Τεχνολογίες Διαδικτύου 2025-26 (DIT 315)

Δρ. Ειρήνη Λιώτου

eliotou@hua.gr

7/10/2025

Διάρθρωση μαθήματος

- **Διαλέξεις:**
  - Θεωρία
  - Knowledge checks
  - Interactive exercises

Διάρθρωση μαθήματος

- **Τελική εξέταση:**
  - Quiz 20 ερωτήσεων (70%)
  - Εργασία παρουσίασης ερευνητικών εργασιών στο state of the art των Τεχνολογιών Διαδικτύου (30%) → *οδηγίες σε ξεχωριστό pdf*
  - Ποσοστό επιτυχίας: > 80%
  - Μ.Ο. ~7.0-7.5
  - Ύλη & Πλάνο εξετάσεων
- **Κυρίως σύγγραμμα:**
  - KUROSE & ROSS, «Δικτύωση Υπολογιστών: Προσέγγιση από Πάνω προς τα Κάτω», 8η έκδοση, Εκδόσεις Γκιούρδας (Κεφάλαια 2, 6)
  - KUROSE & ROSS, «Δικτύωση Υπολογιστών: Προσέγγιση από Πάνω προς τα Κάτω», 7η έκδοση, Εκδόσεις Γκιούρδας (Κεφάλαια 9)
  - https://gaia.cs.umass.edu/kurose_ross/index.php

Σχεδιάγραμμα μαθήματος (draft)

Διάλεξη Α/Α	ΠΕΡΙΕΧΟΜΕΝΟ ΔΙΑΛΕΞΗΣ	ΗΜΕΡΟΜΗΝΙΑ
2025		
1	**Εισαγωγή στα δίκτυα υπολογιστών και το διαδίκτυο**: Επίπεδα πρωτοκόλλων, ενθυλάκωση **Επίπεδο εφαρμογής**: Αρχές δικτυακών εφαρμογών, αρχιτεκτονική web	7/10
2	**Επίπεδο εφαρμογής**: πρωτόκολλα HTTP, FTP, web caching, cookies	14/10
3	**Επίπεδο εφαρμογής**: Ηλεκτρονικό ταχυδρομείο (SMTP, POP3, IMAP), υπηρεσία καταλόγου διαδικτύου (DNS)	21/10
4	**Επίπεδο εφαρμογής**: P2P διαμοιρασμός αρχείων, Διανομή αρχείων, βίντεο συνεχούς ροής και DASH, Netflix, YouTube	4/11
5	**Επίπεδο εφαρμογής**: Προγραμματισμός socket	18/11
6	**Δικτύωση πολυμέσων**: Δικτυακές εφαρμογές πολυμέσων, UDP συνεχούς ροής, HTTP συνεχούς ροής, Voice over IP	25/11
7	**Δικτύωση πολυμέσων**: SIP, RTP πρωτόκολλα, πολλαπλές κλάσεις υπηρεσίας, ενοποιημένες και διαφοροποιημένες υπηρεσίες (Diffserv,IntServ), Ποιότητα υπηρεσίας (Quality of Service - QoS), πρωτόκολλο δέσμευσης πόρων (RSVP)	2/12
8	Η ζωή μιας ιστοσελίδας (πρωτόκολλα στην «πράξη»), Wireshark	9/12
9	Επανάληψη	16/12
2026		
10	Παρουσίαση εργασιών φοιτητών 1/3	6/1
11	Παρουσίαση εργασιών φοιτητών 2/3	13/1
12	Παρουσίαση εργασιών φοιτητών 3/3 – αν χρειαστεί	20/1

May add more!

Protocol "layers"

Networks are complex, with many "pieces":

