Layered Network Architecture; Network Applications: Overview, EMail

Qiao Xiang, **Congming Gao**

https://sngroup.org.cn/courses/cnns-xmuf23/index.shtml

9/21/2023

This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

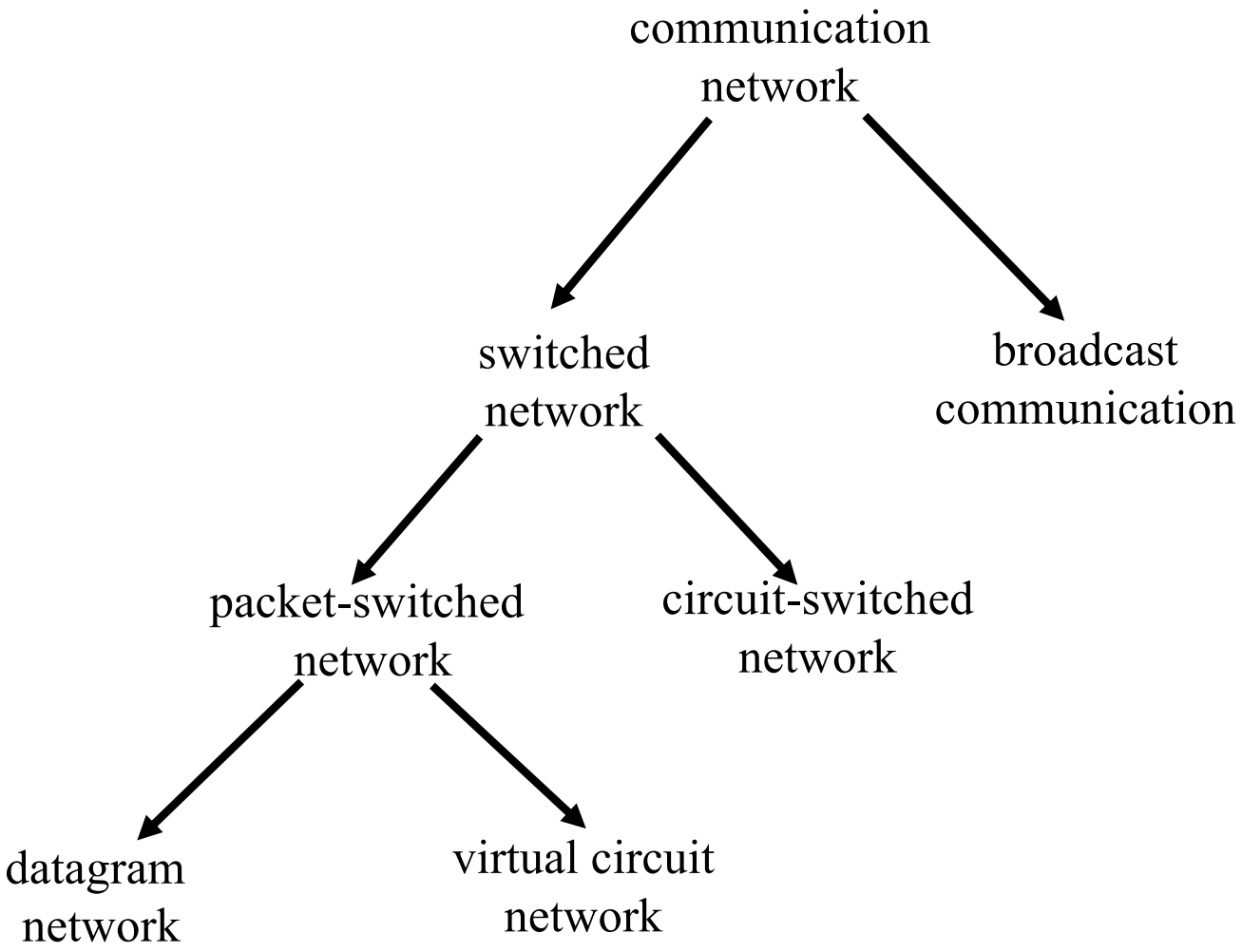
Outline

- Admin. and recap
- Layered network architecture
- Application layer overview
- Network applications
  - Email

Admin

- Questions on Assignment One

Recap: Summary of the Taxonomy of Communication Networks



Recap: Circuit Switching vs. Packet Switching

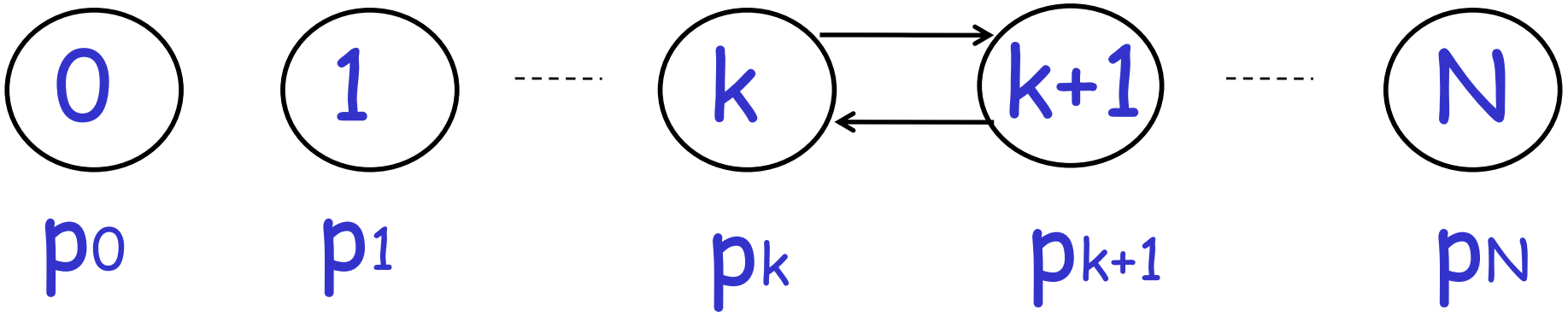
	circuit switching	packet switching
resource usage	use a single partition bandwidth	use whole link bandwidth
reservation/setup	need reservation (setup delay)	no reservation
resource contention	busy signal (session loss)	congestion (long delay and packet losses)
charging	time	packet
header	no per-pkt header	per packet header
fast path processing	fast	per packet processing

Recap: Queueing Theory

- Model system state
- Introduce state transition diagram
- Focus on equilibrium: state trend neither growing nor shrinking

Recap: Queueing Theory Analysis of Circuit-Switching

system state: # of busy lines



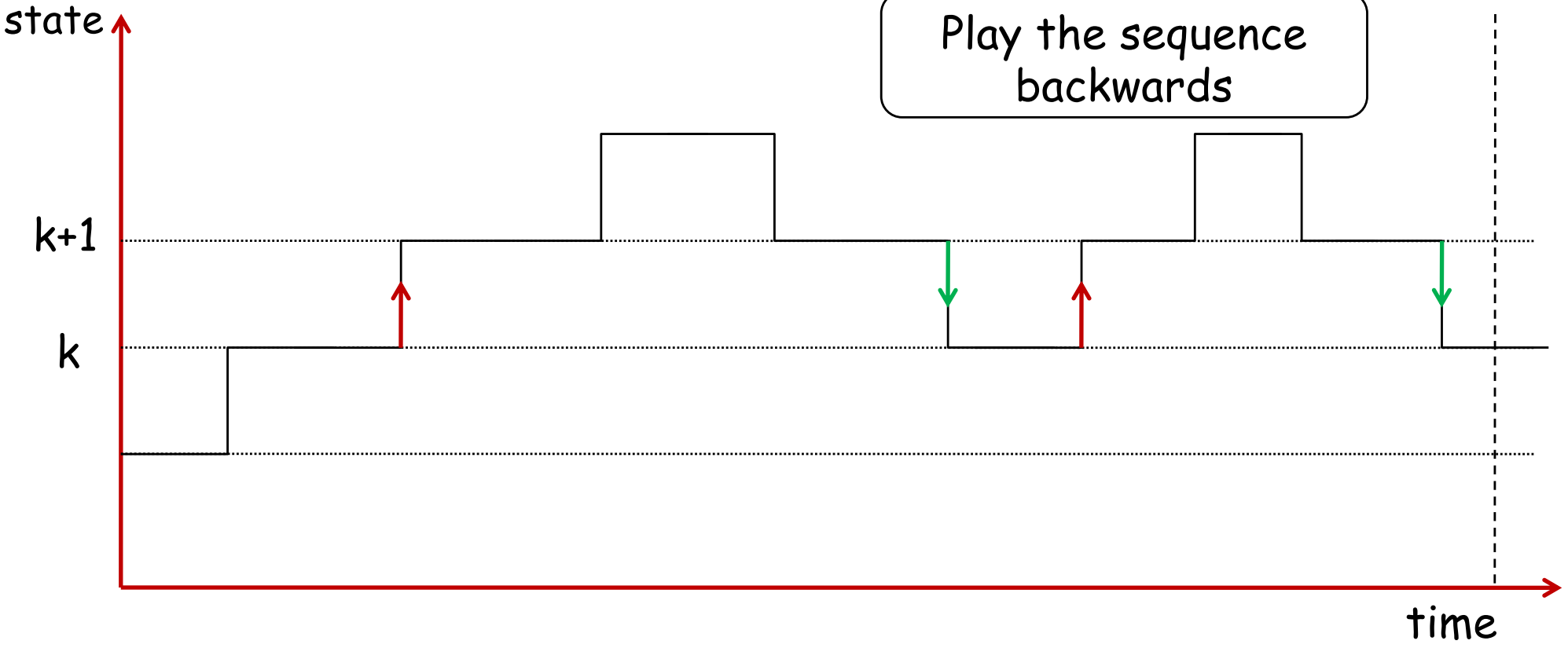
Equilibrium = Time Reversibility [Frank Kelly]

- Statistically cannot distinguish

$$\#f_{k\to k+1} = \#f_{k+1\to k}$$

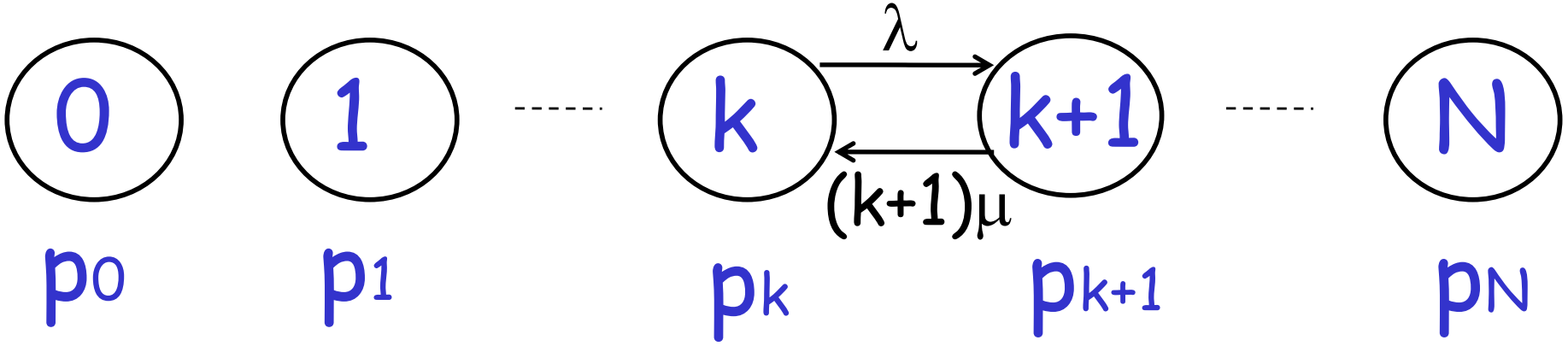
$$\#b_{k\to k+1} \quad \#b_{k+1\to k}$$

Play the sequence backwards



Recap: Queueing Theory Analysis of Circuit-Switching

system state: # of busy lines



at equilibrium (time reversibility) in one unit time:

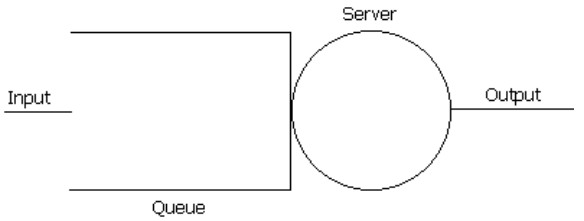
#(transitions k → k+1) = #(transitions k+1 → k)

$$p_k \lambda = p_{k+1}(k+1)\mu$$

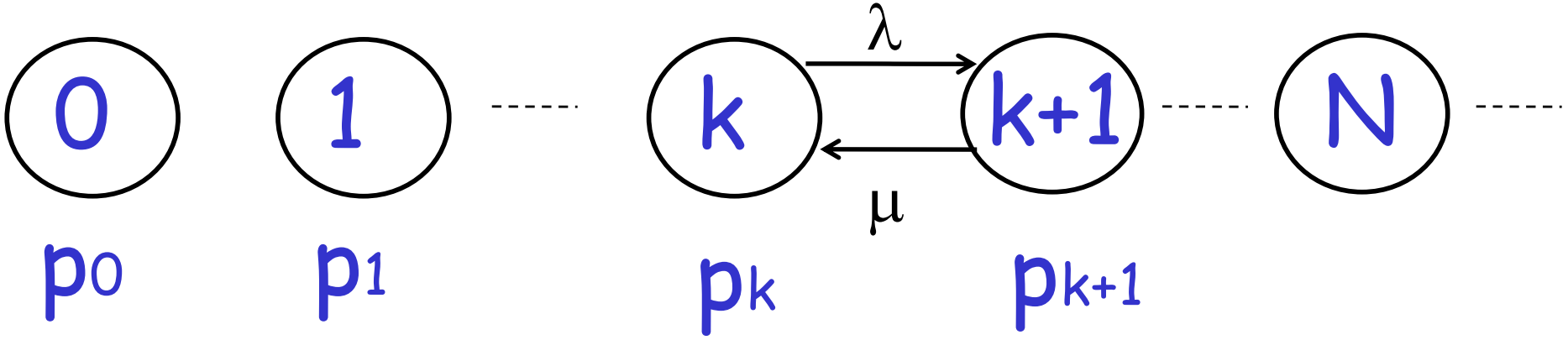
$$p_{k+1} = \frac{1}{k+1}\frac{\lambda}{\mu} p_k = \frac{1}{(k+1)!}\left(\frac{\lambda}{\mu}\right)^{k+1} p_0$$

$$p_0 = \frac{1}{1 + \frac{1}{1!}\frac{\lambda}{\mu} + \frac{1}{2!}\left(\frac{\lambda}{\mu}\right)^2 + \ldots + \frac{1}{N!}\left(\frac{\lambda}{\mu}\right)^N}$$

Recap: Queueing Theory Analysis of Packet Switching



system state: #packets in queue



at equilibrium (time reversibility) in one unit time:

#(transitions k → k+1) = #(transitions k+1 → k)

$$p_k \lambda = p_{k+1} \mu$$

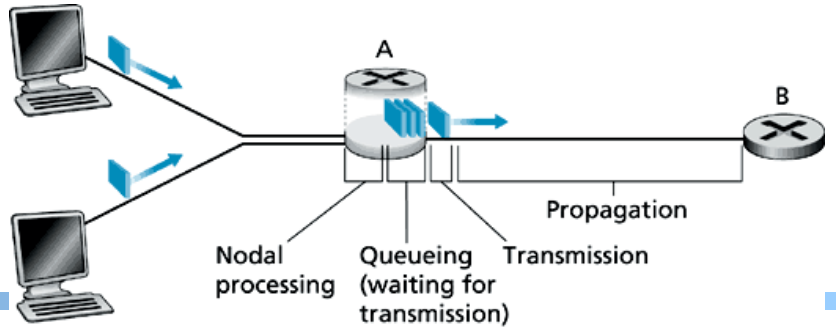
$$\sum_{k=0}^{\infty} p_k = 1$$

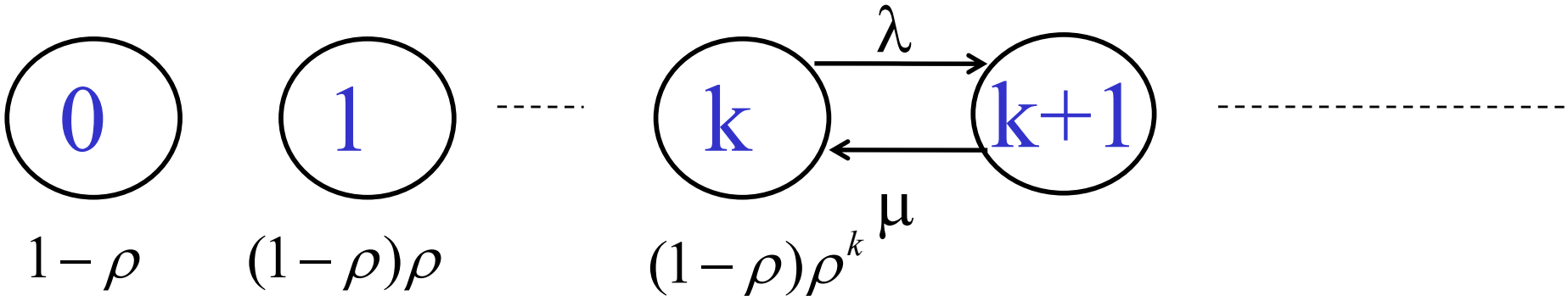
$$p_{k+1} = \frac{\lambda}{\mu} p_k = \left(\frac{\lambda}{\mu}\right)^{k+1} p_0 = \rho^{k+1} p_0$$

$$p_0 = 1 - \rho$$

$$\rho = \frac{\lambda}{\mu}$$

Recap: Analysis of Delay





- Average queueing delay:

$$\sum_{k=0}^{\infty} p_k \cdot k \cdot \frac{1}{\mu} = \sum_{k=0}^{\infty} \rho^k (1-\rho)\, k \frac{1}{\mu}$$

