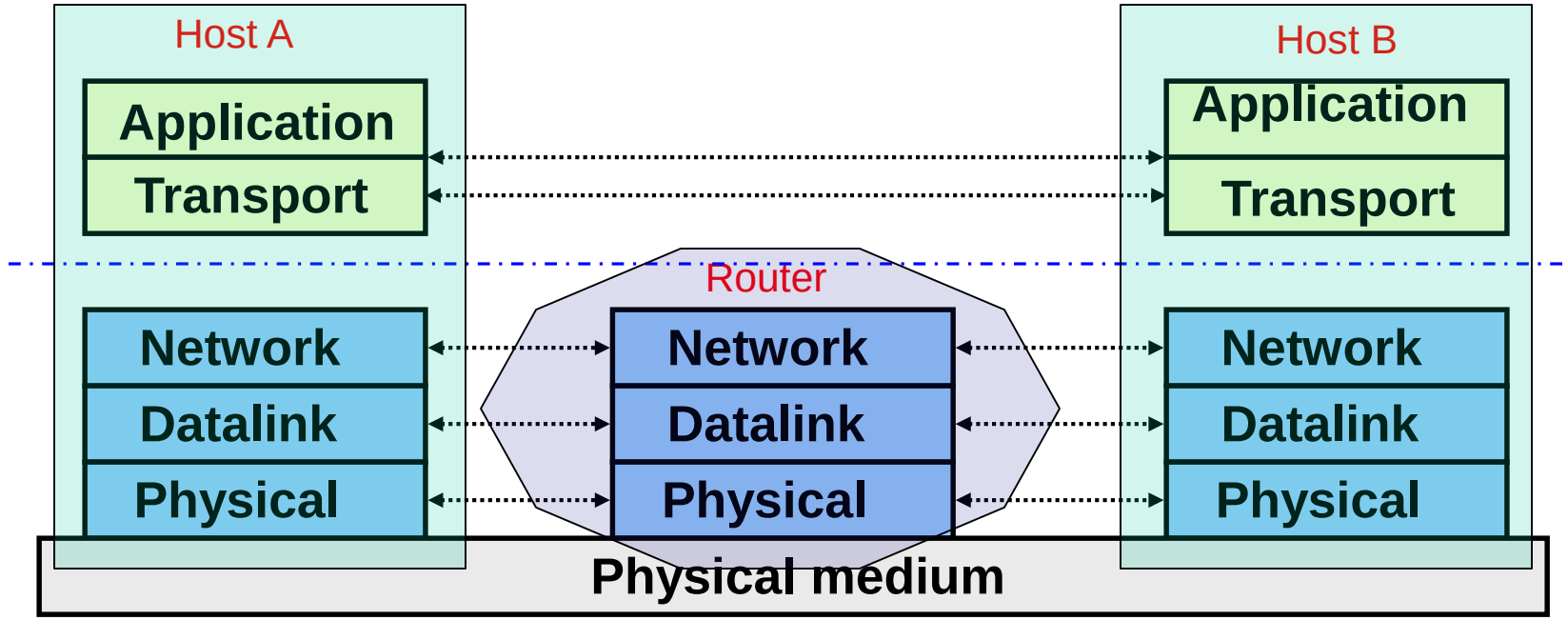
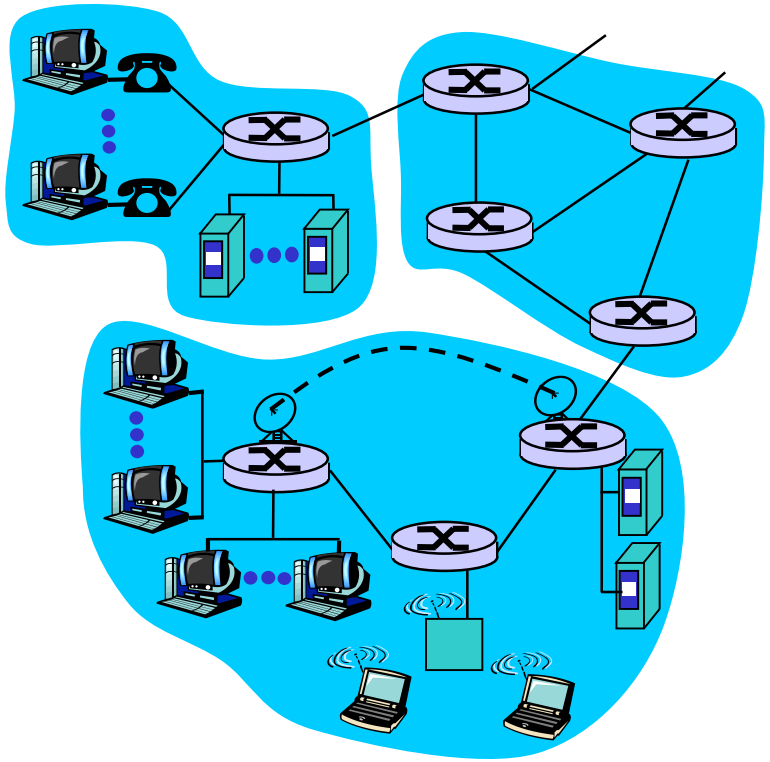
"Dumb" Networks & "Smart" End Systems