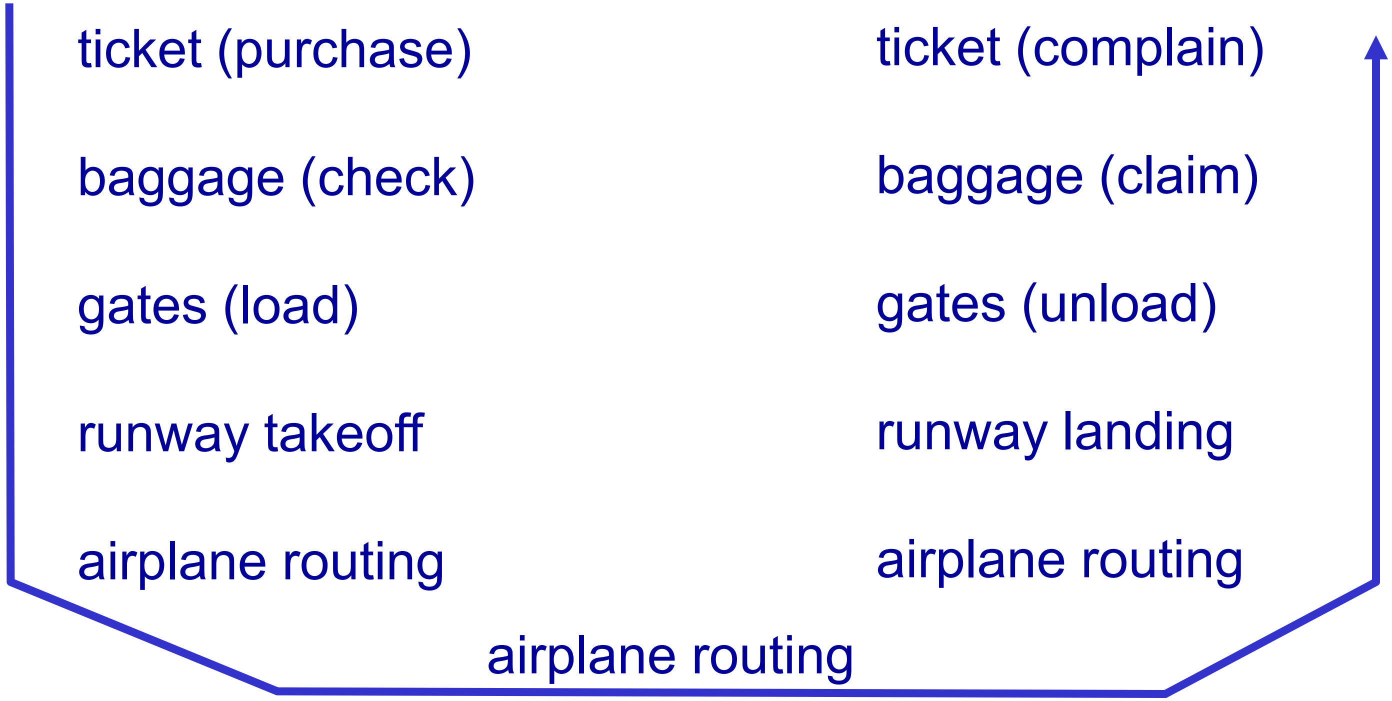
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

is there any hope of *organizing* structure of network?

…. or at least our discussion of networks?

Organization of air travel



ticket (purchase)

baggage (check)

gates (load)

runway takeoff

airplane routing

airplane routing

ticket (complain)

baggage (claim)

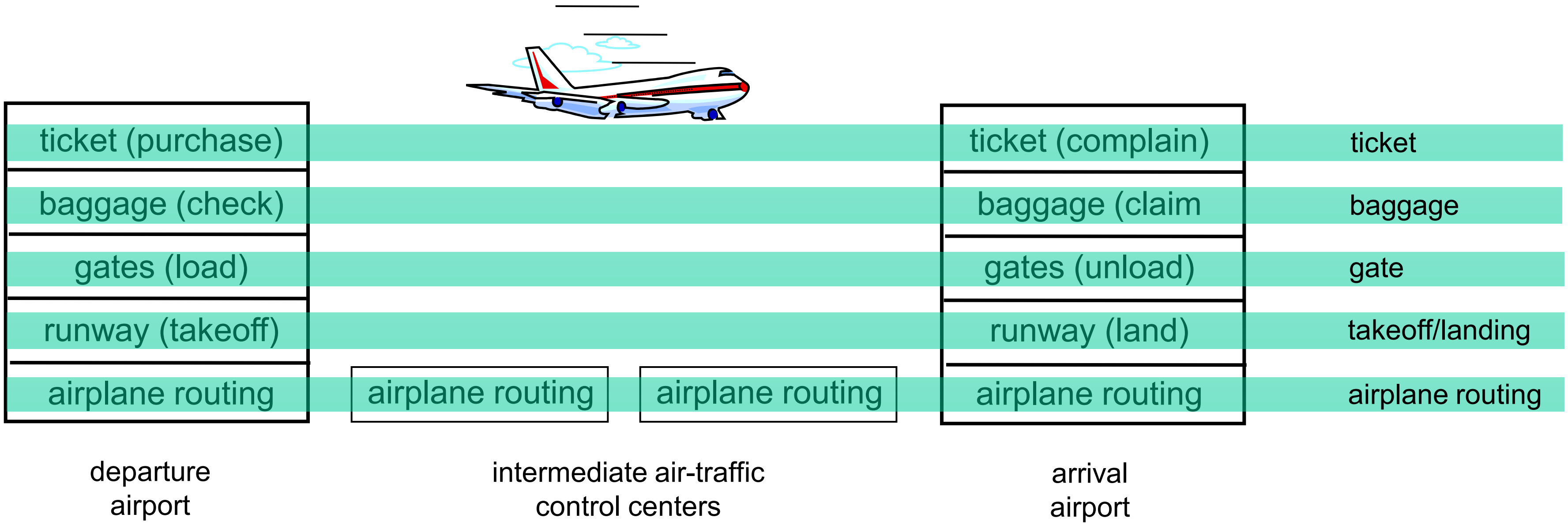
gates (unload)

