Application Layer

Computer Networks

UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

CS4054 - 2025I
PROF.: CWILLIAMS@UTEC.EDU.PE
SRC: KUROSE – COMPUTER NETWORKS BOOK
PETERSON AND DAVIE – COMPUTER NETWORKS BOOK

Executive Summary

- **Motivation: Network conceptual and implementation aspects can lead to complex applications.**
- **Problem: We need details of the layered network abstraction and its relationship with applications.**
- **Overview:**

 - Layered network review.

 - Application layer and transport relationship

 - Review of two important applications: HTTP and STMP

- **Conclusion: We introduced fundamental aspects of well-known applications such as web HTTP and mail.**

Outline

Introduction

Processes and sockets

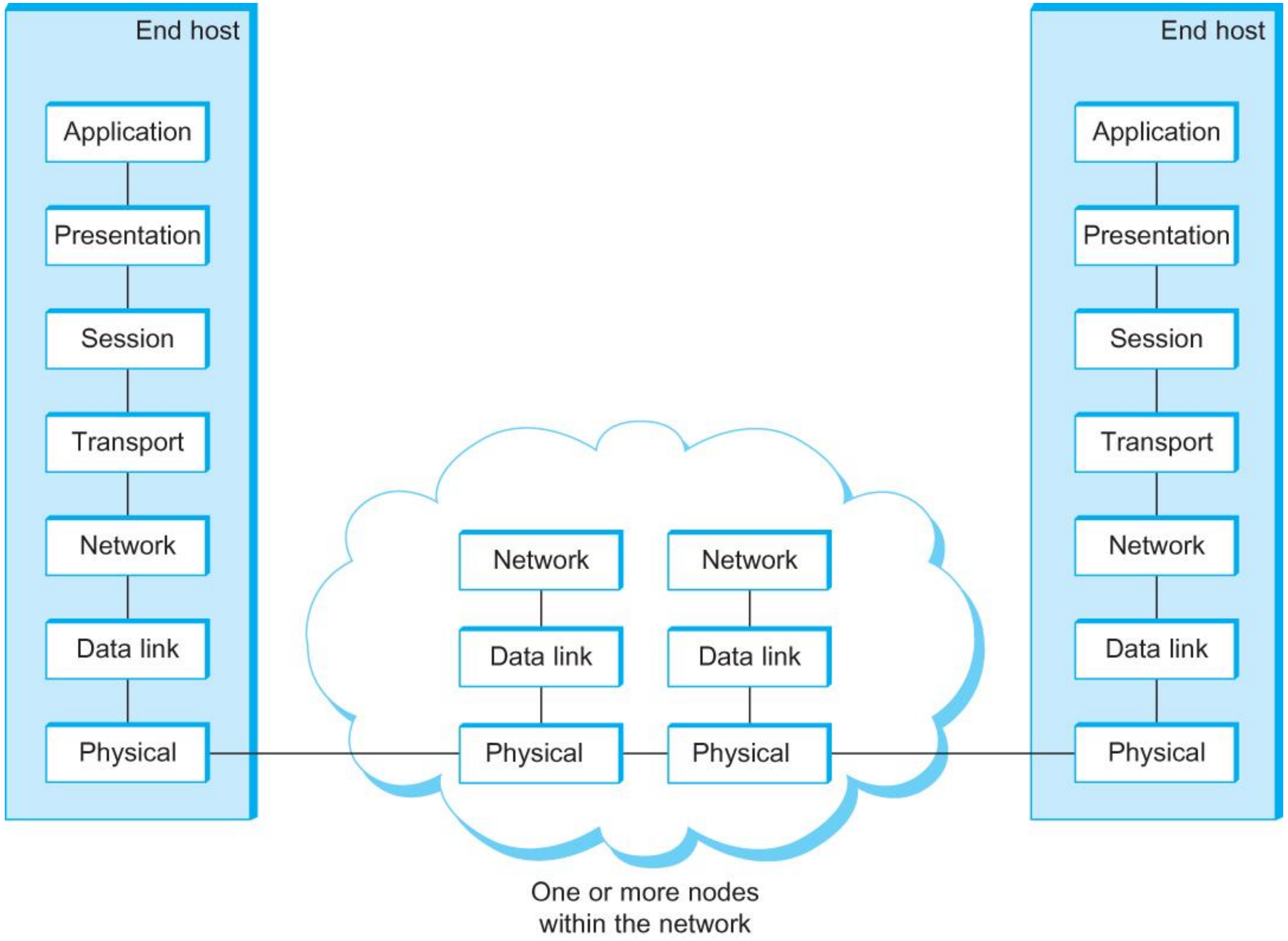
Application layer and protocols

HTTP

SMTP

Conclusions

Recall: OSI Architecture



One or more nodes
within the network

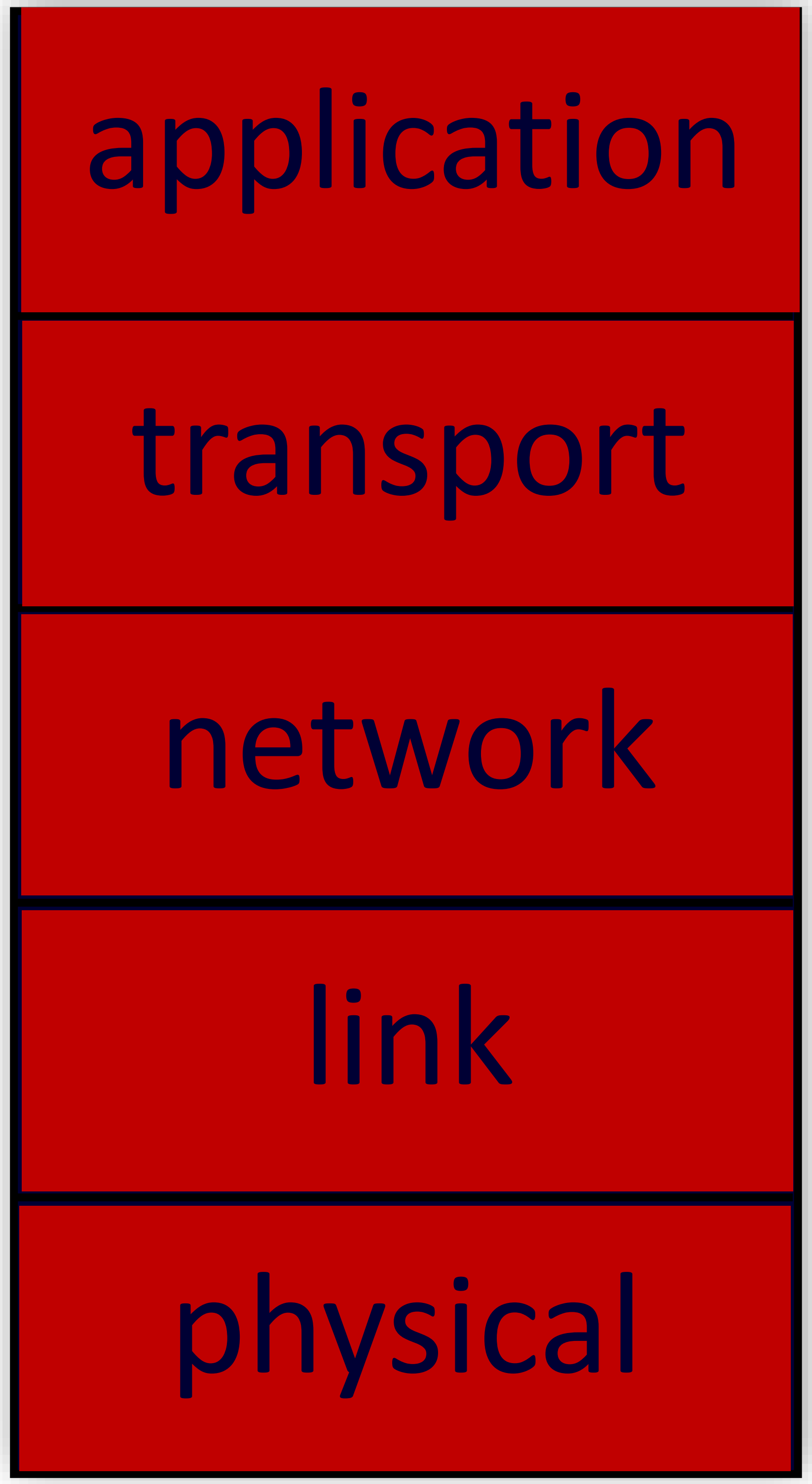
OSI 7-layer Model

OSI – Open Systems Interconnection

- **As a layer standard for network architecture.**

Recall: Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, IMAP, SMTP, DNS
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi), PPP
- *physical:* bits “on the wire”



application

transport

network

link

physical

Some network apps

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)

possible structure of applications:

- client-server: usual architecture
- peer-to-peer (P2P): we will cover this in detail next week