- Transmission delay: $\boxed{S = \frac{1}{\mu}}$

- Queueing + transmission:

Recap: Analysis of Delay

$$\rho = \frac{\lambda}{\mu}$$

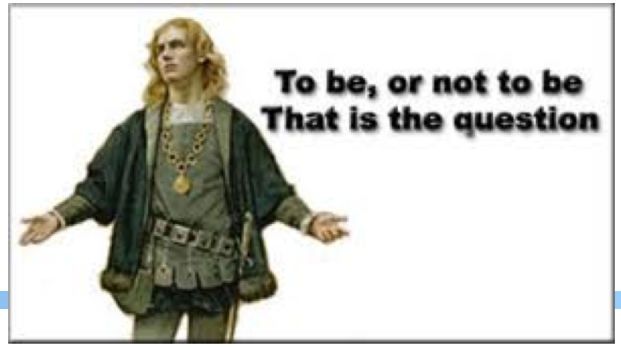
$$S = \frac{1}{\mu}$$

$$\text{average queueing delay: } w = S\frac{\rho}{1-\rho}$$

$$queueing + trans = S\frac{\rho}{1-\rho} + S = S\frac{1}{1-\rho}$$

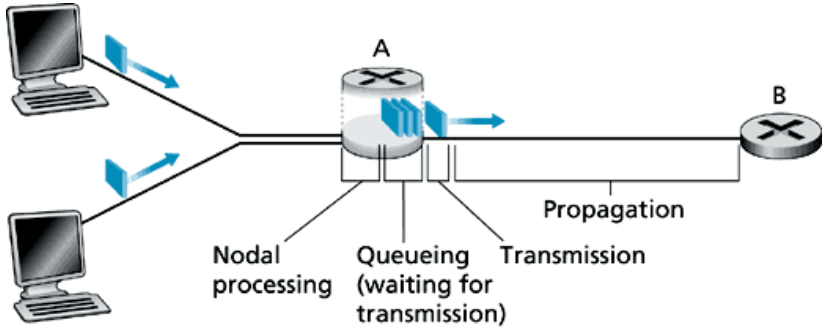
For a demo of M/M/1, see:
http://www.dcs.ed.ac.uk/home/jeh/Simjava/queueing/mm1_q/mm1_q.html

Recap: Statistical Multiplexing



A simple model to compare bandwidth efficiency of
- reservation/dedication (aka circuit-switching) vs
- no reservation (aka packet switching)

setup
- a single bottleneck link with service rate μ
- n flows; each flow has an arrival rate of λ/n



- no reservation: all arrivals into the single link, the queueing delay + transmission delay:

$$S\frac{1}{1-\rho}$$

- reservation: each flow uses its own reserved (sub)link with rate μ /n, the queueing delay + transmission delay:

For each flow i:

$$\rho_i = \frac{\lambda/n}{\mu/n} = \rho \quad \Rightarrow \quad n\,S\frac{1}{1-\rho}$$

$$S_i = \frac{1}{\mu/n} = nS$$

Summary of Progress

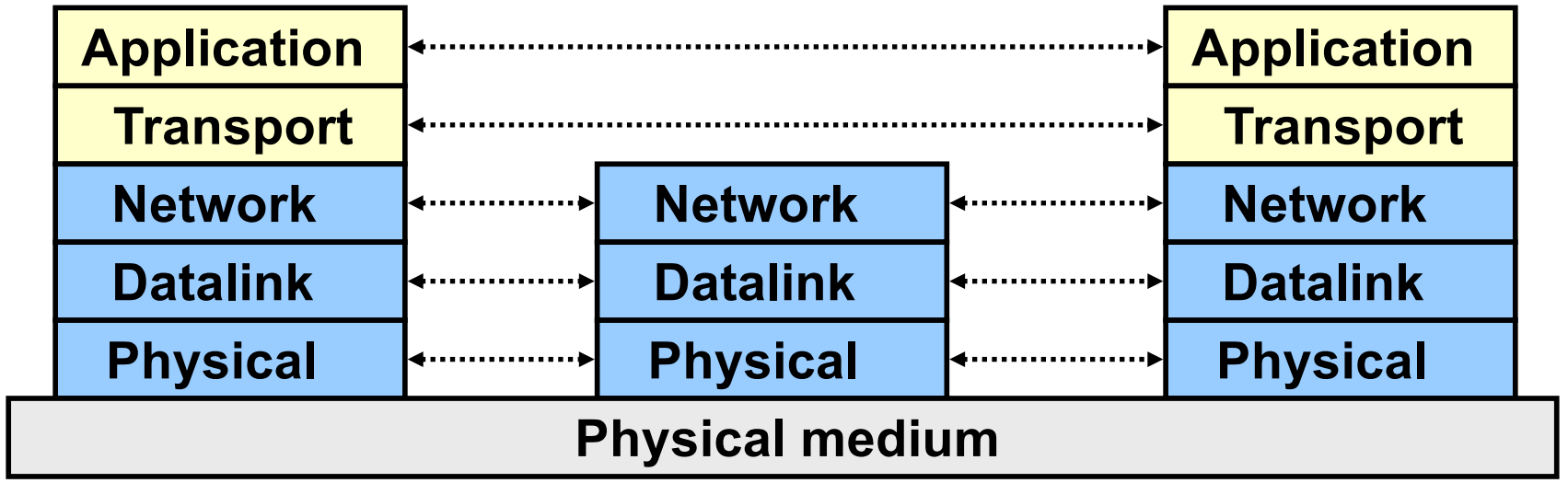
- We have seen the hardware infrastructure, the basic communication scheme, a next key question is how to develop the software system.

Outline

- Admin. and recap
- Layered network architecture
  - *what is layering?*
  - why layering?
  - how to determine the layers?
  - ISO/OSI layering and Internet layering

What is Layering?

- A technique to organize a networked system into a succession of logically distinct entities, such that the service provided by one entity is solely based on the service provided by the previous (lower level) entity.



Outline

- Admin. and recap
- Layered network architecture
  - what is layering?
  - *why layering?*

Why Layering?

Networks are complex !

- many "pieces":
  - hardware
    - hosts
    - routers
    - links of various media
  - software
    - applications
    - infrastructure

Dealing with complex systems:

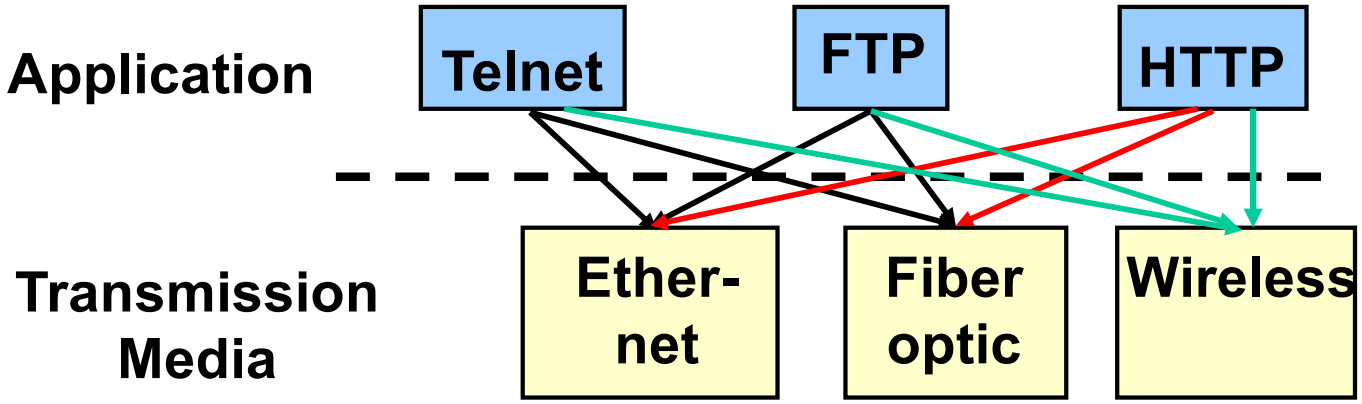
explicit structure allows identification of the relationship among a complex system's pieces

- layered reference model for discussion

Modularization eases maintenance, updating of system:

- change of implementation of a layer's service transparent to rest of system, e.g., changes in routing protocol doesn't affect rest of system

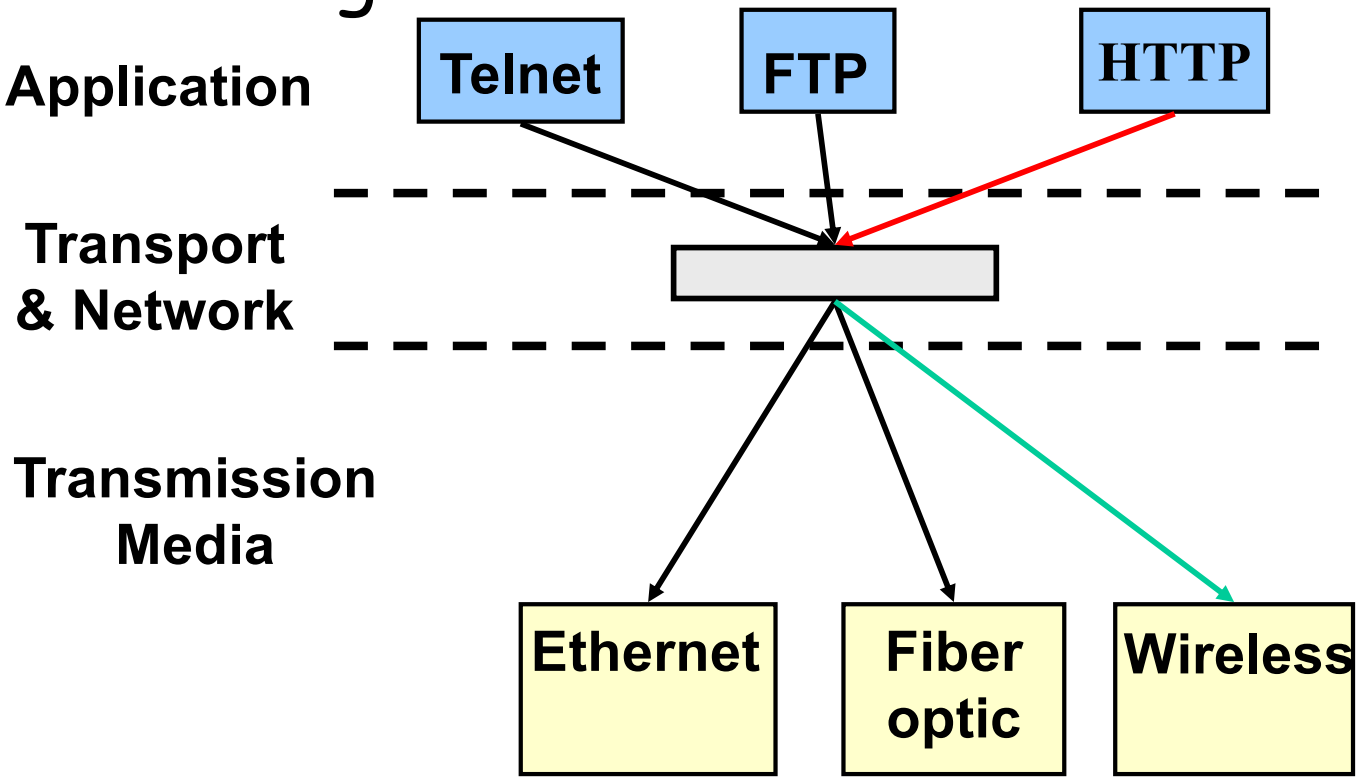
An Example: No Layering



- No layering: each new application has to be re-implemented for every network technology !

An Example: Benefit of Layering

- Introducing an intermediate layer provides a common abstraction for network technologies