runway landing

airplane routing

- a series of steps

Layering of airline functionality



ticket (purchase)			ticket (complain)	ticket
baggage (check)			baggage (claim	baggage
gates (load)			gates (unload)	gate
runway (takeoff)			runway (land)	takeoff/landing
airplane routing	airplane routing	airplane routing	airplane routing	airplane routing
departure airport	intermediate air-traffic control centers		arrival airport	

layers: each layer implements a service

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

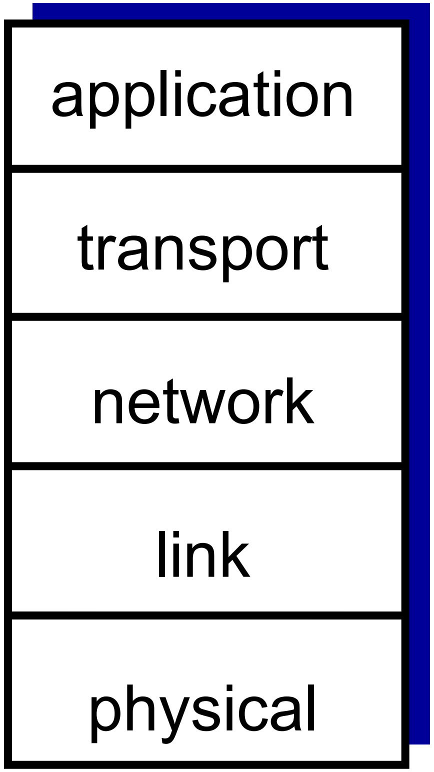
Why layering?

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

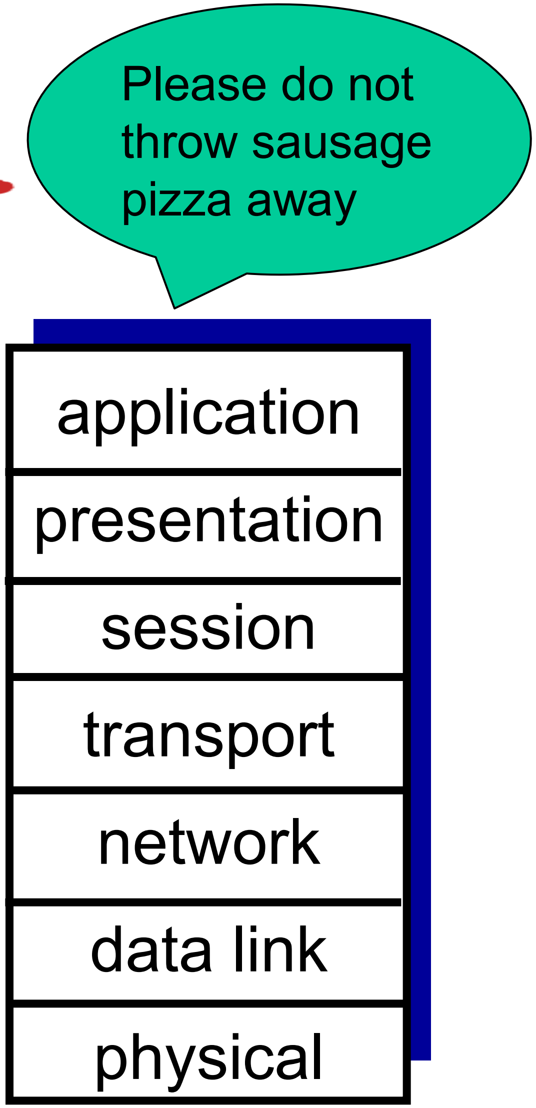
Internet protocol stack

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



ISO/OSI reference model

- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers! Why?
    - these services, *if needed*, must be implemented in application