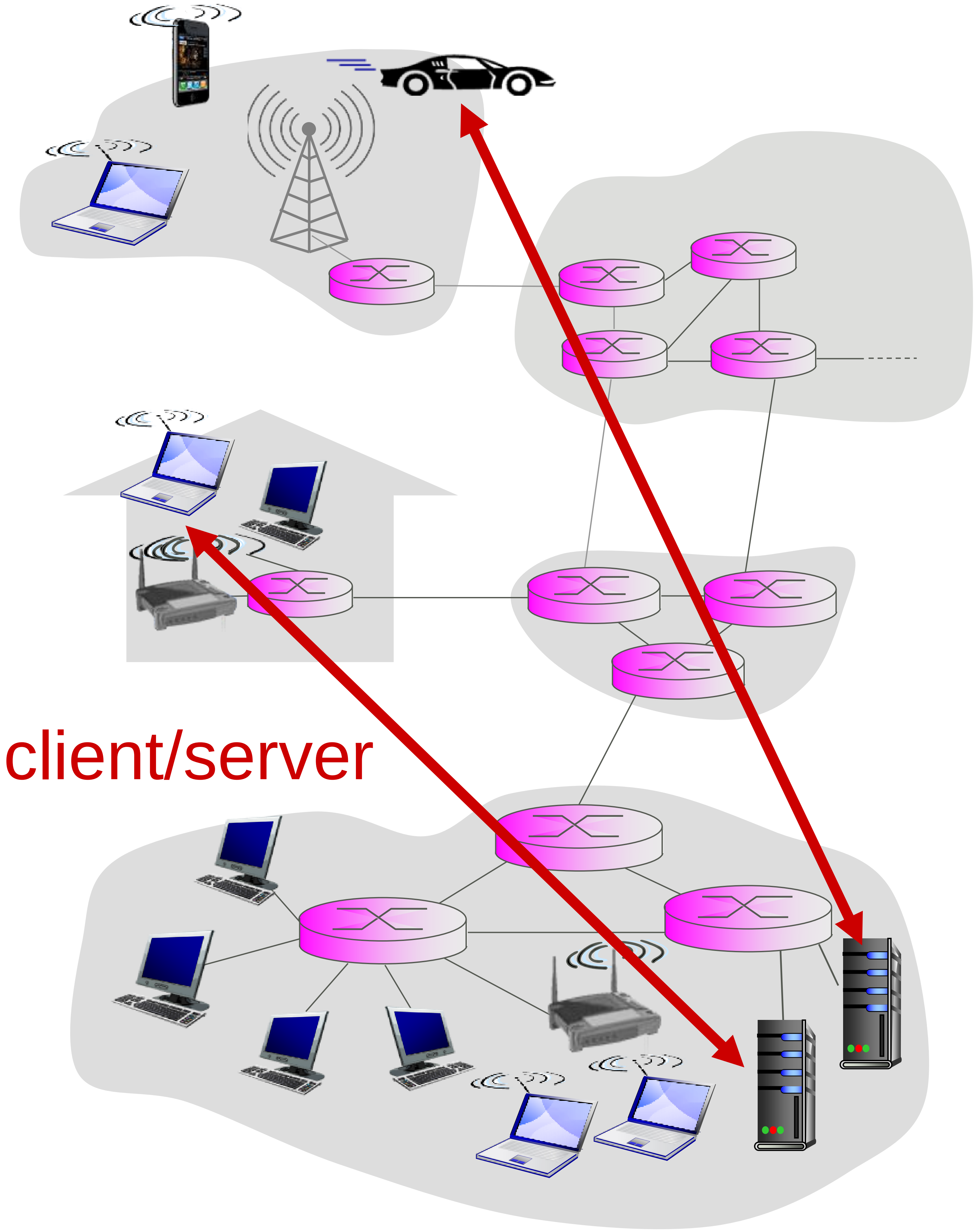
Client-server architecture

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



Outline

Introduction

Processes and sockets

Application layer and protocols

HTTP

SMTP

Conclusions

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

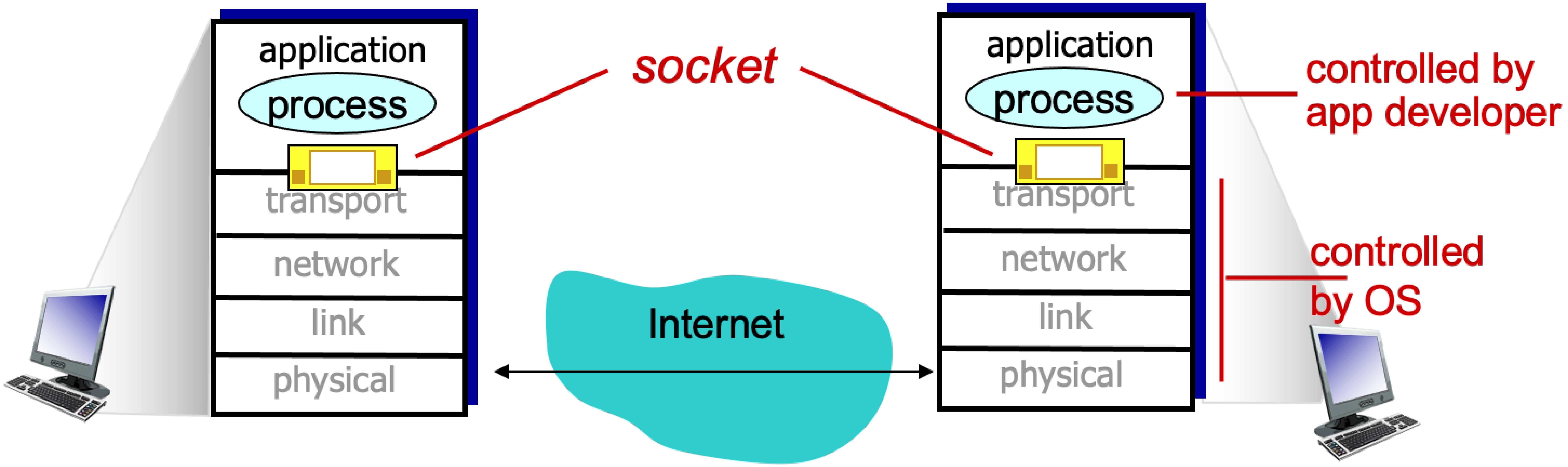
clients, servers

client process: process that initiates communication

server process: process that waits to be contacted

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



Outline

