Computer Networks
컴퓨터네트워크

(Ch. 2: Layer 7 - Application Layer & Network Programming)

Wonjun Lee, Ph.D., IEEE Fellow
Network and Security Research Lab. (NetLab)
http://netlab.korea.ac.kr
http://mobile.korea.ac.kr
Korea University

1

NetLab
Network and Security Research Lab.

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.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR
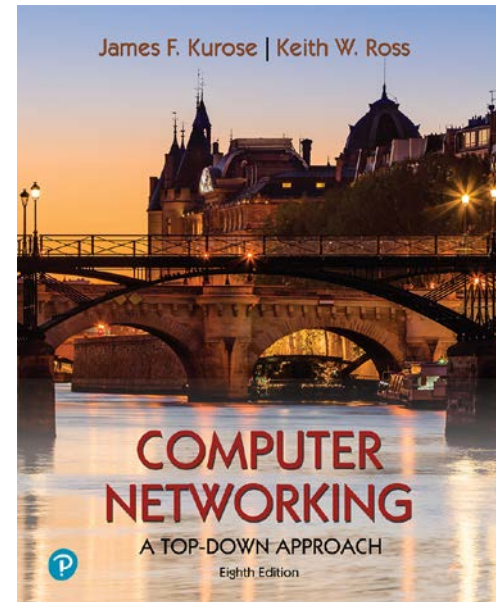
*Computer Networking:
A Top-Down Approach*
8th edition   n
Jim Kurose, Keith Ross
Pearson, 2020

Application layer: overview

- Principles of network applications

- Web and HTTP

- E-mail, SMTP, IMAP

- The Domain Name System DNS

- P2P applications

- video streaming and content distribution networks

- socket programming with UDP and TCP

Application layer: overview

Our goals:

- conceptual *and* implementation aspects of application-layer protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm

- learn about protocols by examining popular application-layer protocols
  - HTTP
  - SMTP, IMAP
  - DNS
- programming network applications
  - socket API

Some network apps

- social networking
- Web
- text messaging
- e-mail
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- P2P file sharing

- voice over IP (e.g., Skype)
- real-time video conferencing
- Internet search
- remote login
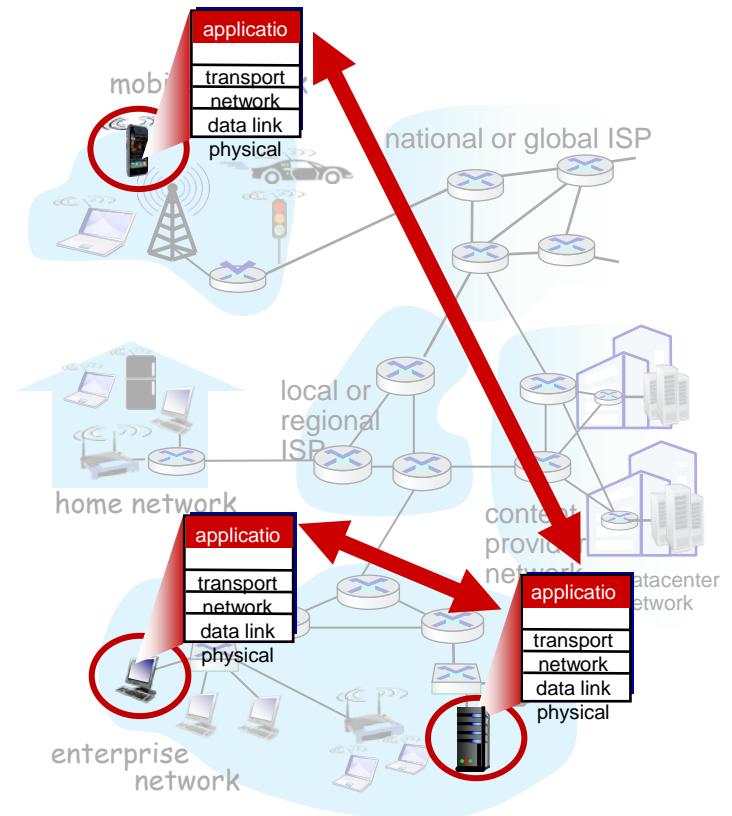- …
  Q: your favorites?

