CSC4200/5200 – COMPUTER NETWORKING
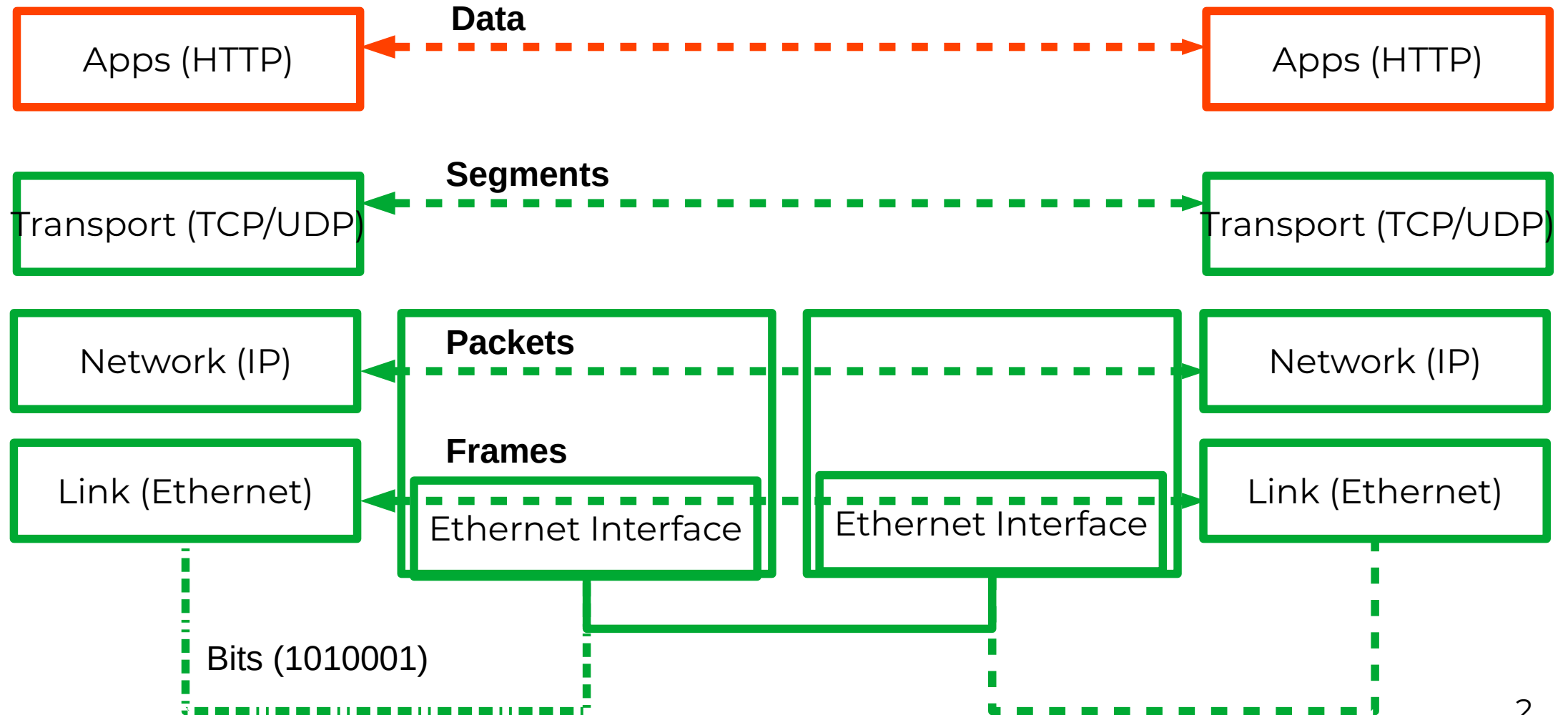
Instructor: Susmit Shannigrahi

NETWORKED APPLICATIONS

sshannigrahi@tntech.edu

GTA: dereddick42@students.tntech.edu

Tennessee
TECH

Apps (HTTP)	**Data** → ←		Apps (HTTP)
Transport (TCP/UDP)	**Segments** → ←		Transport (TCP/UDP)
Network (IP)	**Packets**		Network (IP)
Link (Ethernet)	**Frames**		Link (Ethernet)
	Ethernet Interface	Ethernet Interface	

Bits (1010001)

2

How do you send the cat picture?