Introduction

Processes and sockets

Application layer and protocols

HTTP

SMTP

Conclusions

App-layer protocol defines

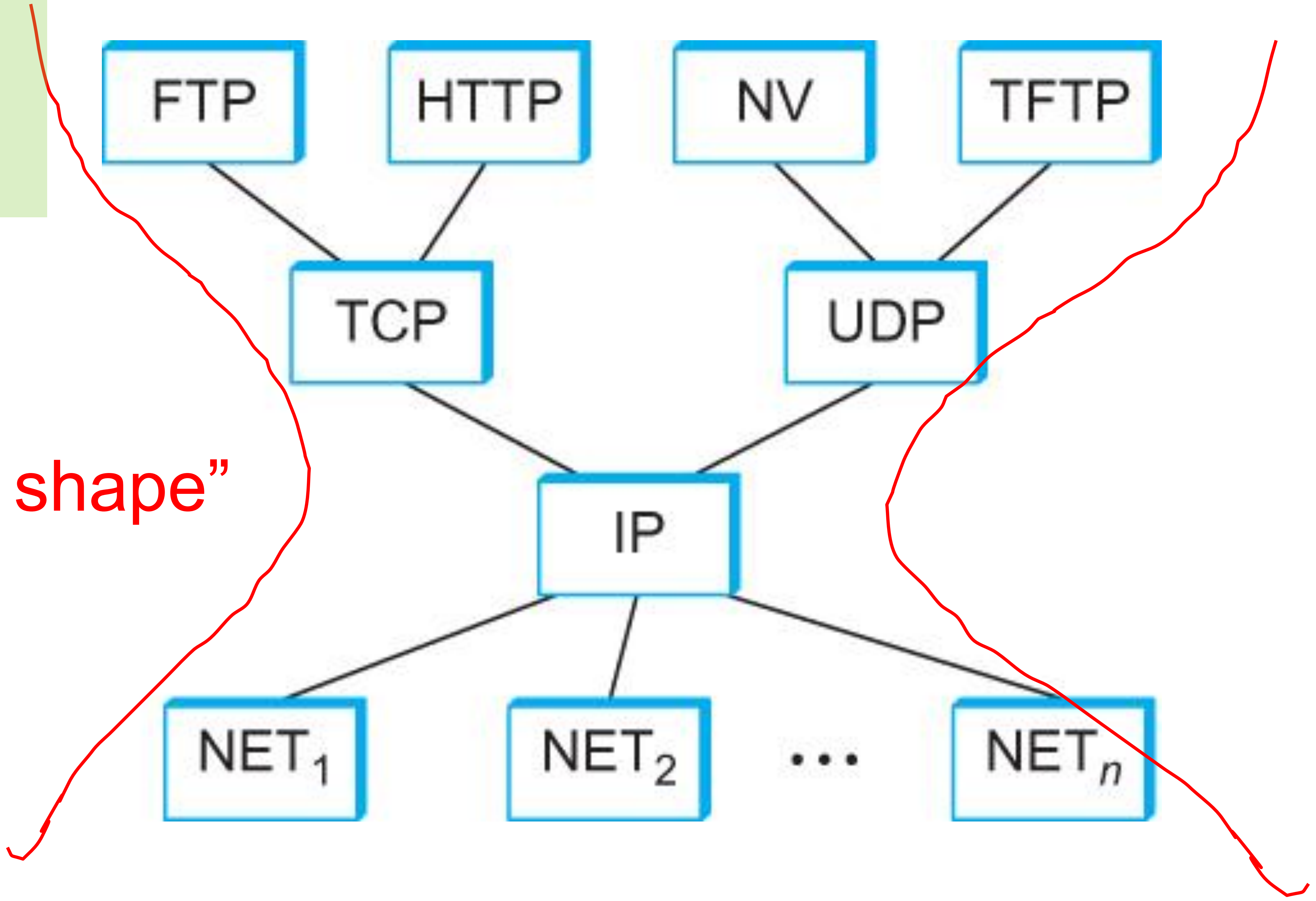
- types of messages exchanged,
  - e.g., request, response
- message syntax:
  - what fields in messages & how fields are delineated
- message semantics
  - meaning of information in fields
- rules for when and how processes send & respond to messages

open protocols:
defined in RFCs
allows for interoperability
e.g., HTTP, SMTP
proprietary protocols:
e.g., Skype

