This lecture will be recorded!

Welcome to

CS 3516:
Computer Networks

Prof. Yanhua Li

Time: 9:00am –9:50am M, T, R, and F

Zoom Lecture

Fall 2020 A-term

Chapter 1: roadmap

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

protocol layers,
service models

Protocol "layers"

Networks are complex, with many "pieces":

- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

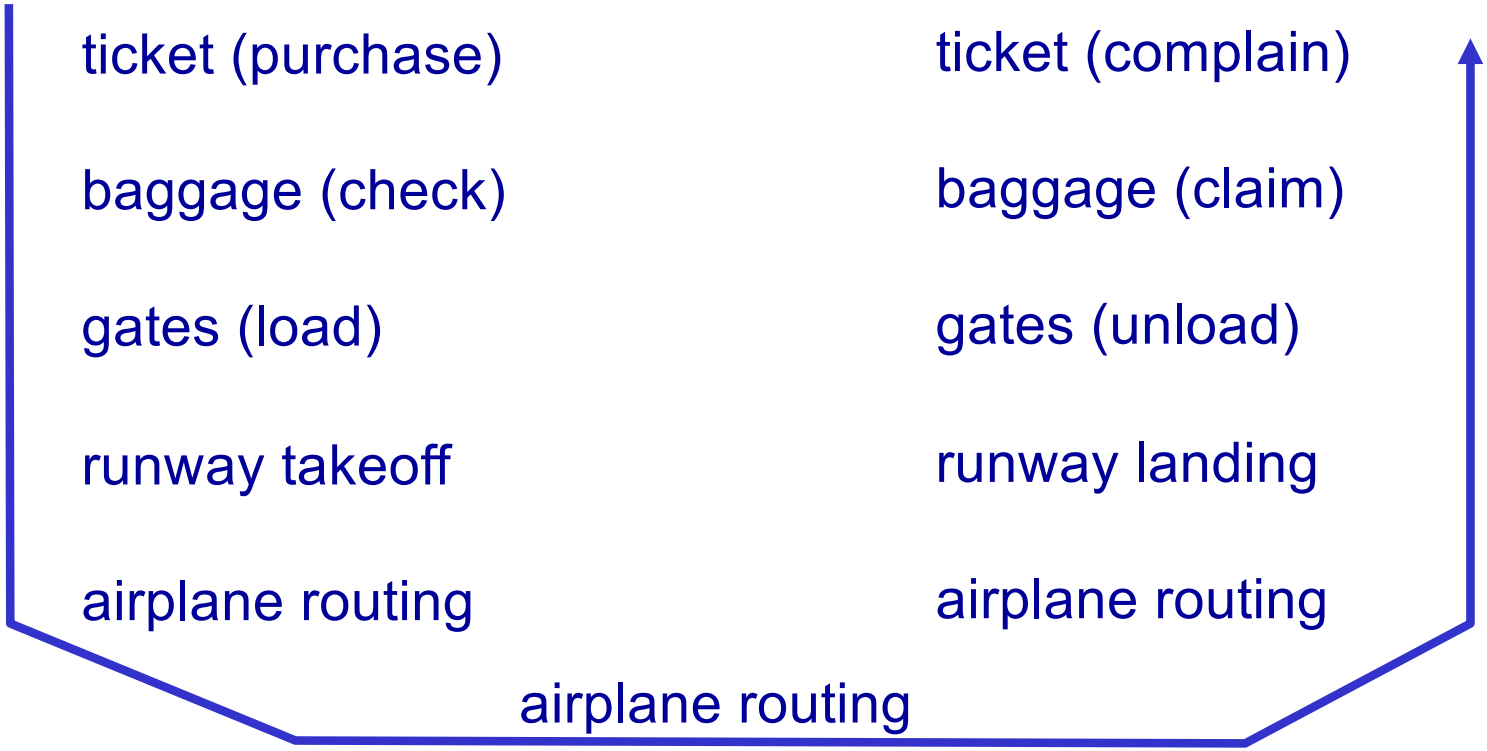
- Performance measures

Question:

is there any hope of *organizing* network processes in a structured way?

