UMASS AMHERST

ECE 671
Introduction to Computer Networks

Lesson 3
Application Layer

Poll: Rationale

- Which aspect of the Computer Engineering field do you think you will work in? (There is no correct answer)
  - -Networking
  - -Software Engineering
  - -Computer Architecture
  - -Embedded Systems
  - -VLSI
  - -Security
  - -Cyber-physical Systems / Internet of Things
  - -Haven't decided

Rationale

- Application layer is highest level protocol used by distributed applications
- Illustrates basic protocol interactions
- Application requirements highlight the need for different transport layer protocols and security

Objectives

- Analyze use of application layer by distributed applications
- Illustrate basic protocol interactions
- Explain how application requirements highlight need for different transport layer protocols and security
- Research the importance of content distribution networks (CDNs)

Video: Prior Knowledge

- Internet protocols

Prior Knowledge

- Protocol stack with layers
  - Each layer provides a service
  - Layering isolates complexity
- Protocols define interactions
  - Message formats
  - Actions in response to messages

Orchestrated Discussion (Hand Raise): Lesson Reflection Feedback

- Discuss questions and comments on Lesson Reflection from prior lesson

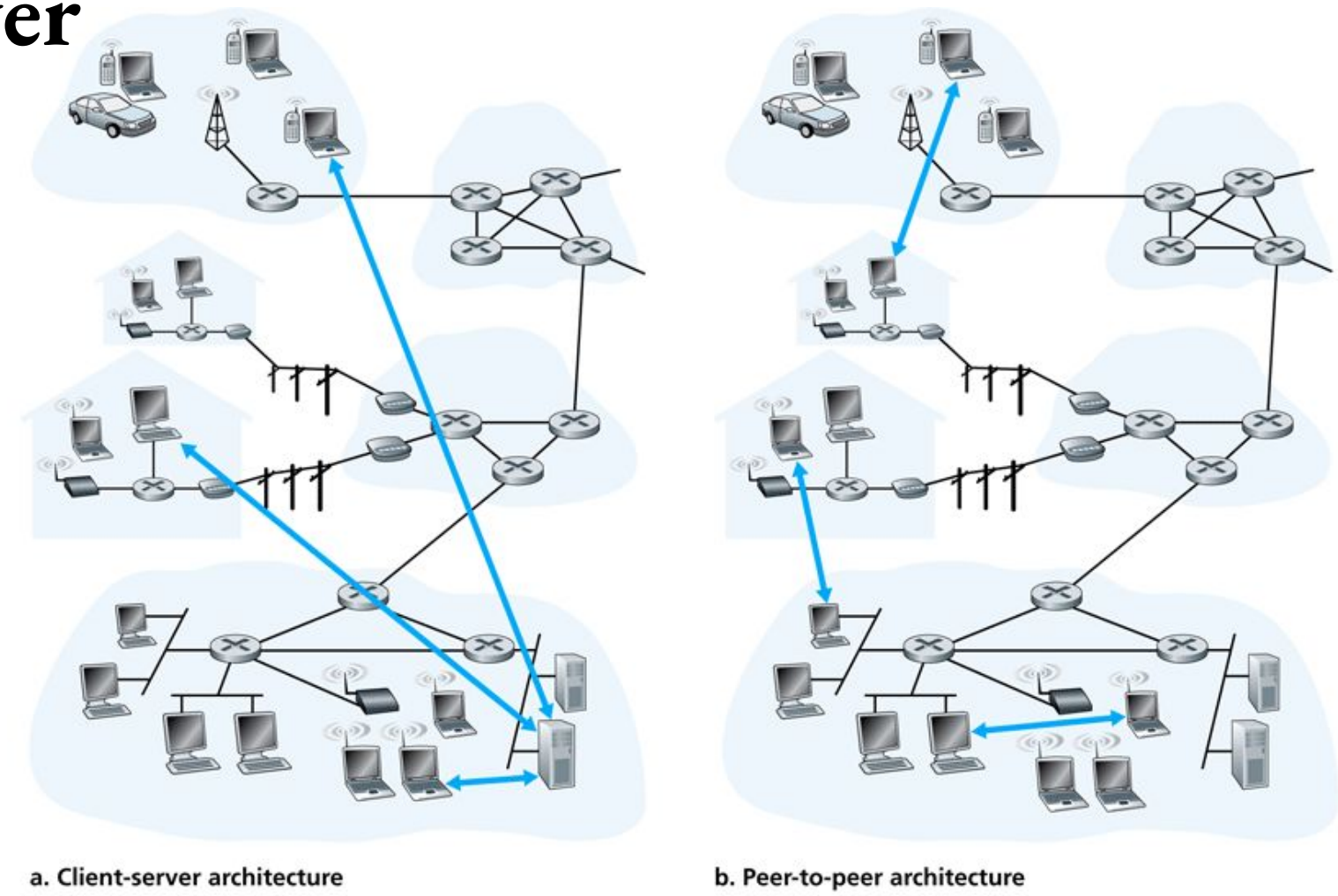
Review of Specific Protocols

- We will briefly review the top four protocols
- For full details
  - Textbook
  - Requests for comments (RFCs)

Layer	Example protocols
Application layer	Hypertext Transfer Protocol (HTTP)
Transport layer	Transmission Control Protocol (TCP)
Network layer	Internet Protocol (IP)
Link layer	Ethernet
Physical layer	1000BASE-T

Application Layer

- Network applications



a. Client-server architecture

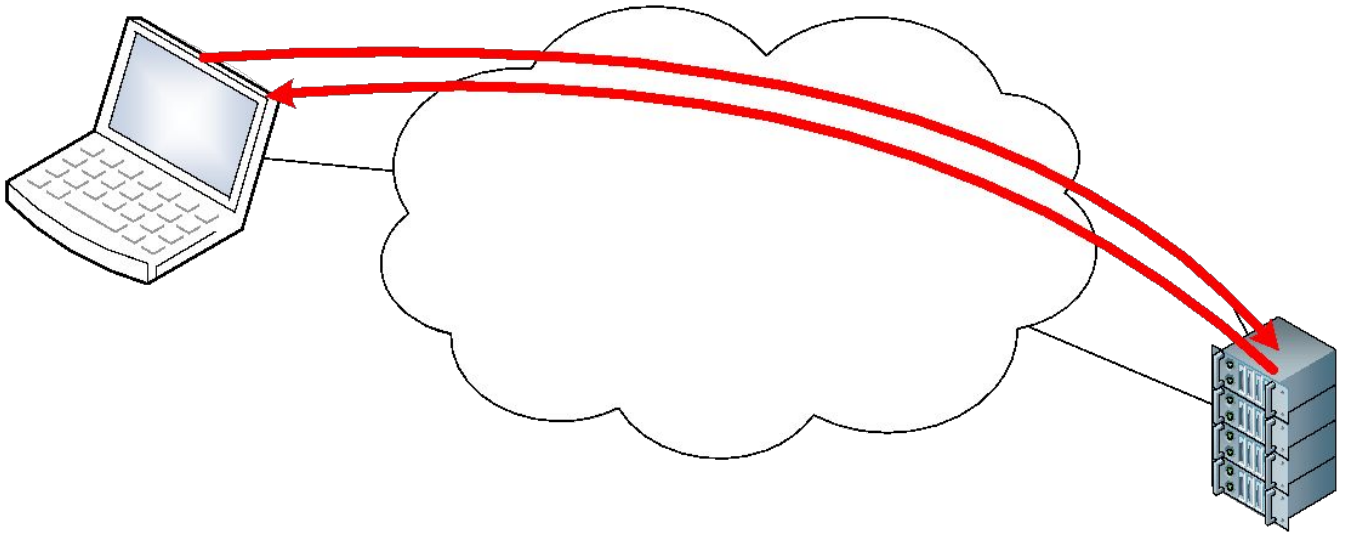
b. Peer-to-peer architecture

Application Layer

- Client-server architecture (most common)
  - Server
    - Provides service (e.g., access to data)
    - Is always on and waits for requests
    - Has well known address
  - Client
    - Initiates communication with server
    - Often determines what is communicated
    - Often unknown to server
  - Communication via “socket”
- Peer-to-peer architecture
  - End-systems act as clients and servers
  - Indexing system figures out which end-system to connect to

