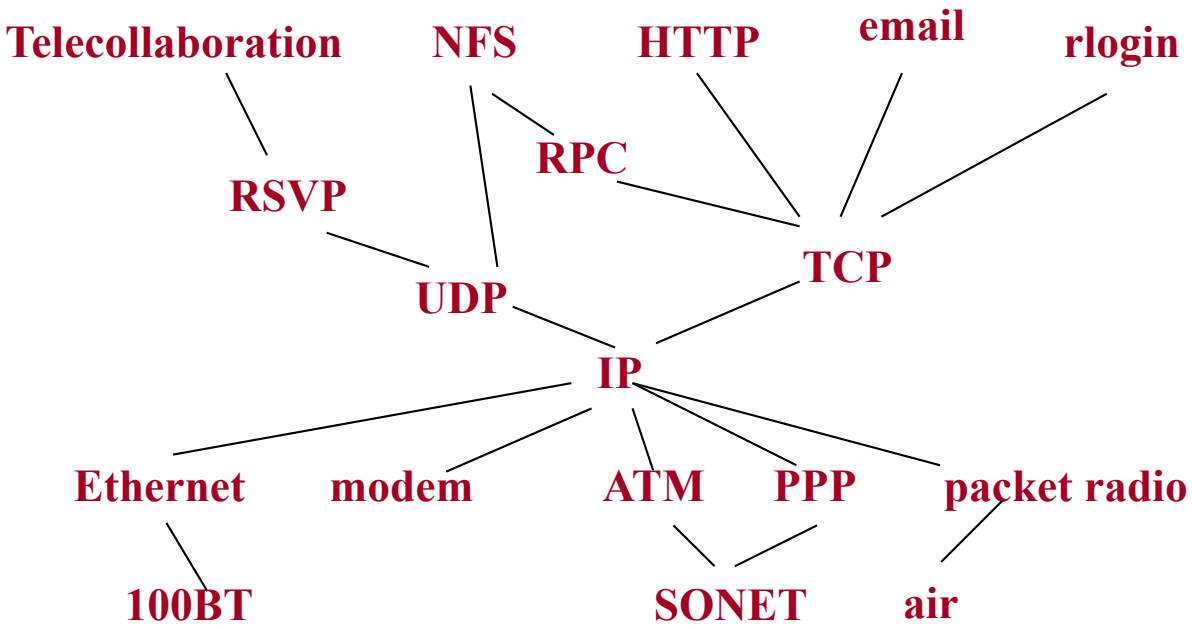
OSI Model Discussion

- OSI standardized before implemented
  - IETF philosophy: “We reject kings, presidents and voting. We believe in rough consensus and working code”
  - IETF requires two working/interoperable versions before considering a standard
- Modular design, but some boundaries are arbitrary
  - Why seven layers?
  - What exactly is the session layer?
  - Much basic network functionality at multiple layers