- Write your own cat picture transfer app
- In an email
- Upload to a webserver and download using FTP
- Upload to dropbox/AWS/Google cloud
- Use a bit-torrent like protocol
- Use a CDN
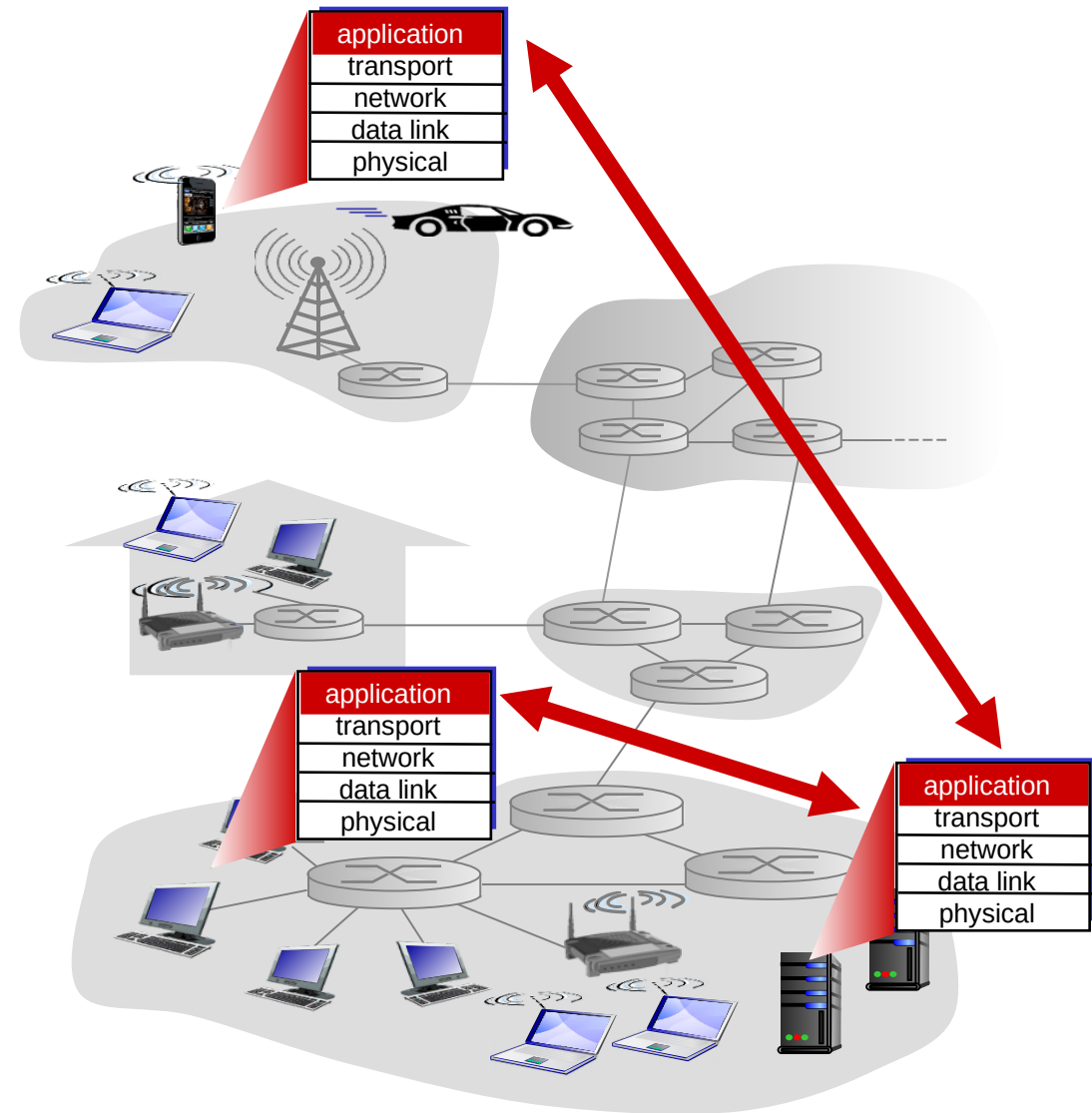- And many other ways....
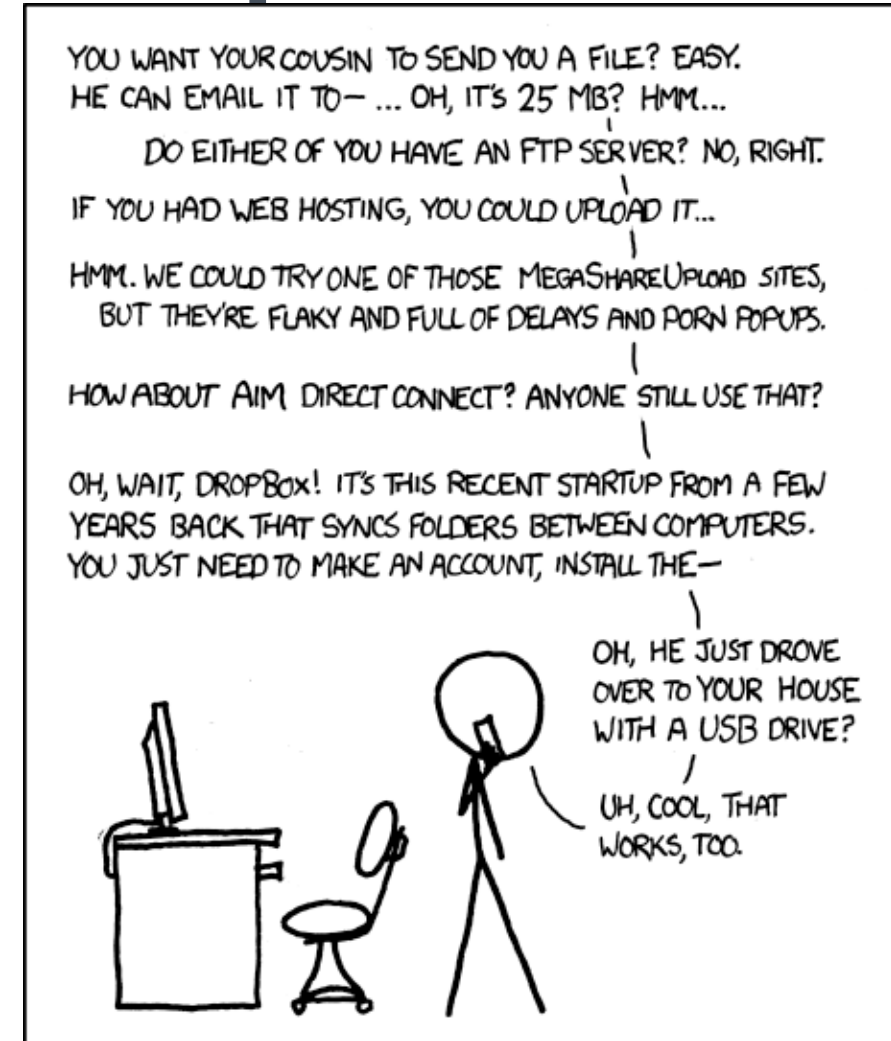
https://xkcd.com/949/