- Five Layer Architecture:
  - Lower three layers are implemented everywhere
  - Top two layers are implemented only at hosts



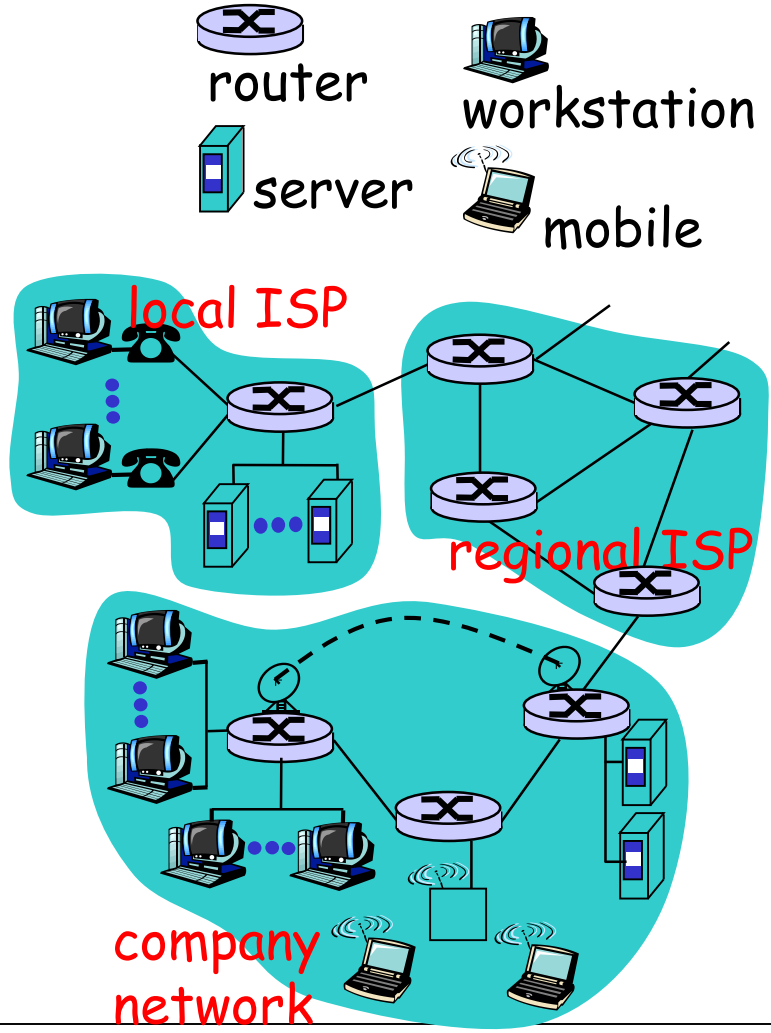
An Overview of Network Structure:
a "horizontal view"

- network edge:
  applications and hosts
- network core:
  - routers
  - network of networks
- access networks, physical media:
  communication links