    Reliability, flow control, security

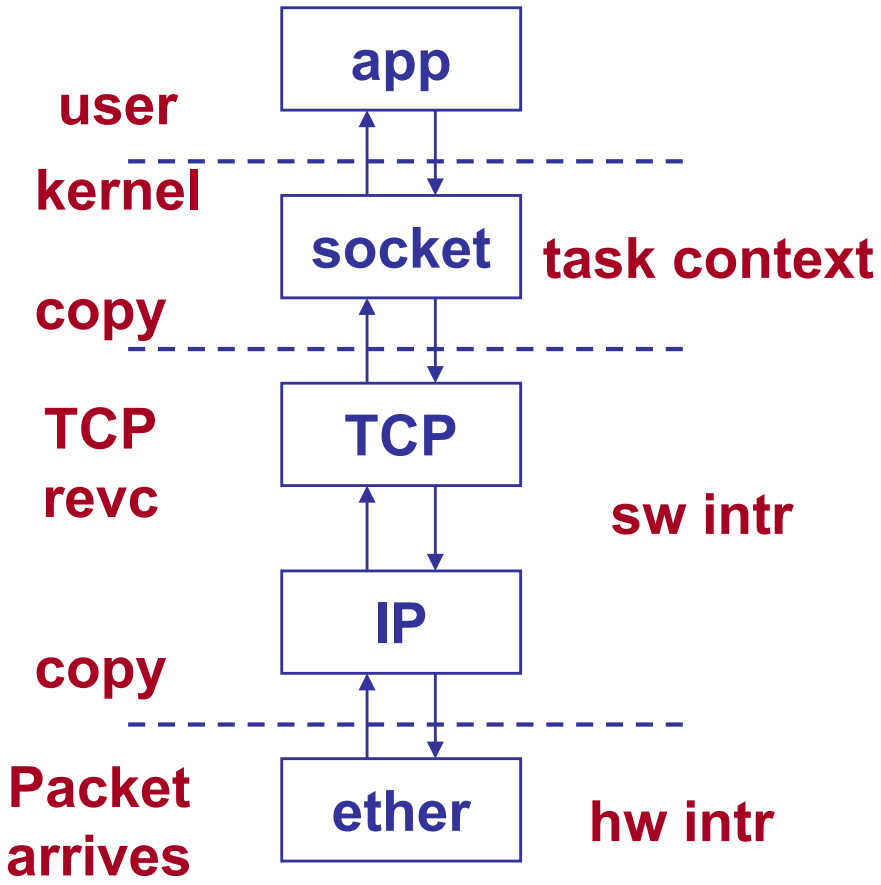
Internet Architecture

- IP Hourglass:



- Layering not strict
  - Can define new abstractions on any existing protocol

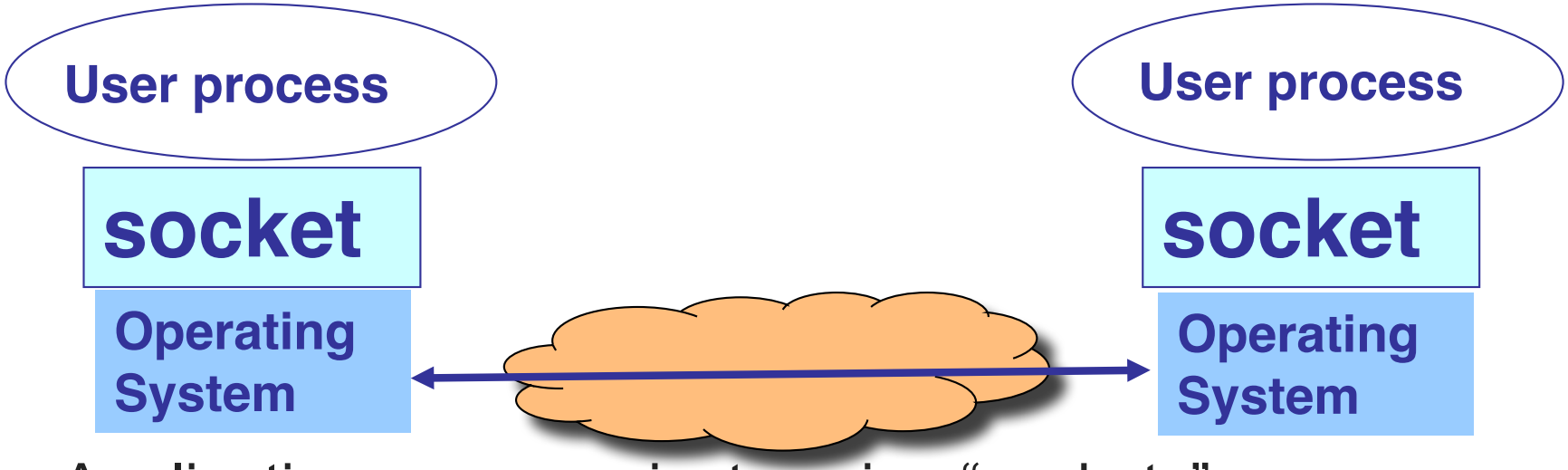
Layering in Applications



- Bottlenecks
  - Boundary crossings
  - Copies
  - Context switches
- Layering nice way to logically consider protocols
  - May not lead to fastest implementation
  - But! Processors are getting faster… people are getting more expensive

