

The Application Layer: The Web and HTTP

Smith College, CSC 249
February 5, 2008

slides mostly from J.F Kurose and K.W. Ross, copyright
1996-2005

Chapter 2: Application layer

OVERVIEW of Chapter

- Basic definition of network applications
- Applications we will investigate
 - ❖ The Web & HTTP
 - ❖ FTP - file transfer protocol
 - ❖ Electronic Mail
 - SMTP, POP3, IMAP
 - ❖ DNS - domain name system (translate names into IP addresses)
 - ❖ P2P - peer-to-peer
- Socket programming

2

Chapter 2: Application Layer

Our goals:

- Understand the conceptual aspects of network applications and application protocols
 - ❖ client-server architecture
 - ❖ Peer-to-peer architecture
 - ❖ transport-layer "service models"

3

Some network apps

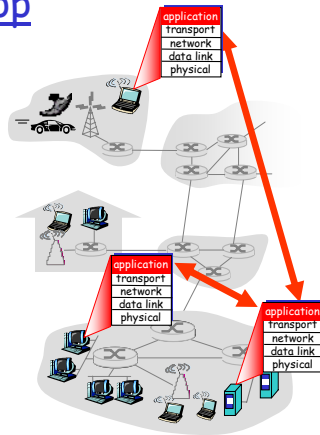
- E-mail
- Web
- Instant messaging
- Remote login
- P2P file sharing
- Multi-user network games
- Streaming stored video clips
- Internet telephone
- Real-time video conference
- ...
- ...

4

Creating a network app

Write programs that

- ❖ run on different end systems and
- ❖ communicate over a network.
- ❖ e.g., A web browser communicates with web server software



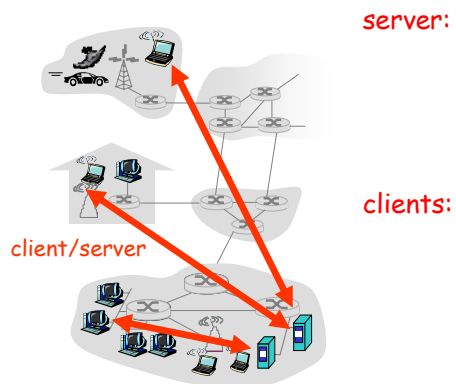
5

Three major application architectures

- ❑ Client-server
- ❑ Peer-to-peer (P2P)
- ❑ Hybrid of client-server and P2P

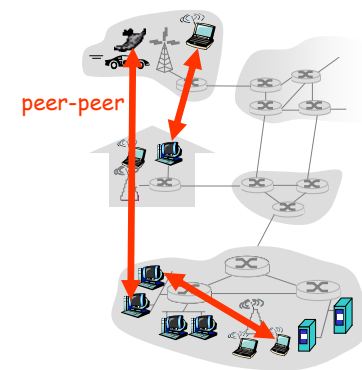
6

Client-server architecture



7

Pure P2P architecture



8

Pros & Cons of client-server v. P2P

- ❑ What are the benefits and drawbacks of using the client-server architecture versus peer-to-peer?
- ❑ Why might you want to develop an application with a hybrid architecture?

9

Hybrid of client-server and P2P

Skype

- ❖ voice-over-IP P2P application
- ❖ centralized server: finding address of remote party:
- ❖ client-client connection: direct (not through server)

Instant messaging

- ❖ chatting between two users is P2P
- ❖ centralized service: client presence detection/location
 - user registers its IP address with central server when it comes online
 - user contacts central server to find IP addresses of buddies

10

Application layer protocols define

- ❑ **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

11

- ❑ The goal in writing an application is to have a (client) program on one host communicate with a (server) program on a second host
- ❑ When an application program is running, it is called a 'process'
- ❑ So, how do these two programs, or processes, communicate?

12

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using the operating system
- processes in different hosts communicate by exchanging **messages**
 - ❖ This course examines how applications exchanged messages over a network

Client process: process that initiates communication

Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

13

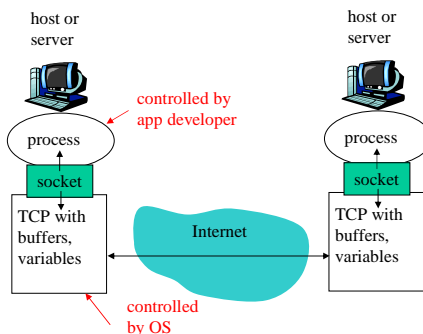
Addressing processes

- To receive messages, a process must have an *identifier*
- Each host device has a unique 32-bit IP address
- **Q:** What is needed, in addition to the IP address of a host on which process runs, to identify a process?

14

Sockets

- A process sends/receives messages to/from its **socket**
- socket analogous to door
 - ❖ The sending process pushes the message out the door
 - ❖ The sending process relies on transport to bring the message to the socket at receiving process
- Socket = "API:" (1) choice of transport protocol - TCP v. UDP; (2) ability to fix a few parameters (*more later*)



15

What transport service does an application need?

