CS3640

Application Layer (1): Principles & Architecture

Prof. Supreeth Shastri
Computer Science
The University of Iowa

IOWA

Announcements

Updates to spot quizzes

- *50% more time (now, 30 mins)*
- *Balance between questions that refer directly to the lecture material vs those that need applying the concepts*
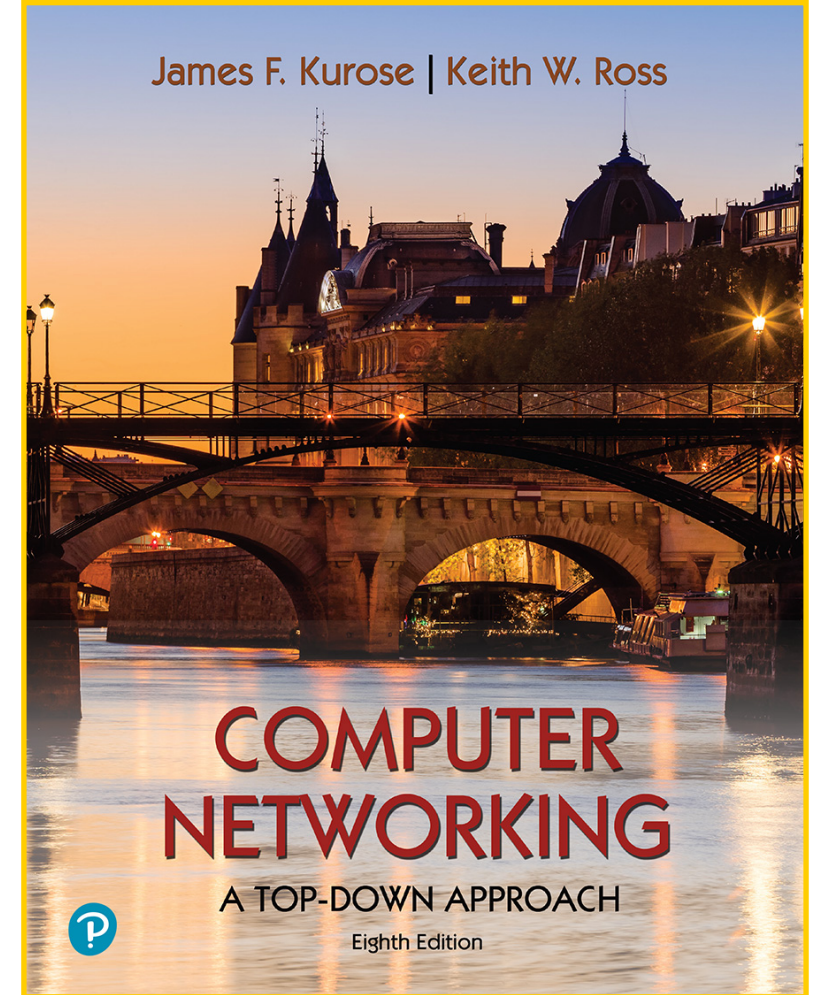
Written **assignment 1**

- *Open now; submission deadline: Feb-16*
- *Part-1 requires reading a research paper on end-to-end principle*
- *Part-2 covers the overview lectures*

IOWA

Lecture goals

Learn the conceptual and implementation framework of network applications

- *Design and architecture*
- *Utilizing transport services*
- *Understanding protocols*

James F. Kurose | Keith W. Ross

COMPUTER NETWORKING

A TOP-DOWN APPROACH

Eighth Edition

Chapter 2.1

IOWA

Architecture of Networked Apps

Network applications are ubiquitous in modern world

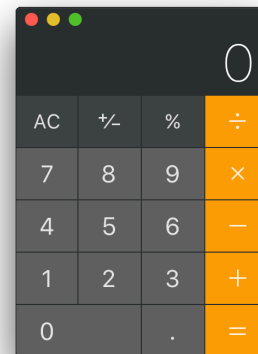**Socializing**

Communicating

Shopping

Entertainment

Traveling

Challenge: name three applications that are not networked

Convert values

- In the Calculator app 🖩 on your Mac, enter the original value, choose Convert in the menu bar, then choose a category, such as Temperature or Currency.