Socket Abstraction

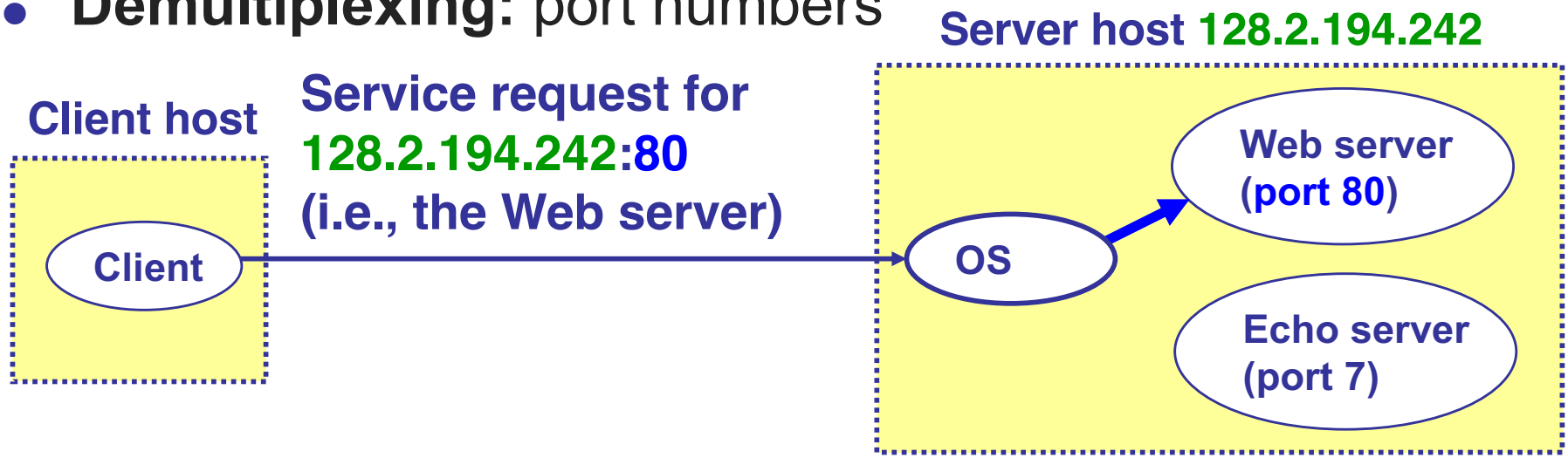
- Best-effort packet delivery is a clumsy abstraction
  - Applications typically want higher-level abstractions
  - Messages, uncorrupted data, reliable in-order delivery



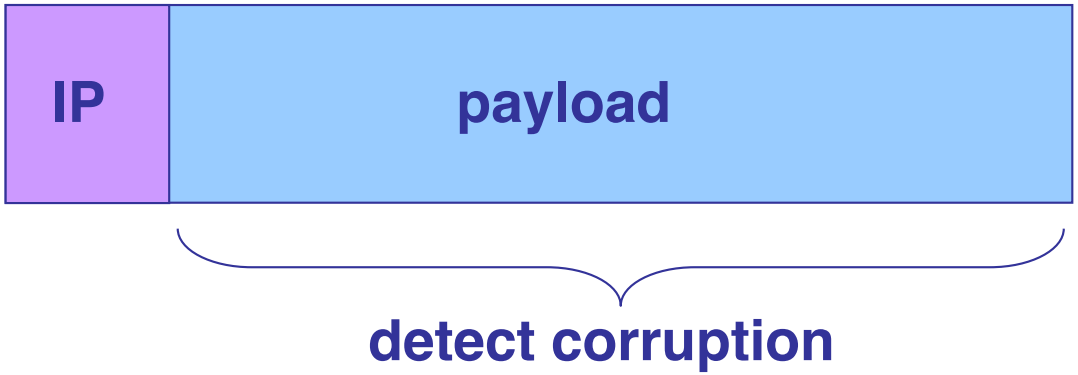
- Applications communicate using “sockets”
  - Stream socket: reliable stream of bytes (like a file)
  - Message socket: unreliable message delivery