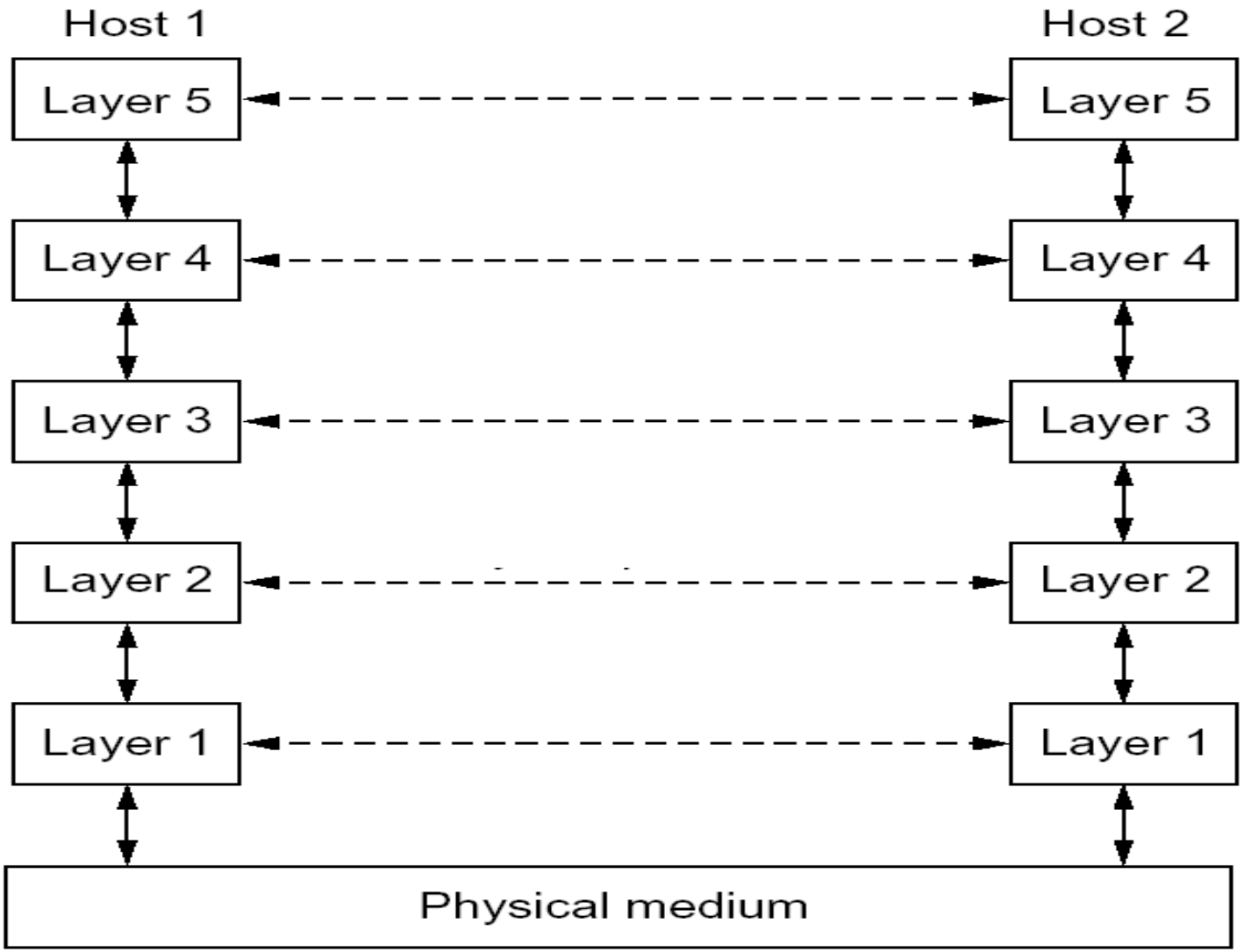


ISO/OSI Concepts

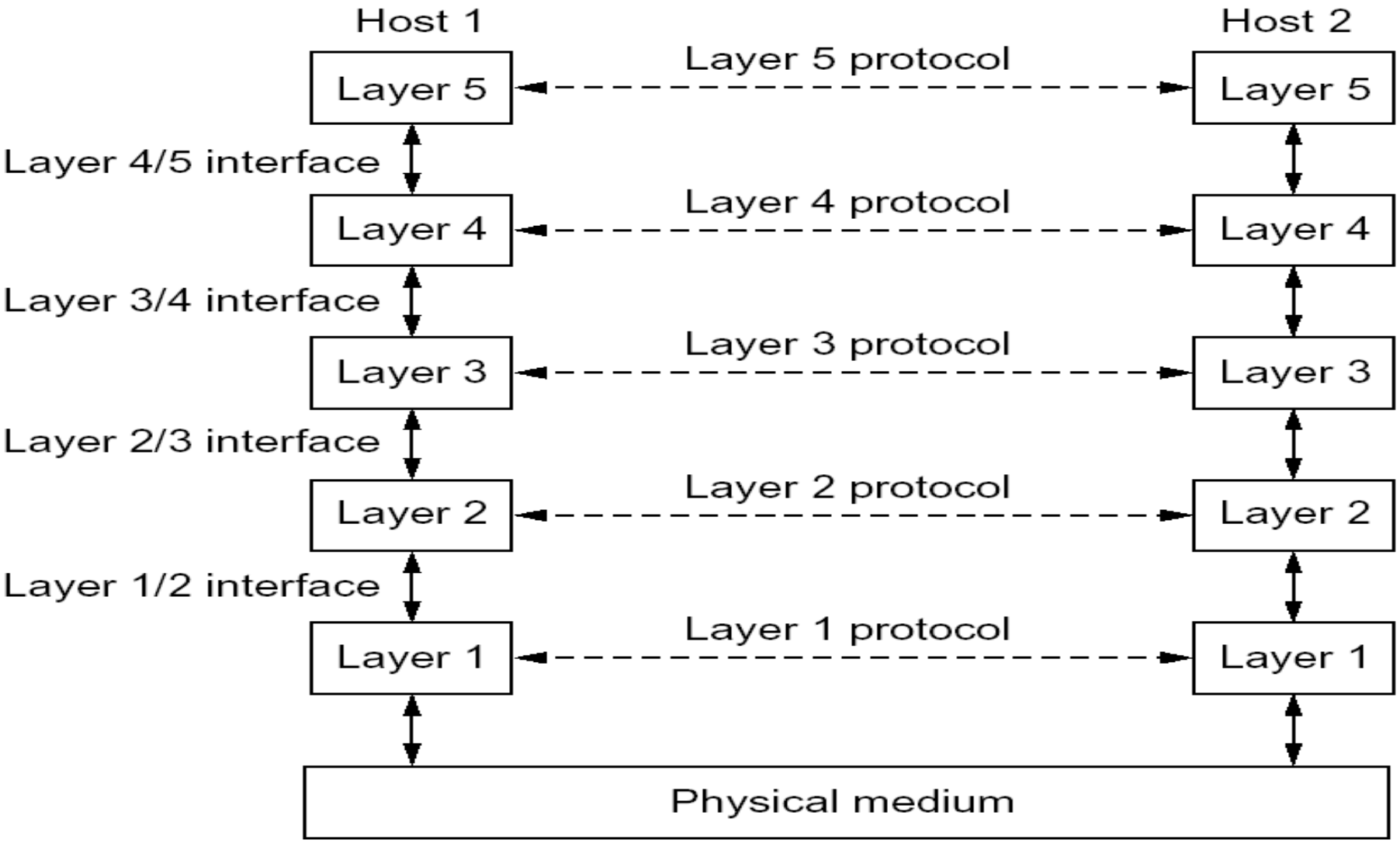
- ISO – International Standard Organization
- OSI – Open System Interconnection

- Service – says what a layer does
- Interface – says how to access the service
- Protocol – specifies how the service is implemented
  - a set of rules and formats that govern the communications between two or more peers

An Example of Layering



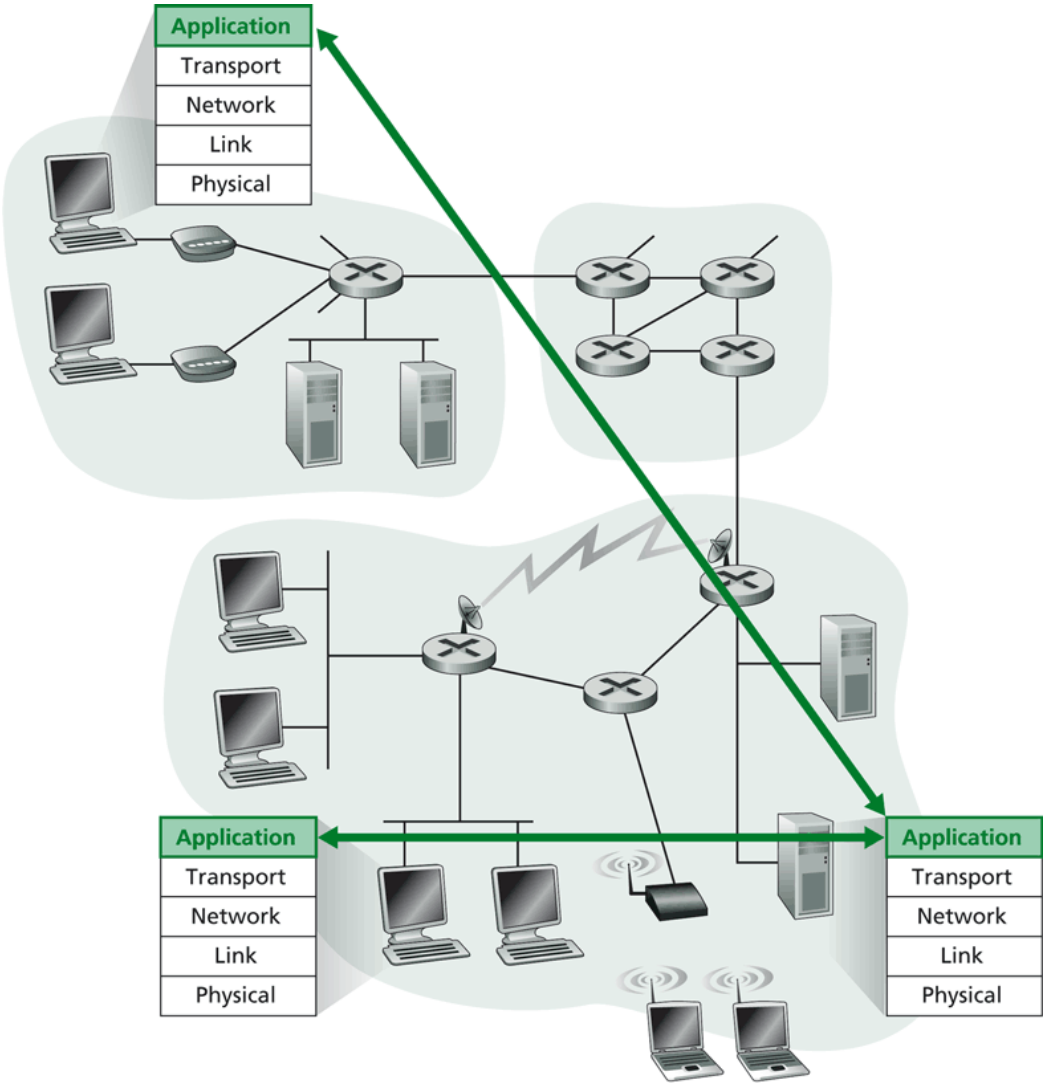
An Example of Layering



Layering -> *Logical* Communication

E.g.: application

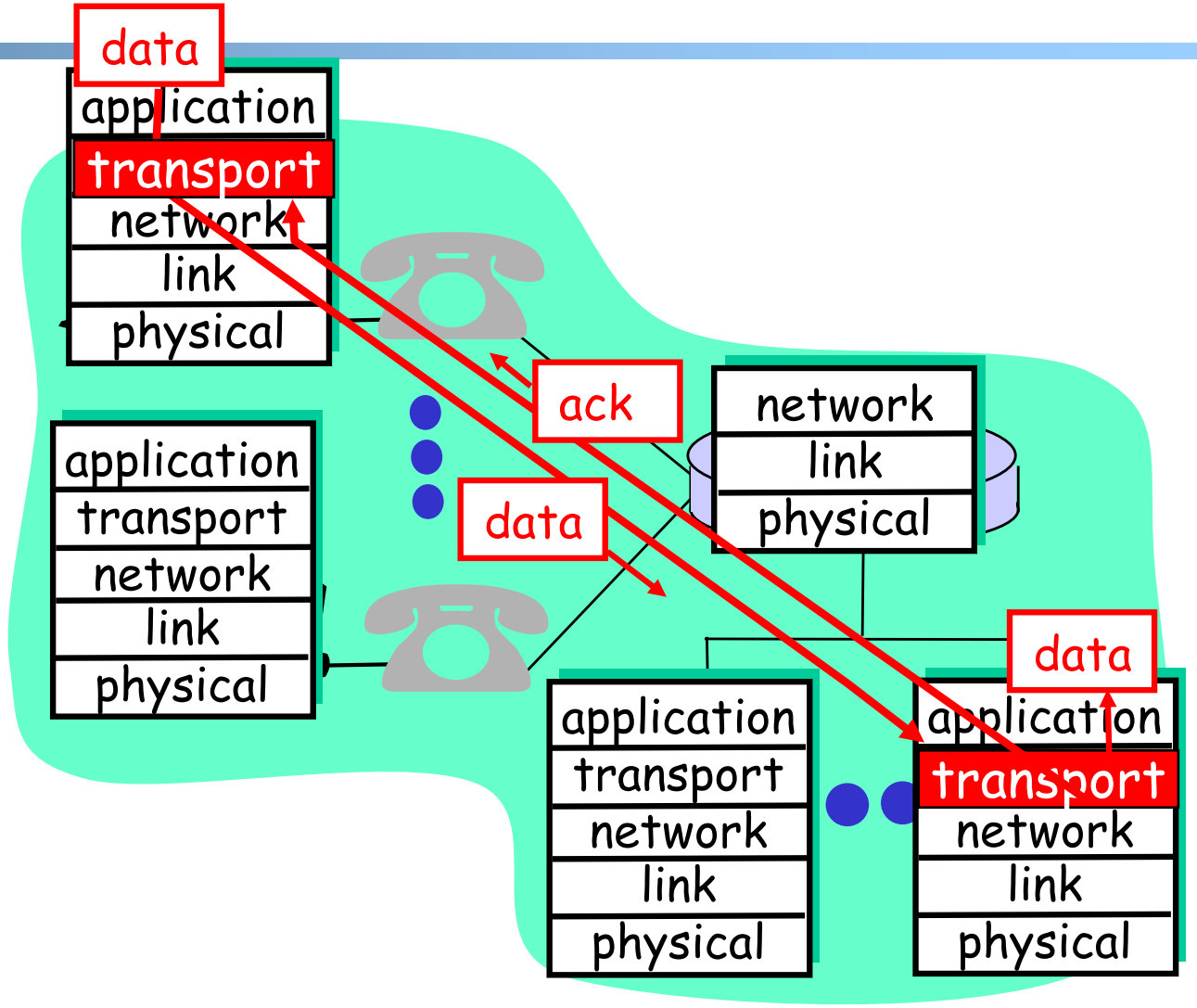
- provide services to users
- application protocol:
  - send messages to peer
  - for example, HELO, MAIL FROM, RCPT TO are messages between two SMTP peers



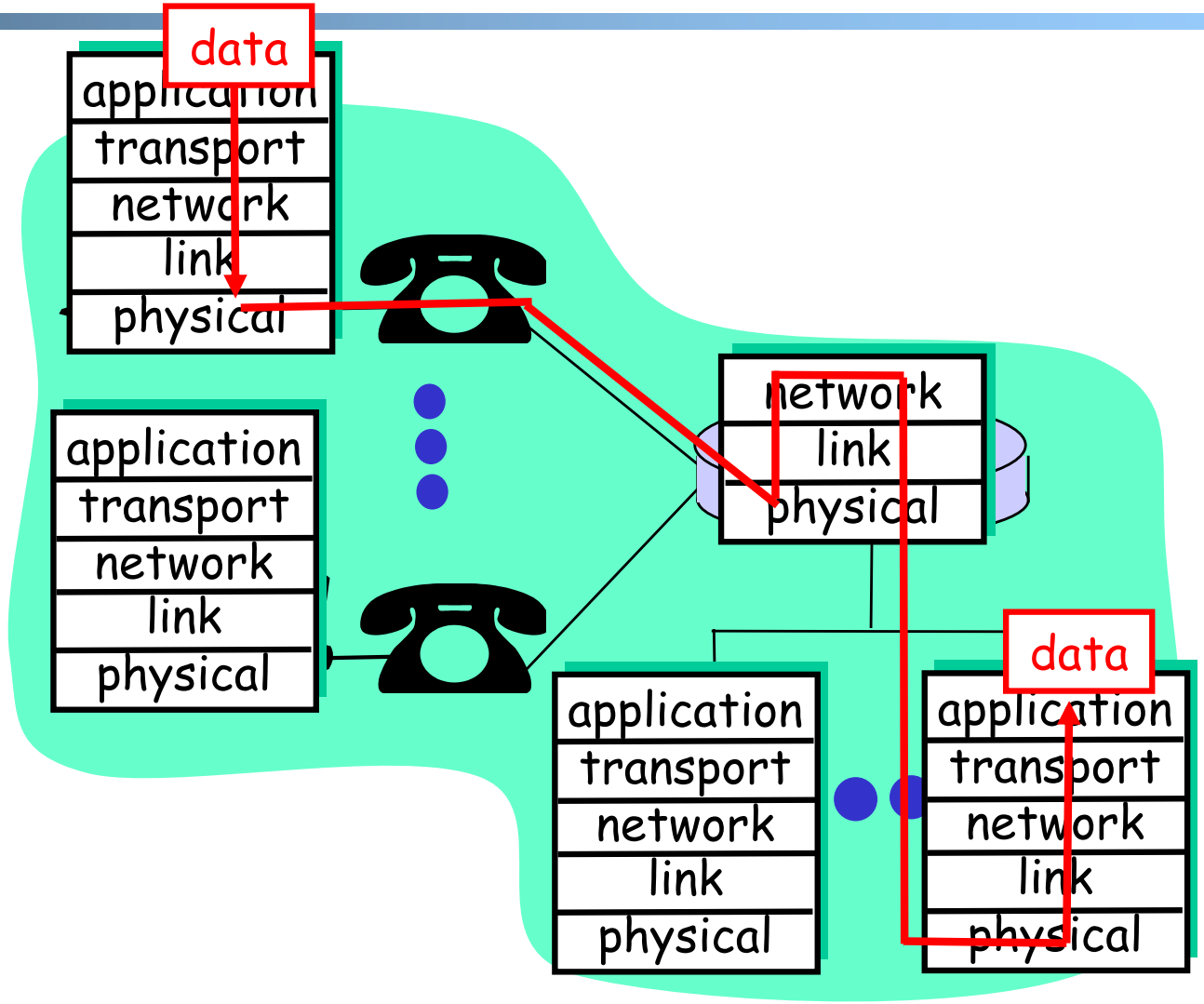
Layering: *Logical* Communication

E.g.: transport

- Trans. msg for app
- Transport protocol
  - add control info to form "segment"
  - send segment to peer
  - wait for peer to ack receipt; if no ack, retransmit