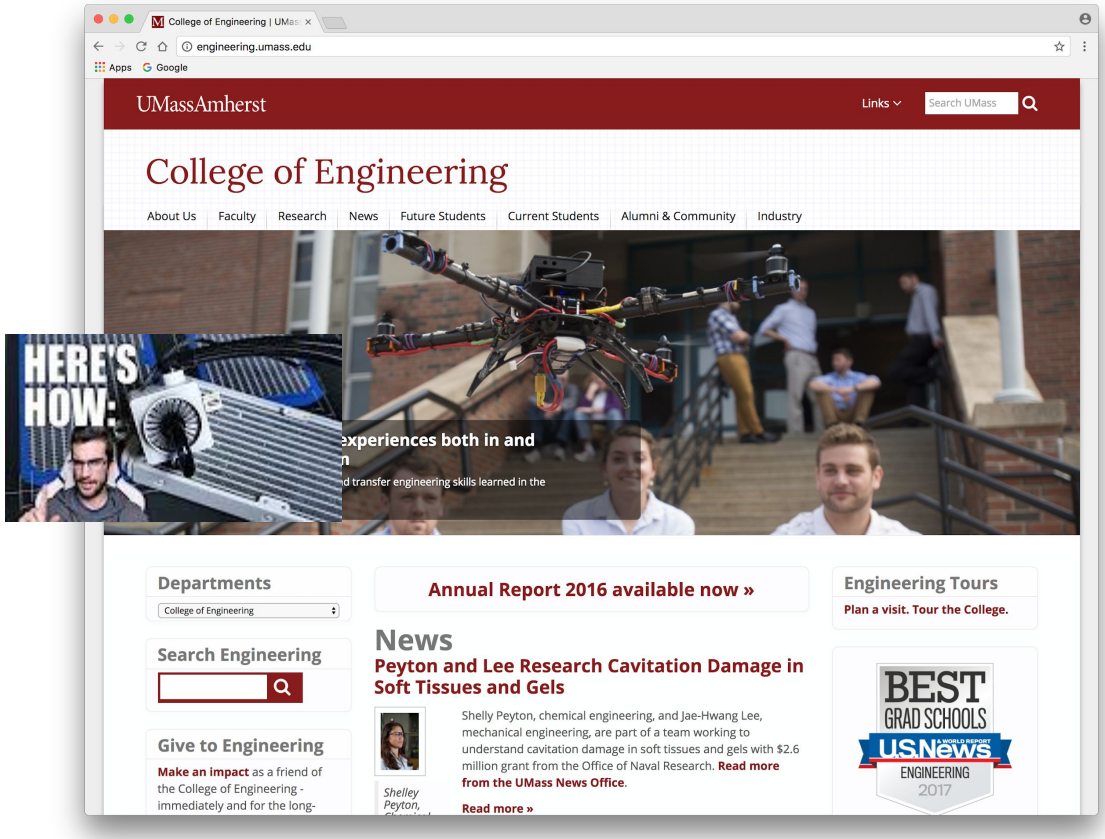
World-Wide Web

- Distributed application to exchange information
- Client (web browser) fetches document from server (document source)
- Good example for application layer interactions



Connected Device: Web Page Load

- Browser connects to server to load requested web page
  - Seems simple
  - We do this many times a day
- We assume that we can establish connection
  - Transport layer service

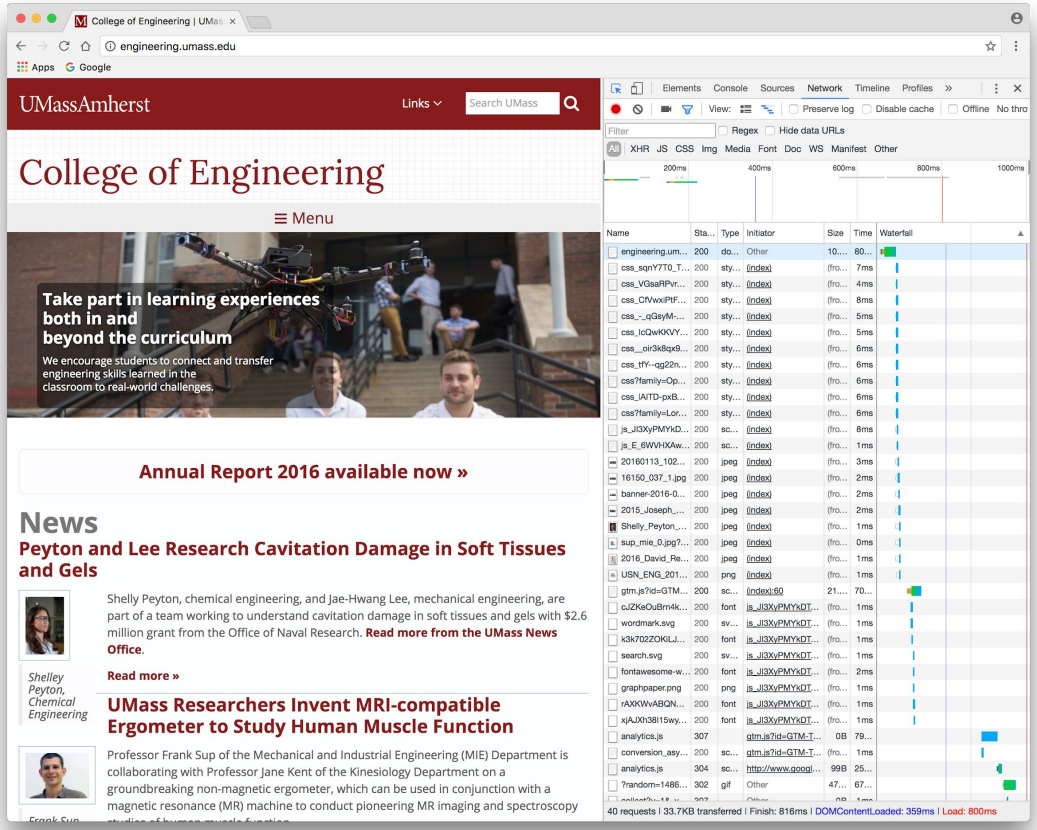


Connected Device: Web Document

- Web documents use Hypertext Markup Language (HTML)
  - Describes to browser how to render content
  - Allows rendering on different types of displays
- HTML document components
  - Meta-information
  - Content
  - References and links to images, videos, other documents, etc.