Two Basic Transport Features

- **Demultiplexing:** port numbers



- **Error detection:** checksums



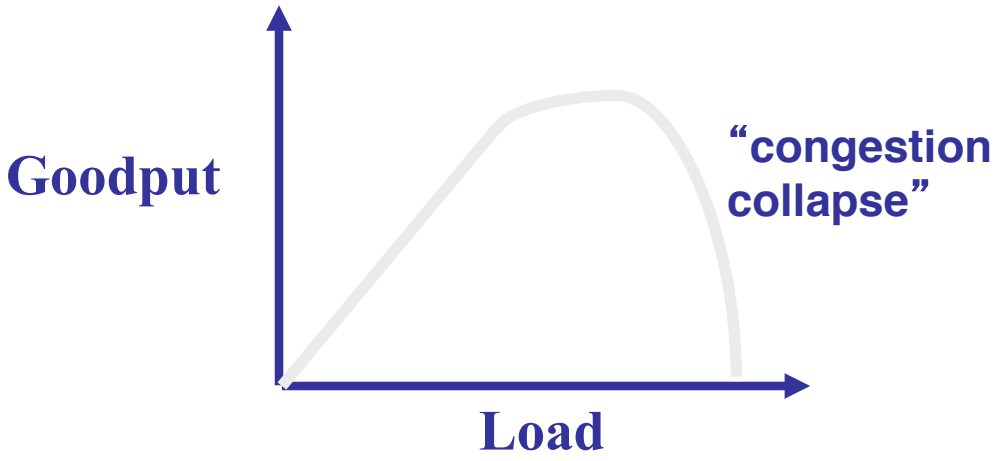
Two Main Transport Layers

- User Datagram Protocol (UDP)
  - Just provides demultiplexing and error detection
  - Header fields: port numbers, checksum, and length
  - Low overhead, good for query/response and multimedia

- Transmission Control Protocol (TCP)
  - Adds support for a “stream of bytes” abstraction
  - Retransmitting lost or corrupted data
  - Putting out-of-order data back in order
  - Preventing overflow of the receiver buffer
  - Adapting the sending rate to alleviate congestion
  - Higher overhead, good for most stateful applications

Sharing the 'Net

- Best-effort network easily becomes overloaded
  - No mechanism to "block" excess calls
  - Instead excess packets are simply dropped
- Examples
  - Shared Ethernet medium: frame collisions
  - Ethernet switches and IP routers: full packet buffers
- Quickly leads to congestion collapse

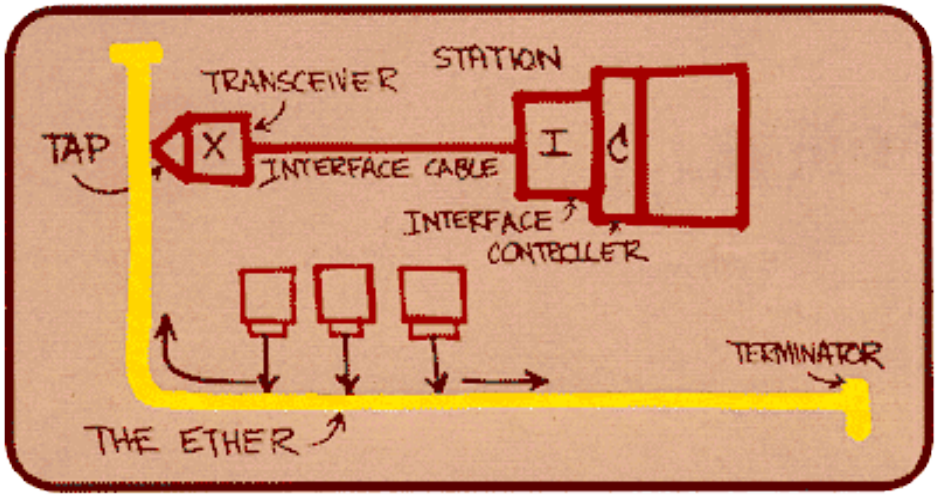


Increase in load that results in a *decrease* in useful work done.