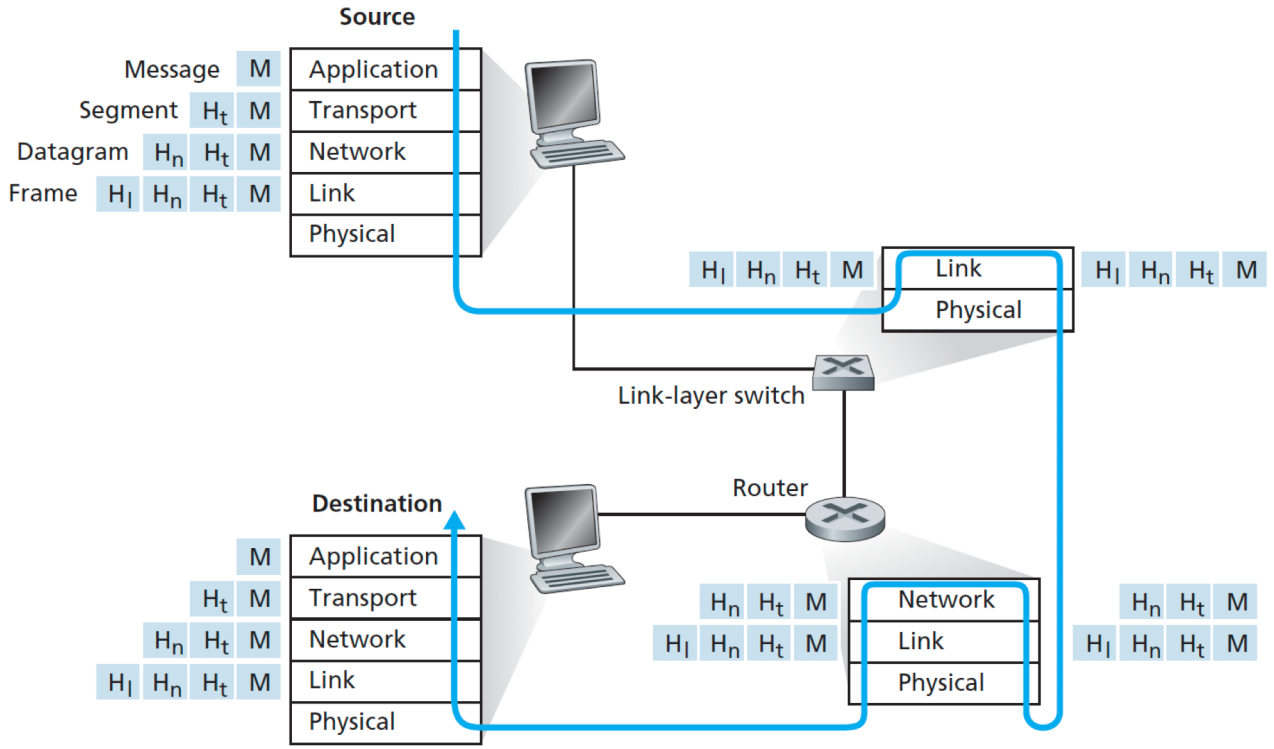
Layering: *Physical* Communication



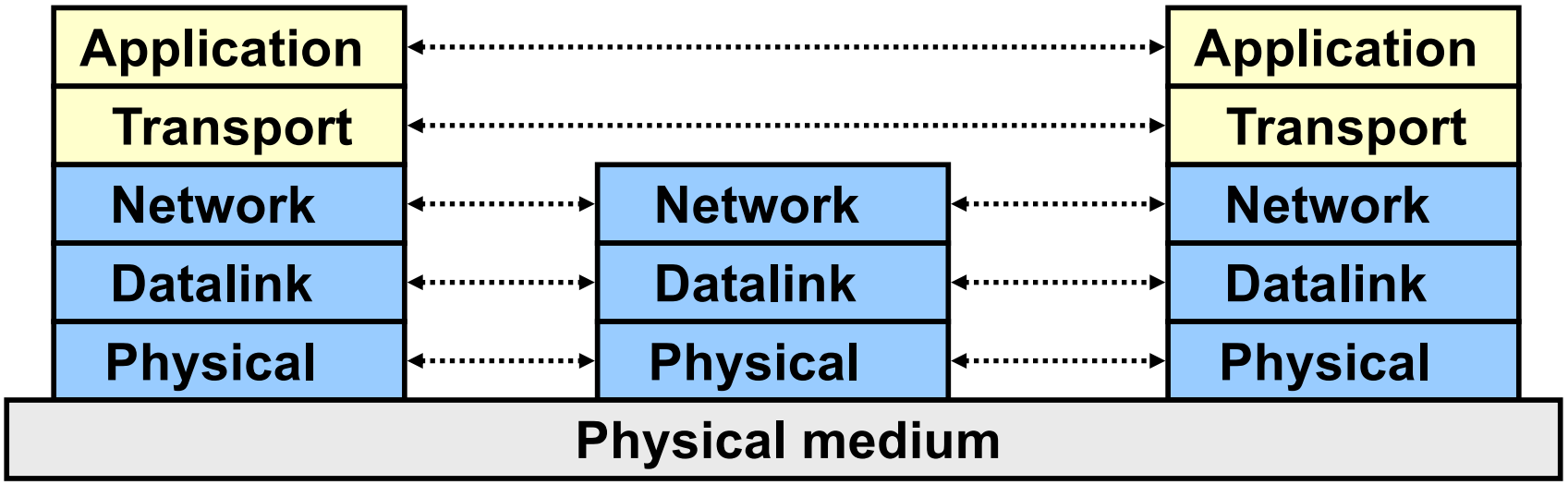
Protocol Layering and Meta Data

Each layer takes data from above

- adds header (meta) information to its peer to create new data unit
- passes new data unit to layer below

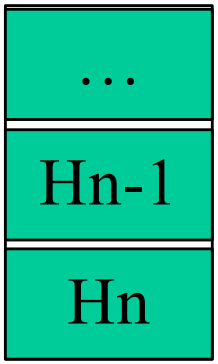


Packet as a Stack in a Layered Architecture

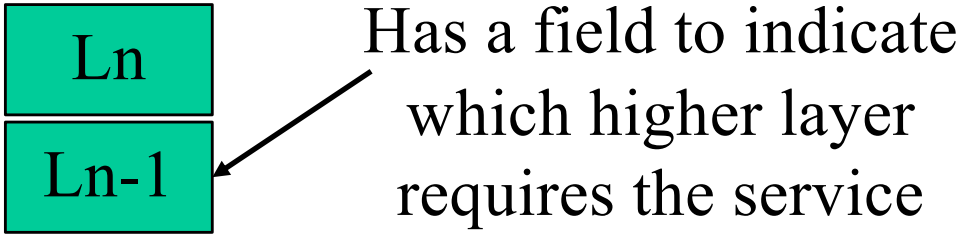


Some Implications of Layered Architecture

- A packet as a stack container



- Each layer needs multiplexing and demultiplexing to serve layer above



Key design issue:

How do you *divide* functionalities among the layers?

Outline

- Admin. and recap
- Layered network architecture
  - what is layering?
  - why layering?
  - *how to determine the layers?*

The End-to-End Arguments

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication systems. Therefore, providing that questioned function as a feature of the communications systems itself is not possible.

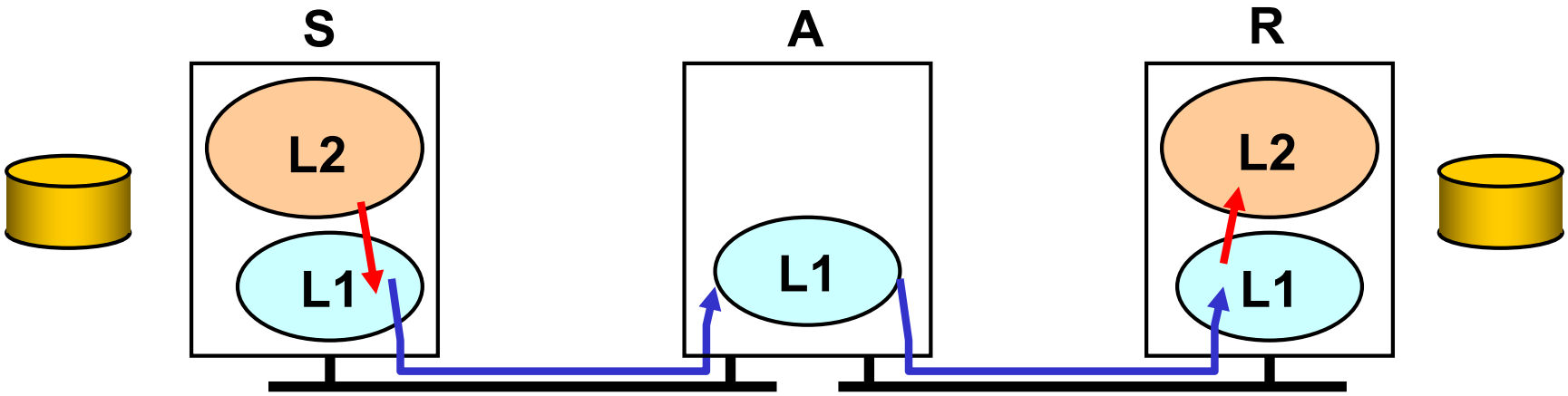
J. Saltzer, D. Reed, and D. Clark, 1984

What does the End-to-End Arguments Mean?

- The application knows the requirements best, place functionalities as high in the layer as possible

- Think twice before implementing a functionality at a lower layer, even when you believe it will be useful to an application

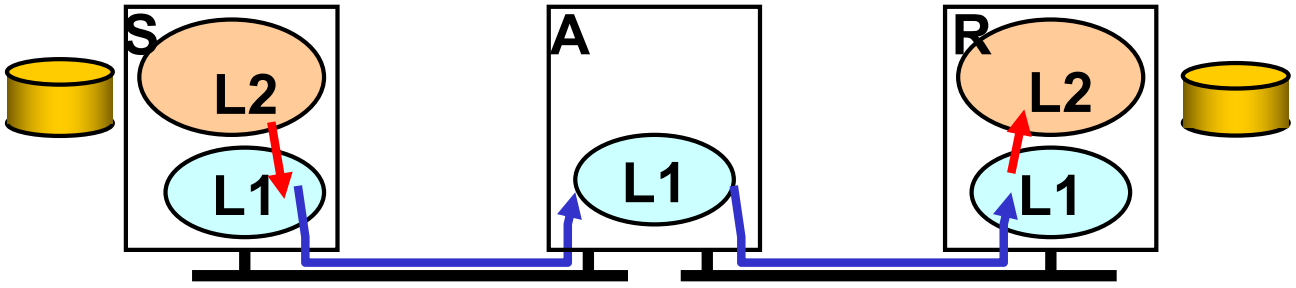
Example: Where to Provide Reliability ?



- Solution 1: the network (lower layer L1) provides reliability, i.e., each hop provides reliability
- Solution 2: the end host (higher layer L2) provides reliability, i.e., end-to-end check and retry

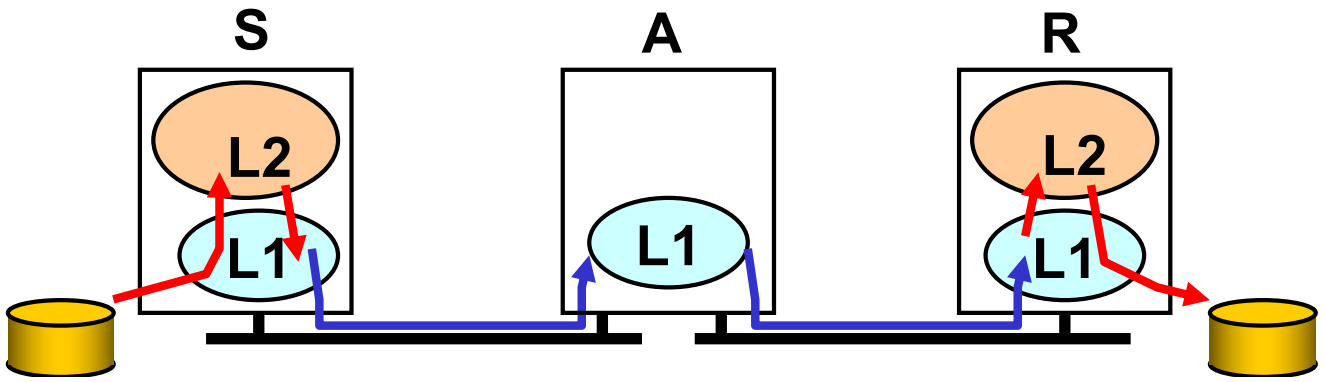
What are Reasons for Implementing Reliability at Higher Layer ?

- The lower layer cannot completely provide the functionality
  - the receiver has to do the check anyway !
- Implementing it at lower layer increases complexity, cost and overhead at lower layer
  - shared by all upper layer applications → everyone pays for it, even if you do not need it
- The upper layer
  - knows the requirements better and thus may choose a better approach to implement it



Are There Reasons Implementing Reliability at Lower Layer ?

- Improve performance, e.g., if high cost/delay/… on a local link
  - improves efficiency
  - reduces delay
- Share common code, e.g., reliability is required by multiple applications



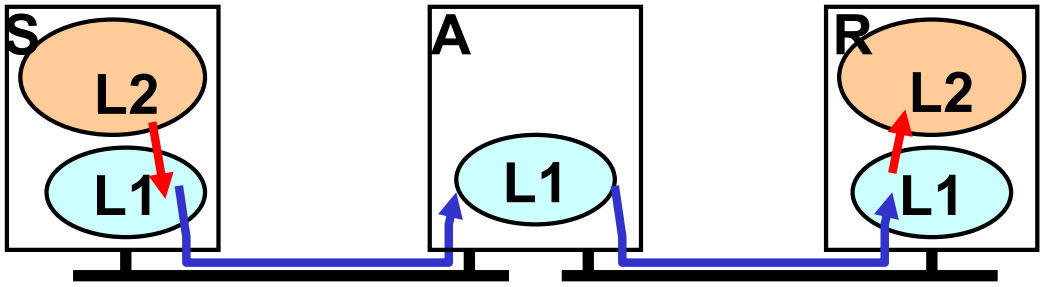
Summary: End-to-End Arguments

- If a higher layer can do it, don't do it at a lower layer -- the higher the layer, the more it knows about the best what it needs
- Add functionality in lower layers iff it
  - (1) is used by and improves performance of a large number of (current and potential future) applications,
  - (2) does not hurt (too much) other applications, and
  - (3) does not increase (too much) complexity/overhead
- Practical tradeoff, e.g.,
  - allow multiple interfaces at a lower layer (one provides the function; one does not)

Examples

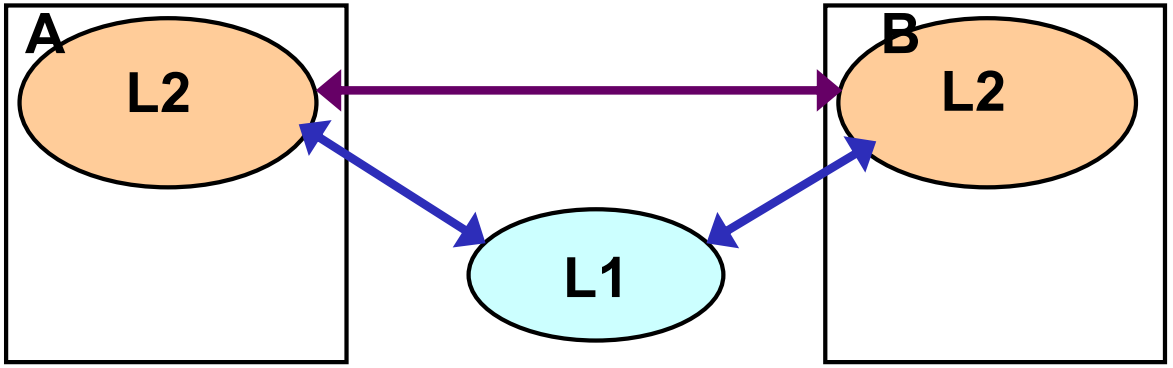
- We used reliability as an example

- Assume two layers (L1: network; L2: end-to-end). Where may you implement the following functions?
  - security (privacy of traffic)
  - quality of service (e.g., delay/bandwidth guarantee)
  - congestion control (e.g., not to overwhelm network links or receiver)



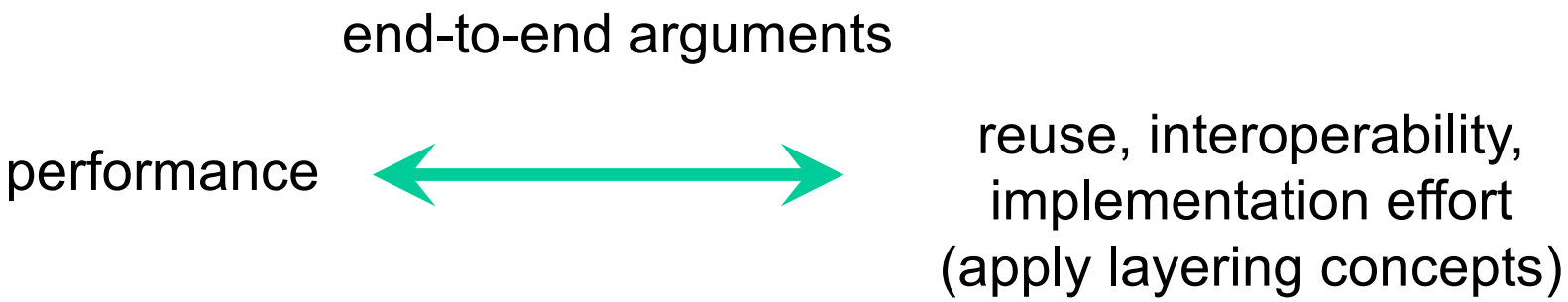
Example

- Consider the presence service in a social networking system: shows which contacts are online (e.g., skype)
  - implementing by end user's host app or through a third party service?