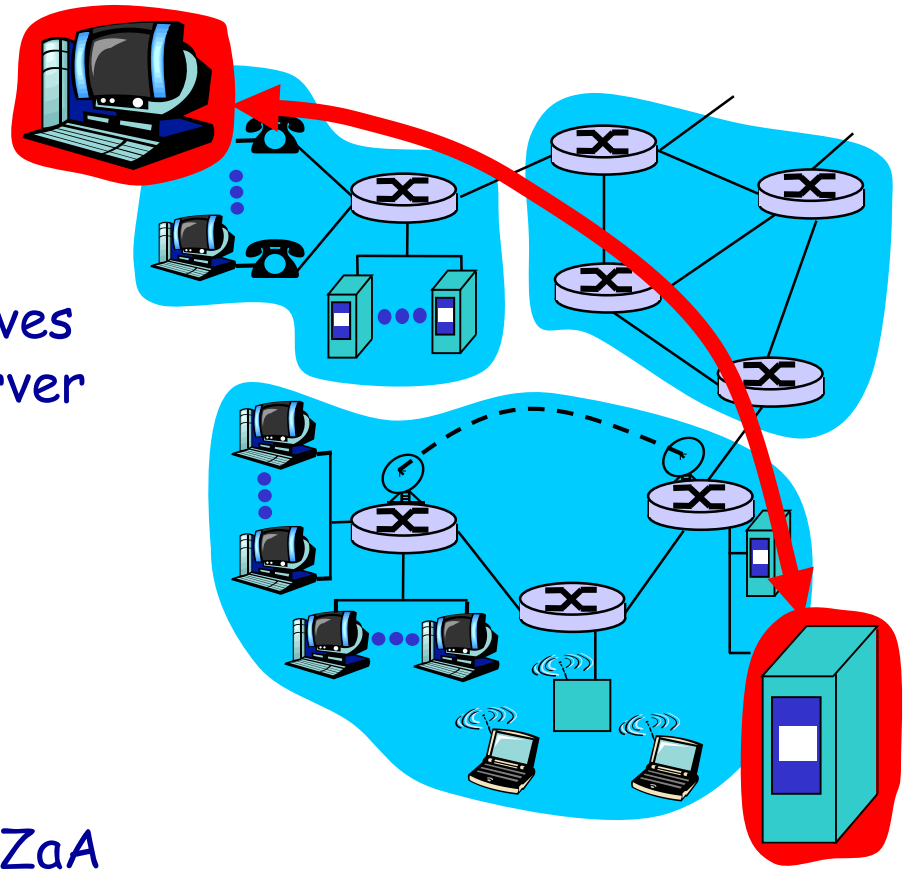
What's the Internet: "nuts and bolts" view

- millions of connected computing devices: *hosts = end systems*
- running *network apps*
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = ***bandwidth***
- *routers:* forward packets (chunks of data)



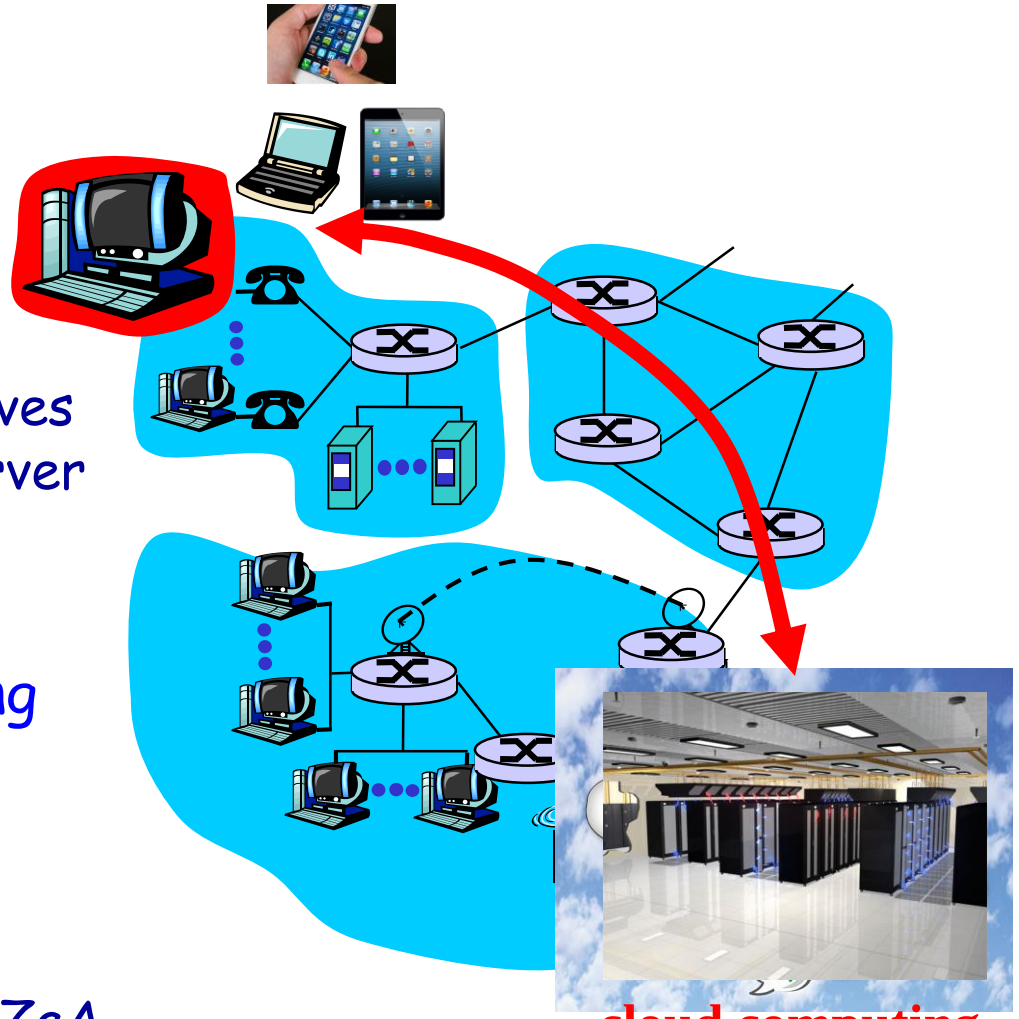
The network edge:

- end systems (hosts):
  - run application programs
  - e.g. Web, email
  - at "edge of network"
- client/server model
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
- peer-peer model:
  - minimal (or no) use of dedicated servers
  - e.g. Skype, BitTorrent, KaZaA



The network edge:

- end systems (hosts):
  - run application programs
  - e.g. Web, email
  - at "edge of network"
- client/server model
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
  - Cloud & Mobile Computing
- peer-peer model:
  - minimal (or no) use of dedicated servers
  - e.g. Skype, BitTorrent, KaZaA



cloud computing

Network edge: connection-oriented service

Goal: data transfer between end systems

- *handshaking*: setup (prepare for) data transfer ahead of time
  - Hello, hello back human protocol
  - *set up "state"* in two communicating hosts
- TCP - Transmission Control Protocol
  - Internet's connection-oriented service

TCP service [RFC 793]

- *reliable, in-order* byte-stream data transfer
  - loss: acknowledgements and retransmissions
- *flow control:*
  - sender won't overwhelm receiver
- *congestion control:*
  - senders "slow down sending rate" when network congested

Network edge: connectionless service

Goal: data transfer between end systems
- same as before!

- UDP - User Datagram Protocol [RFC 768]:
  - connectionless
  - unreliable data transfer
  - no flow control
  - no congestion control

App's using TCP:

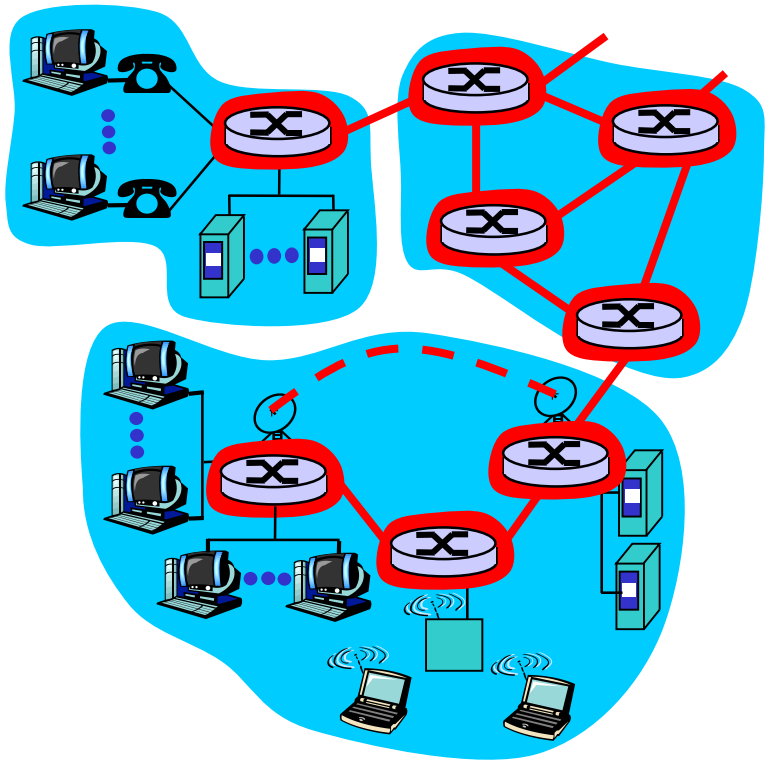
- HTTP (Web), FTP (file transfer), Telnet (remote login), SMTP (email), Flash videos, DASH stream videos

