UIC UNIVERSITY OF ILLINOIS AT CHICAGO Department of Computer Science
COLLEGE OF ENGINEERING

CS450 – Introduction to Networking

Lecture 4 – Network Applications & HTTP

Phu Phung

January 21, 2015

News: Discussion board

- Piazza

  https://piazza.com/uic/spring2015/cs450/home

- There will be bonus points up to 5%
  - for Piazza forums
    - contribution of helpful code to the common good of the class (e.g. test cases and/or testing scripts)
  - Thoughtful discussions during lecture.

A quick check on assignment 1 progress

A. I have completed/nearly completed the project

B. I have done about >= 75%

C. I have done about >= 50%

D. I have done about >= 25%

E. I want the deadline extended

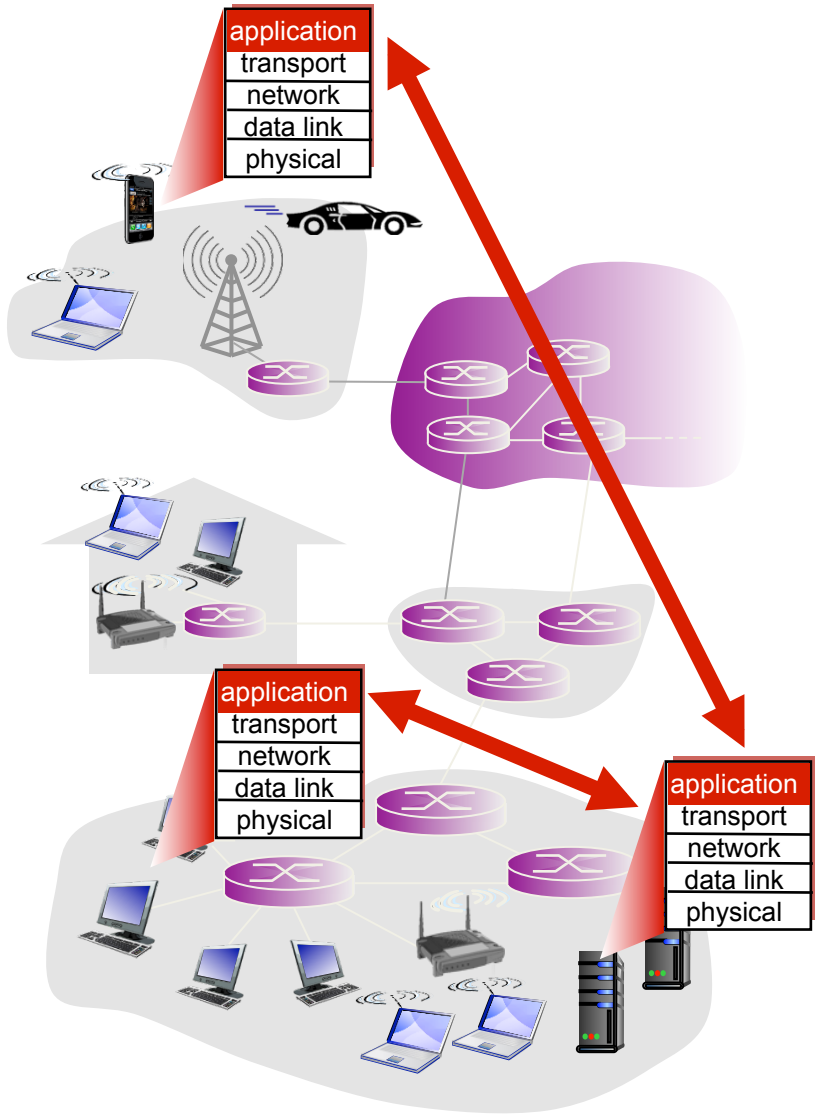
Creating a network app

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

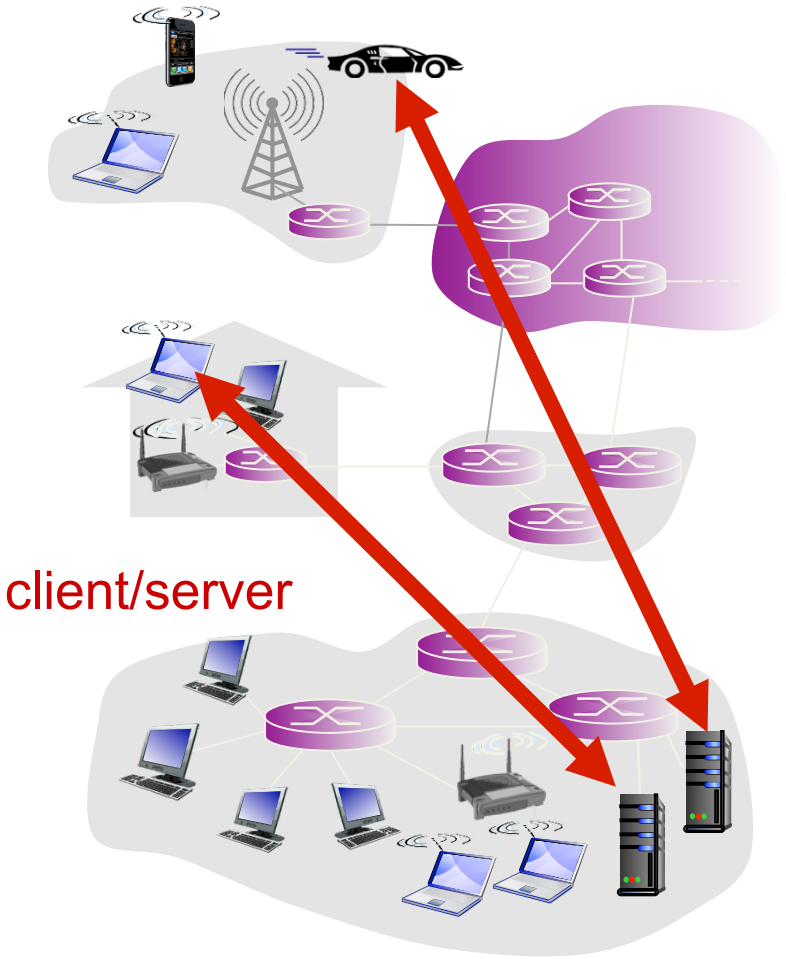


Application architectures

possible structure of applications:

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

Client-server architecture



server:

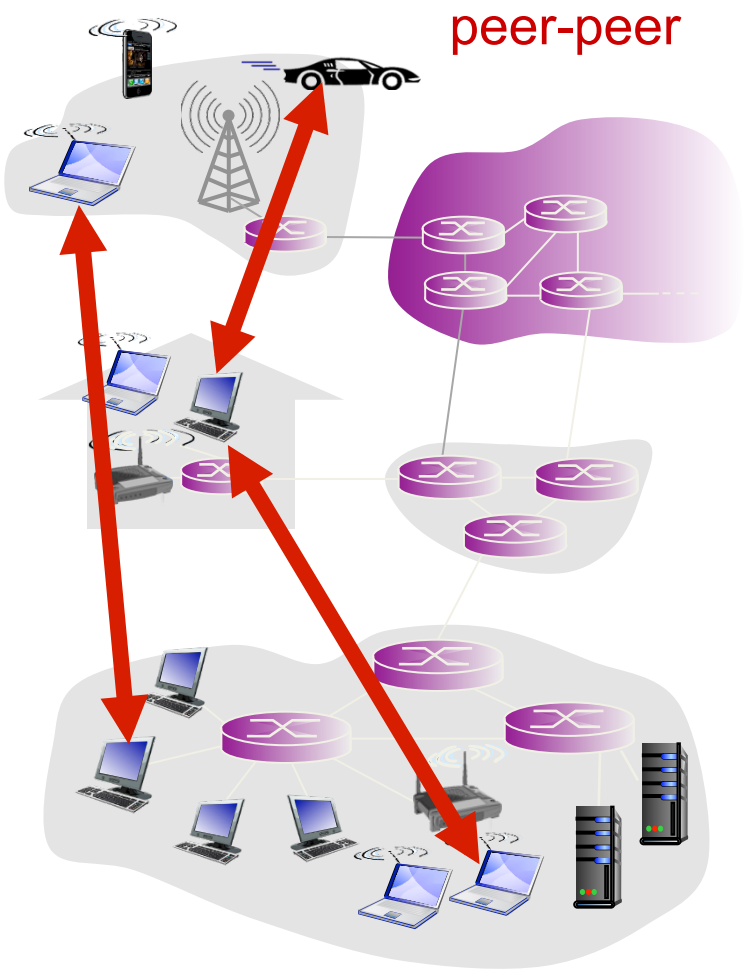
- always-on host
- permanent IP address
- data centers for scaling