Challenges

- Challenges to build a good (networking) system: find the right balance between:



No universal answer: the answer depends on the goals and assumptions!

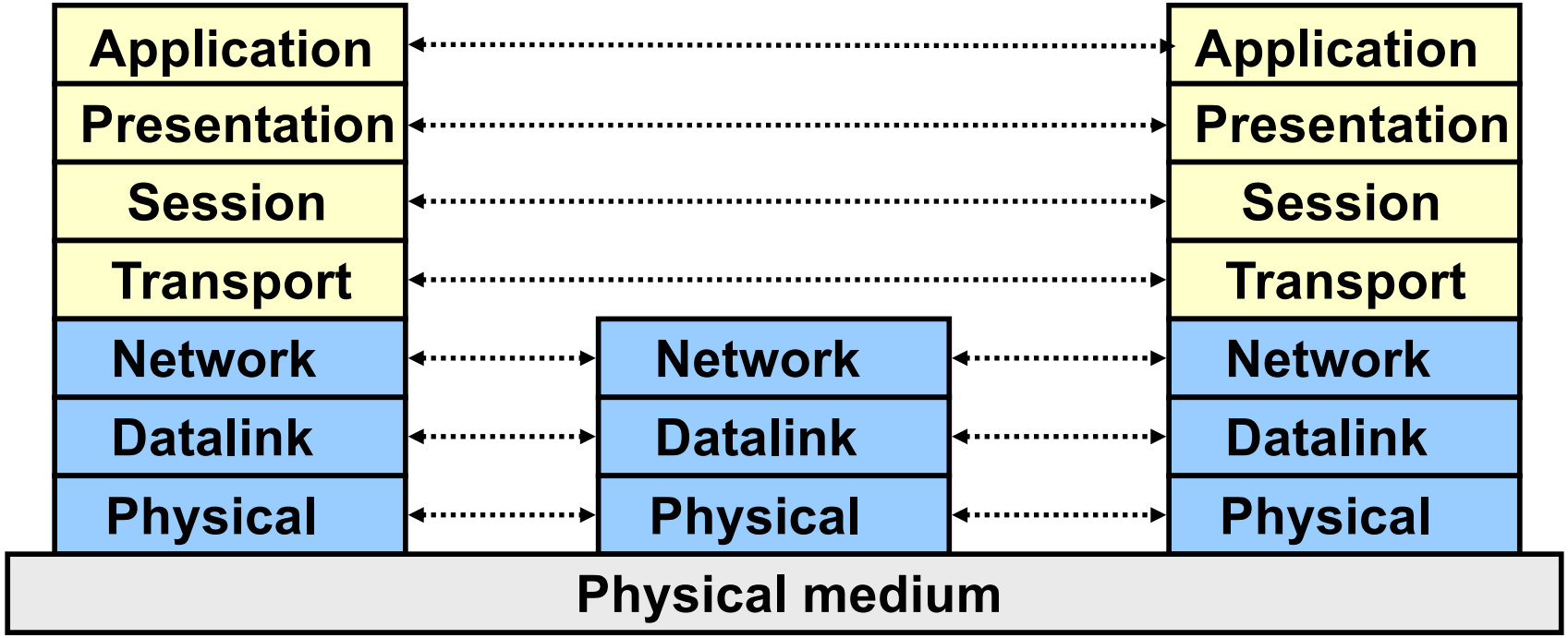
Discussion: Limitations of Layered Architecture

Outline

- Admin. and recap
- Layered network architecture
  - *what is layering?*
  - why layering?
  - how to determine the layers?
    - *ISO/OSI layering and Internet layering*

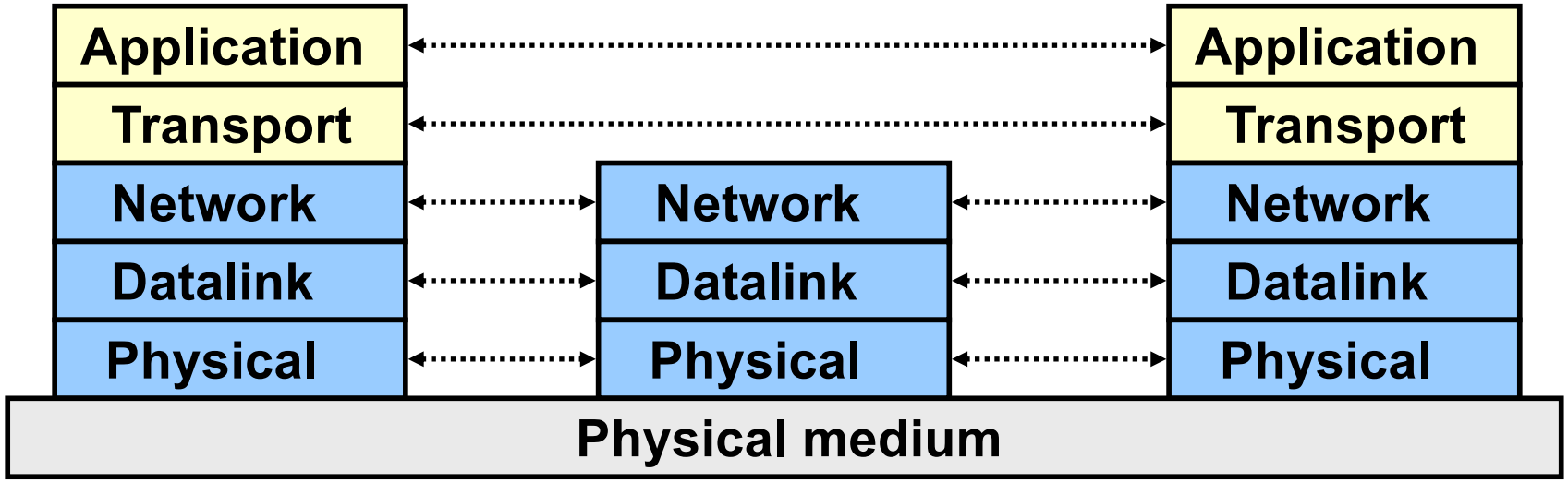
ISO/OSI Reference Model

- Seven layers
  - highest four layers are implemented in host



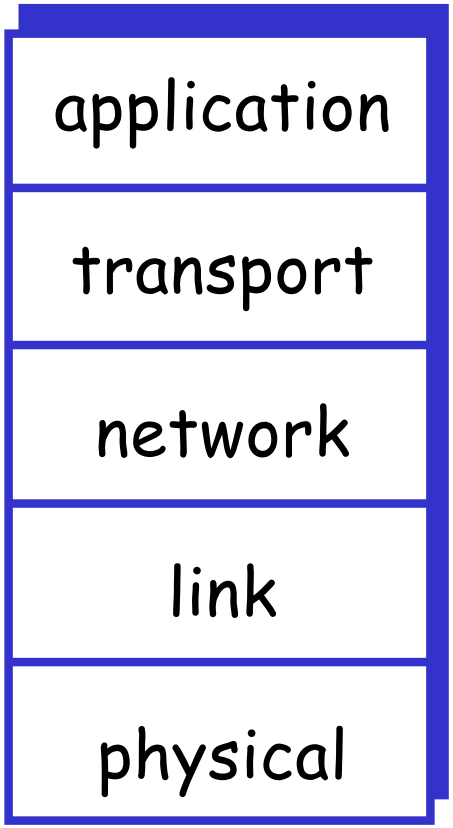
Internet Layering

- Five layers
  - highest two layers are implemented in host

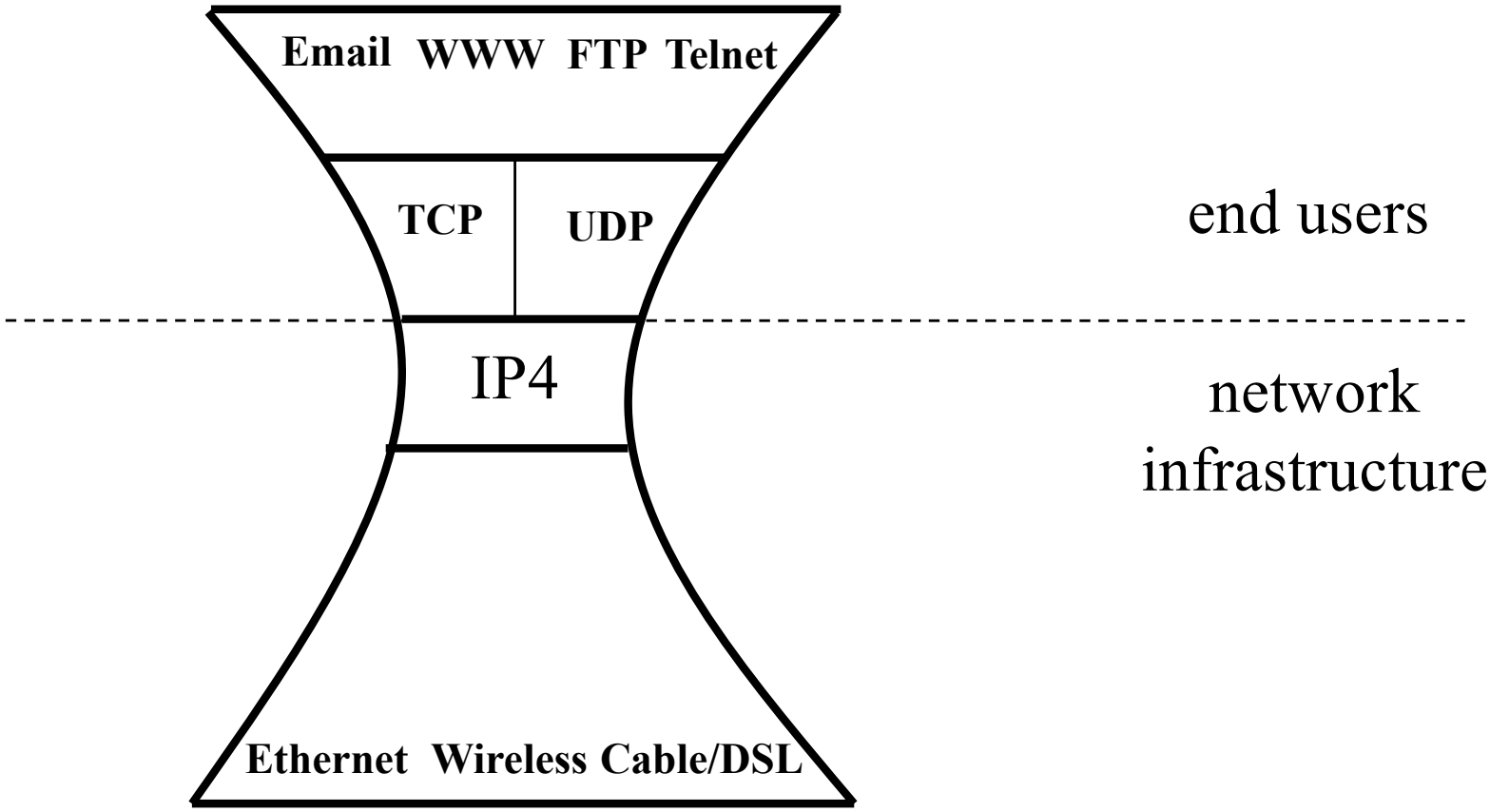


Internet Protocol Layers

- Five layers
  - Application: applications
    - ftp, smtp, http, p2p, IP telephony, blockchain, MapReduce, …
  - Transport: host-host data transfer
    - tcp (reliable), udp (not reliable)
  - Network: routing of datagram from source to destination
    - ipv4, ipv6
  - Link: data transfer between neighboring network elements
    - ethernet, 802.11, cable, DSL, …
  - Physical: bits "on the wire"
    - cable, wireless, optical fiber

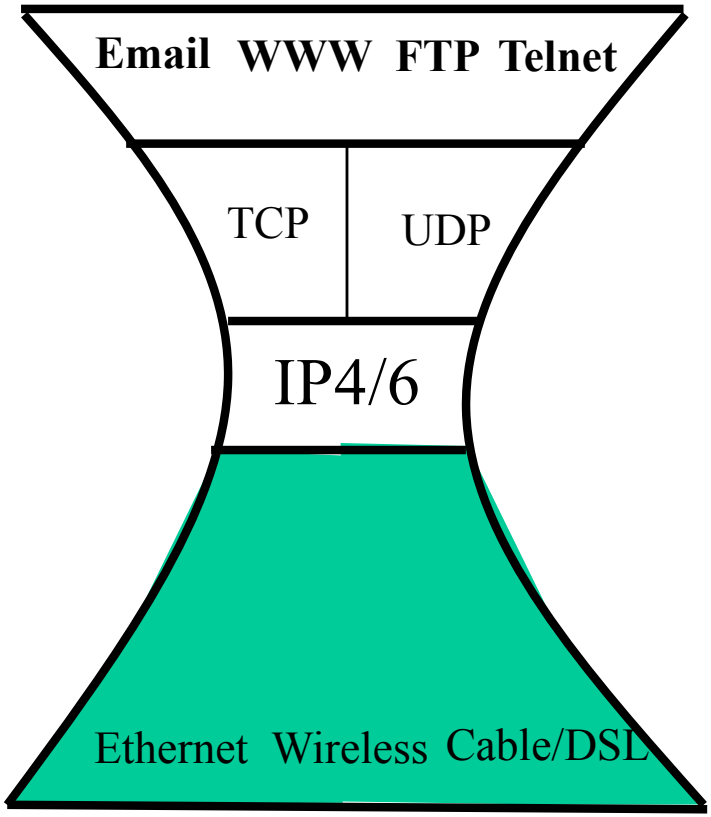


The Hourglass Architecture of the Internet

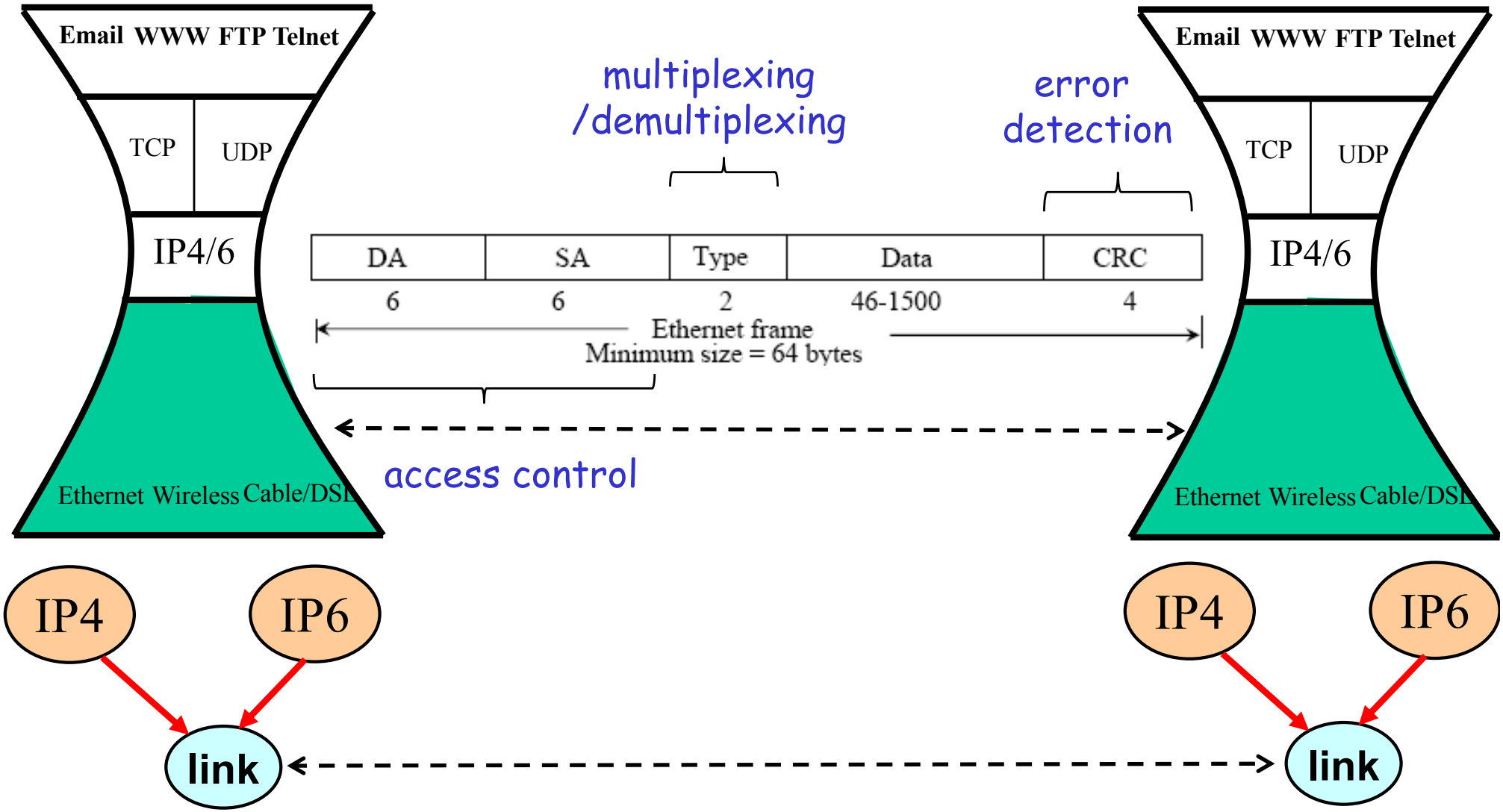


Link Layer (Ethernet)