  Note: You must be connected to the internet to get the most recent currency conversion rate.

0

| AC | +/− | % | ÷ |
7	8	9	×
4	5	6	−
1	2	3	+
0		.	=

Creating a network app

Write programs that

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

What do you need to write for network-core devices?

- nothing! network-core devices do not run user applications
- limiting app building to end systems allows for rapid development and deployment
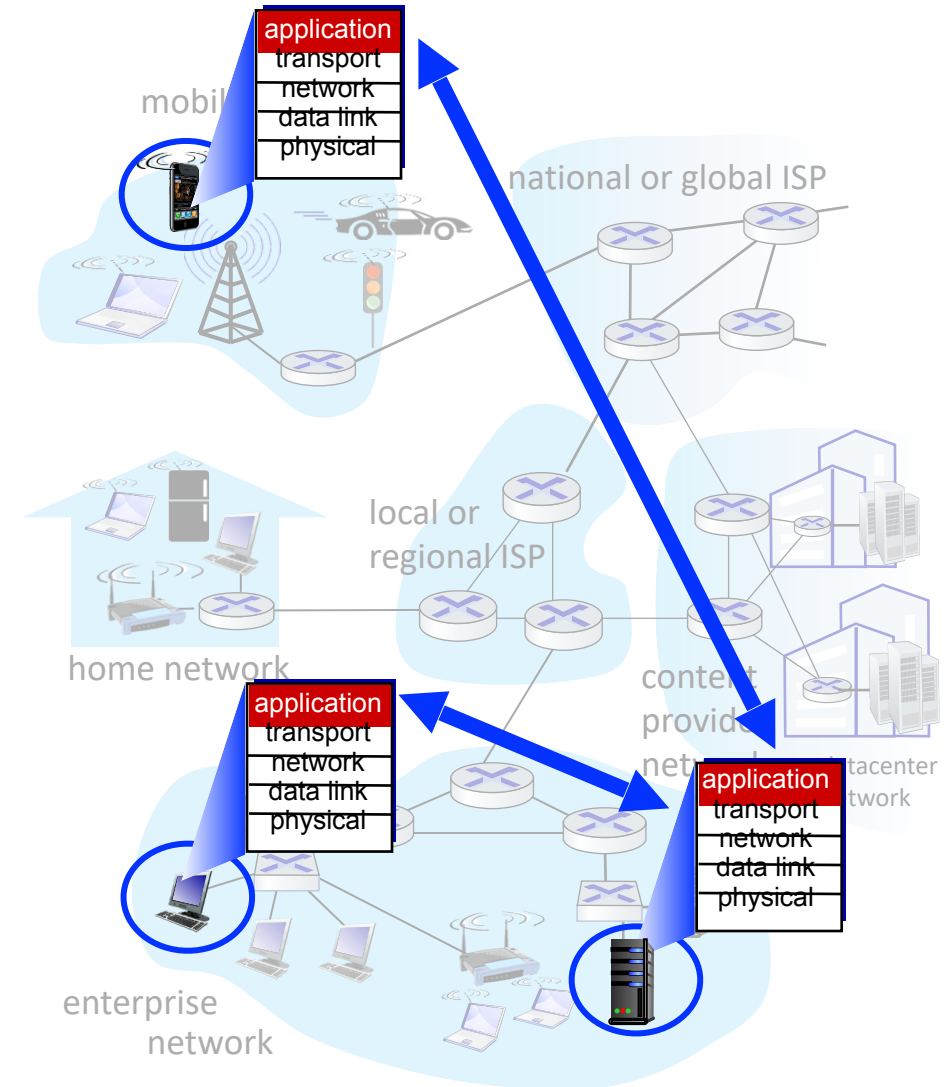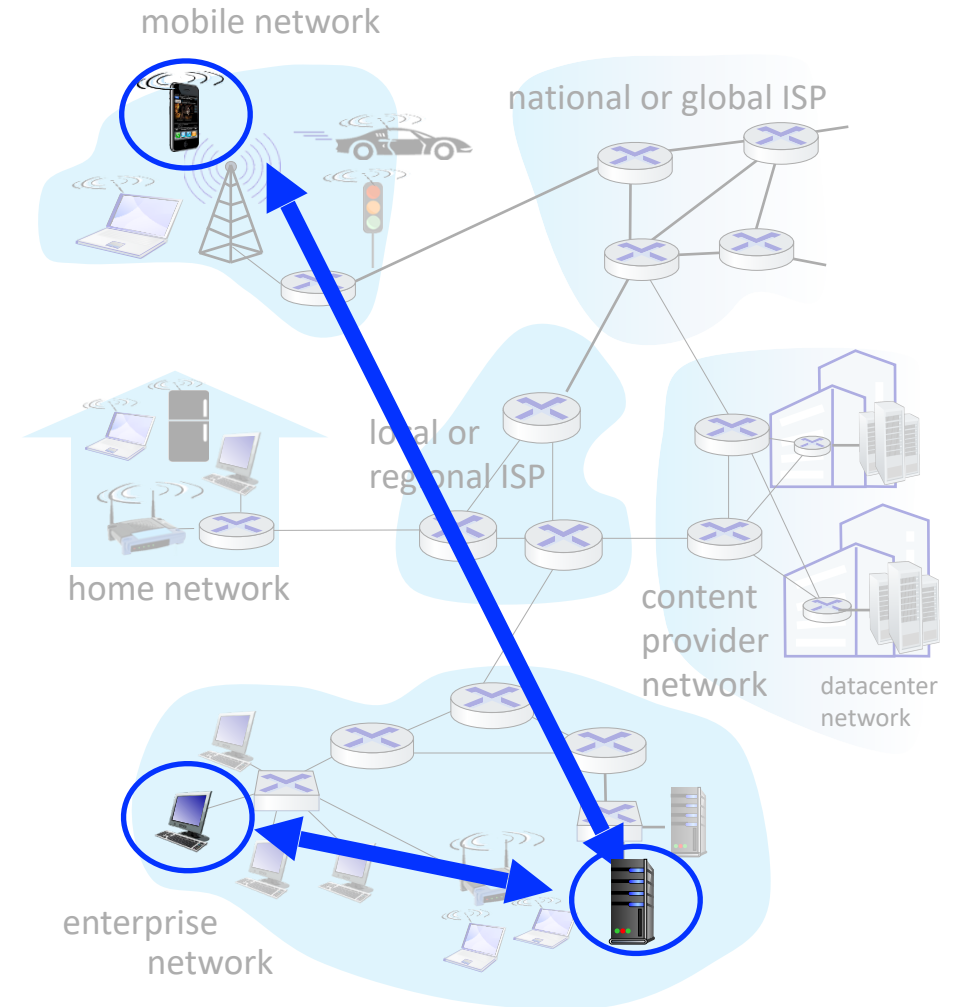
Architecture: client-server model