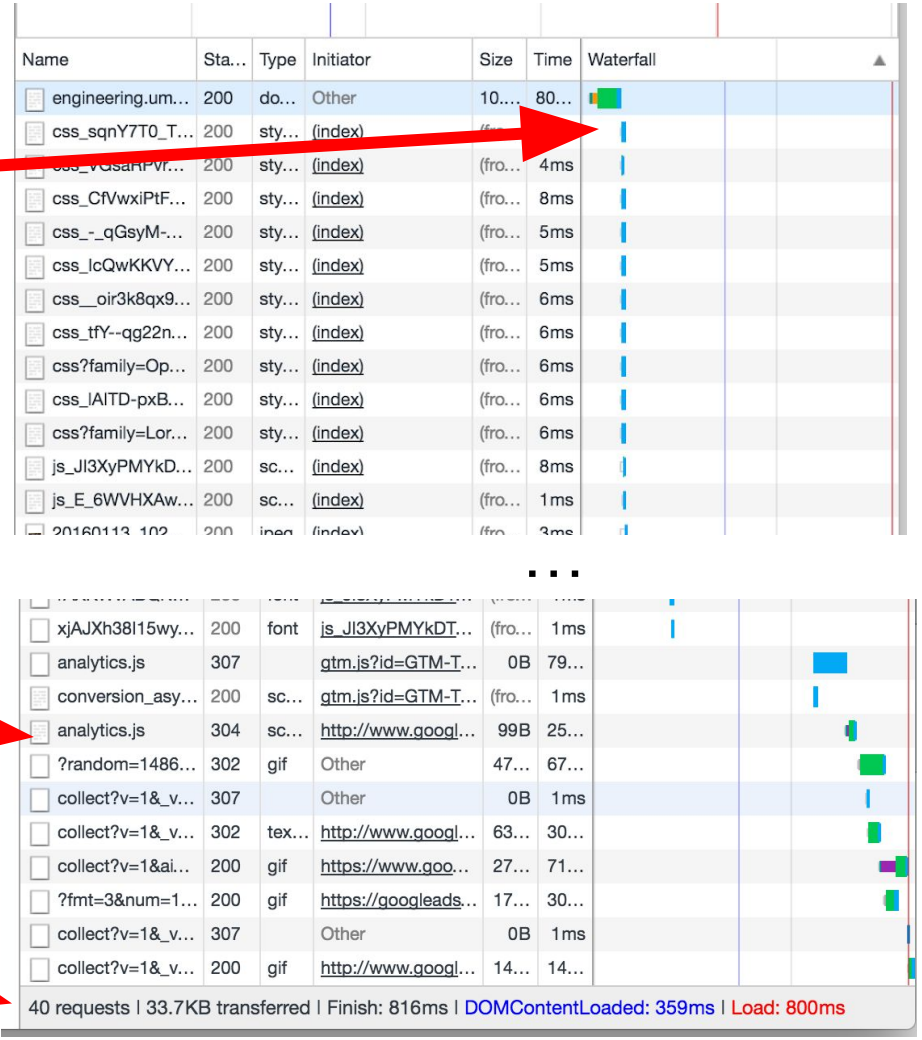
Connected Device: Web Page Load

- All components of web page need to be loaded
  - Only known after initial document is fetched
  - Loads can go to same or different web server
- Demo: Google Chrome browser
  - Developer Mode / Network shows details of load process



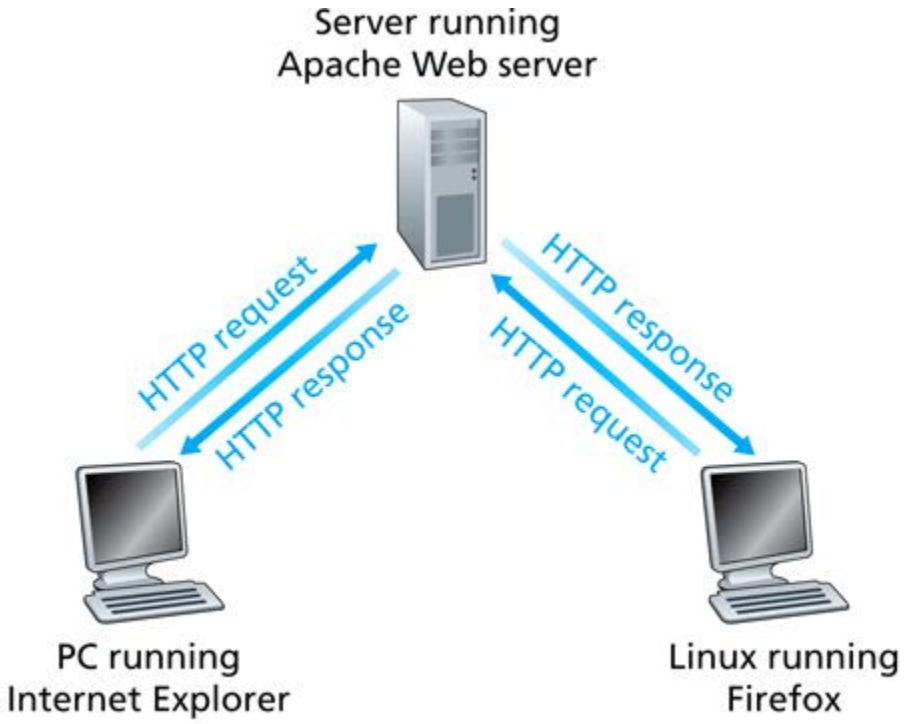
Connected Device: Web Page Load

- Initial content often from same web server
- Some content from other servers
  - Ads
  - Analytics
- Tens to hundreds of loads triggered by one web page