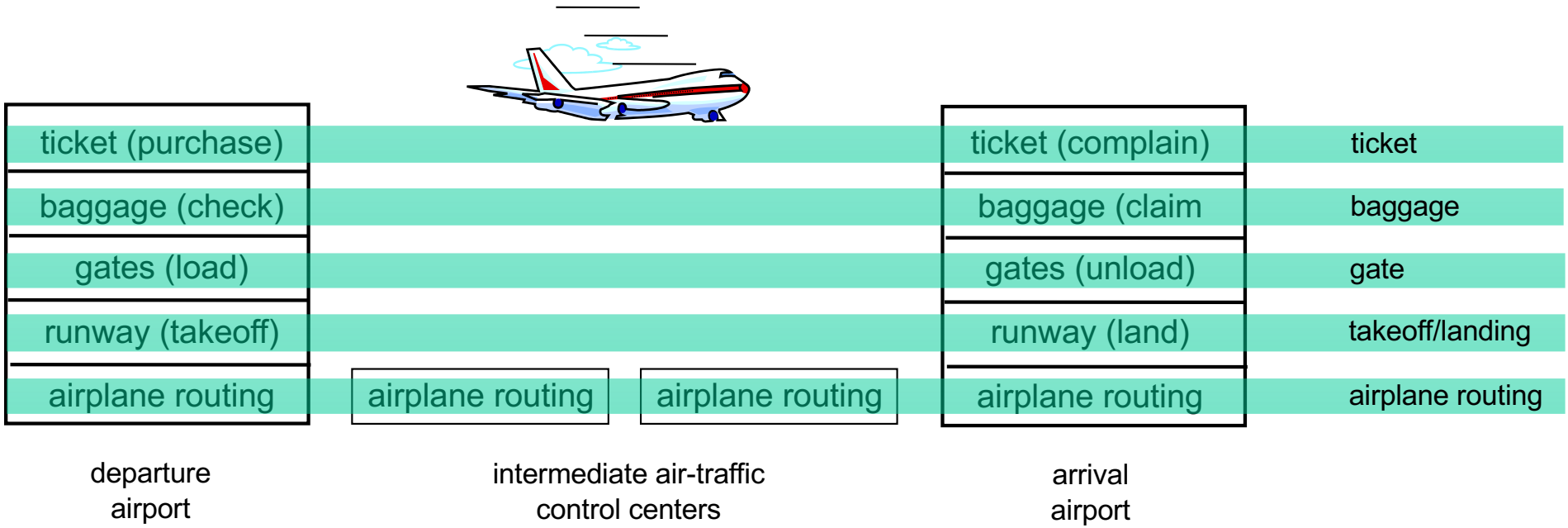
…. or at least our discussion of networks?