Server

- always-on host
- has a permanent IP address
- often located in data centers
- ability to scale to match the load

Clients:

- contact and communicate with server
- do not communicate directly with each other
- may be intermittently connected
- may have dynamic IP addresses
- examples: HTTP, IMAP, FTP

Architecture: peer-peer model

- there is no always-on server

- end systems communicate directly

- 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 can change IP addresses

- service management tends to be complex

- Examples: P2P file sharing, Skype
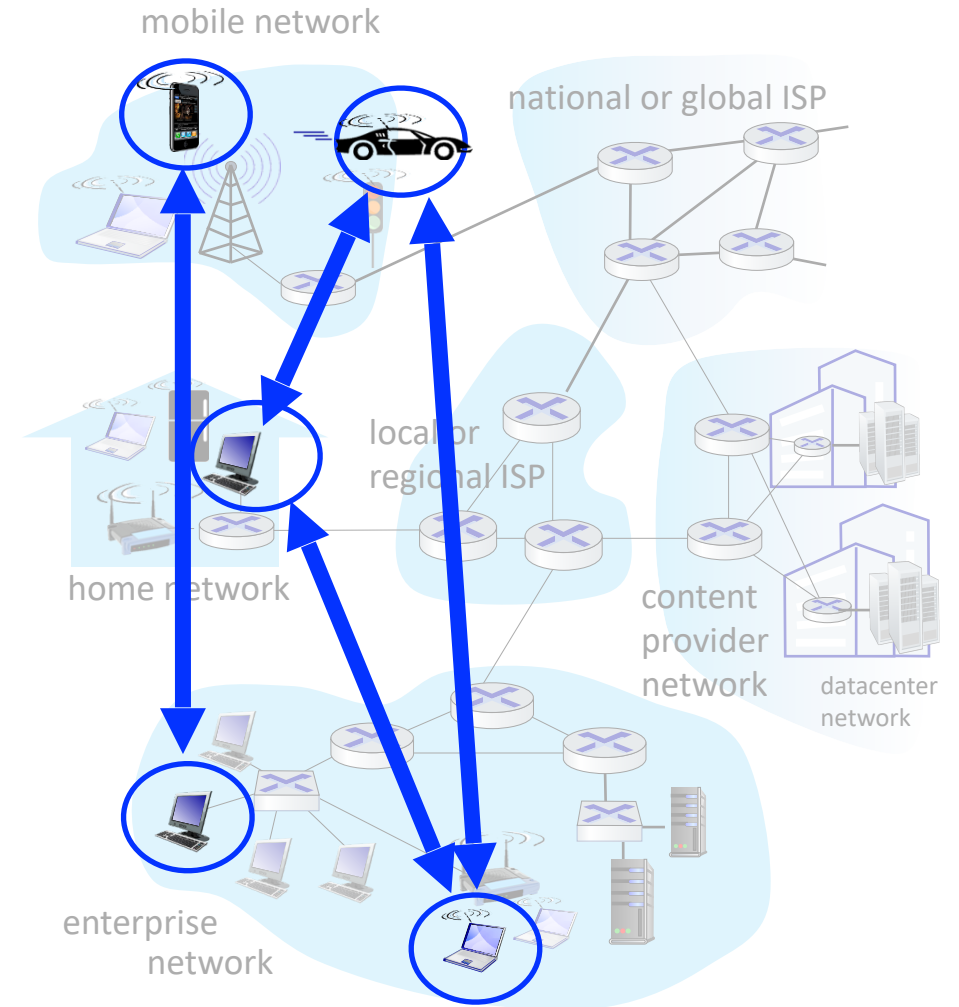
Communication: between the apps

*A primer on **process** and their **communications***

- Process is a program running on a host system

- Two processes within same host typically communicate using inter-process communication

- Processes in different hosts communicate by exchanging messages

clients, servers

client: *the process that initiates communication*

server: *the process that waits to be contacted*

In P2P architectures, an app can be both client and server at the same time

Communication: identifying an app on the network

To exchange messages, a process must have a unique *identifier*

How about using the IP address?

- Every host device has a unique 32-bit IP address

- But, IP address alone cannot identify the process that intends to communicate

IP address: **Port number**

- Ports are 16-bit numbers (0 - 65535) that identify processes running on a given host

- Categories: system ports, registered ports, and ephemeral ports

- *System ports are reserved: HTTP server runs on 80, SSH server on 22, SMTP server on 25*