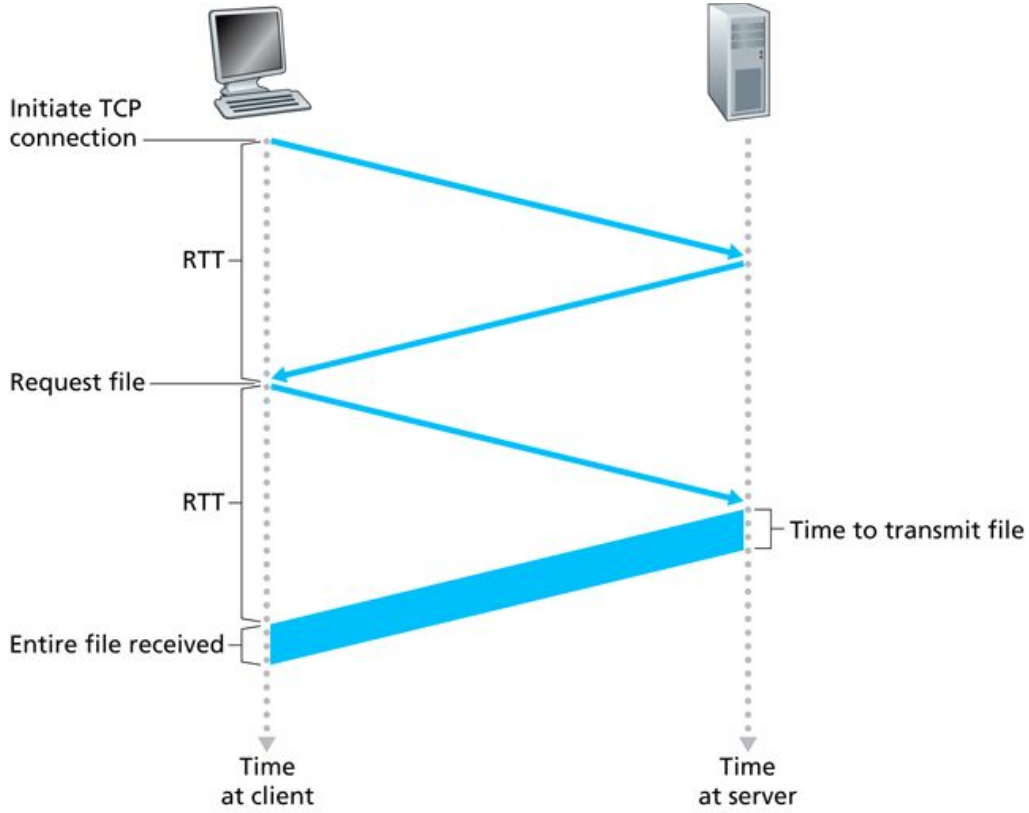
Hypertext Transfer Protocol

- Web browser and server use protocol to exchange web documents
  - Hypertext Transfer Protocol (HTTP)
- HTTP uses plain text for protocol exchange
  - Easy for humans to understand
- Remember: protocol
  - Definition of message format
    - Types of messages
    - Syntax of messages (fields and delineation)
    - Semantics of fields
  - Definition of message exchange
    - When and how to send messages
    - When and how to respond



Hypertext Transfer Protocol

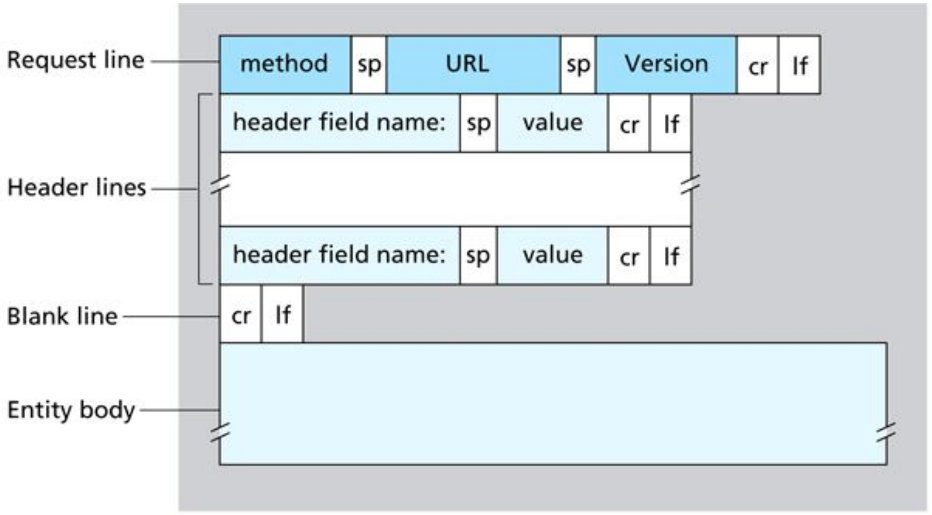
- Sequence of messages
  1. Establish connection (transport layer and below)
  2. Send request for document
  3. Receive document
  4. Repeat 2.-3. or close connection
- Format of messages
  - Plain text



Hypertext Transfer Protocol

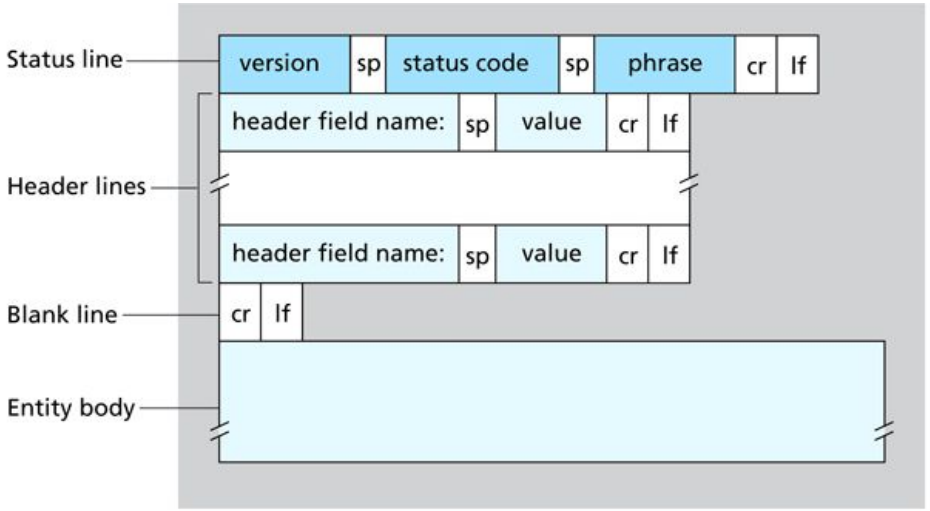
- **Example:**

Request




```
GET / HTTP/1.1
Host: engineering.umass.edu
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10_12_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

Response

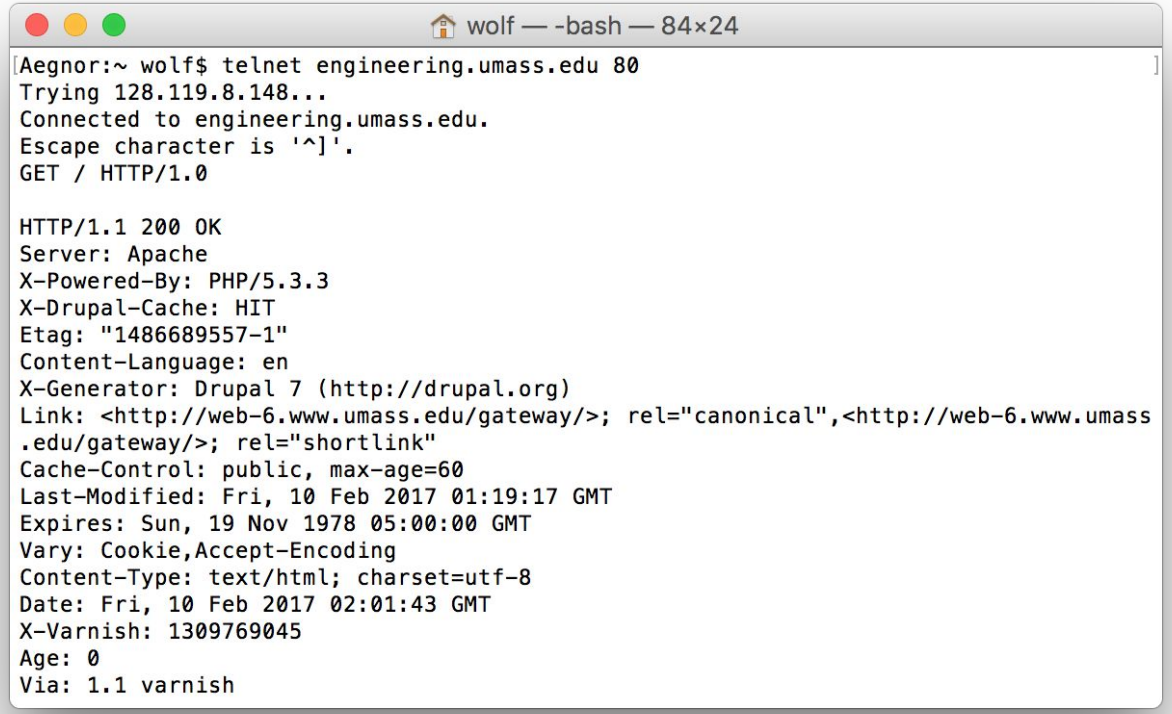