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

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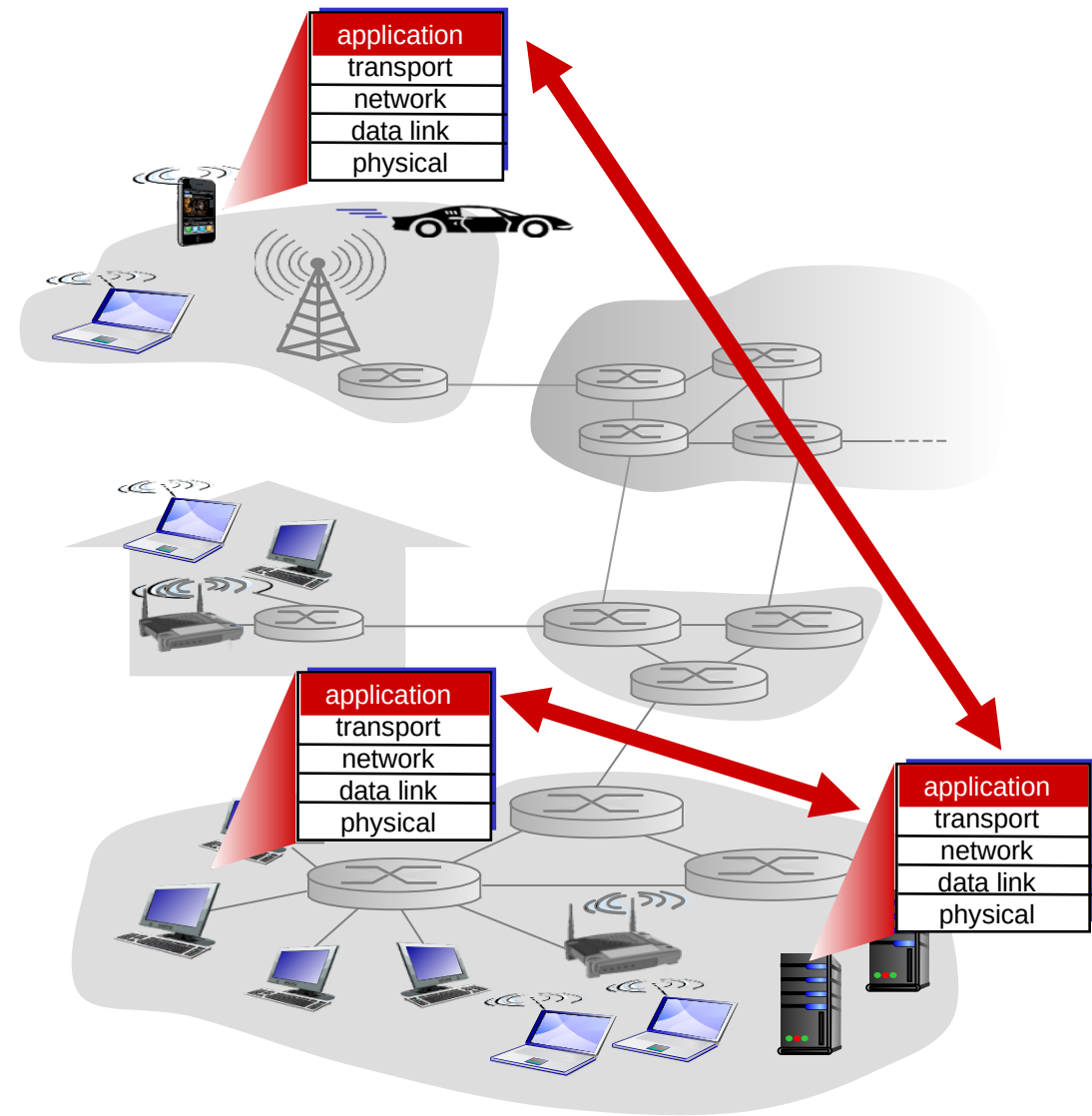