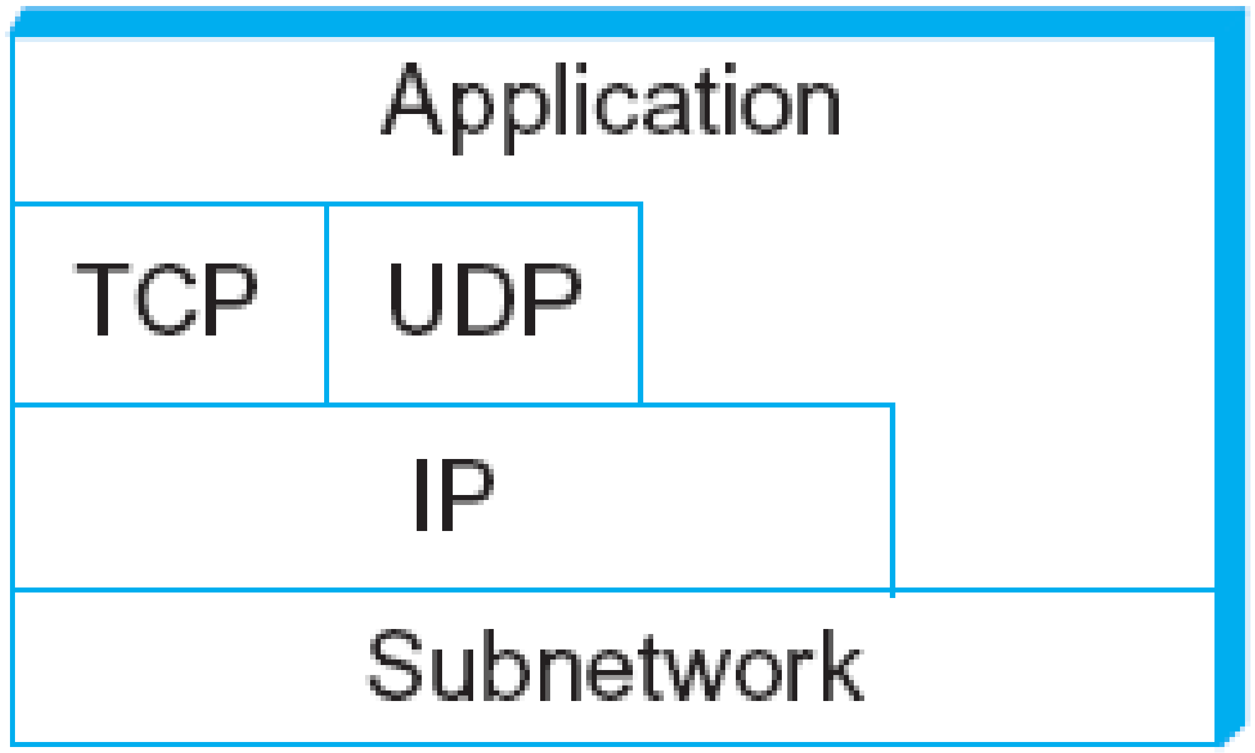
Recall: Internet Architecture Protocol Graph

We are here

“Hourglass shape”