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

clients, servers

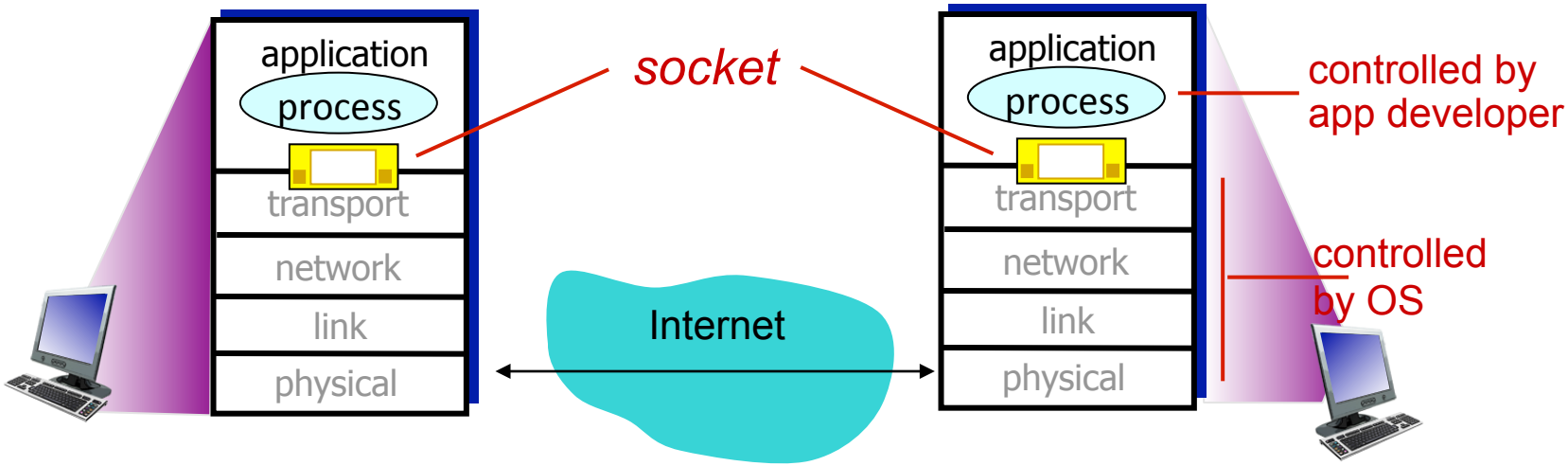
client process: process that initiates communication