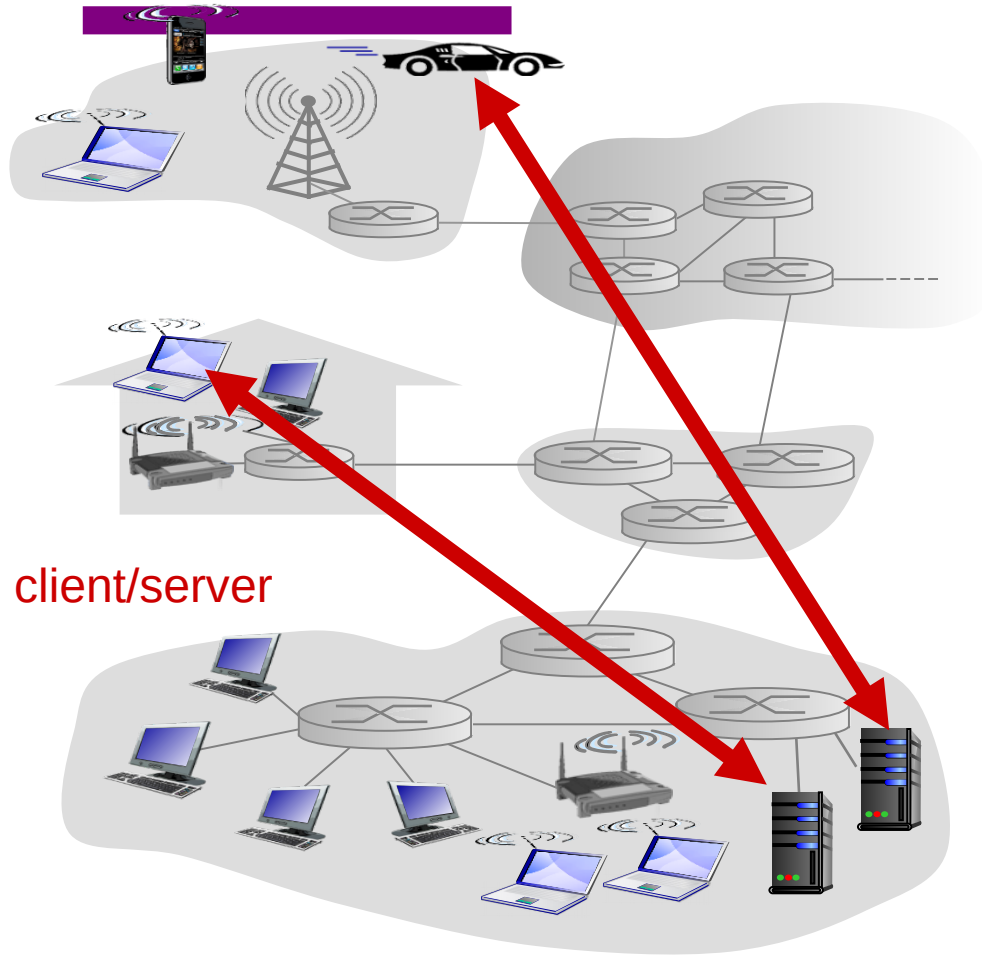- applications on end systems allows for rapid app development, propagation