Internet Protocol Graph

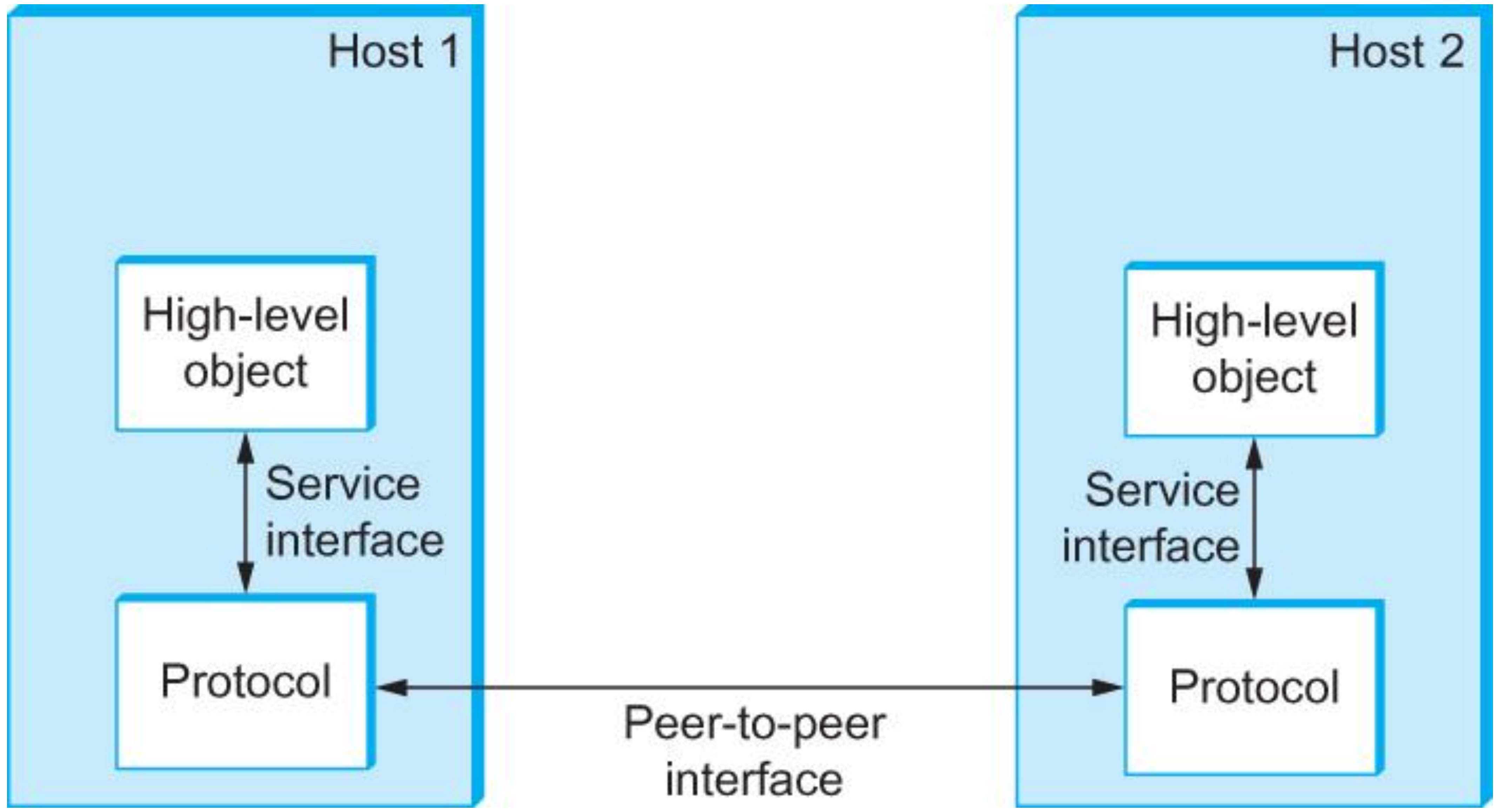


Alternative view of the Internet architecture. The “Network” layer shown here is sometimes referred to as the “sub-network” or “link” layer.

Recall: Protocols and Interfaces

- Defines the interfaces between **the layers (more later)** in the same system and with the layers of peer system
- **Building blocks** of a network architecture

- **Each protocol object has** two different interfaces
  1. **service interface:** operations on this protocol
  2. **peer-to-peer interface:** messages exchanged with peer



Service and Peer Interfaces

Transport layer provides to the App layer

1. data integrity

some apps (e.g., file transfer, web transactions) require 100% reliable data transfer

other apps (e.g., audio) can tolerate some loss

2. security

encryption, data integrity, …

3. 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

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

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	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), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Outline

Introduction

Processes and sockets

Application layer and protocols

HTTP

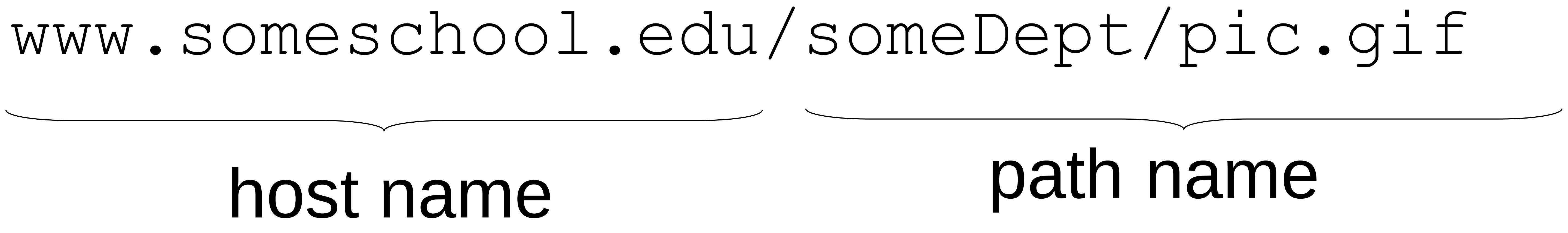
SMTP

Conclusions

Web and HTTP

First, a review…

- *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.,

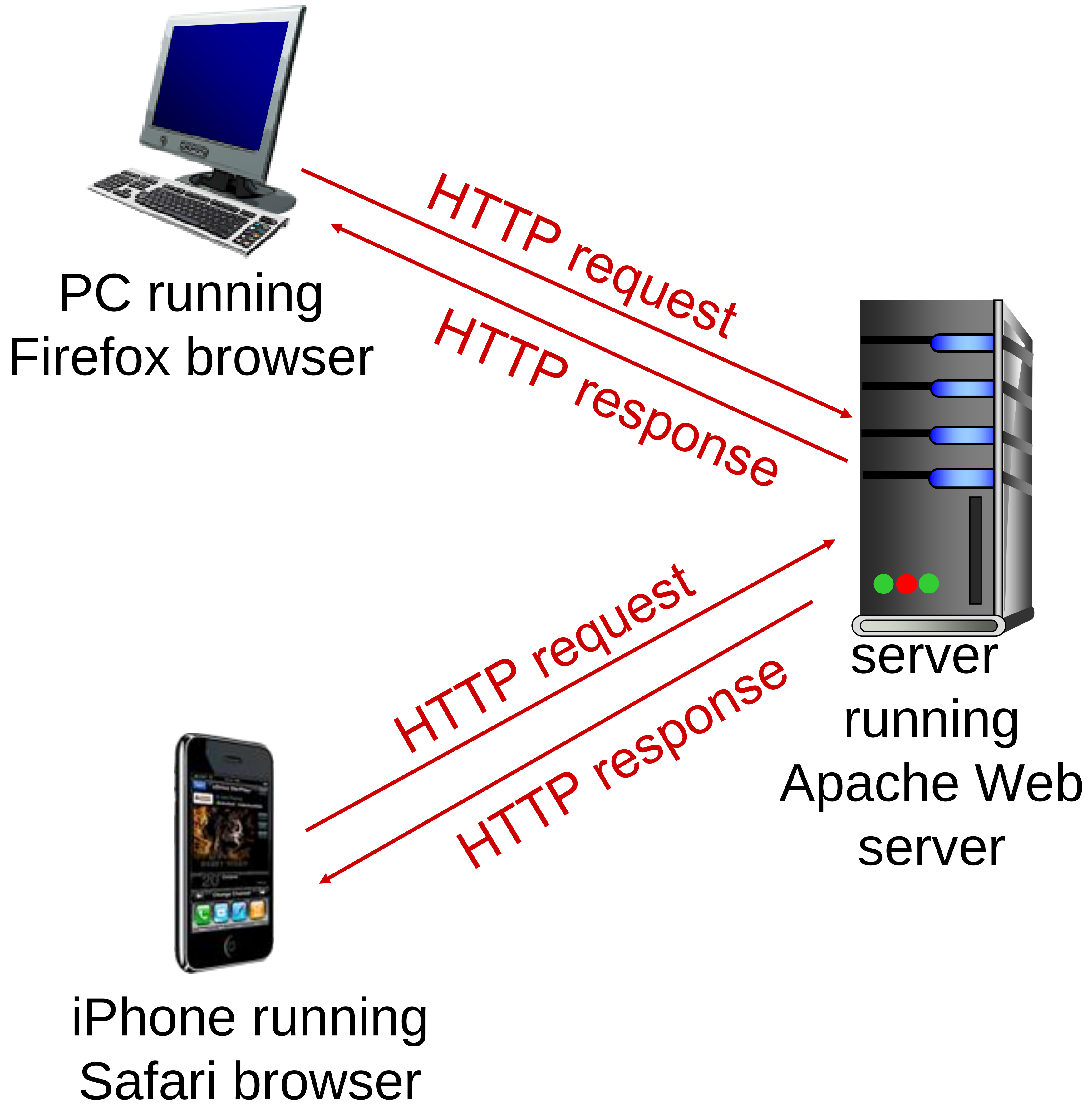


```
www.someschool.edu/someDept/pic.gif
```