Please do not throw sausage pizza away

application

presentation

session

transport

network

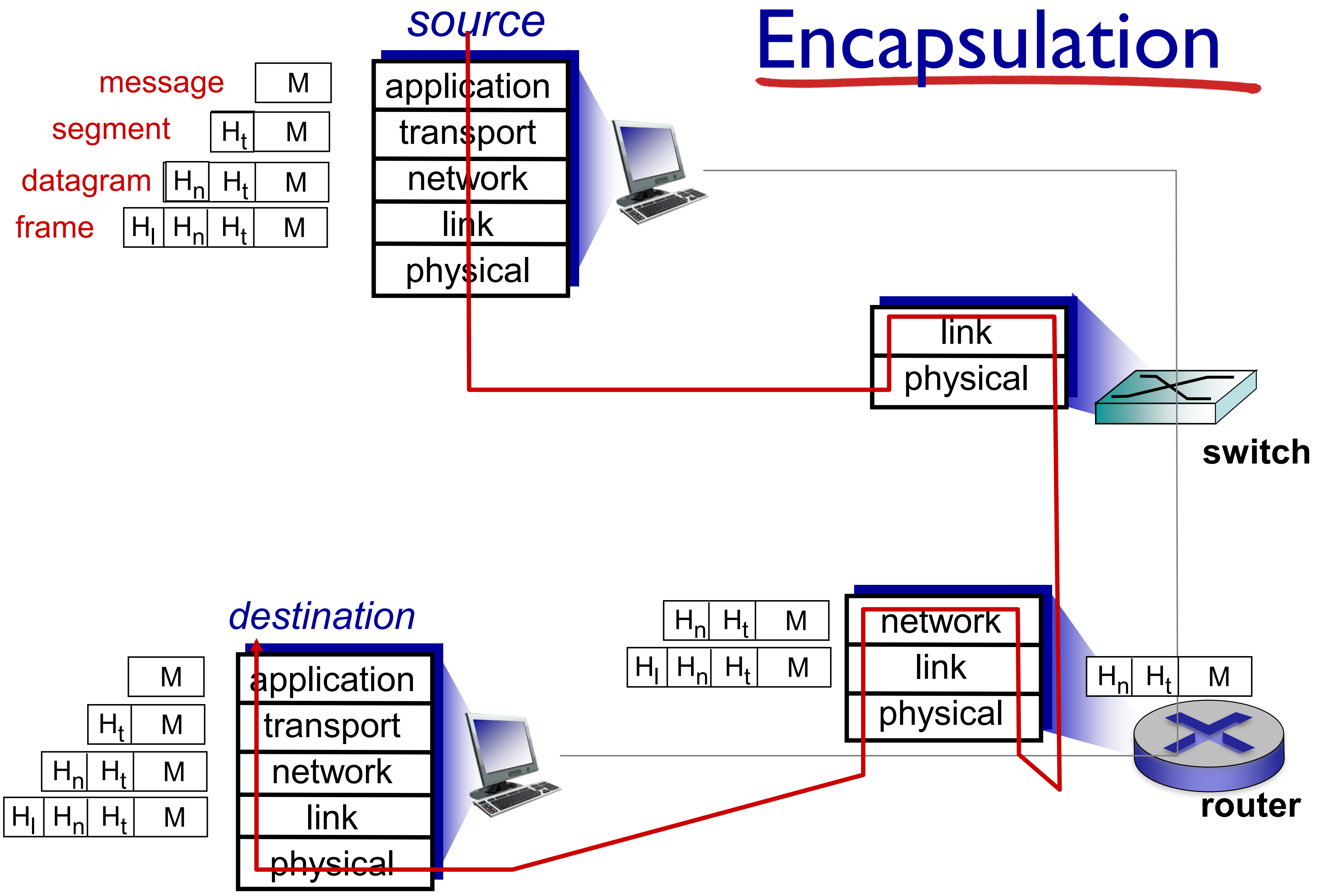
data link

physical

7 Layers of the OSI Model

Application	• End User layer • HTTP, FTP, IRC, SSH, DNS
Presentation	• Syntax layer • SSL, SSH, IMAP, FTP, MPEG, JPEG
Session	• Synch & send to port • API's, Sockets, WinSock
Transport	• End-to-end connections • TCP, UDP
Network	• Packets • IP, ICMP, IPSec, IGMP
Data Link	• Frames • Ethernet, PPP, Switch, Bridge
Physical	• Physical structure • Coax, Fiber, Wireless, Hubs, Repeaters