Adjusting to Congestion

- End hosts adapt their sending rates
  - In response to network conditions
- Learning that the network is congested
  - Shared Ethernet: carrier sense multiple access
    - » Seeing your own frame collide with others
  - IP network: observing your end-to-end performance
    - » Packet delay or loss over the end-to-end path
- Adapting to congestion
  - Slowing down the sending rate, for the greater good
  - But, host doesn't know how bad things might be…

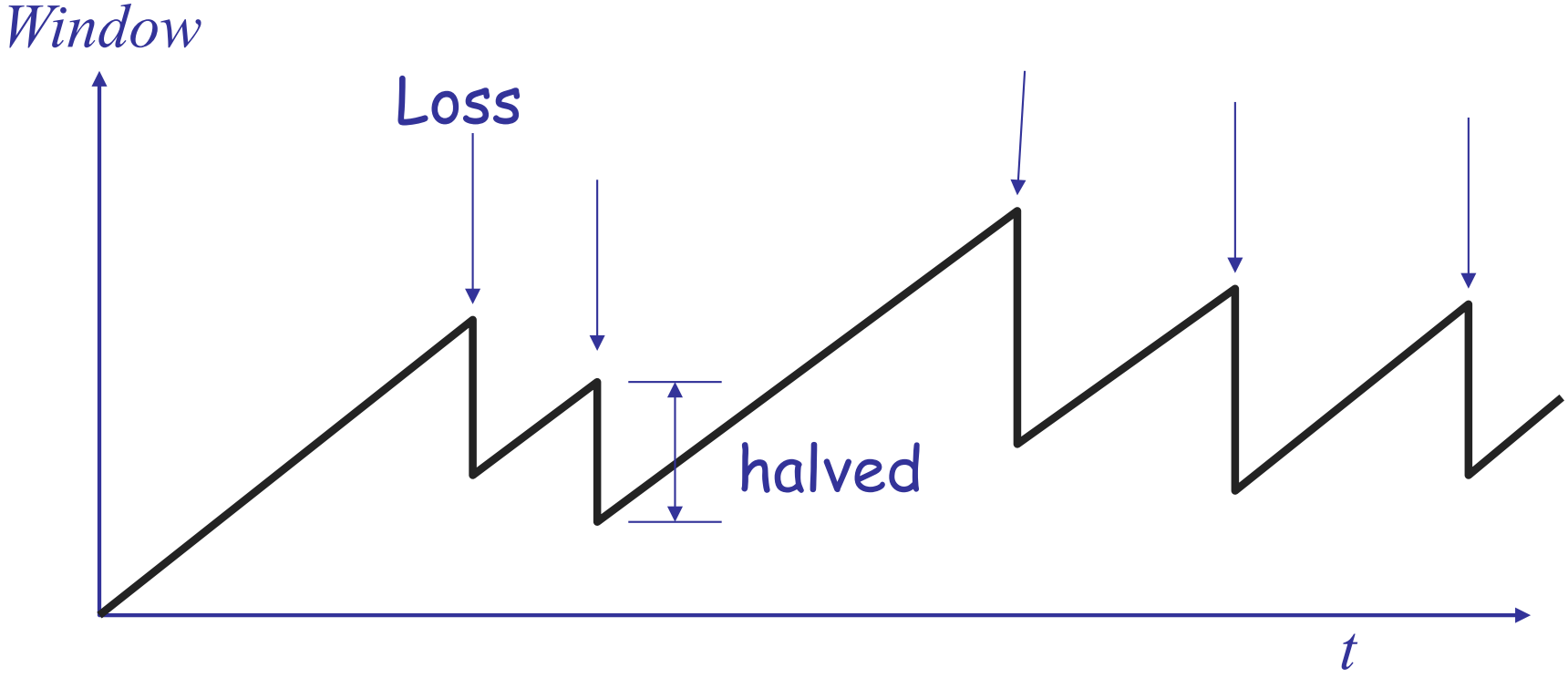
Ethernet Back-off Mechanism



