

Application Layer: Introduction

CSE 310: Aruna Balasubramanian

Slides adapted from Kurose and Rose

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
- ...
- ...

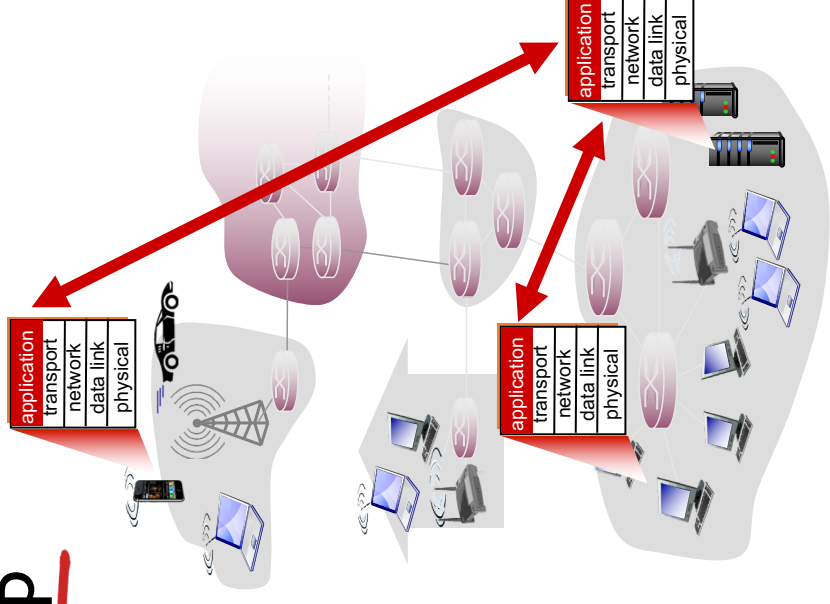
Creating a network app

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

- applications on end systems allows for rapid app development, propagation

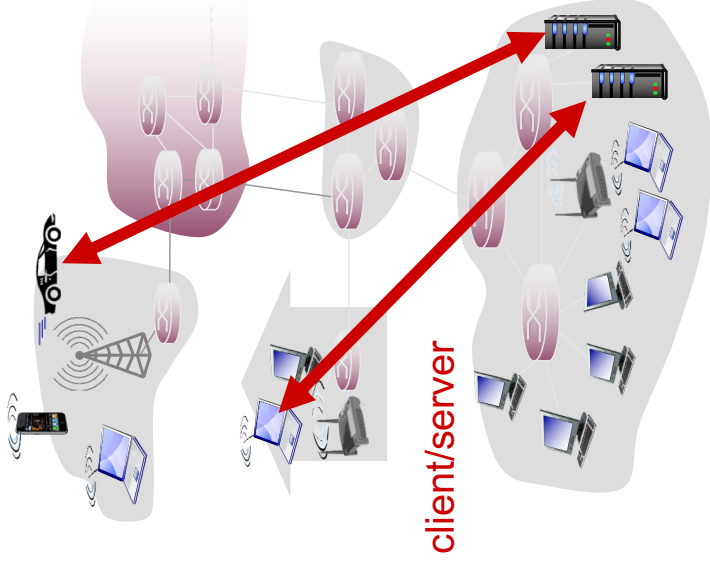


Application architectures

possible structure of applications:

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

Client-server architecture



server:

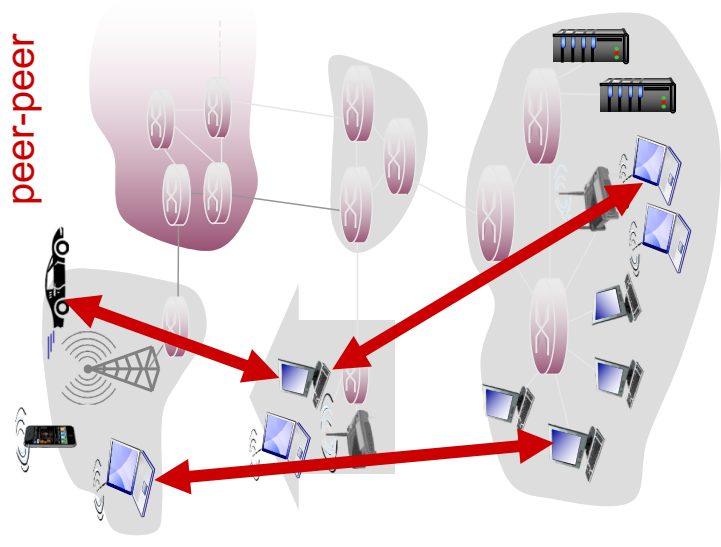
- always-on host
- permanent address

clients:

- communicate with server
- may be intermittently connected
- may have dynamic addresses
- do not communicate directly with each other

P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- Example?



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

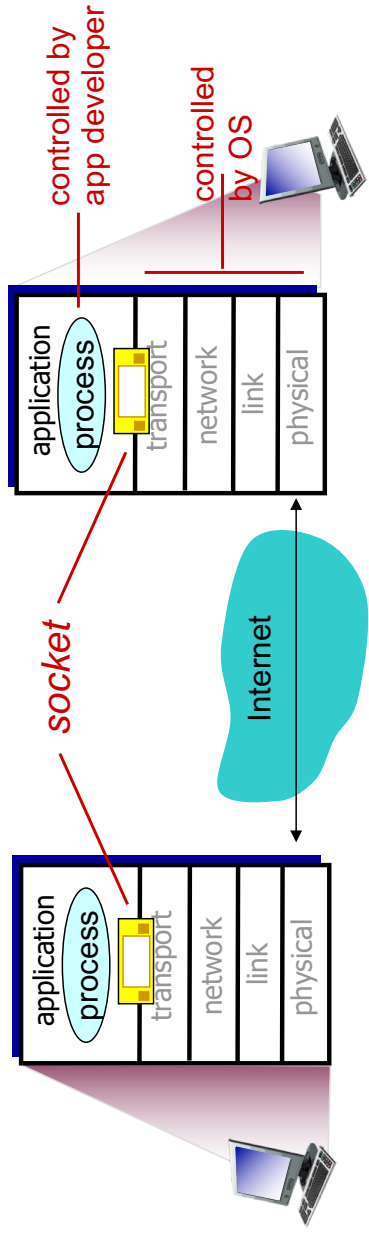
client process: process that initiates communication

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



Two things you need for app communication

- Addressing: We will discuss this in the context of DNS
- Defining an application protocol: Define this now.
- Important: Application protocol is different from the Application itself. The protocol simply provides a programming framework for applications to be written.

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 abstractions does an app need?

We will discuss this as part of transport layer

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, ...