```
HTTP/1.1 200 OK
Server: Apache
Content-Language: en
Cache-Control: public, max-age=180
Last-Modified: Thu, 09 Feb 2017 23:18:58 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Content-Length: 10656
Accept-Ranges: bytes
Date: Fri, 10 Feb 2017 00:17:19 GMT
Connection: keep-alive

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; …
```

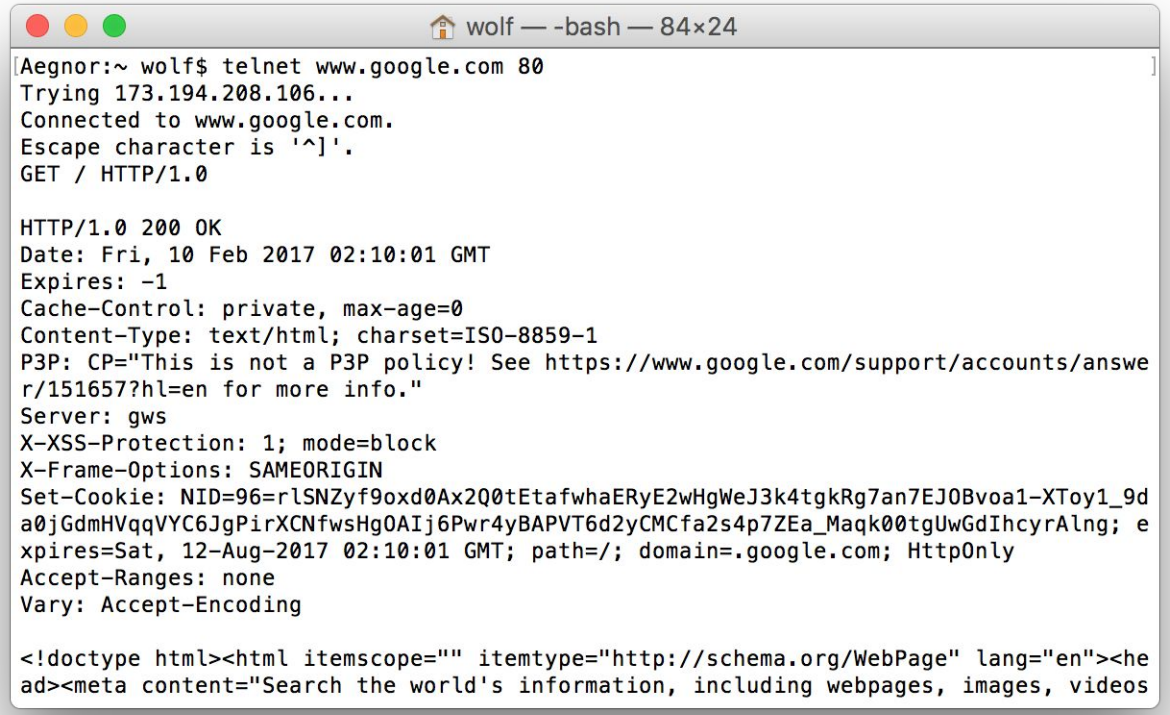
Connected Device: Hypertext Transfer Protocol

- Demo: fetch web page via telnet in terminal
  - Connect to server: “telnet engineering.umass.edu 80”
    - 80 is port number (process identifier) of web servers
  - Request document: “GET / HTTP/1.0” (hit return twice)



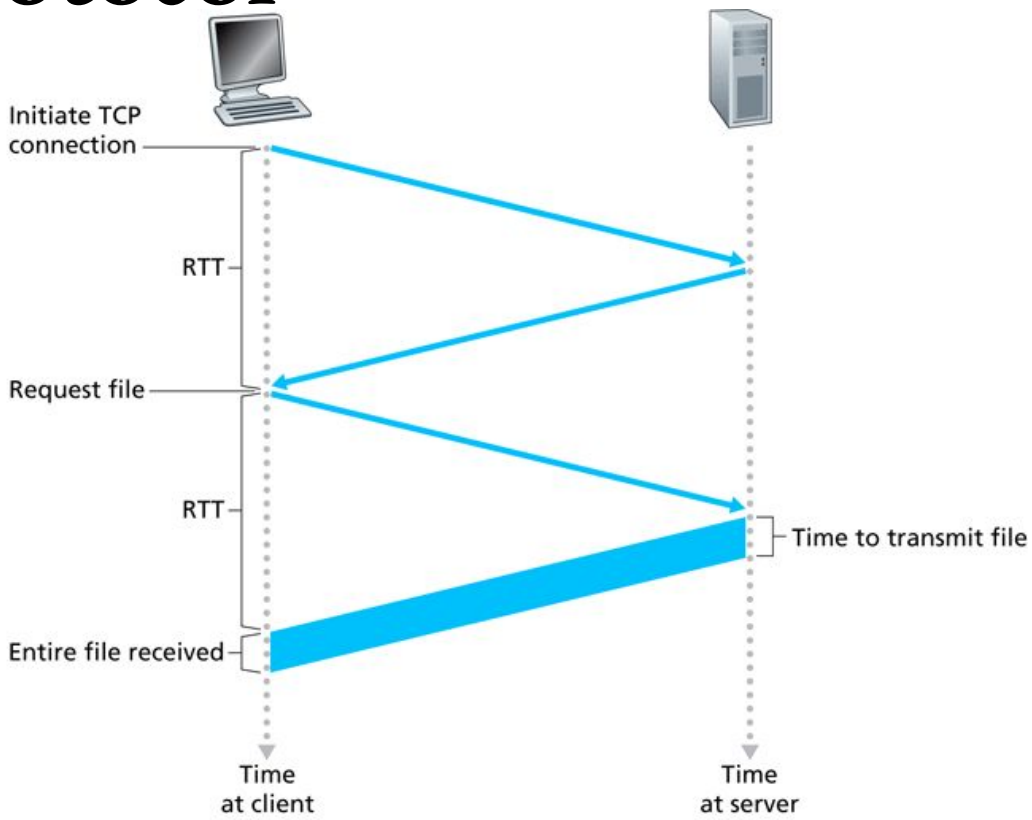
Connected Device: Hypertext Transfer Protocol

- Demo: fetch web page via telnet in terminal
  - This works with many web sites (e.g., www.google.com)
  - In practice, there is a lot of meta-information



Hypertext Transfer Protocol

- What happens in network when requesting document?
  - Transport layer establishes connection
    - Requires one round-trip time (RTT)
  - HTTP requests document
    - Requires one RTT plus transmission time

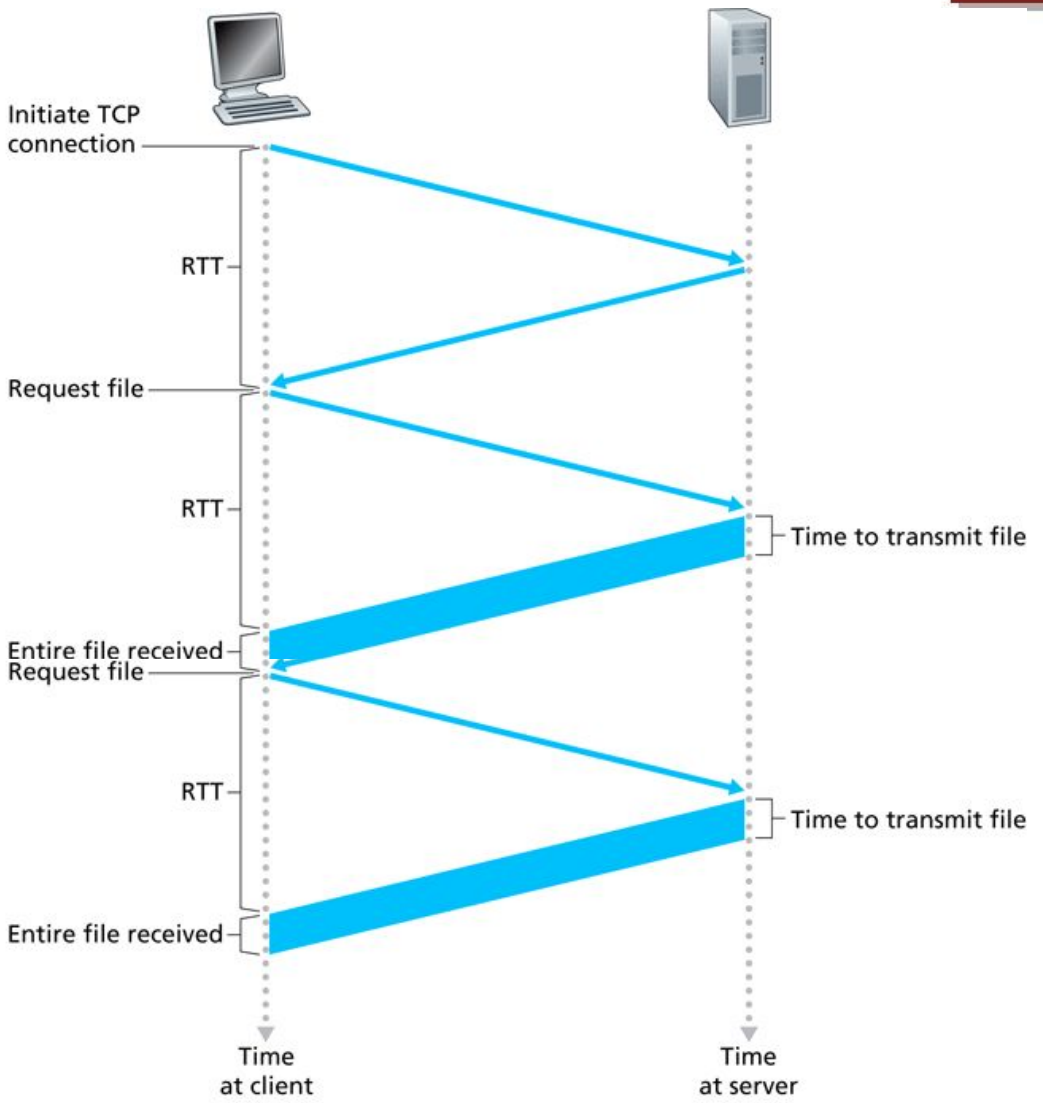


Orchestrated Discussion (Short Answer): HTTP Performance

- What are some HTTP performance bottleneck issues for a website with many embedded images?
- How could download speed be improved?

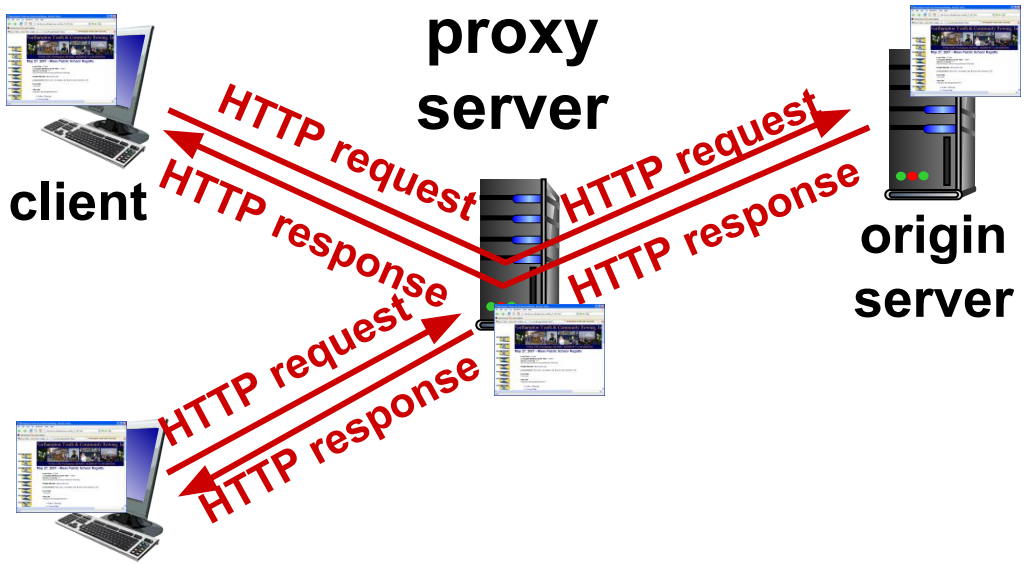
Persistent Connections

- Persistent HTTP connection reuses connection
  - Multiple files on same connection
  - "Pay" for RTT only once
- Example of how protocol design can affect performance



Content Caching