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
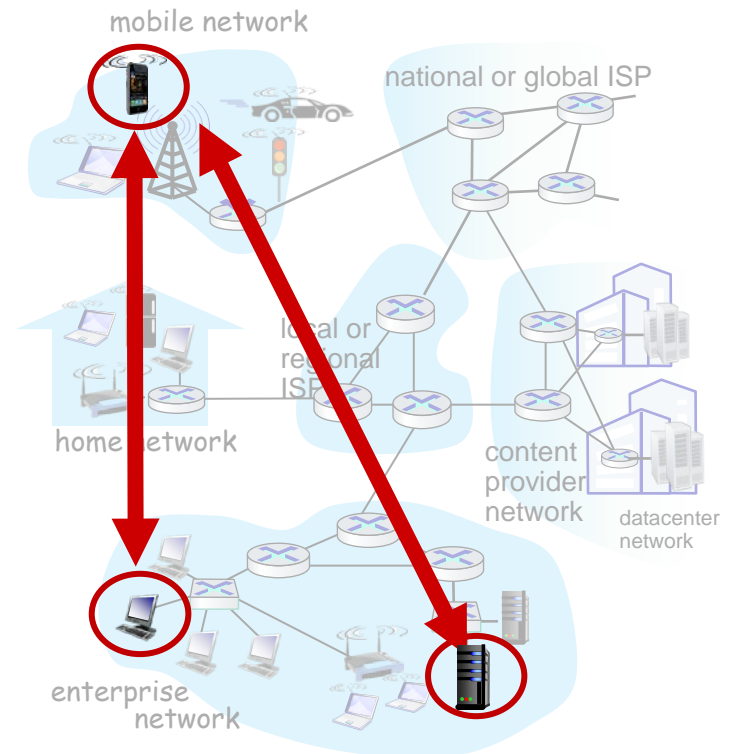
Client-server paradigm

server:
- always-on host
- permanent IP address
- often in data centers, for scaling

clients:
- contact, communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do *not* communicate directly with each other

- examples: HTTP, IMAP, FTP

Peer-peer 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
- example: P2P file sharing
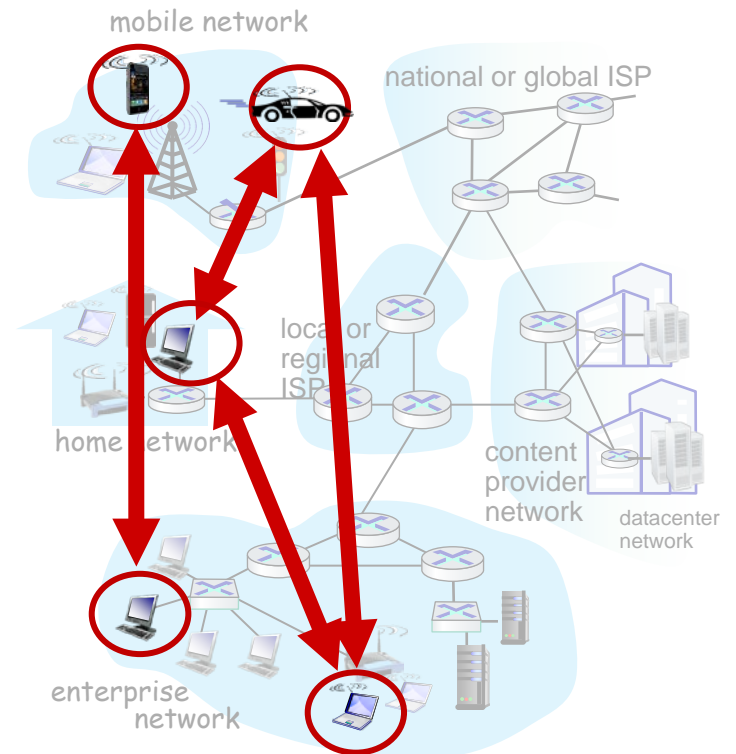
Processes communicating

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

- note: 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
  - two sockets involved: one on each side