Organization of air travel



- a series of steps

Layering of airline functionality

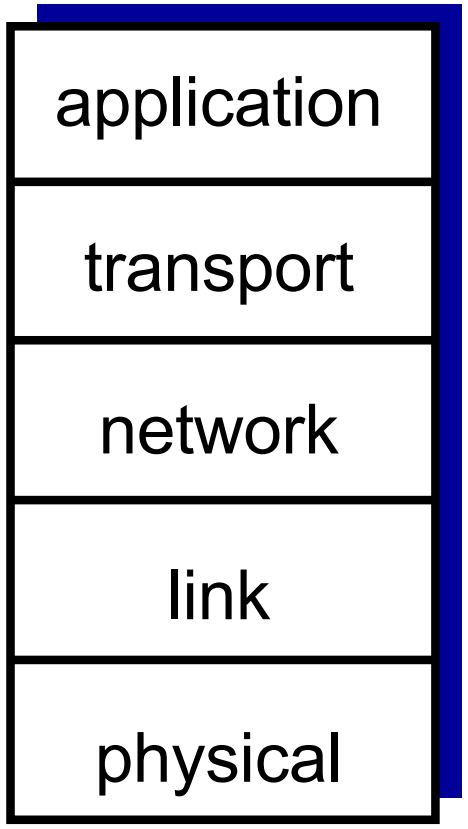


layers: each layer implements a service

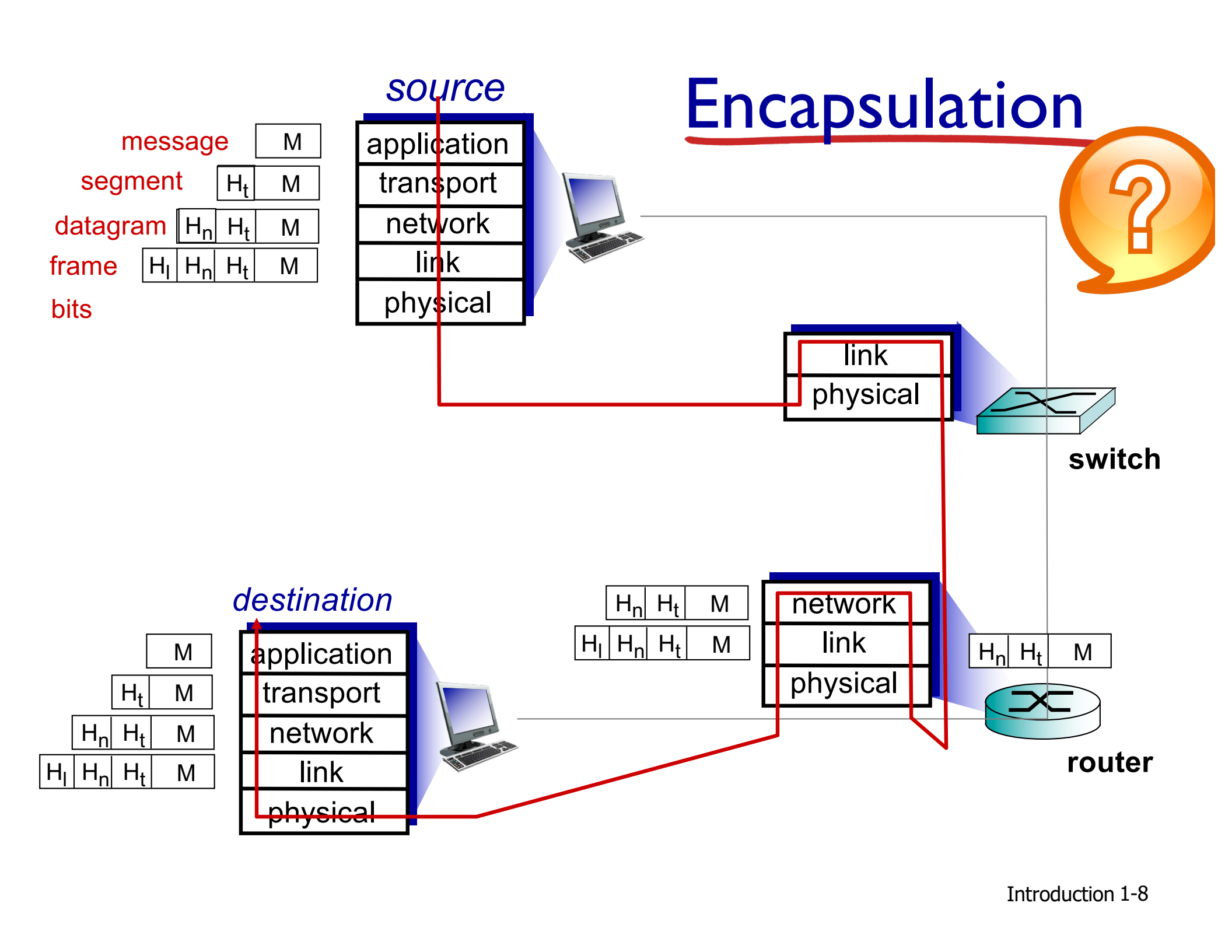
- via its own internal-layer actions
- relying on services provided by layer below

Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP, DNS
- *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
  - Ethernet, 802.11 (WiFi), PPP
- *physical:* bits "on the wire"



Encapsulation



Introduction: summary

covered a "ton" of material!