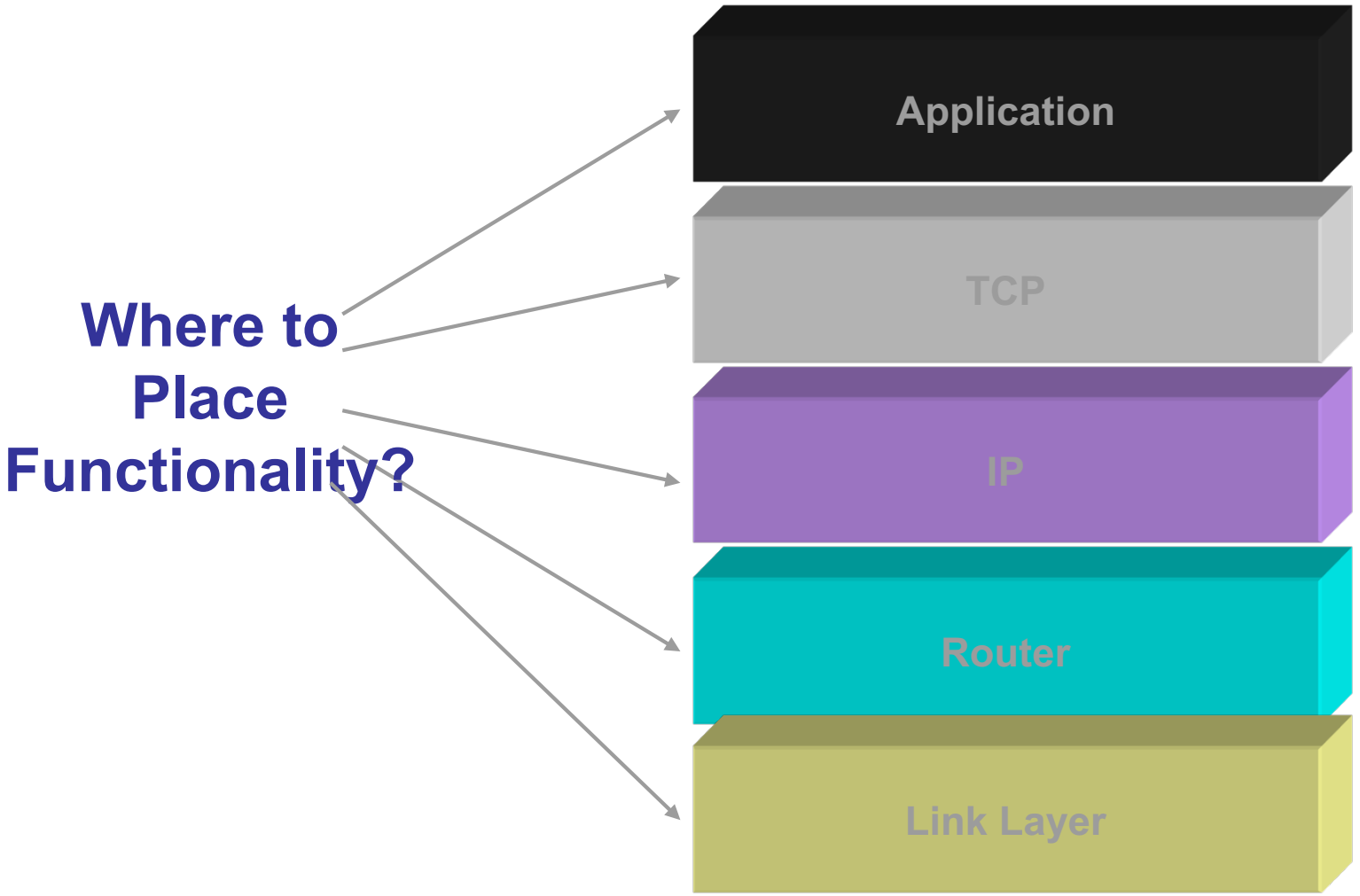
- Carrier sense: wait for link to be idle
  - If idle, start sending; if not, wait until idle
- Collision detection: listen while transmitting
  - If collision: abort transmission, and send jam signal
- Exponential back-off: wait before retransmitting
  - Wait random time, exponentially larger on each retry

TCP Congestion Control

- Additive increase, multiplicative decrease
  - On packet loss, divide congestion window in half
  - On success for last window, increase window linearly