Encapsulation



source

message M

segment H_t M

datagram H_n H_t M

frame H_l H_n H_t M

application

transport

network

link

physical

link

physical

switch

destination

M

H_t M

H_n H_t M

H_l H_n H_t M

application

transport

network

link

physical

H_n H_t M

H_l H_n H_t M

network

link

physical

H_n H_t M

router

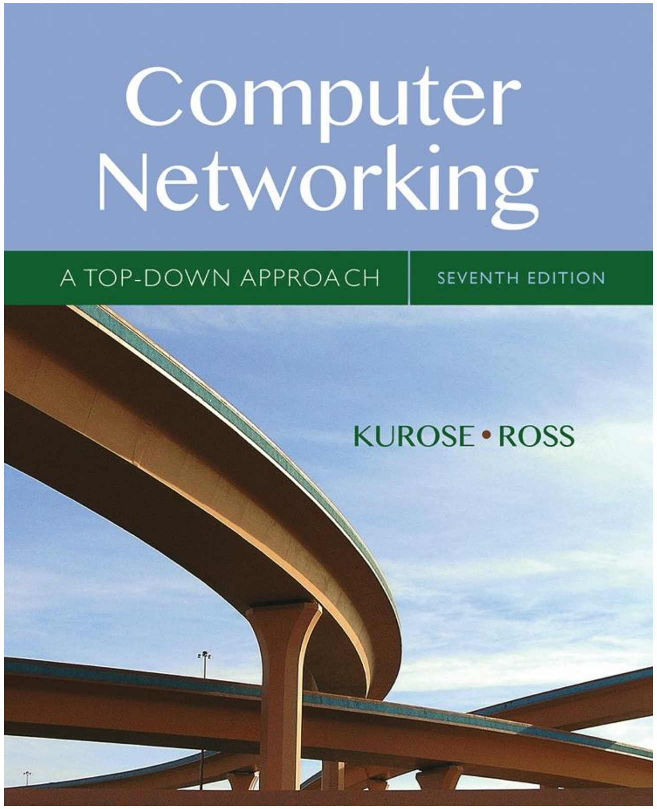
Chapter 2
Application Layer

A note on the use of these Powerpoint slides:

We' re making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we' d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Chapter 2: application layer

our goals:

- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
  - content distribution networks
- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- creating network applications
  - socket API