- Reduce RTT by fetching content from “closer” server
- Caching proxy
  - Keeps copy of fetched web documents
  - Serves content when other local users request document
- Caching works well for frequently requested content



Orchestrated Discussion (Short Answer): User State in HTTP

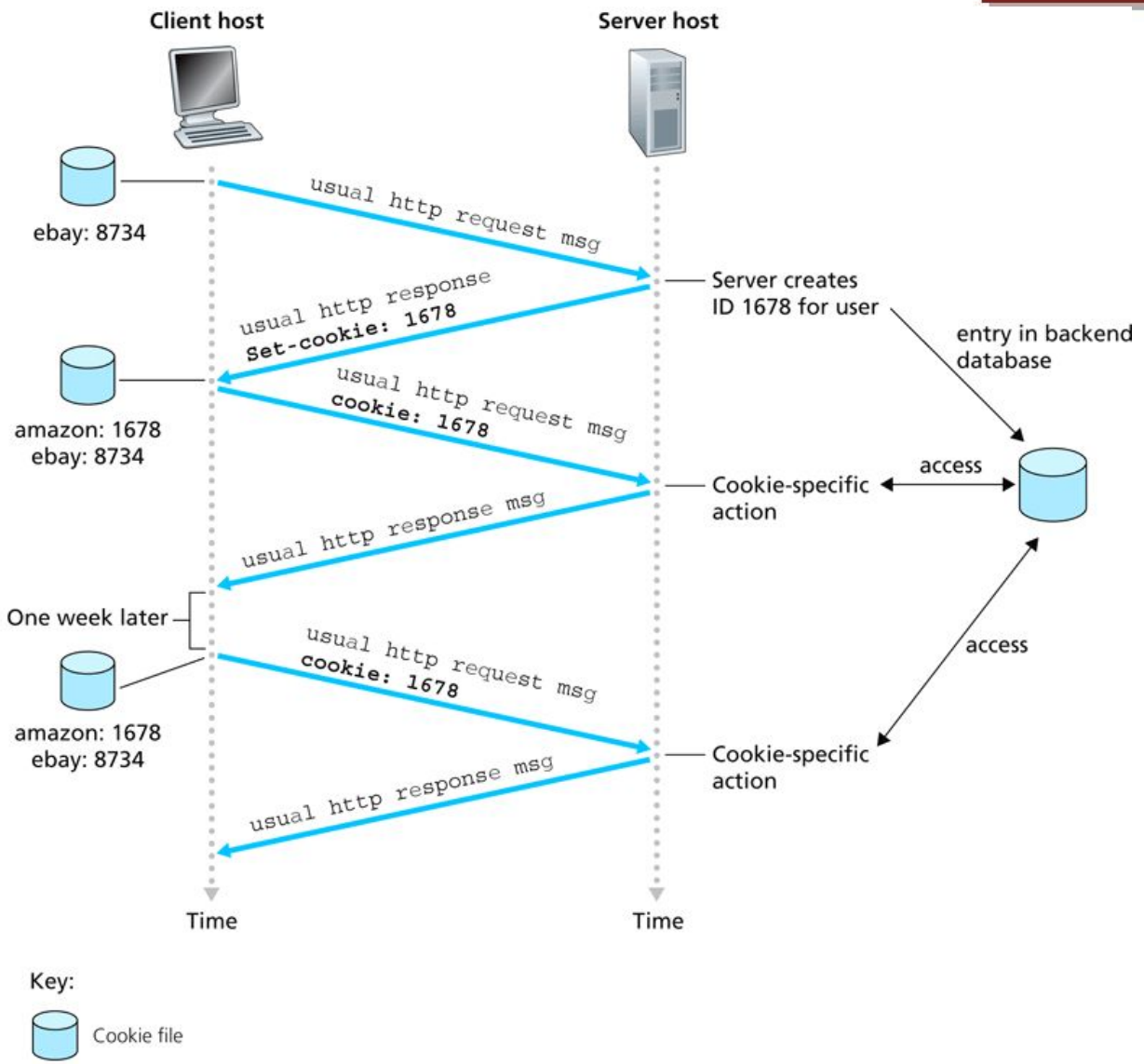
- HTTP is a “stateless” protocol
  - Any request is independent from any other
  - In practice, state is necessary for user experience
- How can a shopping site remember what I have in my shopping cart?

User State in HTTP

- HTTP is a “stateless” protocol
  - Any request is independent from any other
  - In practice, state is necessary for user experience
- How can a shopping site remember what I have in my shopping cart?
  - Identify browser / session with server-provided id
  - Associate shopping cart with session

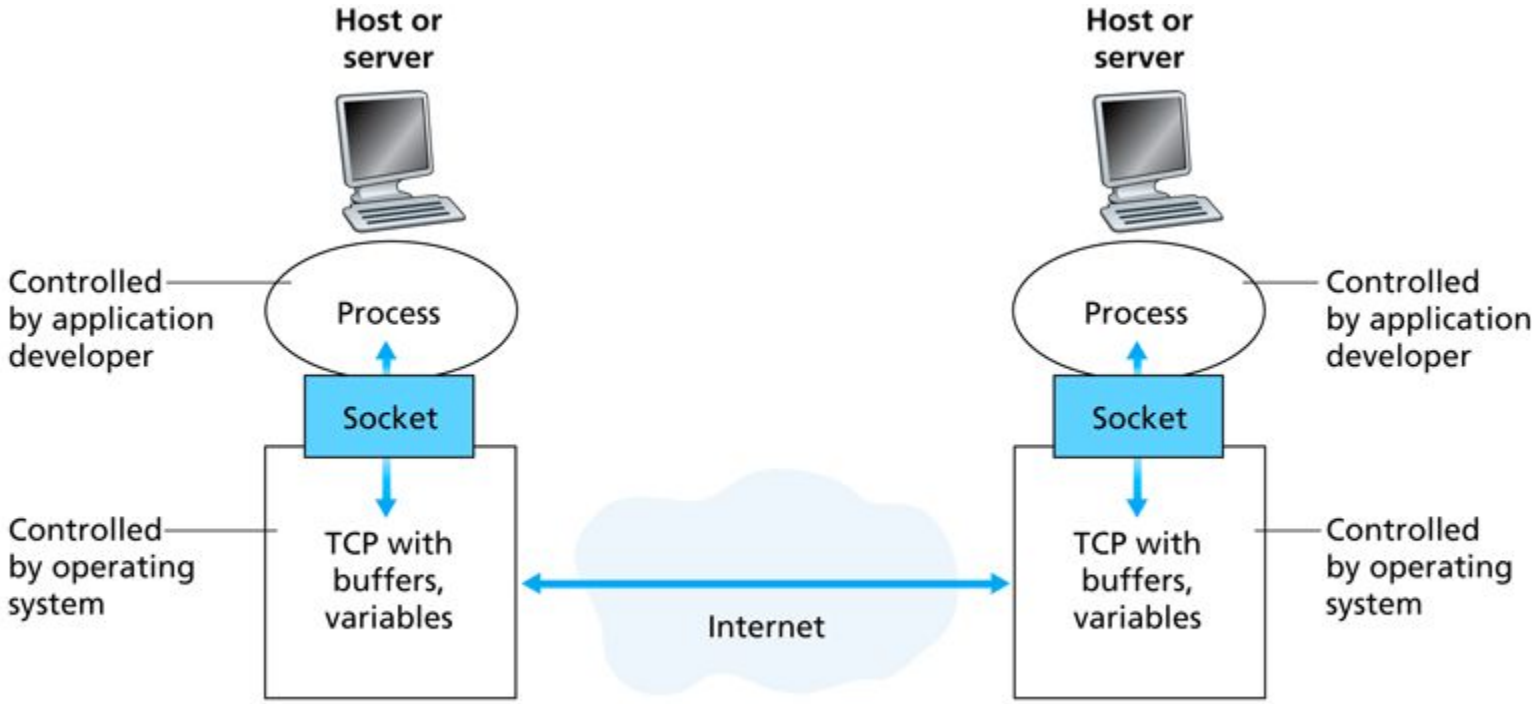
Connected Device: User State in HTTP

- Cookies in HTTP
  - Identifier that is sent with every request



Application Layer

- Connection provided layer below (transport layer):
  - UNIX socket (=“bit pipe”)
  - “telnet” is one protocol that provides this functionality



Transport Layer Service

- 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, security
  - 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
- Why bother with UDP at all?

Application Requirements

- Not all applications have same needs
  - No loss / loss tolerant, elastic bandwidth / fixed, time-sensitive / not time sensitive

Application	Data Loss	Bandwidth	Time-Sensitive
File transfer			
E-mail			
Web documents			
Internet telephony/ Video conferencing			
Stored audio/video			