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



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

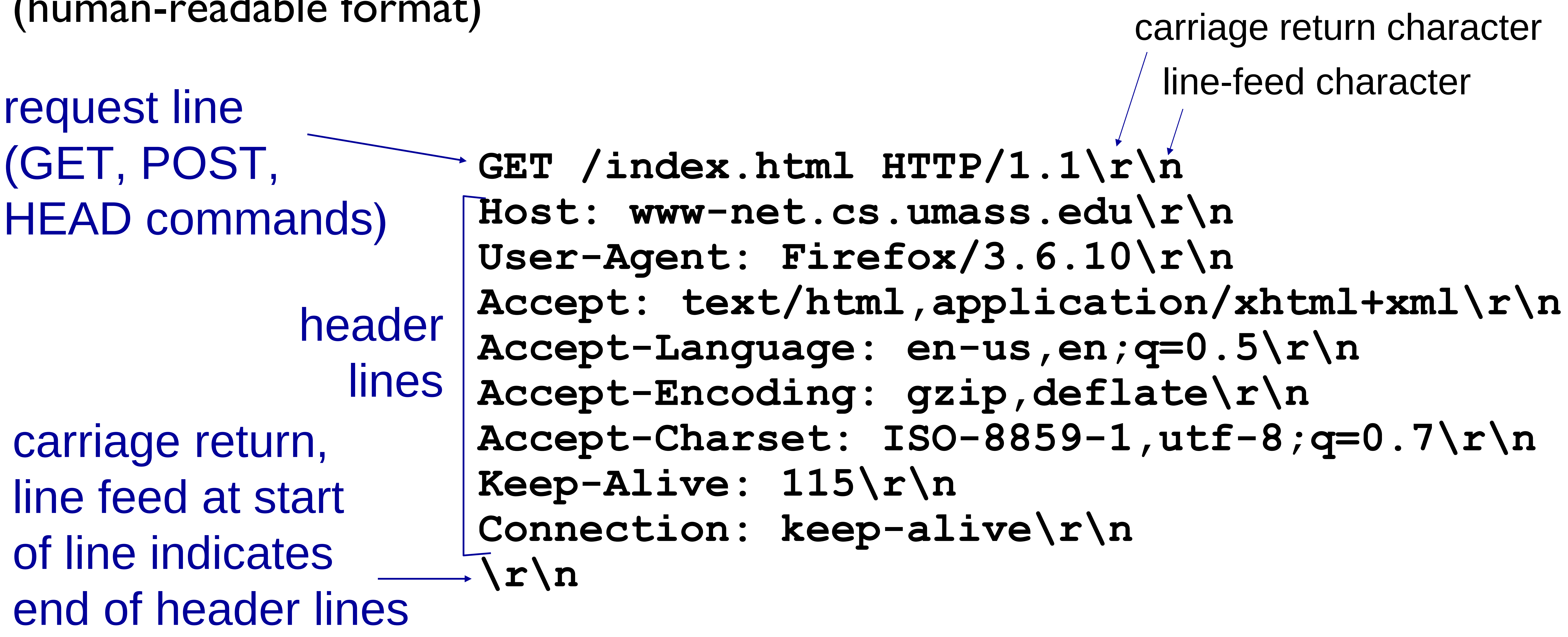
aside

protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

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 ...
```

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

User-server state: cookies

Why?

HTTP is "stateless"
server maintains no information about past client requests

many Web sites use cookies

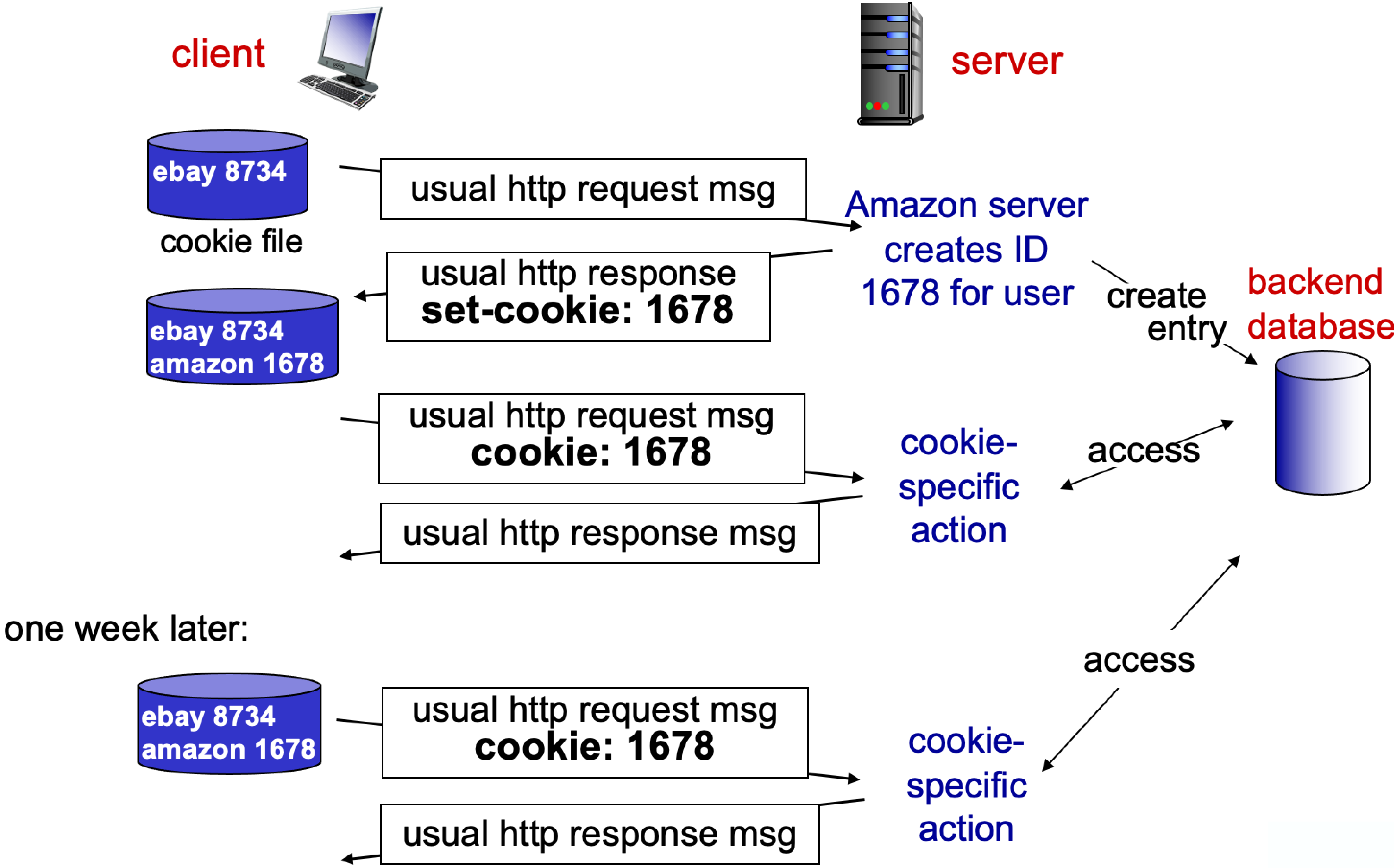
four components:

1) cookie header line of HTTP *response* message
2) cookie header line in next HTTP *request* message
3) cookie file kept on user' s host, managed by user' s browser
4) back-end database at Web site

example:

- **Susan always access Internet from PC**