- Services (to network layer)
  - multiplexing/demultiplexing
    - from/to the network layer
  - error detection
  - multiple access control
    - arbitrate access to shared medium
- Interface
  - send frames to a directly reachable peer

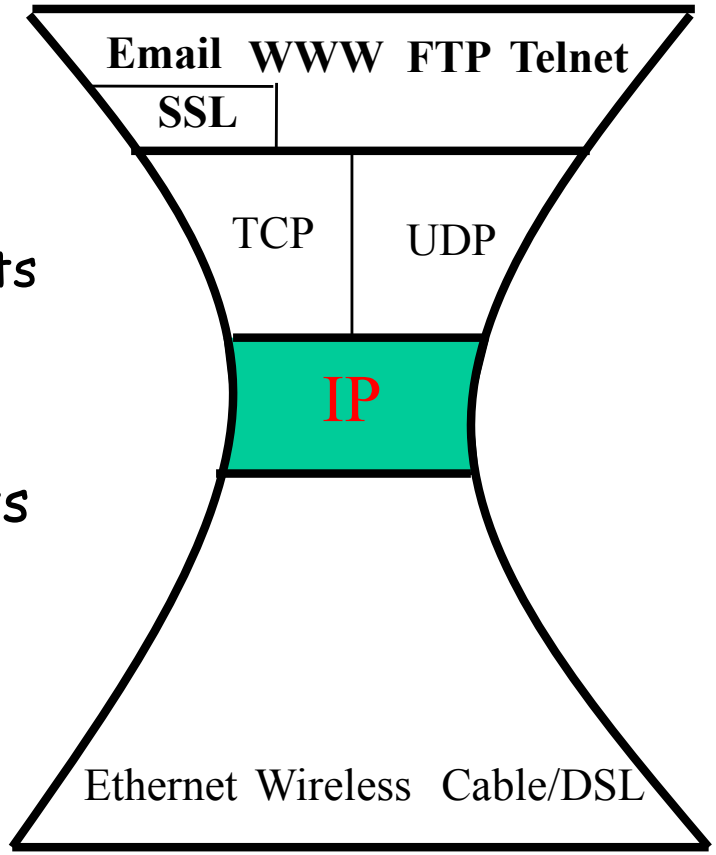


Link Layer: Protocol Header (Ethernet)

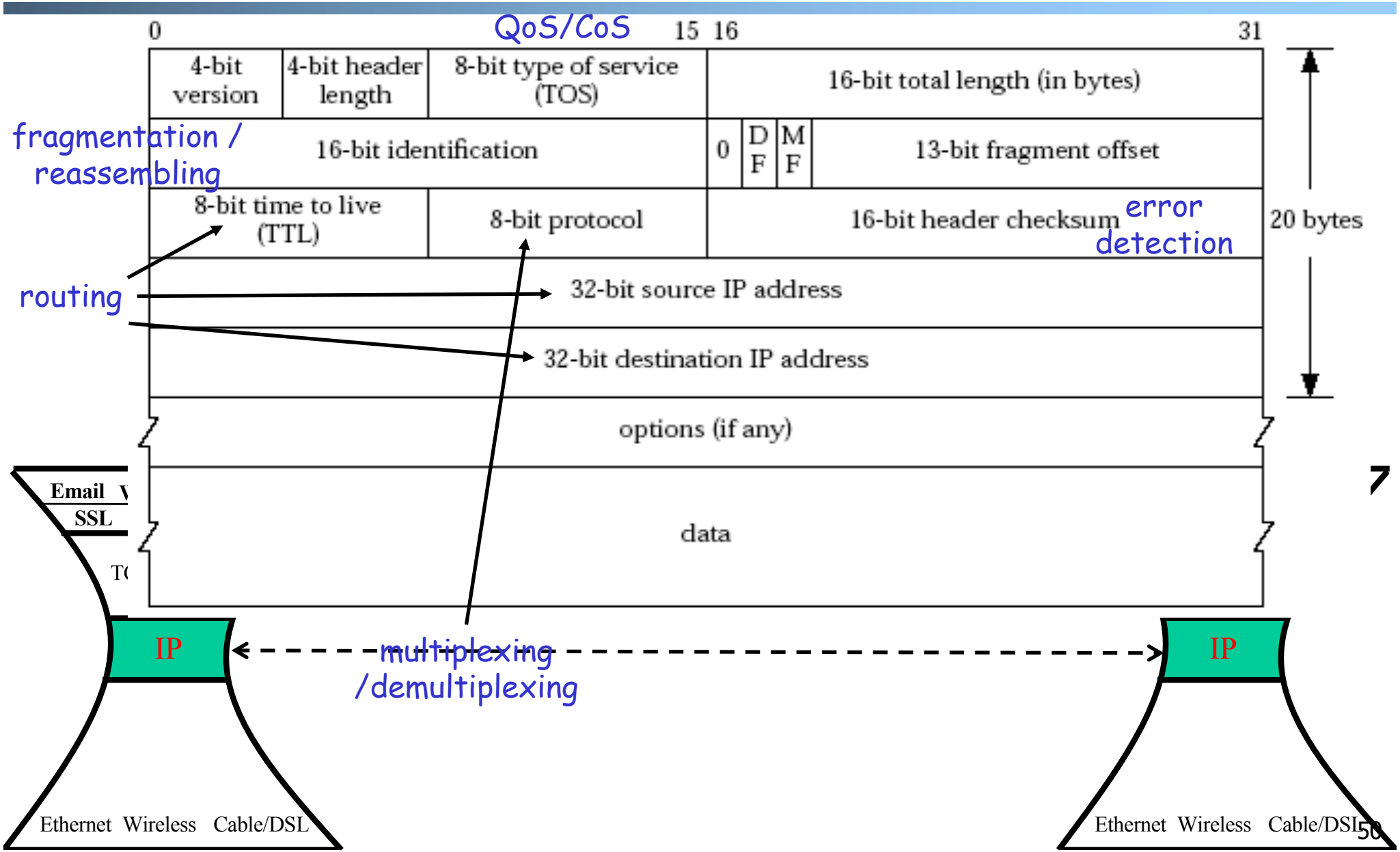


Network Layer: IP

- Services (to transport layer)
  - multiplexing/demultiplexing from/to the transport
  - fragmentation and reassembling: partition a fragment into smaller packets
    - removed in IPv6
  - error detection
  - routing: best-effort to send packets from source to destination
  - certain QoS/CoS
  - does not provide reliability or reservation
- Interface:
  - send a packet to a (transport-layer) peer at a specified global destination, with certain QoS/CoS

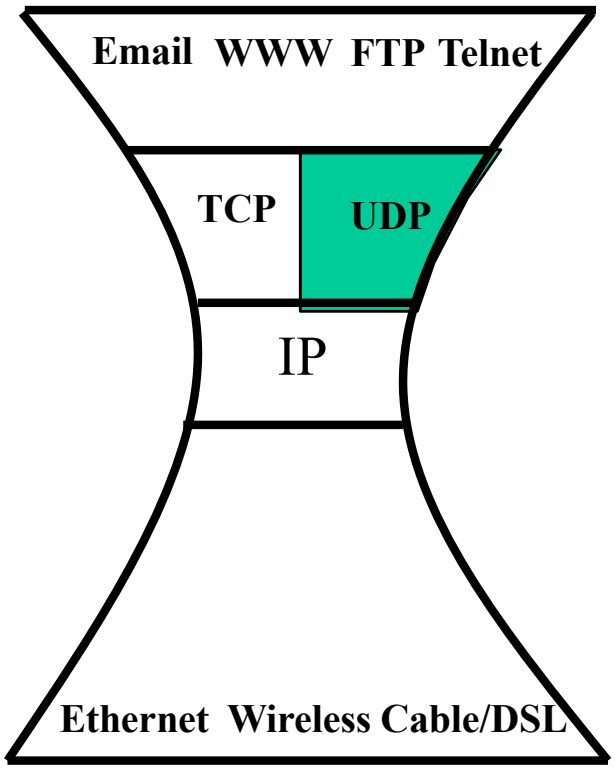


Network Layer: IPv4 Header



Transport Layer: UDP

- A connectionless service
- Does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee
  - why is there a UDP?

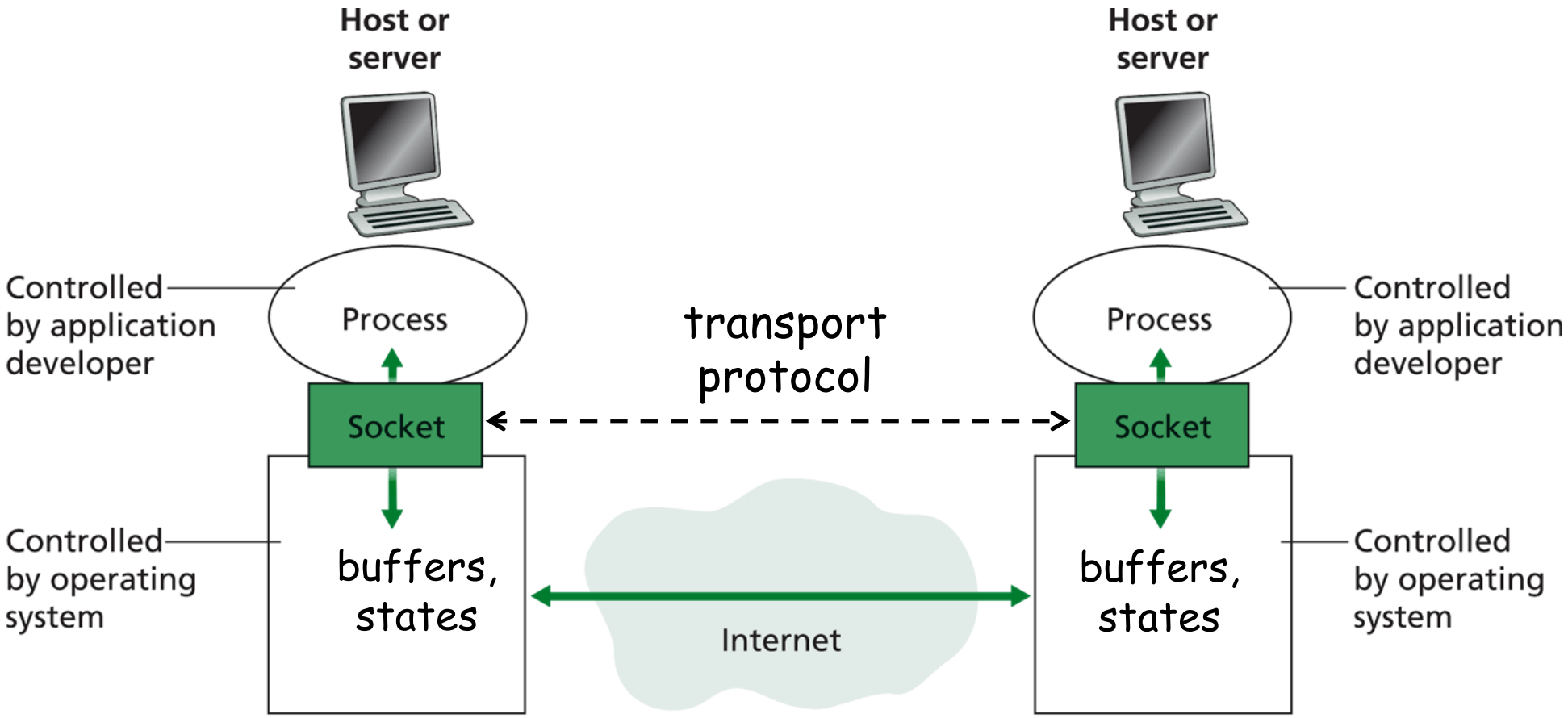


Transport Services and APIs

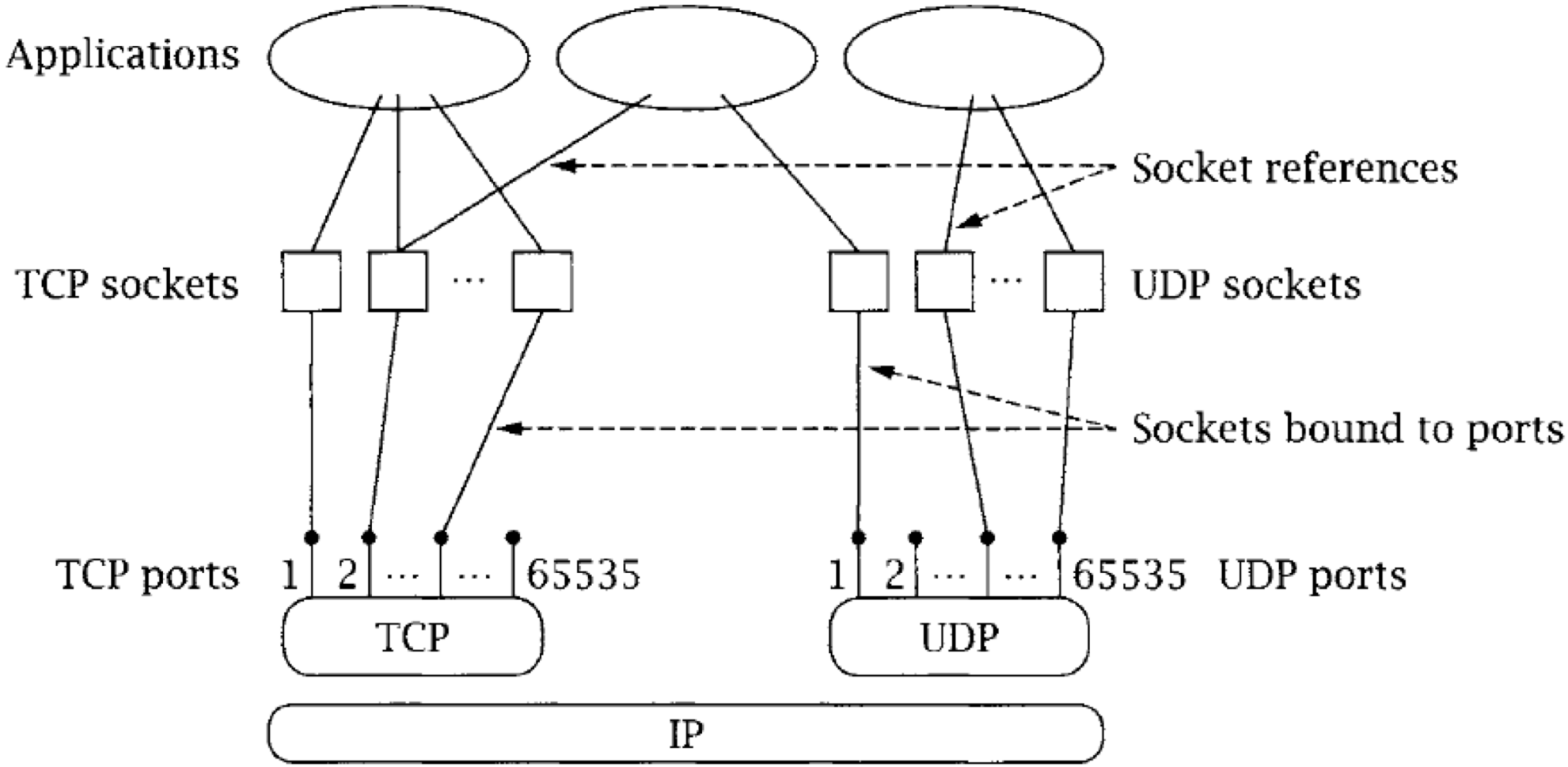
- Multiple services and APIs proposed in history
  - XTI (X/Open Transport Interface), a slight modification of the Transport Layer Interface (TLI) developed by AT&T.