Interactive games			
Instant messaging			

Whiteboard: Application Requirements

Fill in table requirements: reliability, bandwidth, time-sensitivity

Application	Data Loss	Bandwidth	Time-Sensitive
File transfer			
E-mail			
Web documents			
Internet telephony/ Video conferencing			
Stored audio/video			
Interactive games			
Instant messaging			

Application Requirements

Application	Data Loss	Bandwidth	Time-Sensitive
File transfer	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
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

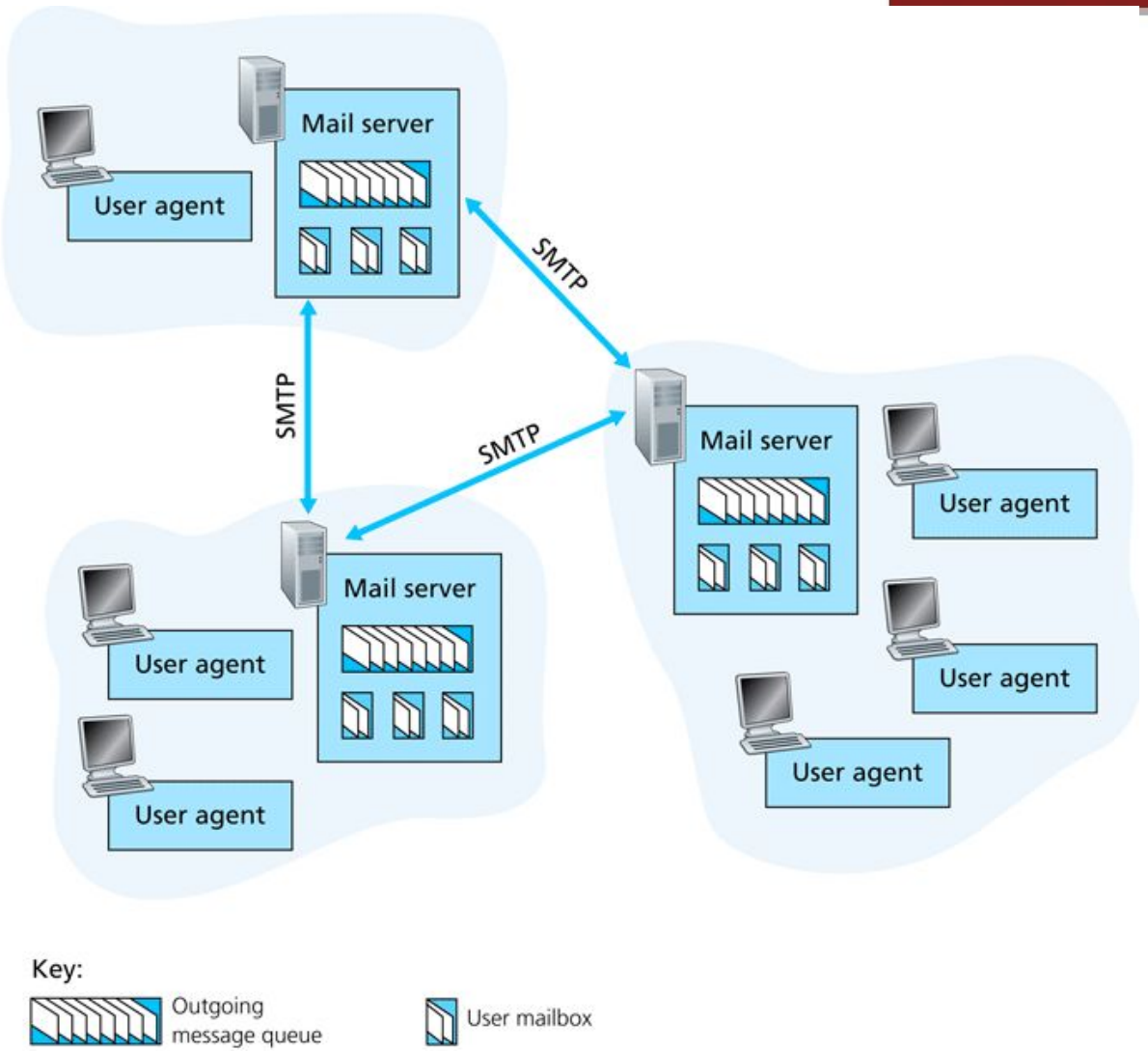
Application Protocol Examples

- Many different application layer protocols
  - Use different transport layer services
  - Follows ”hourglass architecture”

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic 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	TCP or UDP
Internet telephony	SIP, RTP, or proprietary (e.g., Skype)	Typically UDP

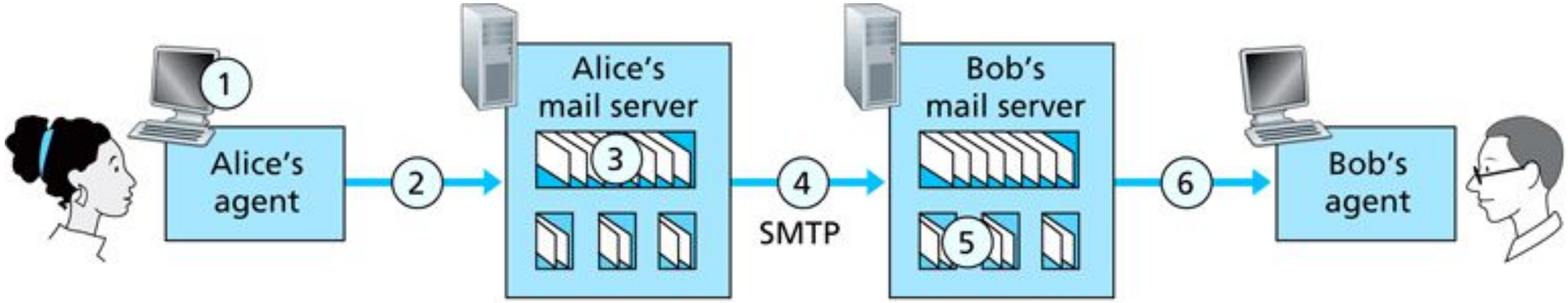
Electronic Mail

- Mail servers
  - Transfer mail
  - Store mail in
  - mailboxes
- User agents
  - Access mail from server
  - Transmit new mail
- Protocols
  - Simple Mail Transfer Protocol (SMTP)
  - Post Office Protocol (POP)
  - Internet Mail Access Protocol (IMAP)
  - Web-based email access (HTTP)



Electronic Mail

- Sending of email:



- SMTP "pushes" email to destination mail server
- POP/IMAP/ HTTP "pulls" email from mail server
- SMTP messages are in cleartext

Securing TCP

- TCP & UDP
  - No encryption
  - Cleartext passwords sent into socket traverse Internet in cleartext
- Secure Socket Layer (SSL)
  - provides encrypted TCP connection
  - data integrity