- **visits specific e-commerce site for first time**
- **when initial HTTP requests arrives at site, site creates**
  - unique ID
  - entry in backend database for ID

Cookies: keeping "state" (cont.)



Outline

Introduction

Processes and sockets

Application layer and protocols

HTTP

SMTP

Conclusions

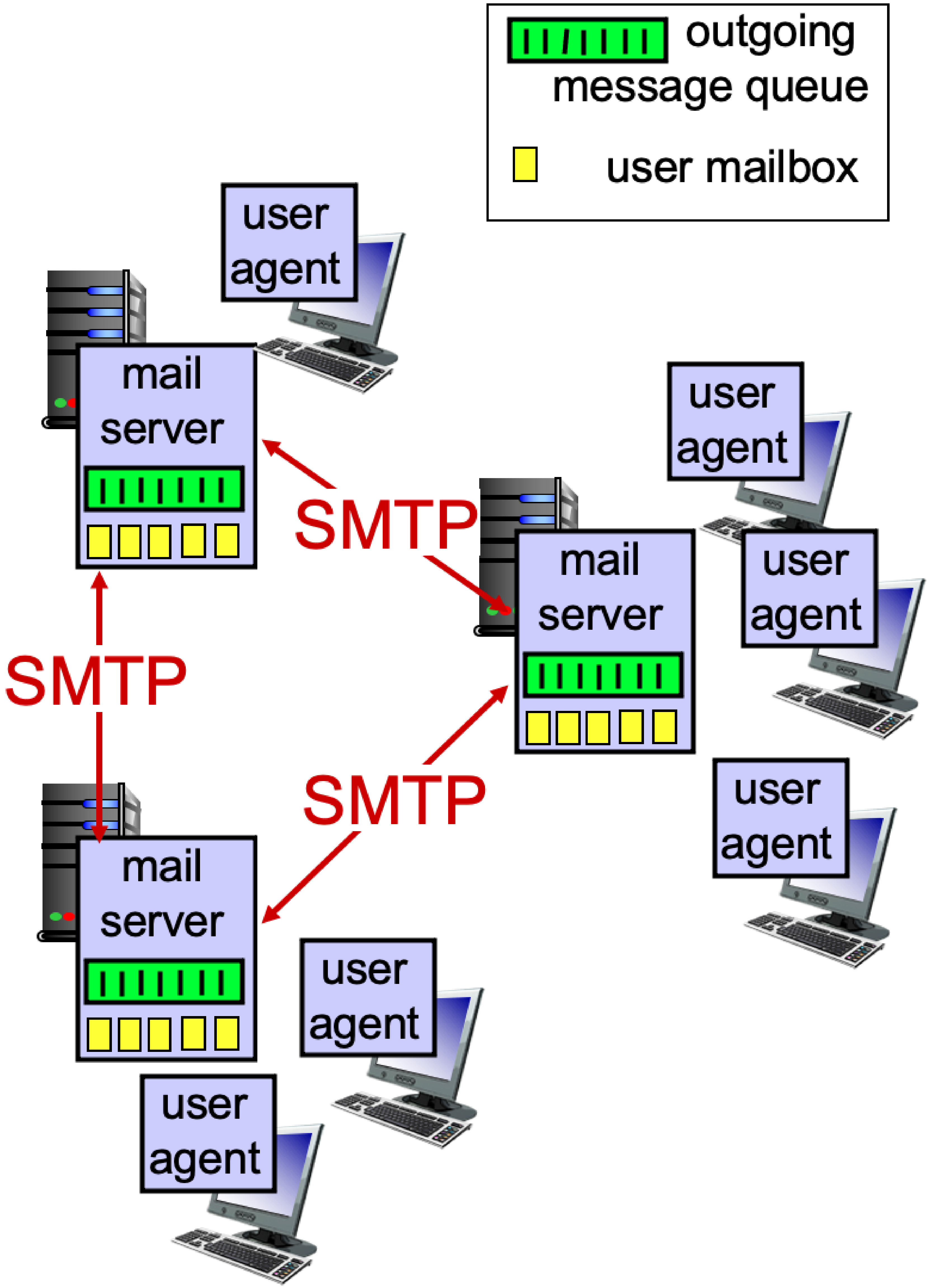
Electronic mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server



Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction (like HTTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCI

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message

comparison with HTTP:

- **HTTP: pull**
- **SMTP: push**

- **both have ASCII command/response interaction, status codes**

- **HTTP: each object encapsulated in its own response message**
