CSC339 – DATA COMMUNICATIONS AND COMPUTER NETWORKS
LECTURE 08

Course Instructors:
Dr. Zulfiqar Ali
Sara Ali

AGENDA

- Application Layer: Overview
  - Network Application Examples
  - Creating Network app
- Network Application Architectures
  - Client-Server Architecture / Peer-to-Peer Architecture
- Process Communicating
  - Client and Server Processes
  - The Interface Between the Process and the Computer Network – Socket
  - Addressing Processes
- Transport Services Available to Applications:
  - Reliable Data Transfer, Throughput, Timing, Security
- Transport Services Provided by the Internet
  - TCP Services / UDP Services
- Application-Layer Protocols

Chapter 2: Application Layer
Book: Computer Networking: A Top Down Approach.
Author: Jim Kurose, Keith Ross Addison-Wesley

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

- text email, remote access to computers, file transfers, and newsgroups
- text messaging
- Web
  - Web surfing, Search, Electronic Commerce
- voice over IP (e.g., Skype, Whatsapp)
- streaming stored video (YouTube, Netflix)
- multi-user network games
- real-time video conferencing (Zoom, Microsoft Teams, Google Meet, Skype)
- social networking (Facebook, Twitter, WhatsApp)
- …

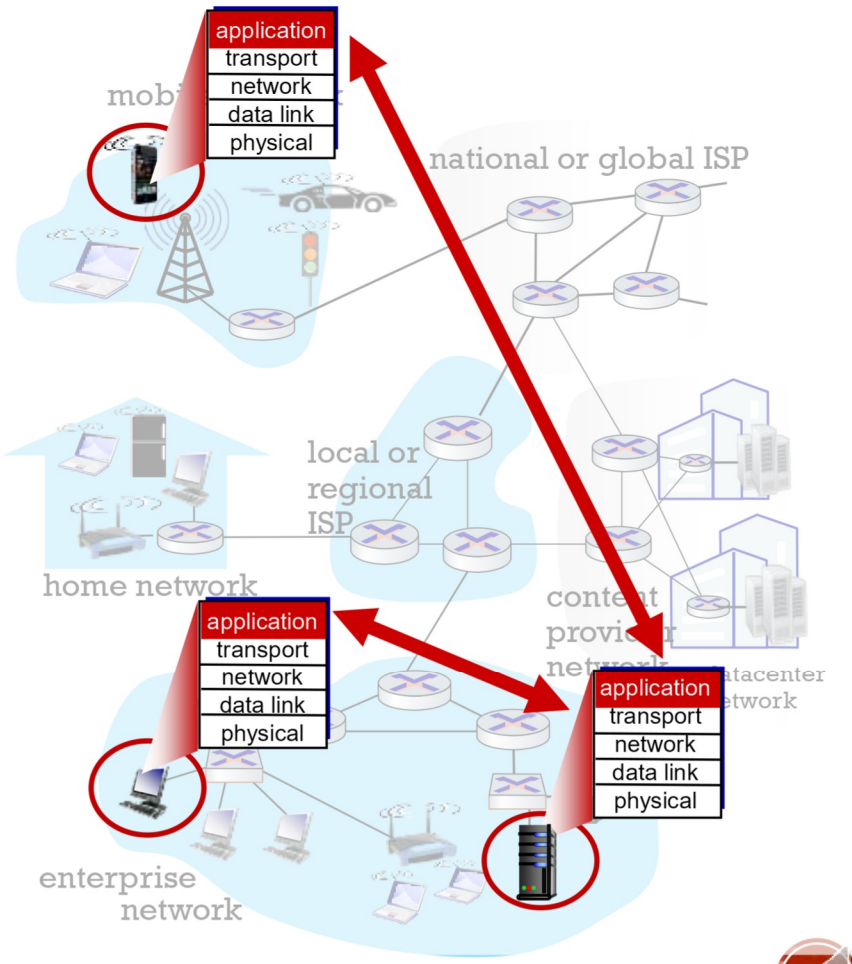
Creating a network App

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



NETWORK APPLICATION ARCHITECTURES

- From the application developer's perspective, the network architecture is fixed and provides a specific set of services to applications.
- The application architecture, on the other hand, is designed by the application developer and dictates how the application is structured over the various end systems.
- Two application architecture
  - Client-Server architecture
  - Peer-to-Peer architecture