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



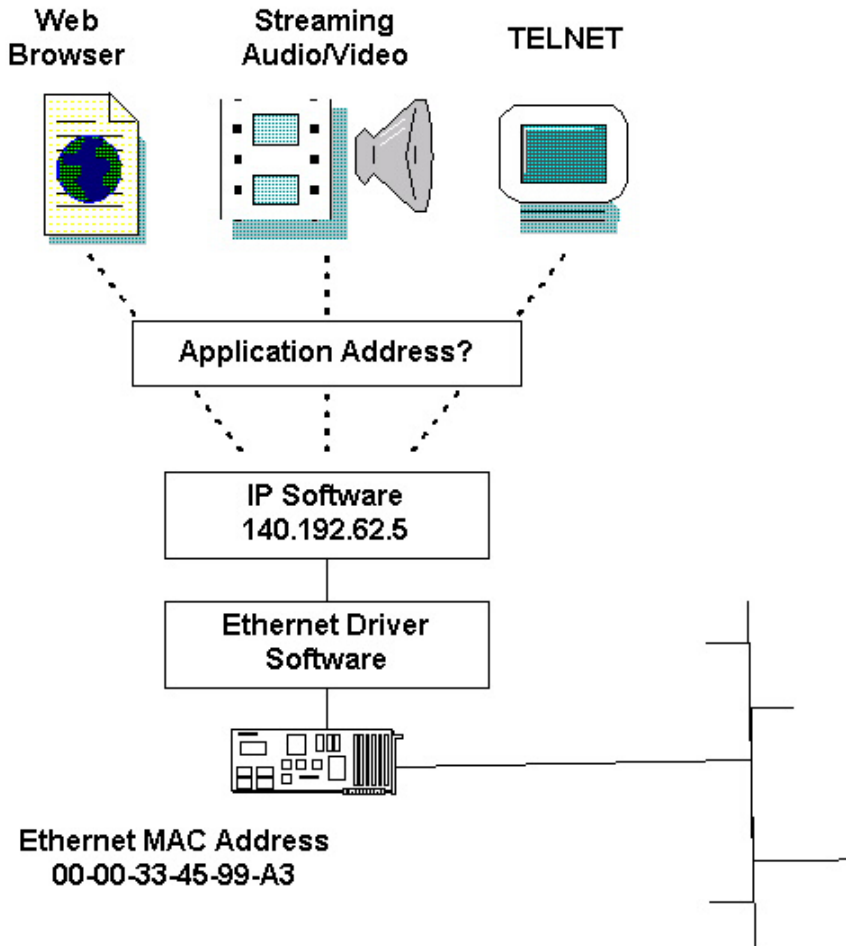
Addressing processes

- to receive messages, 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*

Addressing processes

- *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 www.cs.uic.edu web server:
  - IP address: 131.193.32.29
  - port number: 80
- more shortly…



Process ports

- 16 bit integer
  - *Port numbers: 0..65535*
- 3 ranges
  - 0..1023: for common, well-known services
  - 1024..49151 - Registered port: vendors use for applications
  - >49151 - dynamic / private ports

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

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, …

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, which "talk" to TCP

SSL socket API

- cleartext passwds sent into socket traverse Internet encrypted
- See Chapter 7

Select a wrong statement

A. TCP and UDP are in transport layer

B. TCP provides reliable connections while UDP is unreliable connections

C. HTTP uses TCP

D. FTP uses UDP

E. All network applications rely on either TCP or UDP

A network application process is identified by

A. MAC address

B. IP address

C. IP address and port number

D. Domain name and IP address

E. MAC address and port number

UDP is unreliable, why we still need it?

A. Because TCP is bad

B. Because UDP is good

C. Because in some cases, UDP is better than TCP

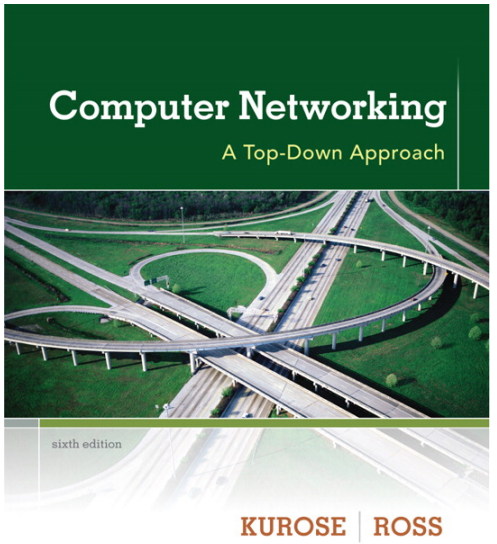
D. Because UDP is speedy

E. C and D are both correct

Next lecture

- HTTP (cont')
  - Readings 2.2

- Questions regarding Assignment 1?



Copy right notice:
These slides are adapted from J.F Kurose and K.W. Ross's ones

Computer Networking: A Top Down Approach

6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012