- Commonly used transport-layer service model and API: Socket
  - sometimes called "Berkeley sockets" acknowledging their heritage from Berkeley Unix
  - a socket has a transport-layer local port number
    - e.g., email (SMTP) port number 25, web port number 80
  - Application can send data into socket, read data out of socket
  - an application process binds to a socket (-a all; -u udp; -n number)
    - %netstat -aun

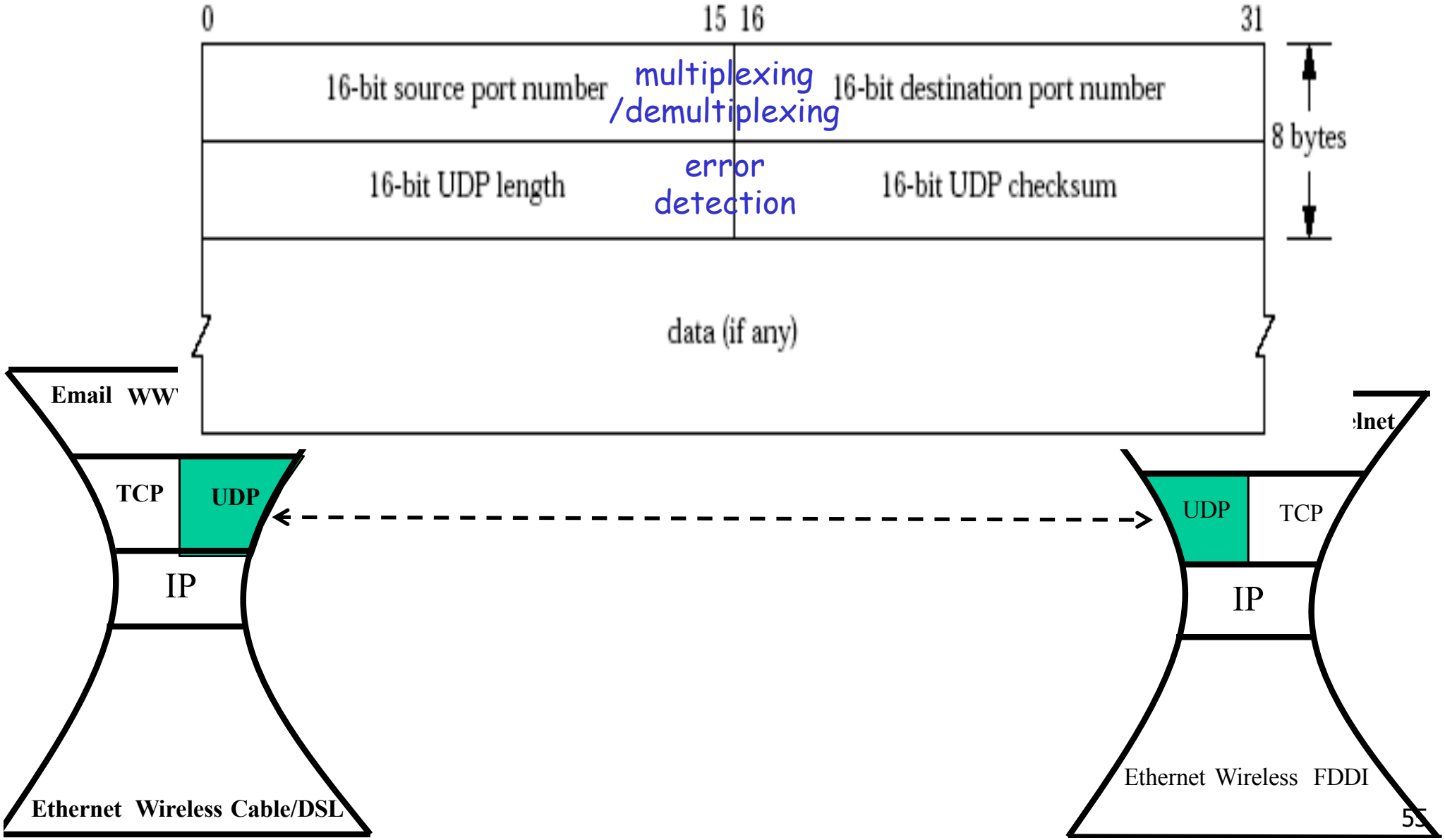
Socket Service Model and API



Multiplexing/Demultiplexing

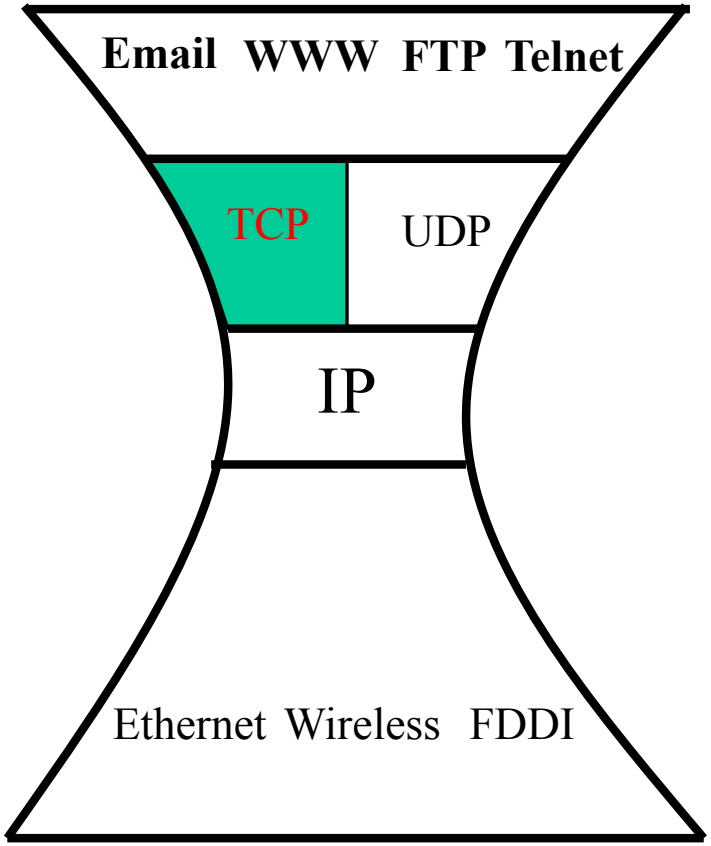


Transport Layer: UDP Header



Transport Layer: TCP

- Services
  - multiplexing/demultiplexing
  - reliable transport
    - between sending and receiving processes
    - setup required between sender and receiver: a connection-oriented service
  - flow control: sender won't overwhelm receiver
  - congestion control: throttle sender when network overloaded
  - error detection
  - does not provide timing, minimum bandwidth guarantees
- Interface:
  - send a packet to a (app-layer) peer