- *Example: to reach the YouTube web server, send the packet to* *172.217.4.78 : 80*
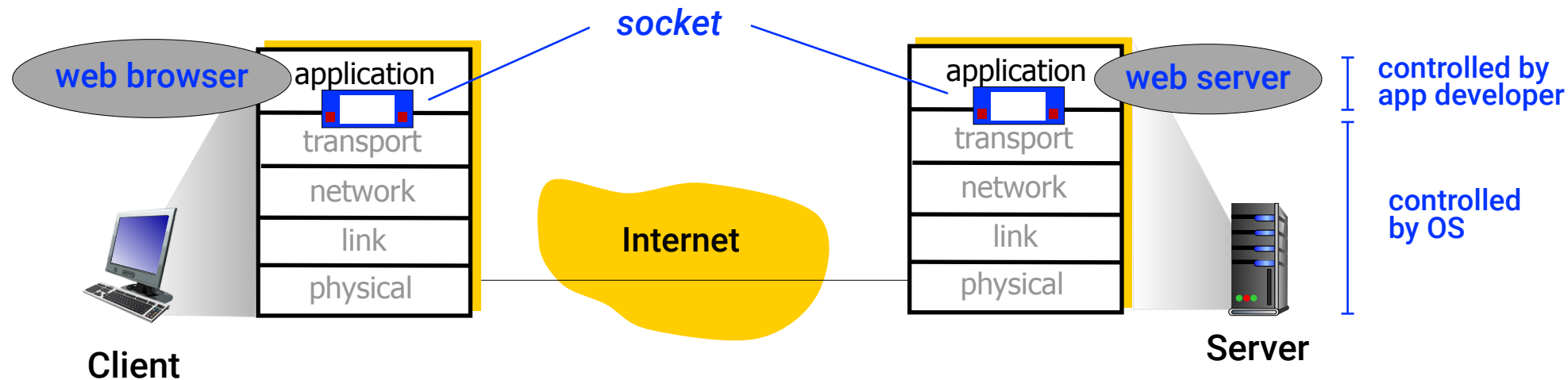
more in **Transport-layer**

Communication: interfacing with the network

Host OS expose APIs for networked applications called sockets

- Host OS implements the network stack from transport to physical layers
- App developers implement the application layer and the app itself

Networked applications use sockets *to send/receive messages*

- Useful to consider the analogy of sockets as doors within a building (i.e., host systems)

socket

web browser

application

transport

network

link

physical

Client

Internet

application

web server

transport

network

link

physical

Server

controlled by
app developer

controlled
by OS

Transport-layer **Services**

What transport services do applications need?

Integrity

- some apps (*e.g., world wide web*) require 100% reliable data transfer
- other apps (*e.g., teleconferencing*) can tolerate some loss

Latency

- some apps (*e.g., interactive games*) require low delay to be effective
- other apps (*e.g., emails*) can tolerate longer delays

Throughput

- some apps (*e.g., streaming*) require a relatively stable throughput
- other apps (*e.g., file transfer*) make use of whatever throughput they get

Other services

- security
- monitoring network conditions
- …

What transport services do applications need?

	Integrity	Latency	Throughput
File transfer	loss intolerant	not sensitive	elastic
Email	loss intolerant	not sensitive	elastic
WWW	loss intolerant	not sensitive	elastic
Teleconferencing	loss tolerant	< 10ms	~Kbps - Mbps
Video streaming	loss tolerant	tens of seconds	~Mbps
Text messaging	loss intolerant	usage dependent	elastic

Services offered by transport protocols

TCP
Transmission Control Protocol

connection-oriented
client and server agree before the flow of packets

reliable transport
lossless and in-order delivery of packets

flow management
so senders won't overwhelm receivers

congestion control
throttling transfer when network is overloaded

what's **not** offered
➡ control over latency
➡ throughput guarantees
➡ security

UDP
User Datagram Protocol

unreliable transport
between sending and receiving processes

what's **not** offered
➡ reliability
➡ flow management
➡ congestion control
➡ control over latency
➡ throughput guarantees
➡ security

Applications and protocols

	Application Protocol	Transport Protocol
File transfer	FTP (RFC 959)	TCP
Email	SMTP (RFC 5321)	TCP
WWW	HTTP (RFC 7320)	TPC
Teleconferencing	SIP (RFC 3261) & RTP (RFC3550)	TCP & UDP
Video streaming	HLS (RFC 8216) or DASH	TCP
Name resolution	DNS (RFC 1034)	UDP

Protocols for Network Apps

Network protocols define the **format** and **order** of **messages** sent and received among network entities, and the **actions taken** on message transmission and receipt

if

then

Applications on **end hosts**

Application protocols

A vast **majority** of the Internet RFCs

are application layer protocols

Application protocols vary significantly in their **size** and **complexity**.