- **SMTP: multiple objects sent in multipart message**

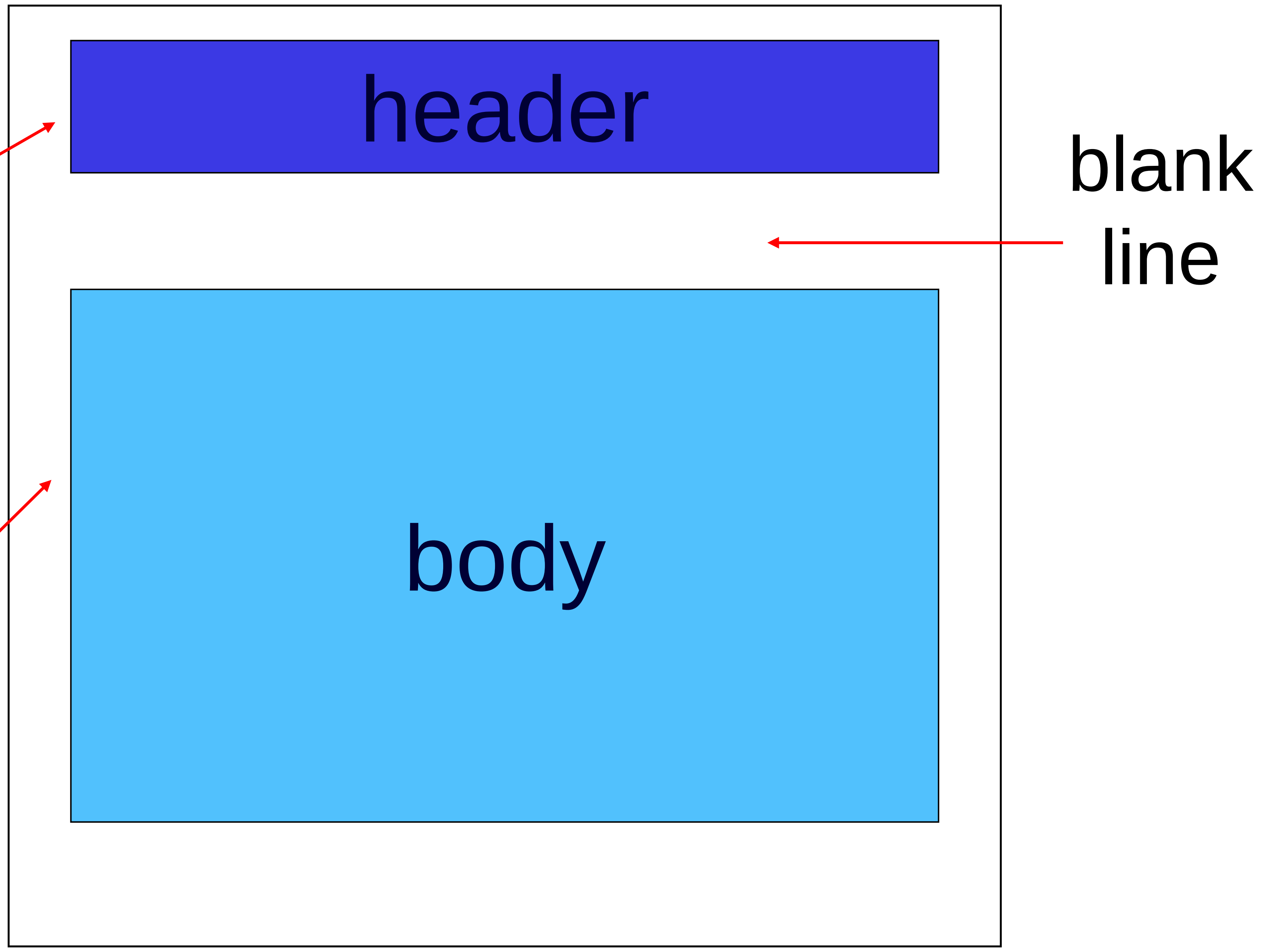
Mail message format

SMTP: protocol for exchanging email messages

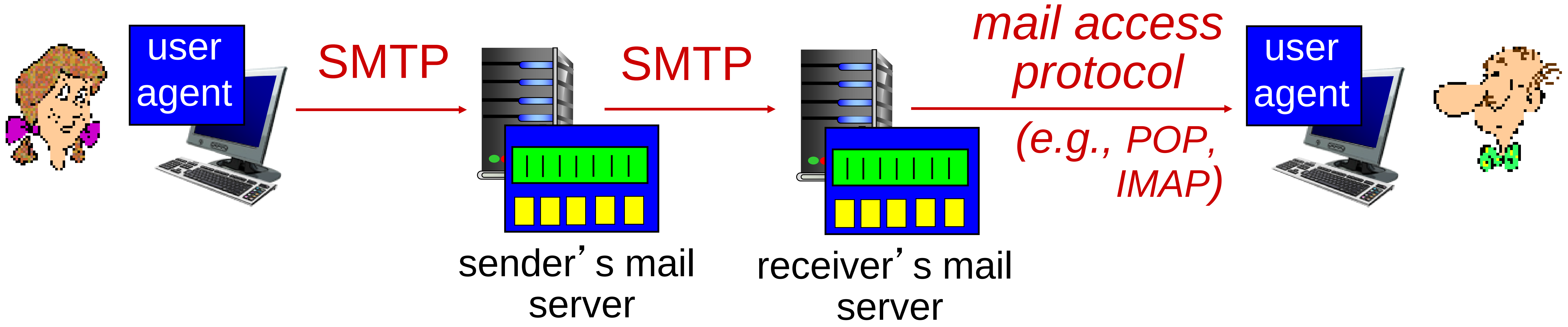
RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the "message"
  - ASCII characters only



Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

Outline

Course Logistics

Introduction

Internet Structure

Internet as a Service

Internet Architecture

Conclusions

Summary

We review the Internet, from the:

- Structure perspective
- Application perspective

- **Both perspectives are important,** each one relies on the other for standarization.
- **Internet is a network of computer network,** that follow protocols and it is organized (by astraction) as layers.