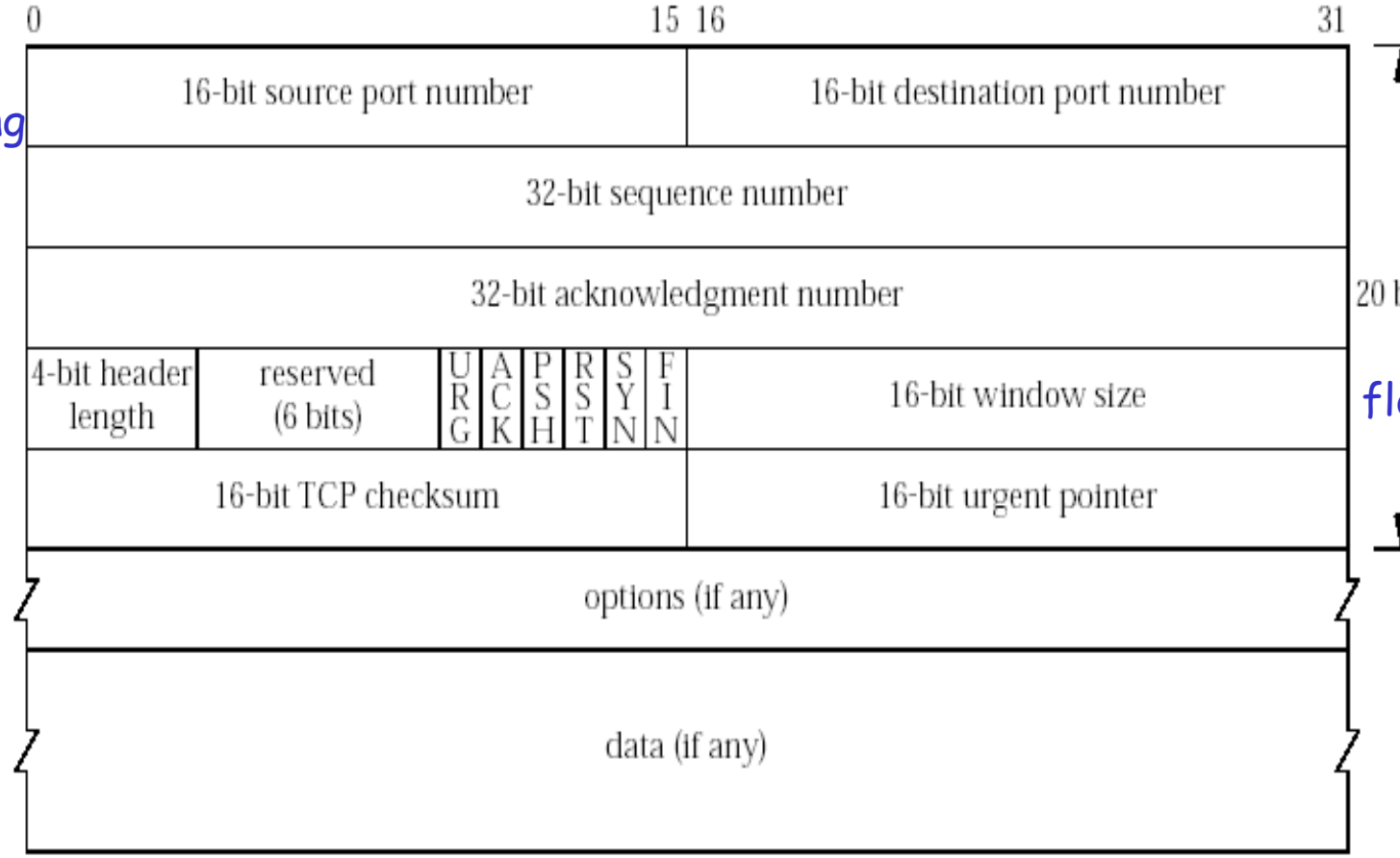


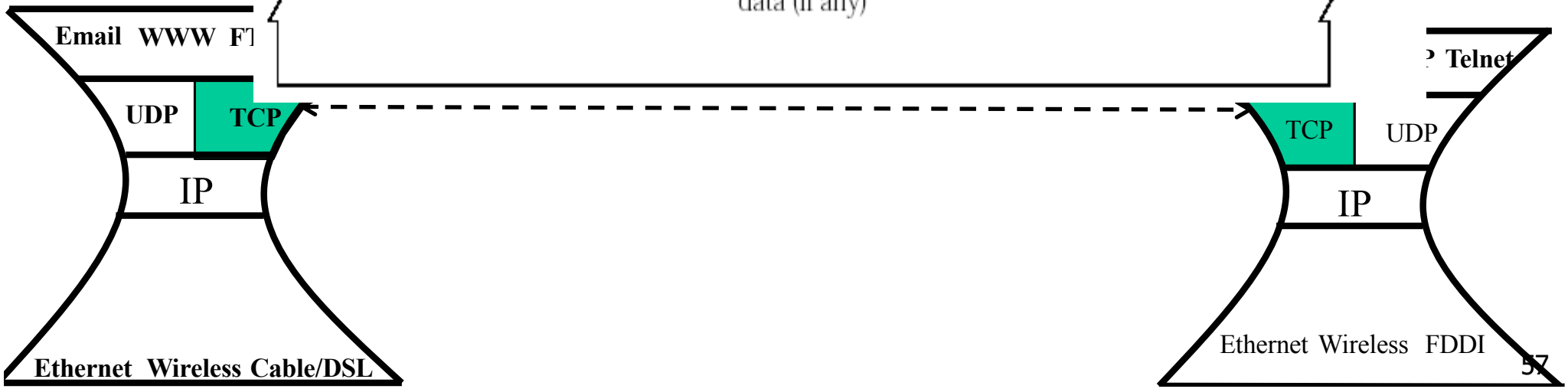
Transport Layer: TCP Header

multiplexing /demultiplexing

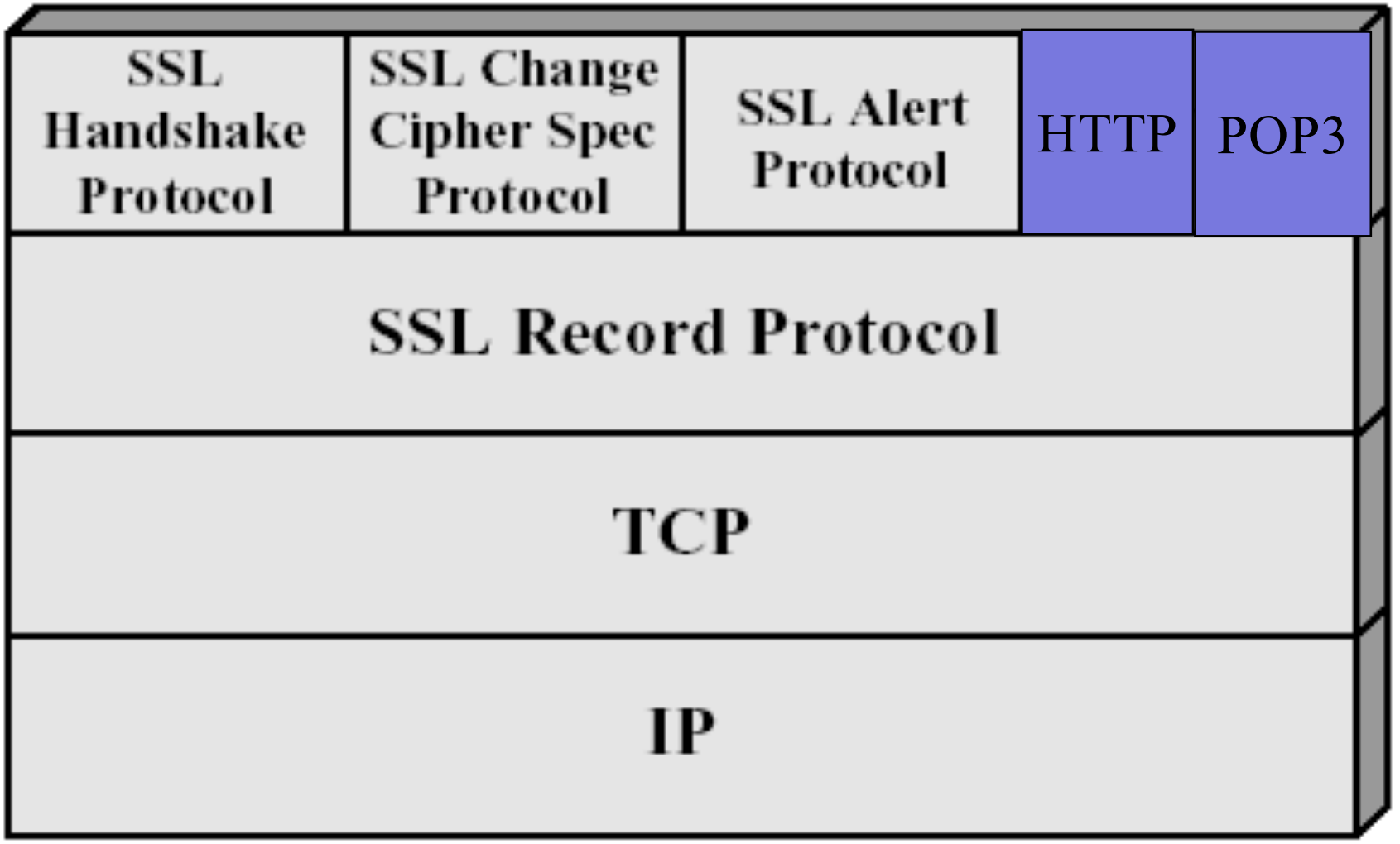
Reliable transport

congestion control error detection

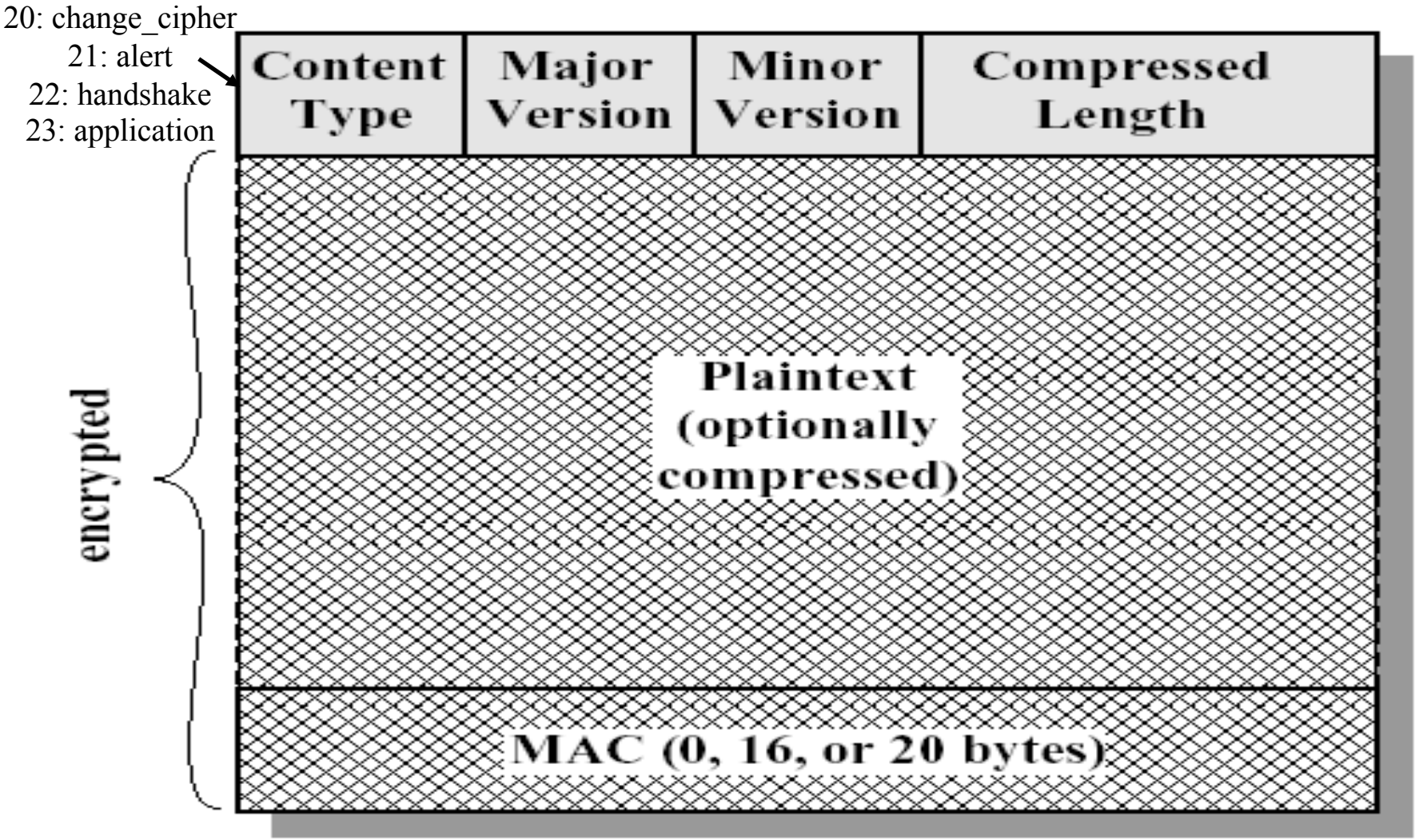




Secure Socket Layer Architecture

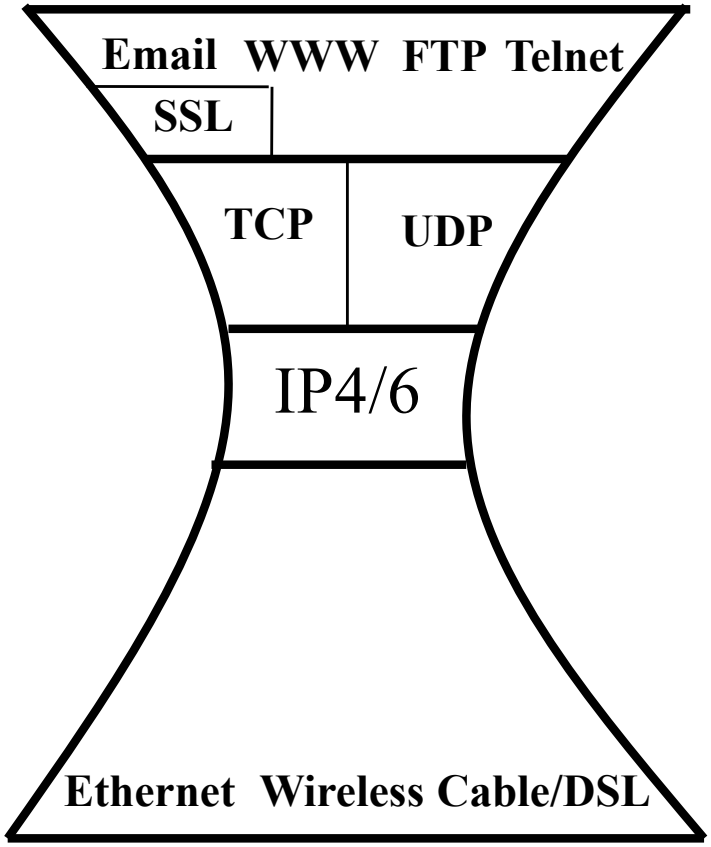


SSL Record-Layer Packet Format

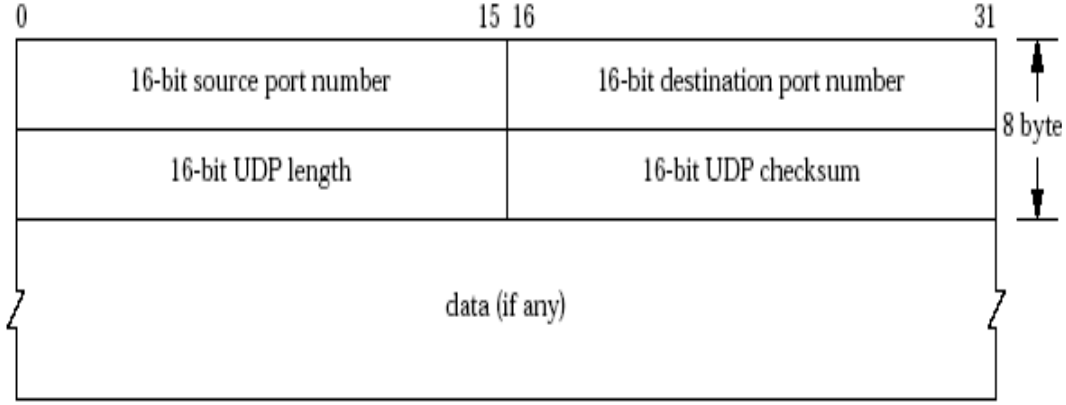


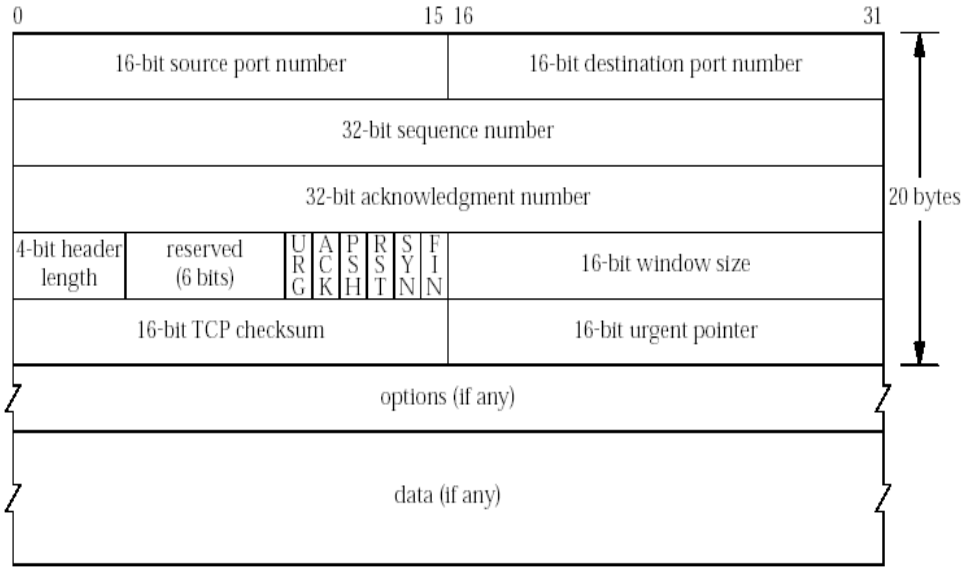
Summary: The Big Picture of the Internet

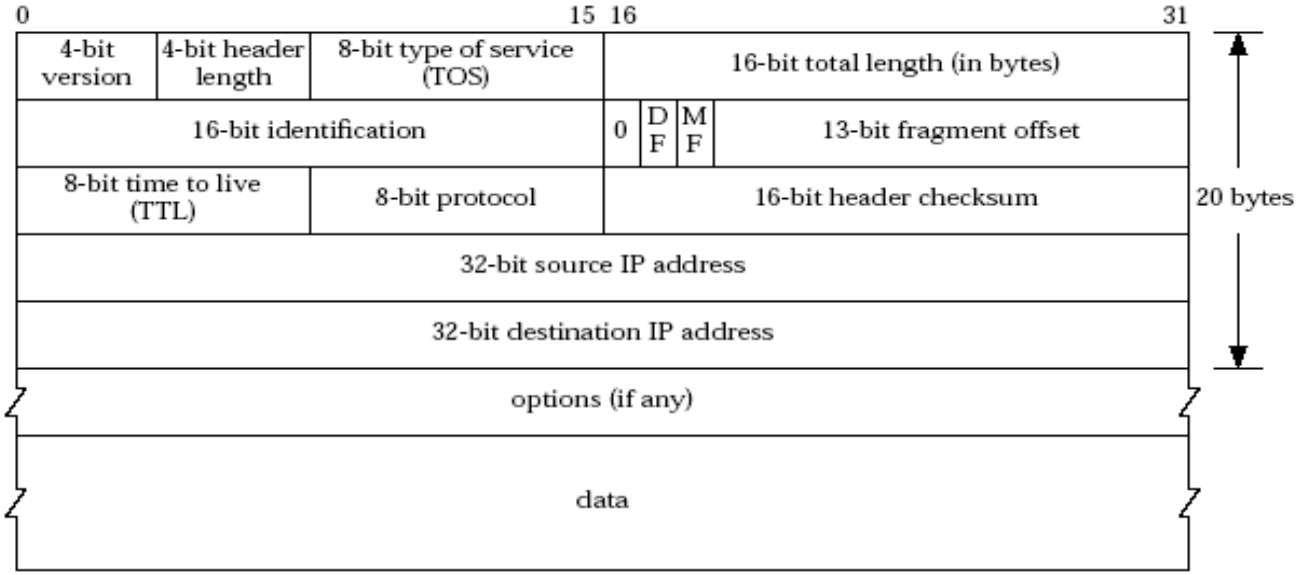
- Hosts and routers:
  - ~ 1 bil. hosts
  - autonomous systems organized roughly hierarchical
  - backbone links at 100 Gbps
- Software:
  - datagram switching with virtual circuit support at backbone
  - layered network architecture
    - use end-to-end arguments to determine the services provided by each layer
  - the hourglass architecture of the Internet

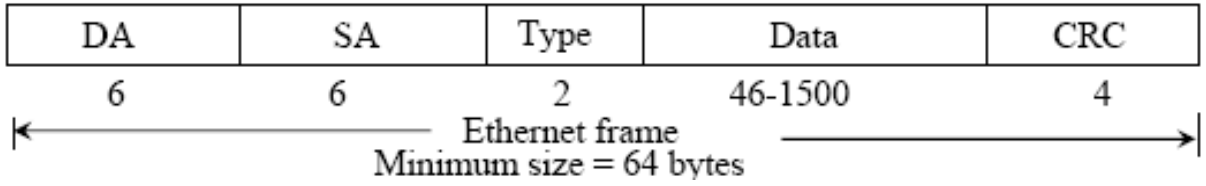


Protocol Formats









DEMO: SMTP

C: auth login
S: 334 VXNlcm5hbWU6
C: eG11Y25ucw==
S: 334 UGFzc3dvcmQ6
C: MzM0ZjU2MDVkZjE1MDRmOQ==
S: 235 OK Authenticated
C: mail from:xmucnns@sina.com
S: 250 ok
C: rcpt to:qiaoxiang@xmu.edu.cn
S: 250 ok
C: data
S: 354 End data with <CR><LF>.<CR><LF>
C: Date:2021-9-22 12:36
C: From:xmucnns@sina.com
C: To:qiaoxiang@xmu.edu.cn
C: Subject:test smtp
C:
C: Hello, Qiao.
C:
C: .
S: 250 ok queue id 11479549283321
C: quit
S: 221 smtp-97-27.smtpsmail.fmail.bx.sinanode.com
S: Connection closed by foreign host.

Backup Slides

The Design Philosophy of the DARPA Internet

Goals

0. **Connect different networks**
1. Survivability in the face of failure
2. Support multiple types of services
3. Accommodate a variety of networks

4. Permit distributed management of resources
5. Be cost effective
6. Permit host attachment with a low level of effort
7. Be accountable

Survivability in the Face of Failure: Questions

- What does the goal mean?
- Why is the goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Survivability in the Face of Failure

- Continue to operate even in the presence of network failures (e.g., link and router failures)
  - as long as the network is not partitioned, two endpoints should be able to communicate…moreover, any other failure (excepting network partition) should be transparent to endpoints
- Decision: maintain state only at end-points (fate-sharing)
  - eliminate the problem of handling state inconsistency and performing state restoration when router fails
- Internet: stateless network architecture

Support Multiple Types of Service: Questions

- What does this goal mean?
- Why is the goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Support Multiple Types of Service

- Add UDP to TCP to better support other types of applications
  - e.g., "real-time" applications
- This was arguably the main reason for separating TCP and IP
- Provide datagram abstraction: lower common denominator on which other services can be built: everything over IP
  - service differentiation was considered (remember ToS?), but this has never happened on the large scale (Why?)

Support a Variety of Networks: Questions

- What does the goal mean?
- Why is this goal important?
- How does the Internet achieve this goal?
- Does the Internet achieve this goal (or in what degree does the Internet achieve this goal)?

Support a Variety of Networks

- Very successful
  - because the minimalist service; it requires from underlying network only to deliver a packet with a "reasonable" probability of success
- …does not require:
  - reliability
  - in-order delivery
- The mantra: IP over everything
  - Then: ARPANET, X.25, DARPA satellite network..
  - Now: ATM, SONET, WDM…

Other Goals

- Permit distributed management of resources
- Be cost effective
- Permit host attachment with a low level of effort
- Be accountable