  - end-point authentication
- SSL is at app layer
  - Apps use SSL libraries that “talk” to TCP
- SSL socket API
  - Cleartext passwords sent into socket traverse Internet encrypted
  - More later in course

Assignment: Content Distribution Networks Presentation

- Case Study: Content Distribution Networks (CDNs)
  - Work in 3-person teams
- Prepare 5-minute presentation
  - Why do we need (CDNs)
  - How do CDNs work
  - Where do we see CDNs in practice
- One team will be selected at random to present at beginning of next lecture

Group Discussion and Report Back (Short Answer): CDN Presentation

- Discuss the topic of this assignment with your group and begin preparing a short presentation. This assignment will be continued outside the classroom. Report any questions your group may have to the Instructor.

Summary of Lesson

- HTML for web documents
- HTTP as application layer protocol example
- HTTP improvements (persistent connections)
- Other application-layer protocols and their performance requirements
- Content Distribution Networks Presentation

Post-work for Lesson 3

Homework #2

- After the Live Lecture, you will complete and submit a homework assignment. Go to the online classroom to view and submit the assignment.

Presentation (Content Distribution Networks)

- After the Live Lecture, you will complete and submit a presentation file. Go to the online classroom to view and submit the presentation file.

To Prepare for the Next Lesson

- Complete and submit the Post-work for Lesson 3.
- Read the Required Readings for Lesson 4.
- Complete the Pre-work for Lesson 4.

Go to the online classroom for details.