Application architectures

possible structure of applications:

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

Client-server architecture

client/server

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

- Services between peers
  – *self scalability*

- peers are intermittently connected and change IP addresses
  – complex management

peer-peer

8
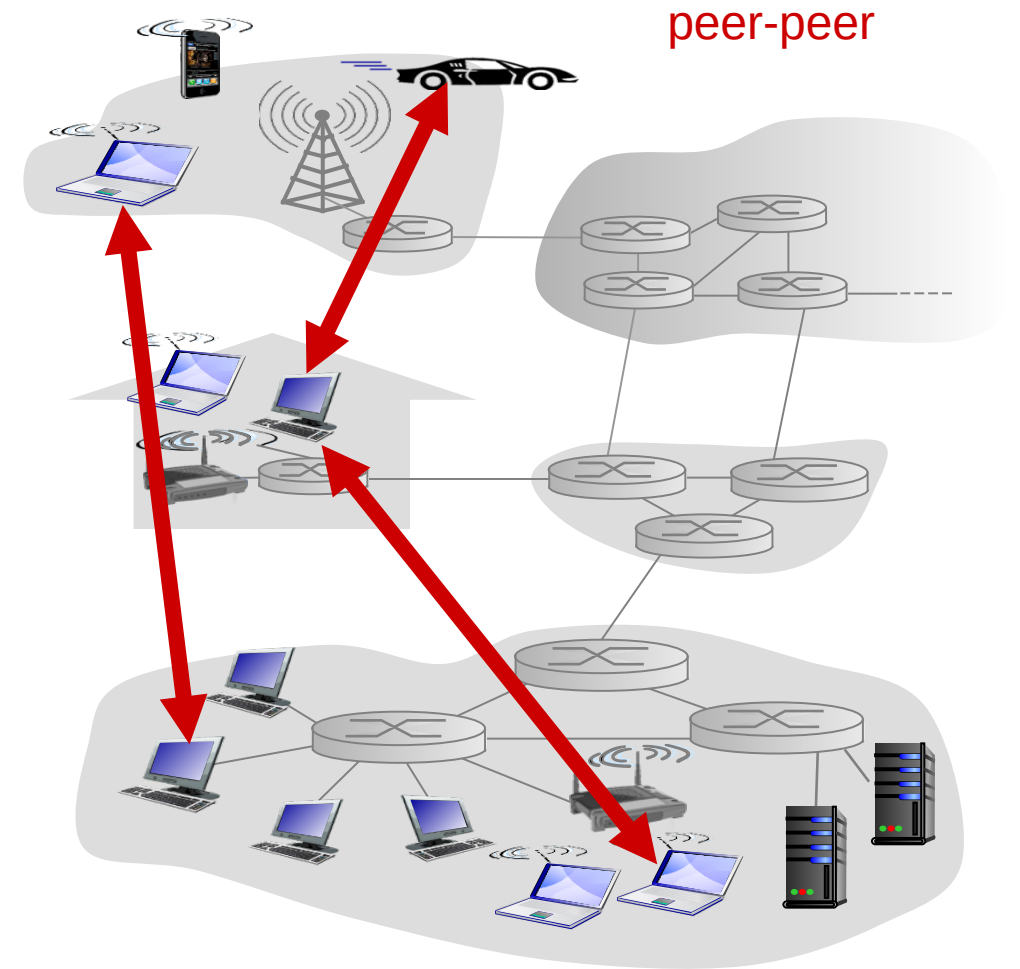
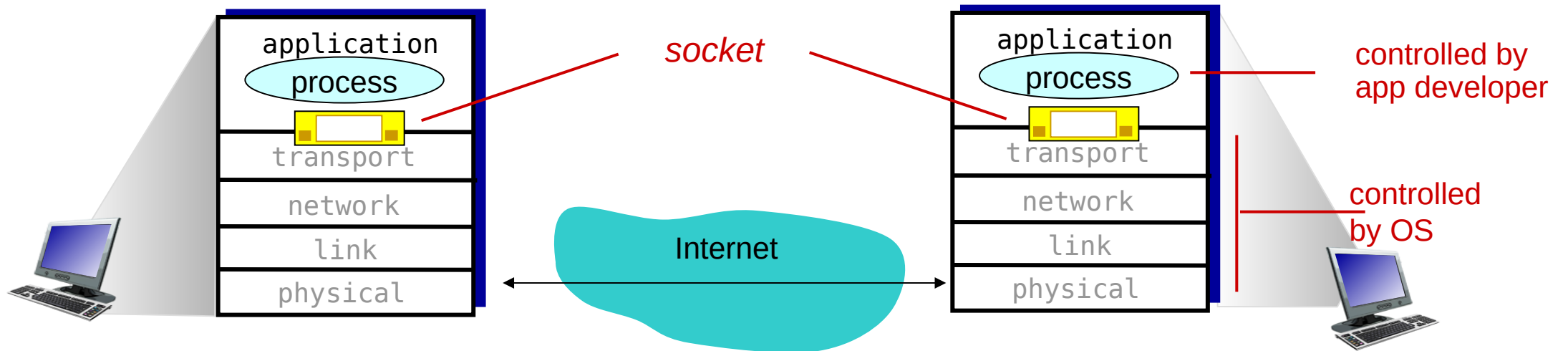# Example of each?

Client server ?

P2P?

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

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

timing
- some apps require low delay to be "effective"

throughput
❖ some apps (e.g., multimedia) require minimum amount of throughput to be " effective"

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	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
text messaging	no loss	elastic	yes and no

Securing Data - Application Layer Function

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
- ❖ More on this later.

Web and HTTP

- *web page* consists of *objects*
- object can be HTML file, JPEG image, Java applet, audio file,...
- web page consists of *base HTML-file* which includes *several referenced objects*
- each object is addressable by a *URL,* e.g.,

`www.someschool.edu/someDept/pic.gif`

host name   path name

Web vs Internet?