End-To-End Argument

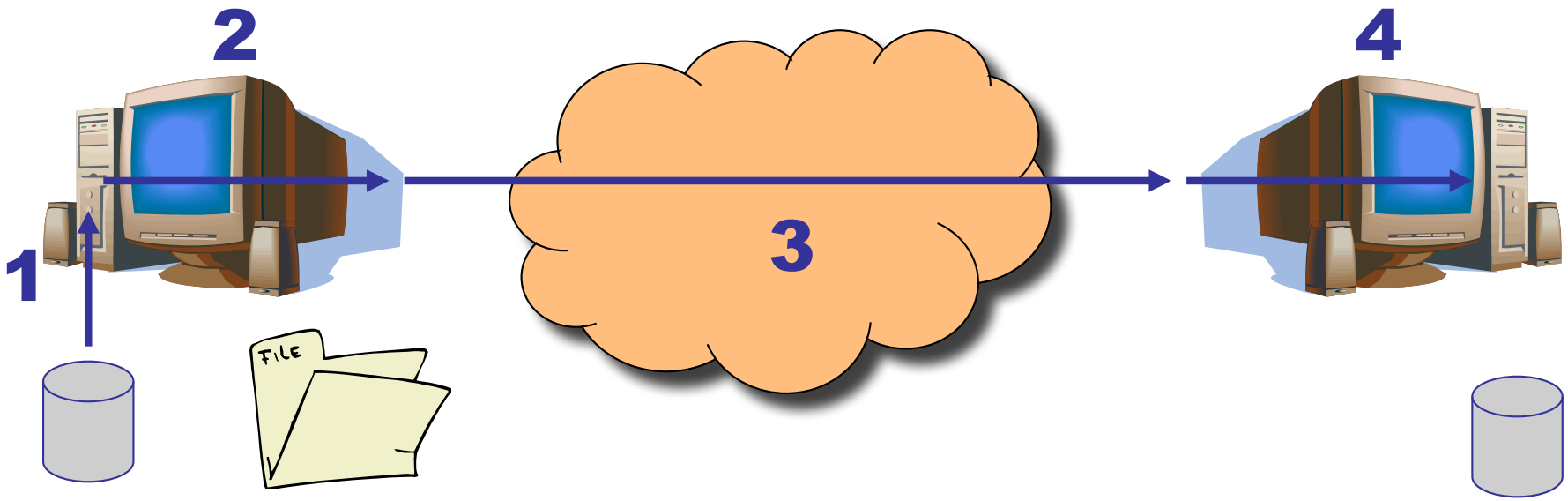


End-to-End Argument

- Functionality should be implemented at a lower layer if and only if it can be correctly and completely implemented there
  - Should not be implemented at lower level if redundant with higher level
  - Performance optimizations are not a violation
- Early example
  - ARPANet provided reliable link transfers between switches
  - Packets could still get corrupted on host-to-switch link, or inside switches
  - Want to know if host *acted* on the request not whether it *received* it

Example: Reliable File Transfer

- From disk on file (web) server over network to client
  - Disk can introduce bit errors
  - Host I/O buses can introduce bit errors
  - Packets can get garbled, dropped, misordered at any node
- Solution: integrity check on file, not per packet or per hop



Hop by Hop for Performance

- Does not violate end to end argument to provide reliability at link layer, even if not required for correct operation
- For file transfer application, consider varying conditions:
  - Prob(corrupted/lost packet per link) = p
  - Prob(packet lost end to end), avg. 15 hops across Internet

    p = 0.0001% => Prob(loss) = 0.0015%

    p = 1% => Prob(loss) = 14%
- Chance of file corruption grows with size of file
  - Potentially retransmit entire file for one lost packet?

Application-Specific Semantics

- Example: move reliability into the network communication protocol (such as TCP)
  - Certain computational and bandwidth overheads to implementing reliable, in-order delivery in the network
  - Not all applications want to pay this overhead
- Real-time voice/audio
  - Better to drop a packet, rather than hold up later packets
  - *On-time* delivery more important than *reliability*
- Applications should be able to pick and choose the semantics they require from underlying system
  - *Active Networks, overlay networks*

Other Examples

- Distributed transactions
  - Exactly once vs. at most once vs. at least once
- Security
  - Security only as strong as weakest assumption/link
- Suppressing duplicate messages
  - Duplicate effort between network and application layer
- Mis-ordered messages

Discussion

- When should the network support a function?
  - E.g., link-layer retransmission in wireless networks?
- Who's interests are served by the e2e argument?
- How does a network operator influence the network without violating the e2e argument?
- Does the design of IP and TCP make it *hard* to violate the e2e argument?
  - E.g., middlebox functionality like NATs, firewalls, proxies
- Should the e2e argument apply to routing?

For Next Class…

- NO CLASS MONDAY
- Read P&D Chapter 3
- Read Cerf & Kahn `74
- Keep thinking about term project ideas/groups
  - Suggestions available next week