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
- …
- …

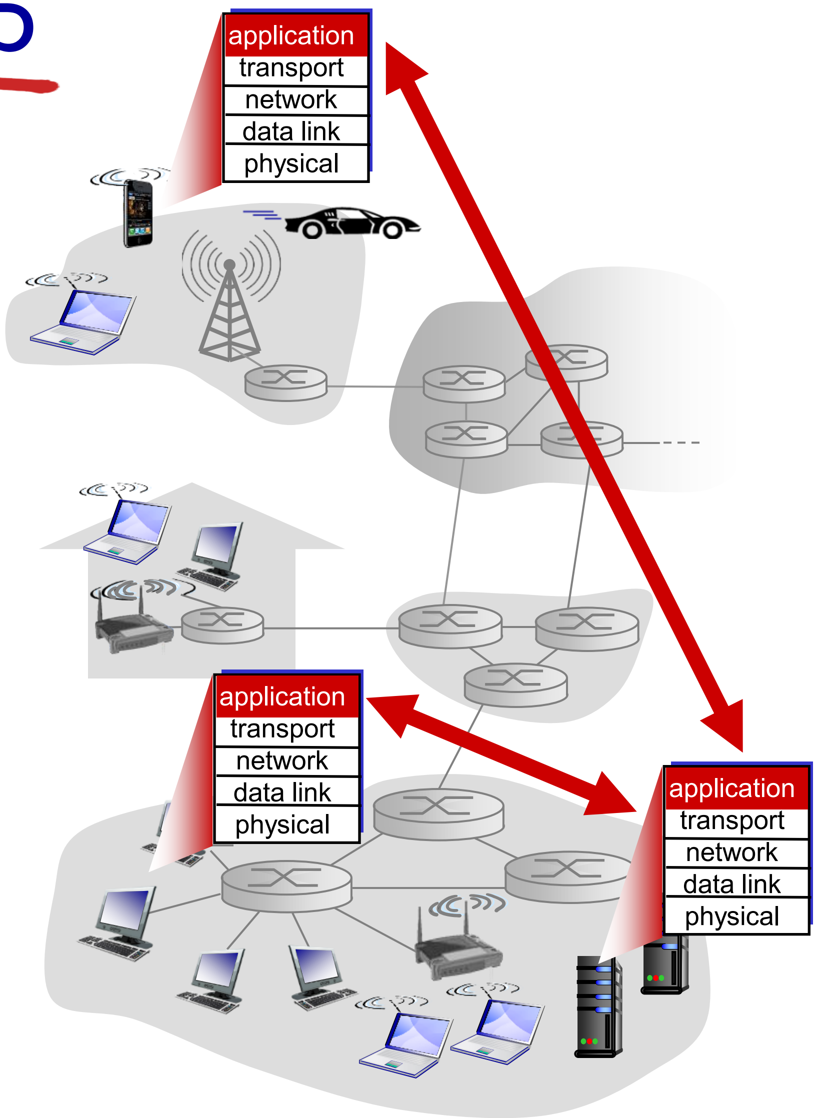
Creating a network app

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 allow for rapid app development, propagation

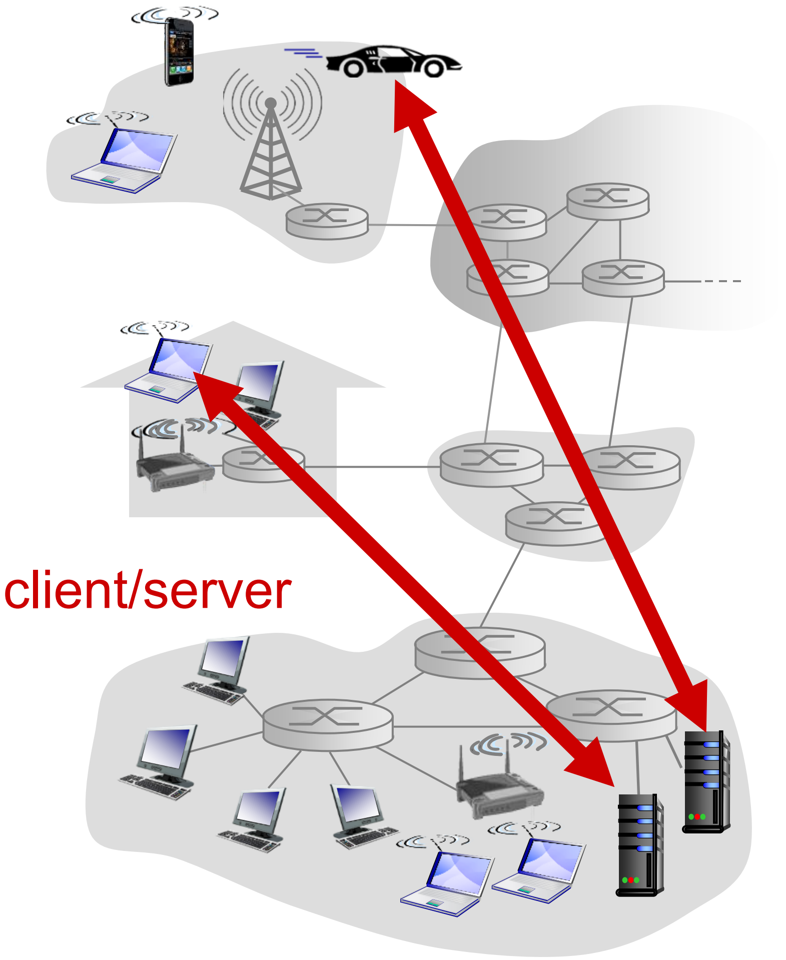


Application architectures

possible structure of applications:

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

Client-server architecture



server:

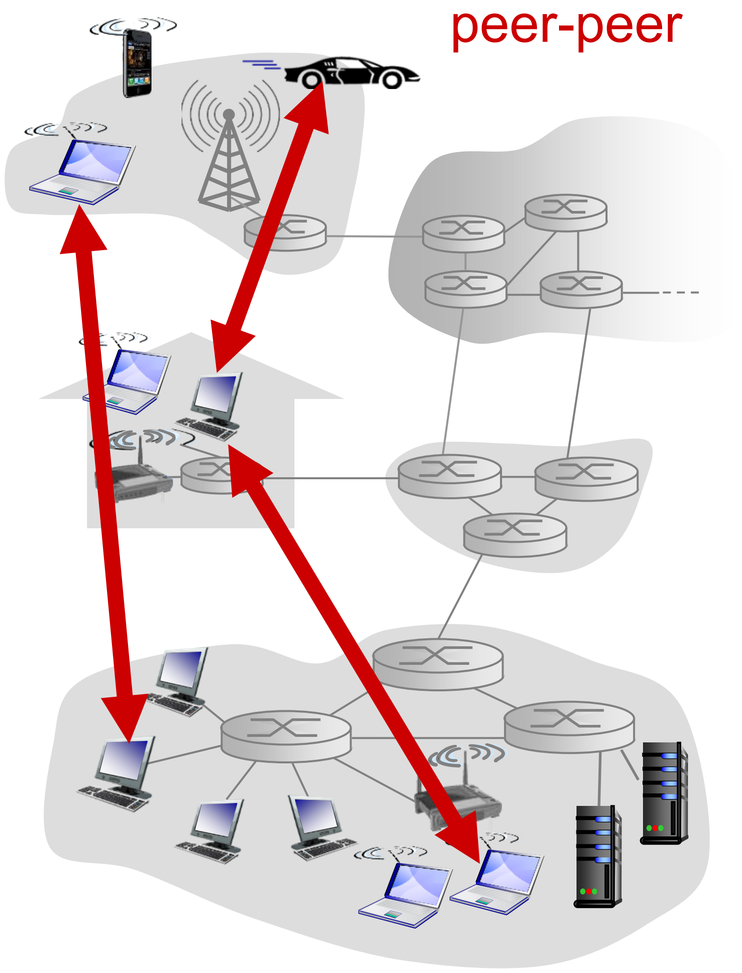
- always-on host
- permanent IP address
- data centers for scaling

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

clients, servers

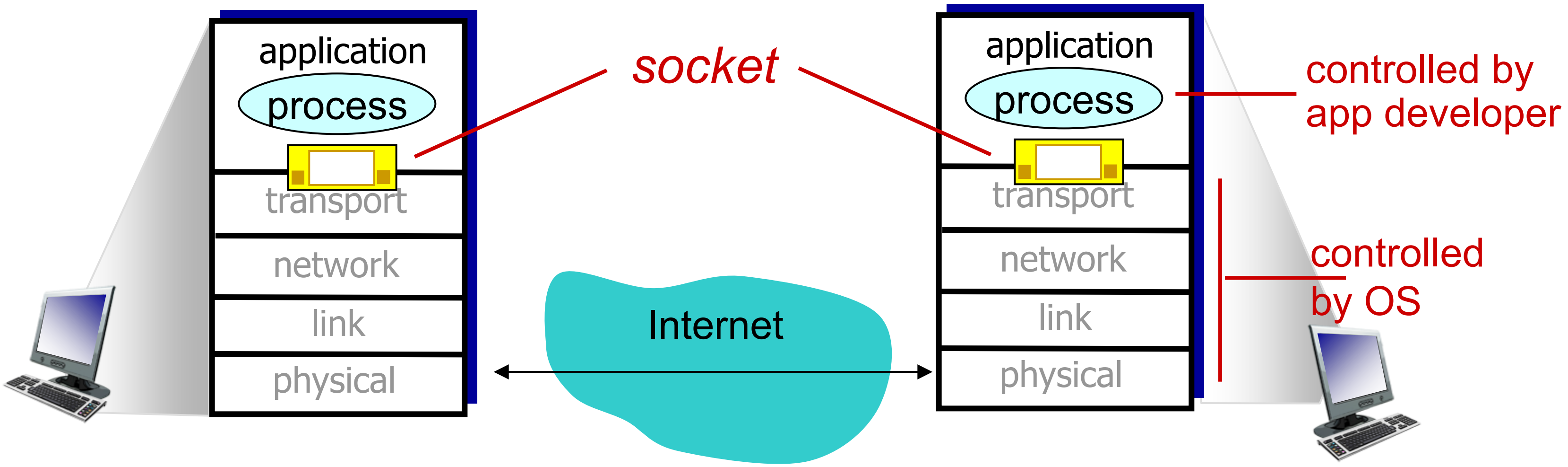
client process: process that initiates communication