*E.g., **Time Protocol** (RFC 868) is just 419 words, whereas **Network File System** (RFC 3530) is 79,803 words*

Application-layer protocols

Components

- Types of messages exchanged. E.g., request, response, status check

- Message syntax and semantics. E.g., header fields and their meanings

- Rules for determining when and how to initiate, process, and respond to messages.
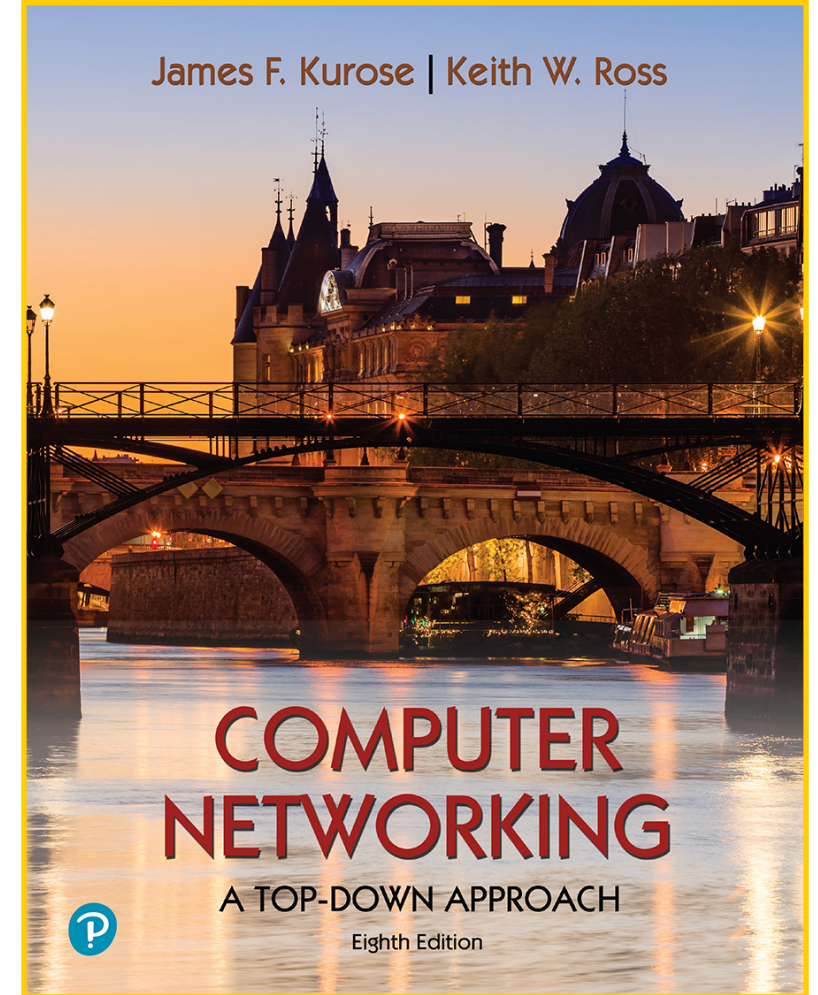
Open vs. proprietary

- Open protocols are defined in RFCs with free and open access to all. E.g., HTTP, SMTP, DNS

- Proprietary ones hide their internals to prevent others from implementing apps or interpreting messages. E.g., Zoom, Skype

- Open protocols allow interoperability and foster innovation

Next lecture

Deep dive into the design and operation of the world wide web

- *HTTP*
- *Web cookies*
- *Web caches*

James F. Kurose | Keith W. Ross

COMPUTER NETWORKING
A TOP-DOWN APPROACH
Eighth Edition

Chapter 2.2

IOWA

Spot Quiz (ICON)