CLIENT-SERVER PARADIGM

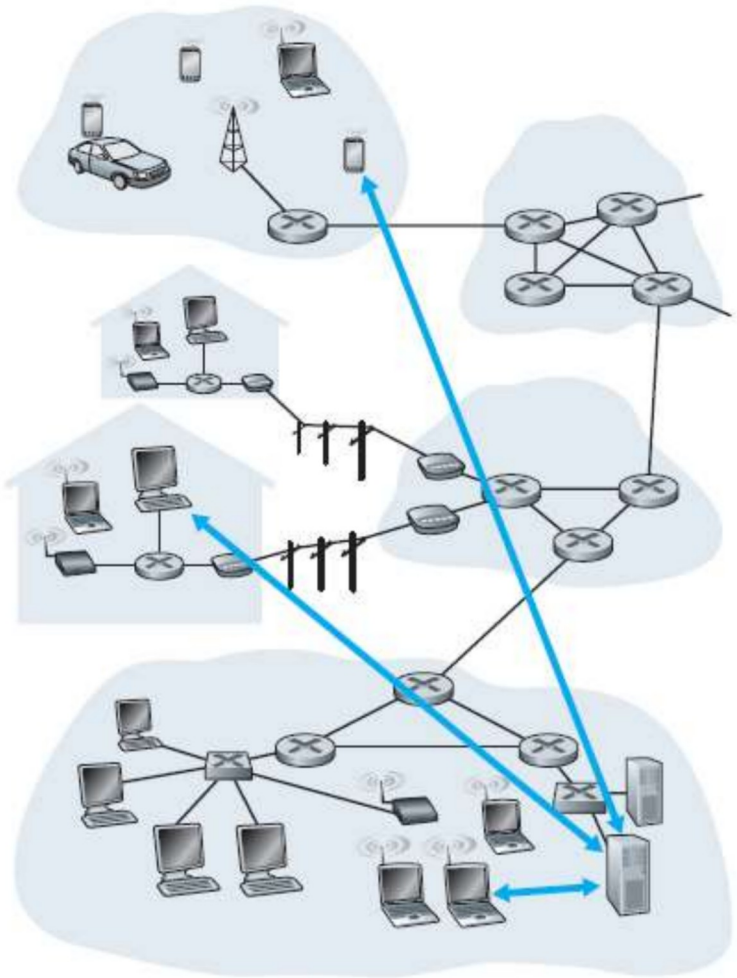
server:

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

clients:

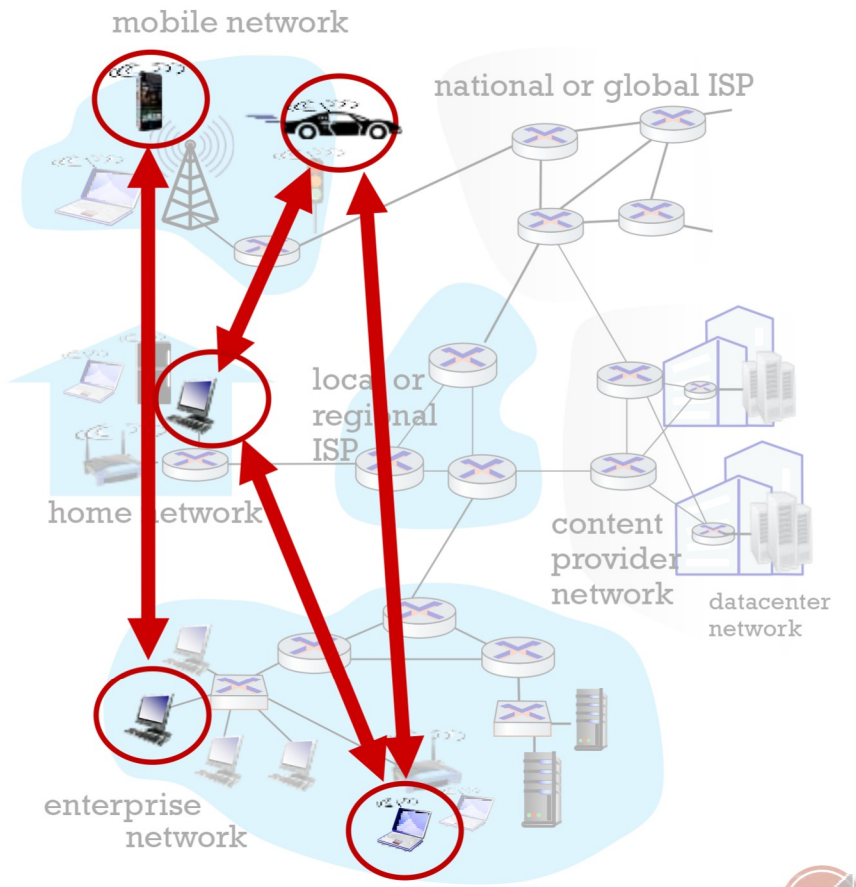
- contact, communicate with server
- may have dynamic IP addresses
- do *not* communicate directly with each other