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
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process

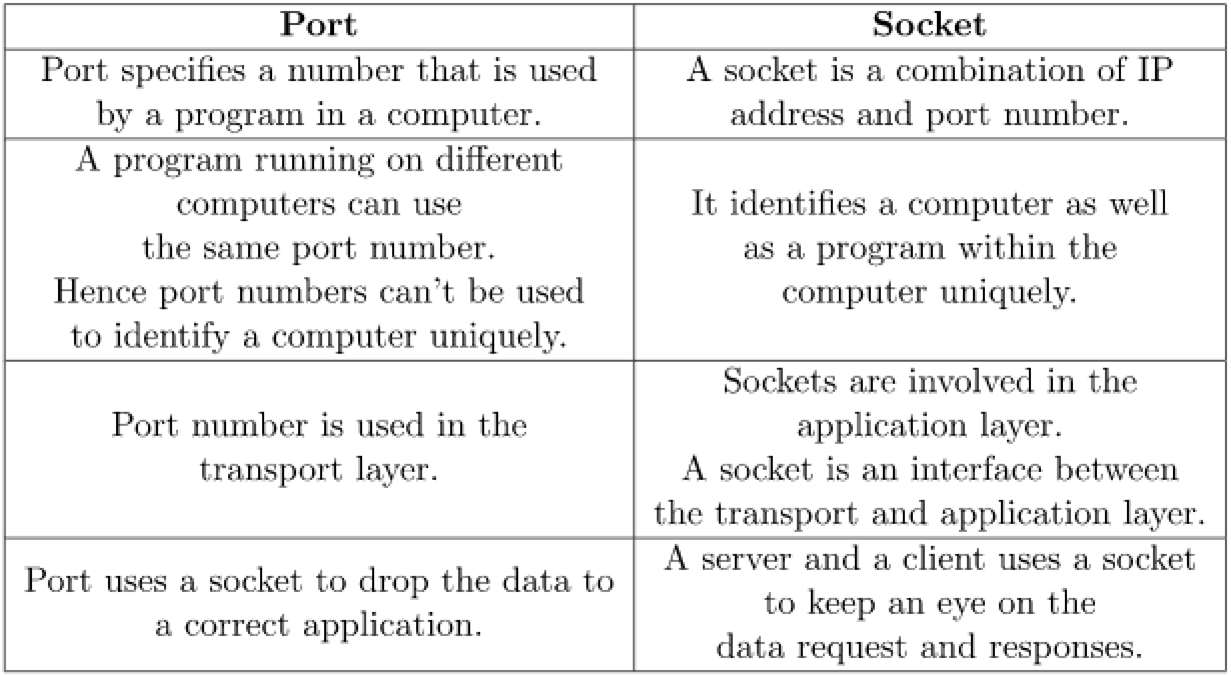


Addressing processes

- to receive messages, a 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

Port versus socket



Port	Socket
Port specifies a number that is used by a program in a computer.	A socket is a combination of IP address and port number.
A program running on different computers can use the same port number. Hence port numbers can't be used to identify a computer uniquely.	It identifies a computer as well as a program within the computer uniquely.
Port number is used in the transport layer.	Sockets are involved in the application layer. A socket is an interface between the transport and application layer.
Port uses a socket to drop the data to a correct application.	A server and a client uses a socket to keep an eye on the data request and responses.

What transport service does an app need?

data integrity

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

security

- encryption, data integrity, authentication…

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100’ s msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
text messaging	no loss	elastic	yes, 100’ s
			msec yes and no

Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won' t overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup,

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Securing TCP

TCP & UDP

- no encryption
- cleartext passwds sent into socket traverse Internet in cleartext

SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- apps use SSL libraries, that "talk" to TCP

SSL socket API

- cleartext passwords sent into socket traverse Internet encrypted
- Now TLS

App-layer protocol defines

- types of messages exchanged,
  - e.g., request, response
- message syntax:
  - what fields in messages & how fields are delineated
- message semantics
  - meaning of information in fields
- rules for when and how processes send & respond to messages

open protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype, Zoom

THANK YOU!