- S1: Internet overview
- S1: what's a protocol?
- S2: network edge, core, access network
  - packet-switching versus circuit-switching
  - Internet structure
- S3: Socket programming
- S4: performance: loss, delay, throughput
- S5: layering, service models

you now have:

- context, overview, "feel" of networking
- more depth, detail *to follow!*

Course Progression

- Week 1-2: Overview
- Week 2-4: Application Layer Protocols
  - P2P, HTTP, SMTP, DNS
- Week 4-5: Transport Layer Protocols
  - UDP and TCP
- Week 6: IP, Routing Protocols
- Week 7: Link Layer Protocols, Wireless & Data Center Networking

Online social networks

Voice call

Online search service

Online shopping

Video Streaming

Some network apps

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- voice over IP (e.g., Skype)
- real-time video conferencing
- social networking
- search
- …
- …

Chapter 2: application layer

our goals:

- conceptual, implementation aspects of network application protocols
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
  - HTTP
  - SMTP
  - DNS
  - P2P
- creating network applications
  - socket API

2.1 principles of network applications

- app architectures
- app requirements

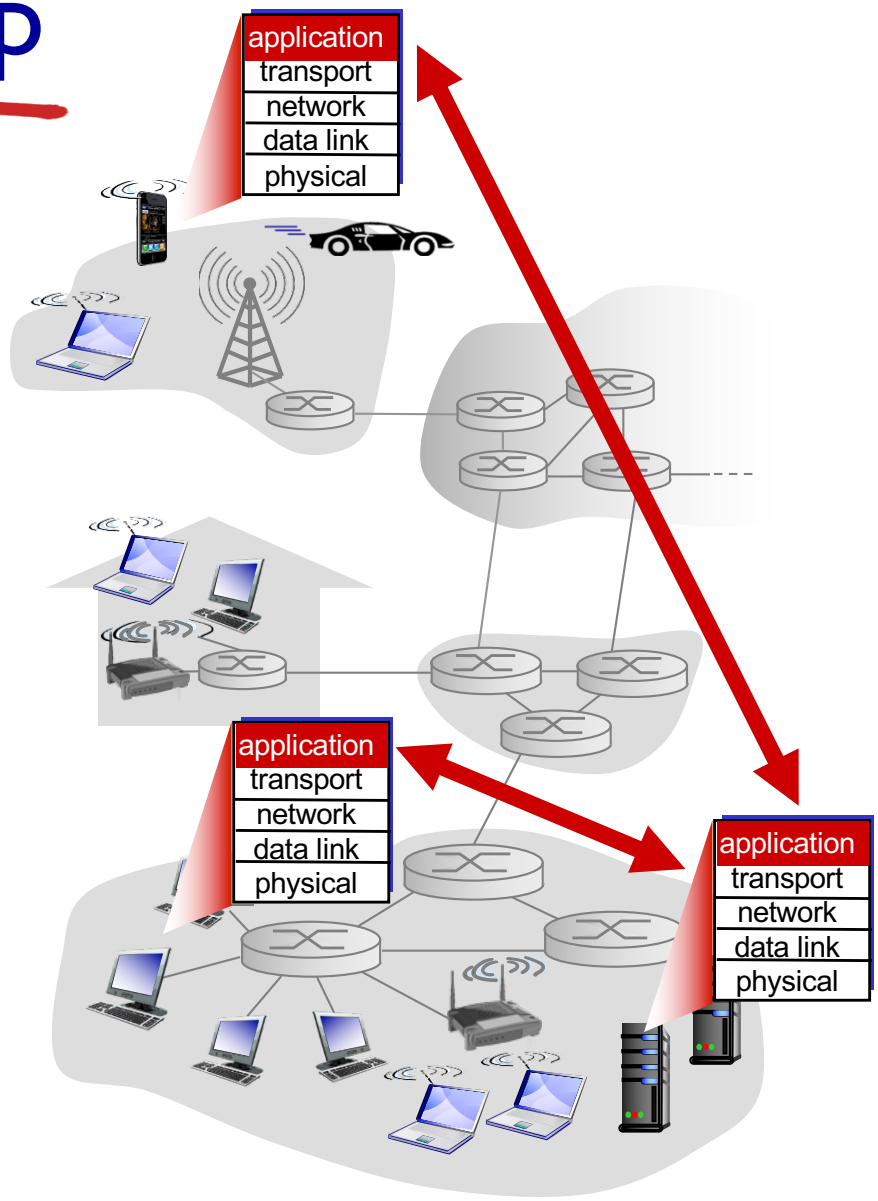
Creating a network app

write programs that:

- run on (different) ***end systems***
- communicate **over network**
- e.g., **web server software** communicates with **browser software**

no need to write software for network-core devices

- **network-core devices** do not run user applications
- applications on end systems allows for **rapid app development, propagation**