App's using UDP:

- streaming media, teleconferencing, DNS, Internet telephony

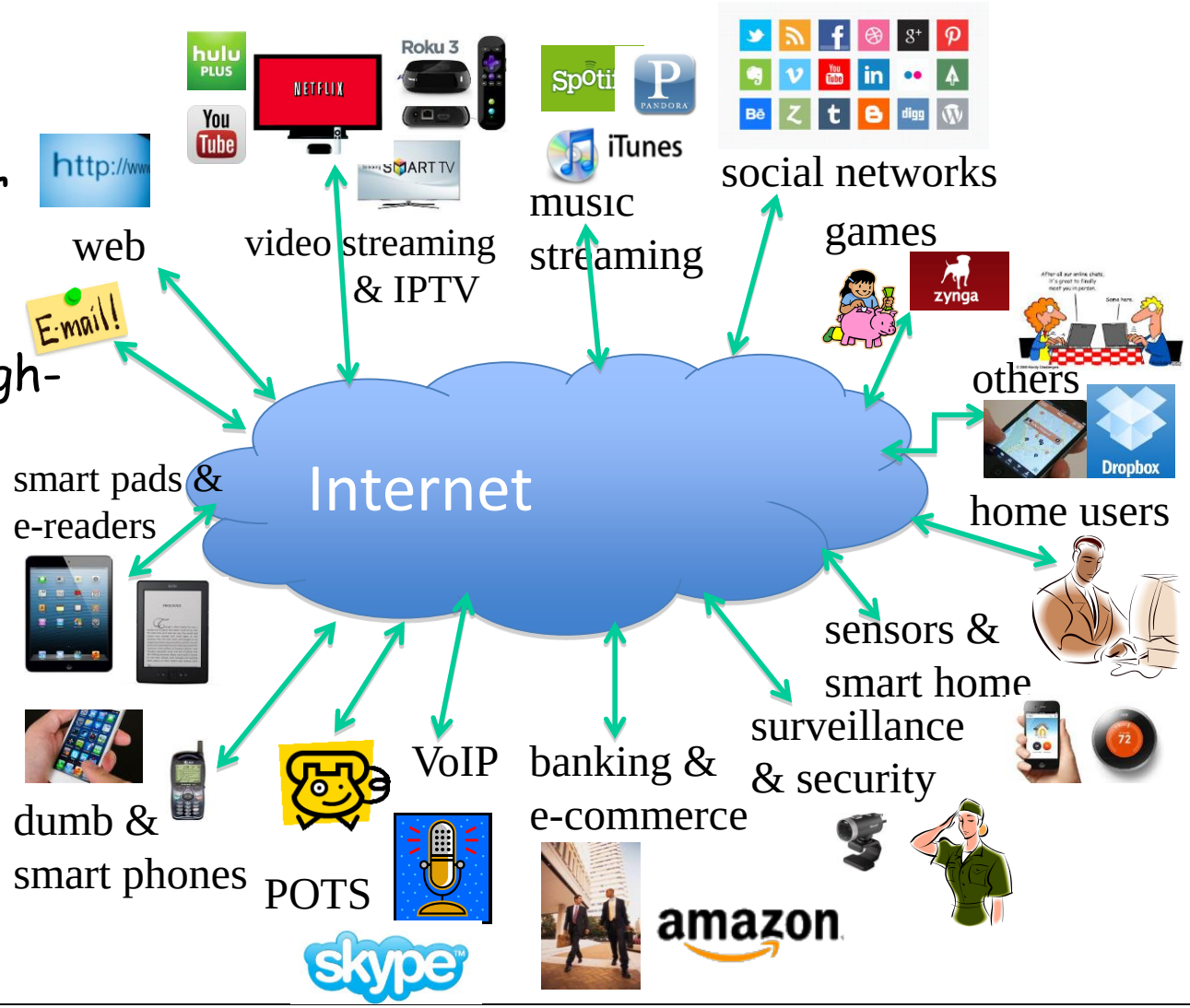
The Network Core

- mesh of interconnected routers shared by many users
- *the* fundamental questions:
  - how network is shared
  - how to find the other party (person, website, …) you want
  - how is data transferred through net?