http://info.cern.ch/
http://info.cern.ch/hypertext/WWW/TheProject.html

HTTP overview

HTTP - hypertext transfer protocol

- Web's application layer protocol

- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests

PC running
Firefox browser

HTTP request

HTTP response

server running Apache Web server

HTTP request

HTTP response

iphone running Safari browser
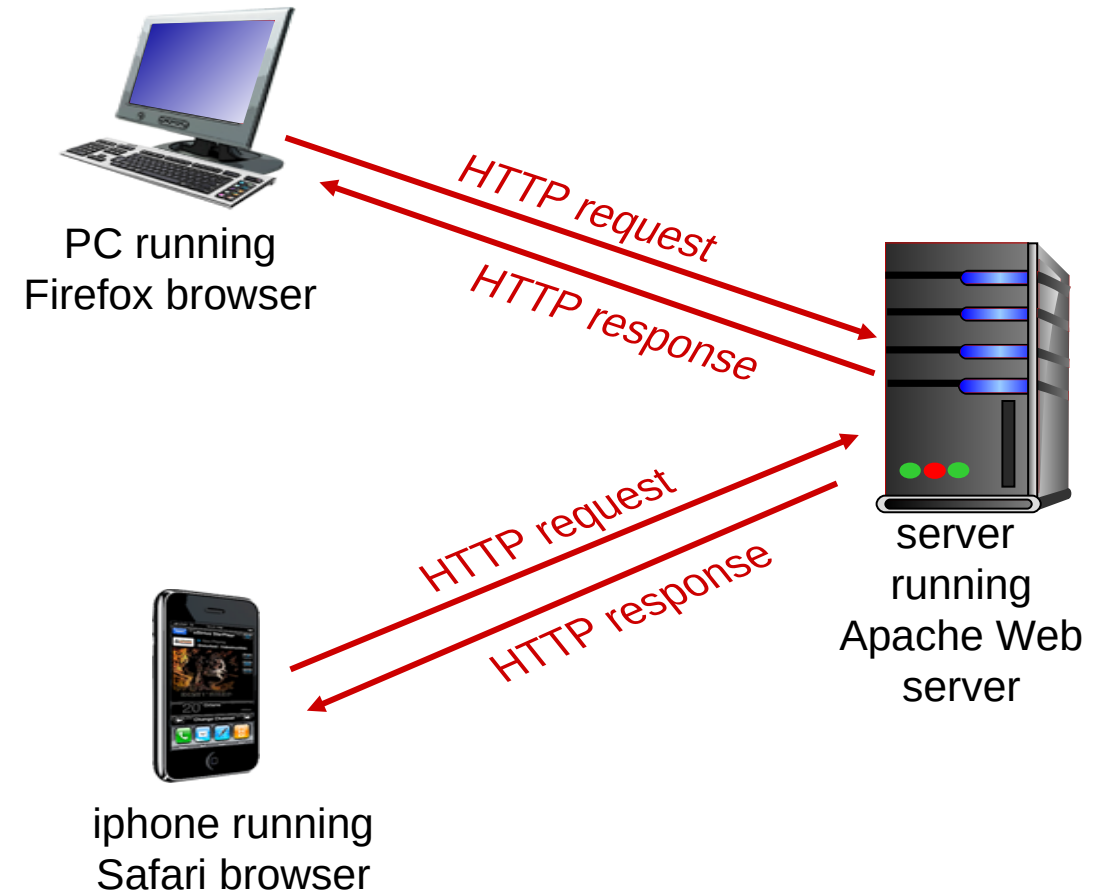
HTTP overview (continued)

uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests
- Applications may make it almost "stateful"

HTTP connections (Remember it uses TCP)

non-persistent HTTP

- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections

persistent HTTP

- multiple objects can be sent over single TCP connection between client, server

HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:
  – ASCII (human-readable format)

carriage return character

line-feed character

request line
(GET, POST,
HEAD commands)

header
lines

carriage return,
line feed at start
of line indicates
end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP response message

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

21

HTTP response status codes

❖ <u>status code</u> appears in 1st line in server-to-client response message.

❖ some sample codes:

200 OK
- request succeeded, requested object later in this msg

301 Moved Permanently
- requested object moved, new location specified later in this msg (Location:)
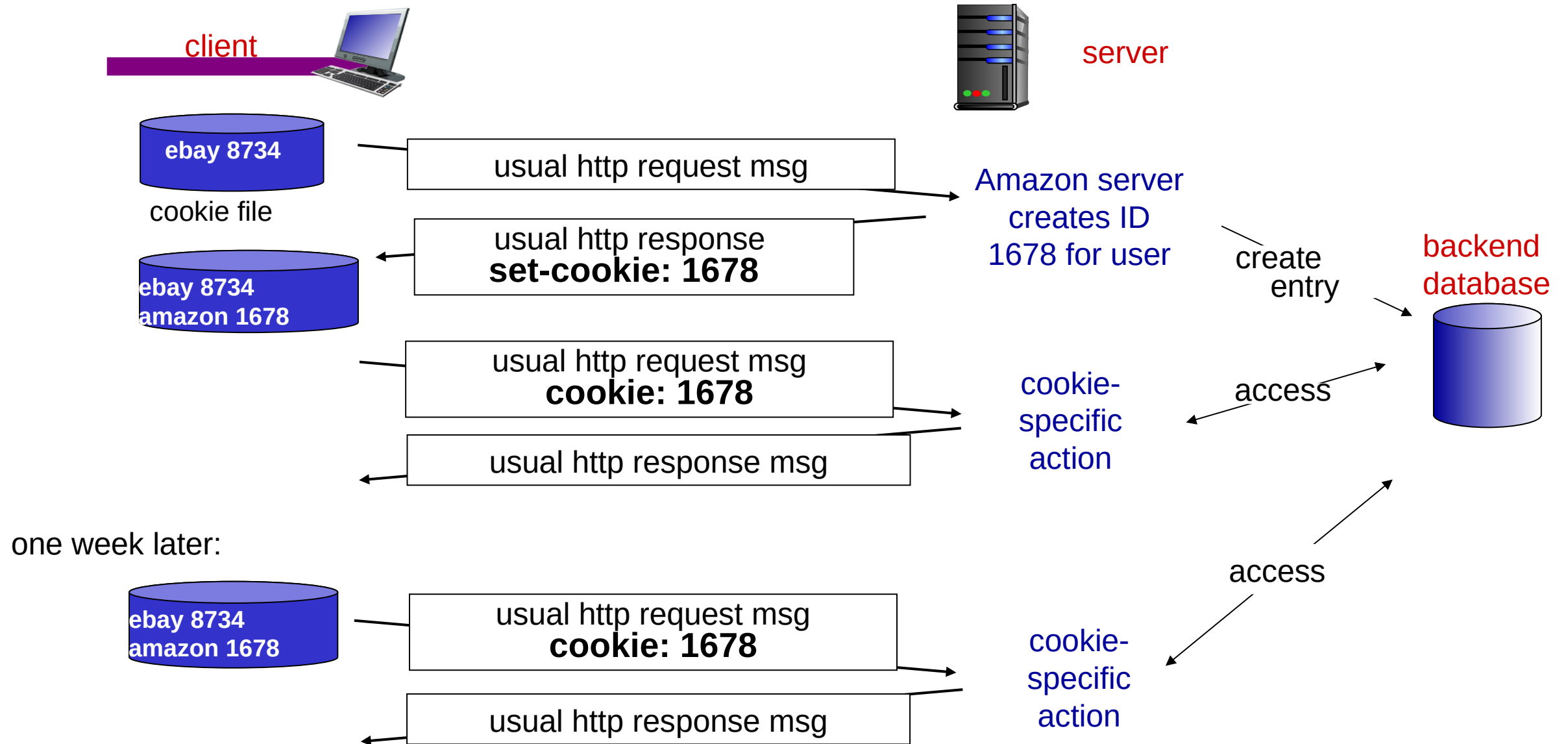
400 Bad Request
- request msg not understood by server

404 Not Found
- requested document not found on this server

505 HTTP Version Not Supported

Cookies: keeping "state"

client

server

ebay 8734

cookie file

usual http request msg

Amazon server
creates ID
1678 for user

usual http response
set-cookie: 1678

create
entry

backend
database

ebay 8734
amazon 1678

usual http request msg
cookie: 1678

cookie-
specific
action

access

usual http response msg

one week later:

ebay 8734
amazon 1678

usual http request msg
cookie: 1678

access

cookie-
specific
action

usual http response msg

Next Steps

DNS and Email
P2P - bittorrent

Apps (HTTP)   **Data**   Apps (HTTP)

Transport (TCP/UDP)   **Segments**   Transport (TCP/UDP)

Network (IP)   **Packets**   Network (IP)

Link (Ethernet)   **Frames**   Link (Ethernet)

Ethernet Interface   Ethernet Interface

Bits (1010001)