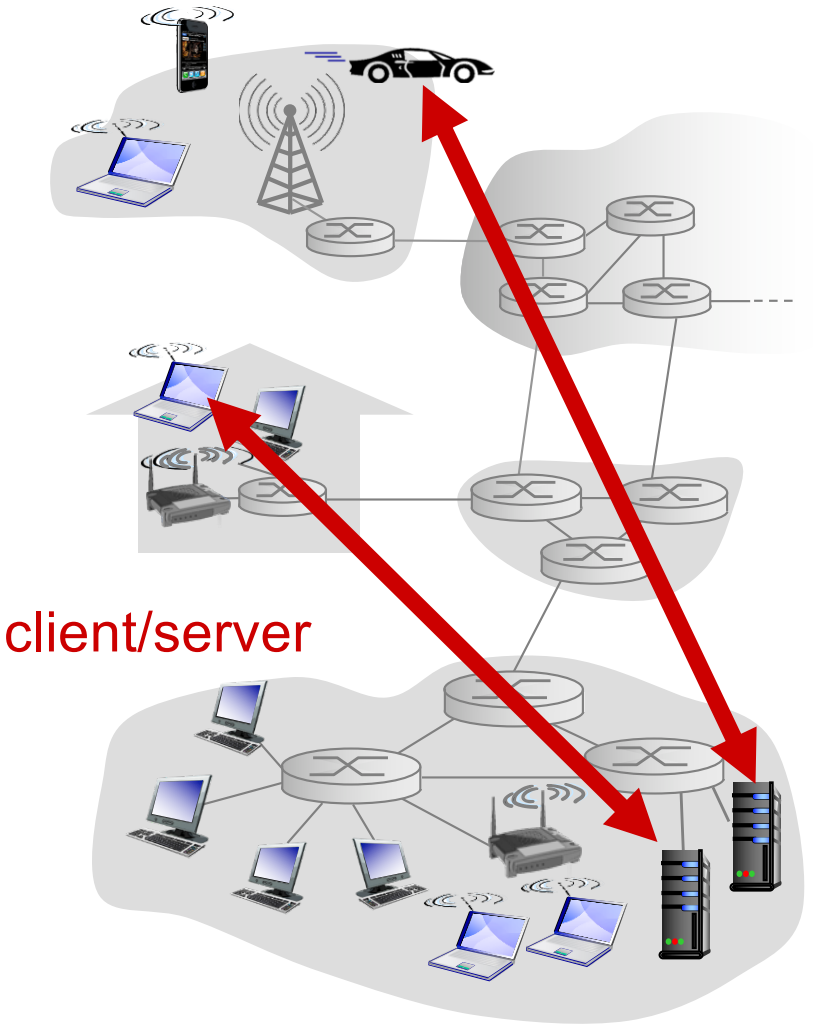


Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)

Client-server architecture



server:

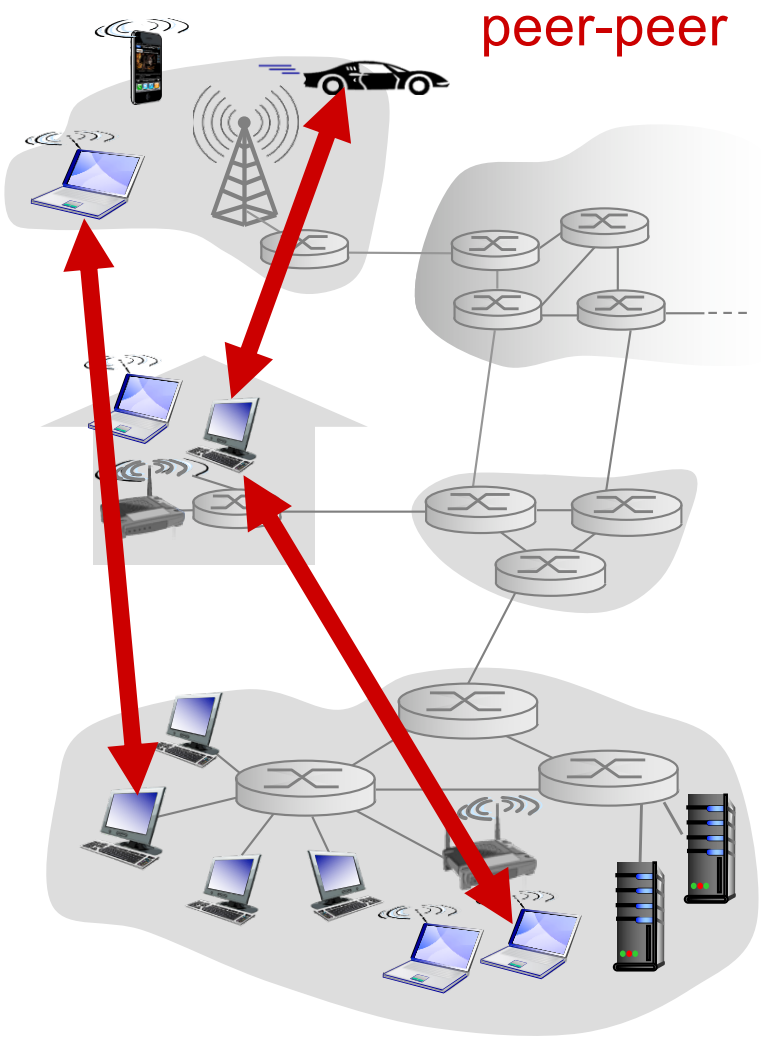
- always-on host
- permanent IP address
- data centers for scaling

clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

P2P architecture

- ***no*** always-on server
- **arbitrary end systems** directly communicate
- **peers** request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are **intermittently** connected and change IP addresses
  - complex management



Processes communicating

process: program running within a host

- **within same host**, two processes communicate using inter-process communication (defined by OS)
- processes **in different hosts** communicate by exchanging messages via sockets

clients, servers

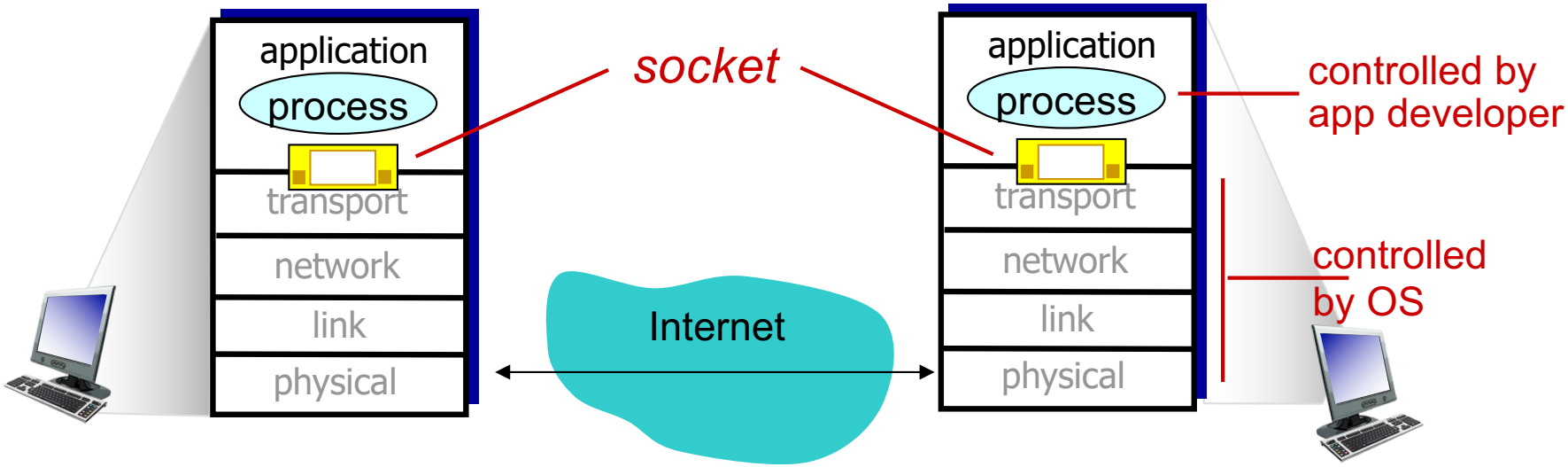
client process: process that initiates communication

server process: process that waits to be contacted

- aside: applications with P2P architectures have client processes & server processes

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door / mail box
  - sending process shoves message out door / drop it to mail box
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
  - *A:* no, *many* processes can be running on same host

- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80
- more shortly…

Questions?