On the Internet Edge …

- Large # of (mobile & stationary) users
- Large # of "dumb" or smart devices & appliances
- Some "always-on," high-speed connection
- Others intermittent connectivity with varying bandwidth
- Diverse applications and services
- Heterogeneous technologies



Within the Internet "Cloud"

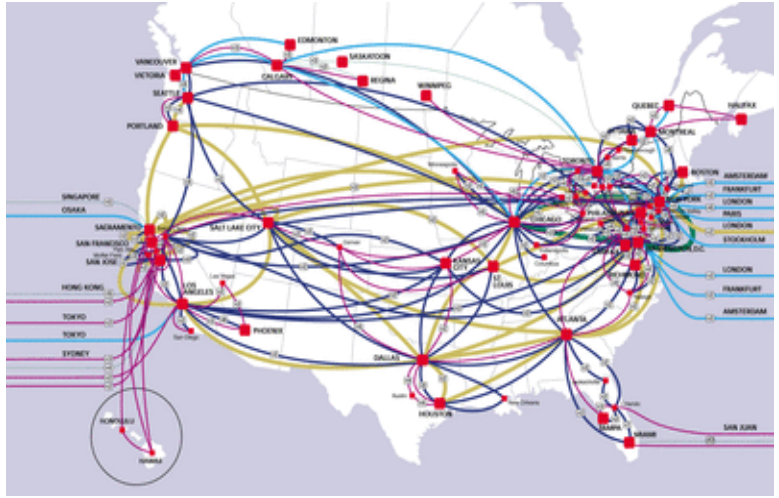
Network Core:

- big ISPs (& cellular providers) with large geographical span
- As well as medium & smaller ISPs

And the "other end/edge":

- big content providers with huge data centers

- High bandwidth, dense and rich topology
- Enormous computing & storage capacities to support cloud, mobile computing/services



Network Architecture
(or organizational principles)

Networks are complex!

- many "pieces":
  - hosts
  - routers
  - links of various media
  - hardware, software
  - applications
  - protocols
  - …..

Question:

Is there any hope of *organizing* structure or principle of network?

Or at least our discussion of networks?

Network architecture:

"blue prints" (or principles) regarding functional division and function placement

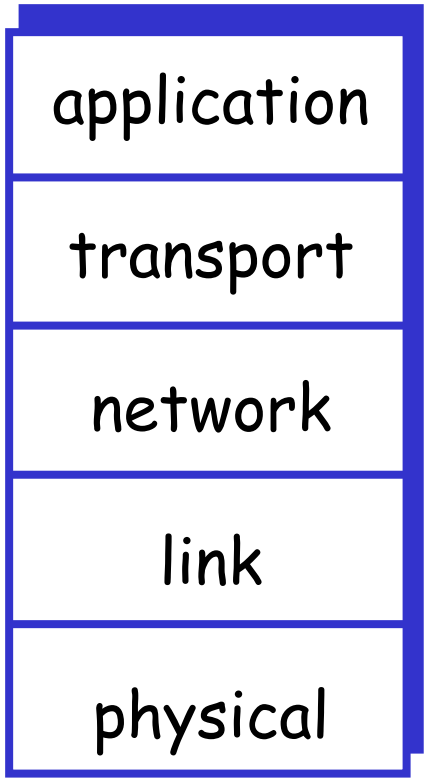
Why Layering?

Dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
  - layered reference model for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system

Internet Protocol Stack

- application: supporting network applications
  - FTP, SMTP, HTTP, DASH, …
- transport: process-process data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - PPP, Ethernet
- physical: bits "on the wire"



Layered Architecture

- *Layering* simplifies the architecture of complex system
- Layer N relies on *services* from layer N-1 to provide a *service* to layer N+1
- *Interfaces* define the services offered
- Service required from a lower layer is independent of its implementation
  - Layer N change doesn't affect other layers
  - Information/complexity hiding
  - Similar to object oriented methodology