- examples: Web, FTP, Telnet, email



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
- With the Web, a browser is a client process and a Web server is a server process.

clients, servers

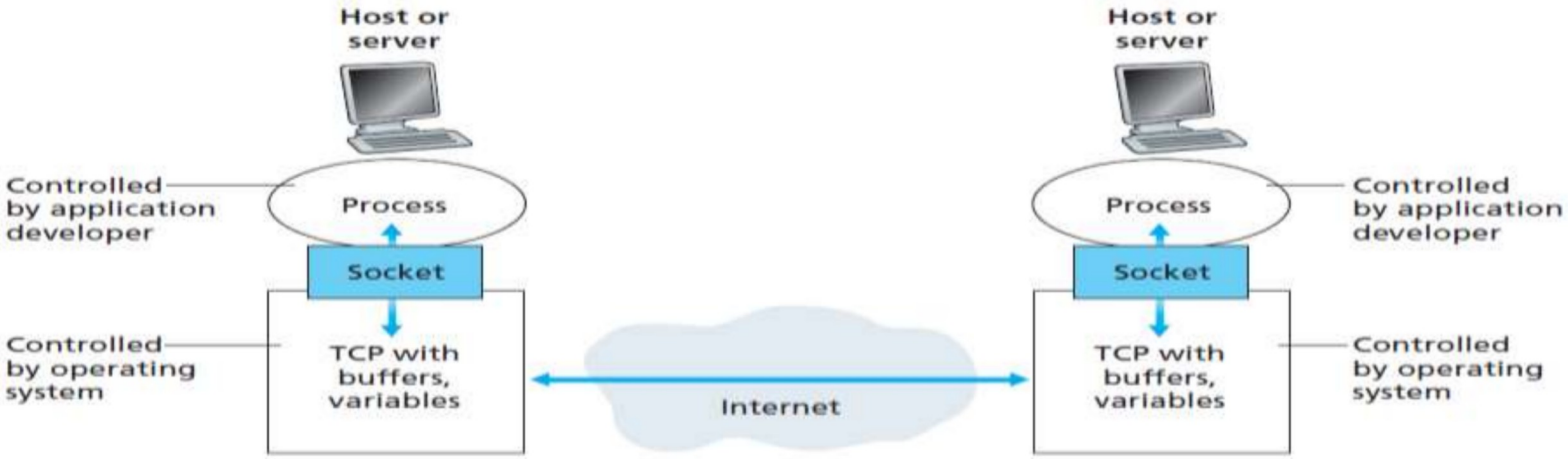
client process: process that initiates communication

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



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

An application-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, everyone has access to protocol definition
- allows for interoperability
- e.g., HTTP, SMTP

TRANSPORT SERVICES AVAILABLE TO APPLICATIONS

- Many networks, including the Internet, provide more than one transport-layer protocol.
- When you develop an application, you must choose one of the available transport-layer protocols.
- How do you make this choice?
  - Most likely, you would study the services provided by the available transport-layer protocols, and then pick the protocol with the services that best match your application's needs

We can broadly classify the possible services along four dimensions: reliable data transfer, throughput, timing, and security.

WHAT TRANSPORT SERVICE DOES AN APP NEED?

Reliable data Transfer

- 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
- Electronic mail, file transfer, and Web transfers are all elastic applications.

security

- encryption, data integrity, …

Transport Service Requirements: common apps

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	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
- *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.

INTERNET TRANSPORT PROTOCOLS SERVICES

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	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)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

NEXT LECTURE TOPICS

- Web and HTTP
- Non-Persistent and Persistent Connections
- HTTP Message Format
- User-Server Interaction: Cookies
- Web Caching