Reliability: Data loss

- some applications (e.g., audio) can tolerate some loss
- other applications (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- some applications (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth/Throughput

- some applications (e.g., multimedia) require minimum amount of bandwidth to be "effective"
- other applications ("elastic apps") make use of whatever bandwidth they get

Security

- TCP enhanced with SSL (secure sockets layer)

16

Chapter 2: Application layer

- 2.1 Principles of network applications
 - ❖ app architectures
 - ❖ app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
 - ❖ SMTP, POP3, IMAP
- 2.5 DNS

17

Web and HTTP

First some jargon

- Web page consists of objects
 - ❖ A base HTML-file which includes
 - ❖ Several referenced objects
- Object can be HTML file, JPEG image, Java applet, audio file,...
- Each object is addressable by a URL
- Example URL:

www.someschool.edu/someDept/pic.gif

host name of server with
requested object

the object's
path name

18

Web and HTTP

- What are the elements of a web application?

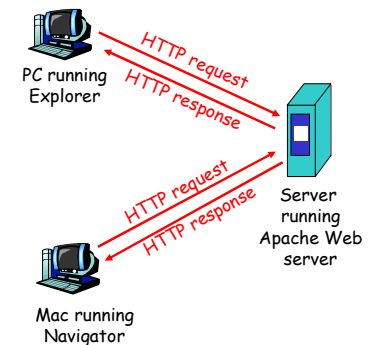
- How do they communicate with each other?

19

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
 - ❖ *client*: browser that requests, receives, "displays" Web objects
 - ❖ *server*: Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



20

HTTP overview (continued)

HTTP Uses TCP:

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

HTTP is "stateless"

- server maintains no information about past client requests

21

HTTP connections

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

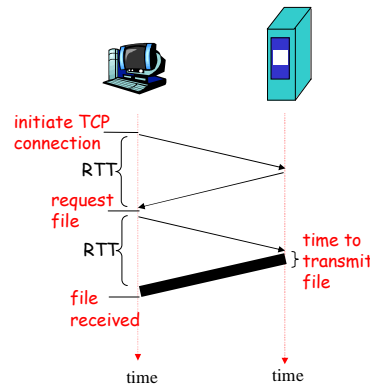
22

Non-Persistent HTTP: Response time

Definition of RTT: time to send a small packet to travel from client to server and back.

Response time:

- one RTT to initiate TCP connection
 - one RTT for HTTP request and first few bytes of HTTP response to return
 - file transmission time
- total = 2RTT+transmit time**



23

Persistent HTTP

Nonpersistent HTTP issues:

- OS overhead for *each* TCP connection
- browsers often open **parallel TCP connections** to fetch referenced objects

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection

Persistent *without* pipelining:

- client issues new request only when previous response has been received

Persistent *with* pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object

24

HTTP messages

□ There are two types of HTTP messages

- ❖ *request*
- ❖ *response*

25

HTTP request message

- **HTTP request message:**
 - ❖ ASCII (human-readable format)

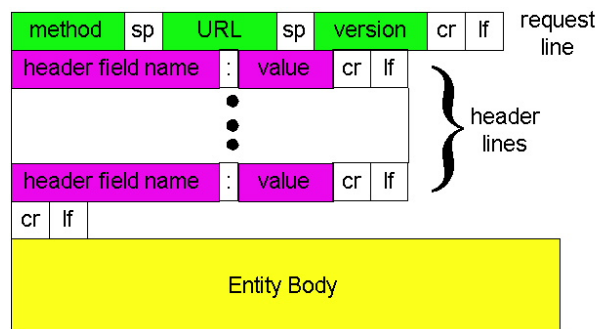
request line
(GET, POST, HEAD commands) → GET /somedir/page.html HTTP/1.1

header lines → Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr

Carriage return line feed indicates end of message → (extra carriage return, line feed)

26

HTTP request message: general format



27

Uploading form input

the basic GET method:

- The 'entity body' in the request is empty

Post method:

- A web page often includes "form" input
- This input information is uploaded to server in the message's "entity body"

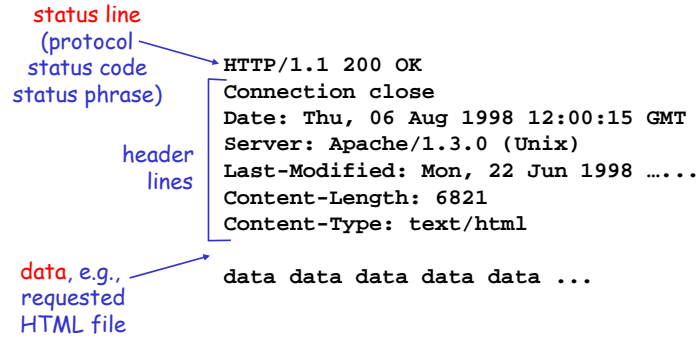
URL method:

- Uses the GET method
- Input is uploaded in URL field of request line:

www.somesite.com/animalsearch?monkeys&banana

28

HTTP response message



Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet cis.poly.edu 80
```

Opens TCP connection to port 80 (default HTTP server port) at cis.poly.edu. Anything typed in sent to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

End with 2 CR. This is a minimal (but complete) GET request to an HTTP server

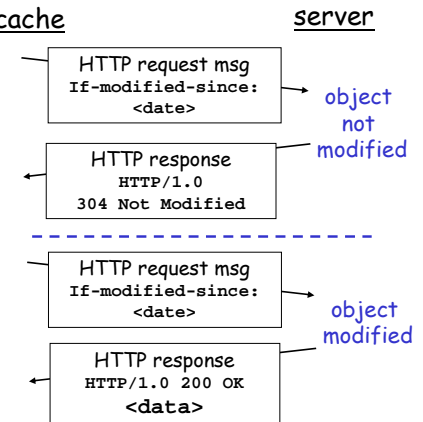
Trying out HTTP (client side) for yourself

3. Look at response message sent by HTTP server

```
HTTP/1.1 200 OK
Date: Fri, 10 Feb 2006 19:15:00 GMT
Server: Apache/1.2.5
Last-Modified: Sat, 28 Jan 2006 12:24:14 GMT
ETag: "15807-225b-43db626e"
Content-Length: 8795
Accept-Ranges: bytes
Content-Type: text/html
```

Conditional GET: HW Problem...

- Goal: do not send the object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
If-modified-since: <date>
- server: response contains no object if cached copy is up-to-date:
HTTP/1.0 304 Not Modified



Examining HTTP in action

- telnet example - page 105 and HW
- Ethereal example - 2nd lab, to come...

33

User-server state: cookies

Many major Web sites use cookies

Four components:

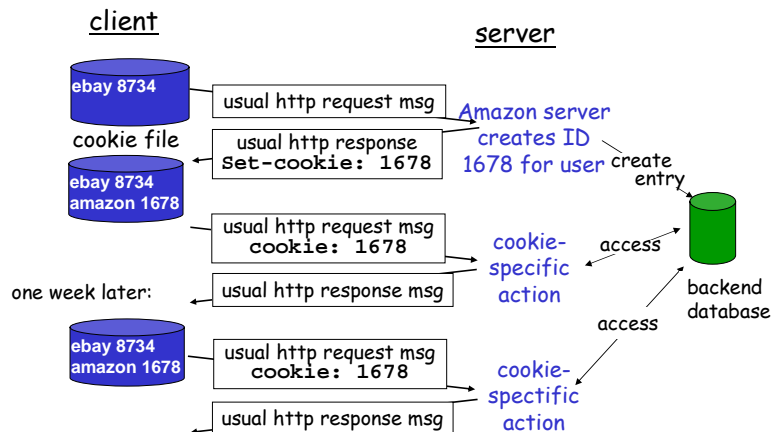
- 1) cookie header line of HTTP *response* message
- 2) cookie header line in HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

Example:

- Susan always access Internet always 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

34

Cookies: keeping "state" (cont.)



35

Cookies (continued)

What cookies can bring:

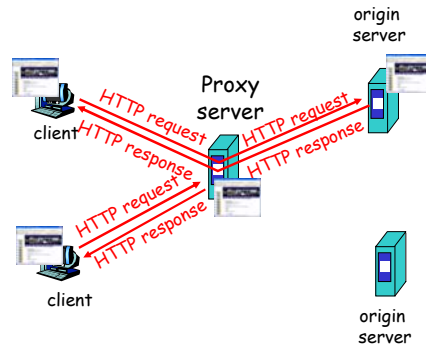
Cookies and privacy:

36

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
 - ❖ object in cache: cache returns object
 - ❖ else cache requests object from origin server, then returns object to client



37

More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link.
- Internet dense with caches: enables "poor" content providers to effectively deliver content (but so does P2P file sharing)

38

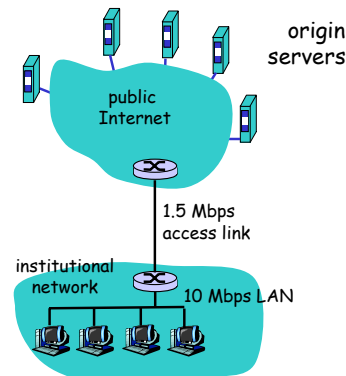
Caching example

Assumptions

- avg object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

Consequences

- utilization on LAN = 15%
 $(15/s)(10^5b/s)/10Mb/s = 15\%$
- utilization on access link = 100%
 $(15/s)(10^5b/s)/1.5Mb/s = 1 = 100\%$
- total delay = Internet delay + access delay + LAN delay
 = 2 sec + minutes + milliseconds



traffic intensity = $(\#req/s)(\#b/req) / \text{bandwidth}$
 As intensity $\rightarrow 1$, Delay $\rightarrow \infty$

39

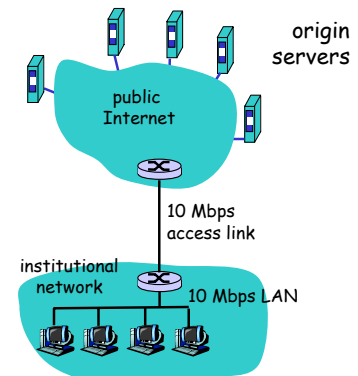
Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay
 = 2 sec + msec + msec
- often a costly upgrade



40

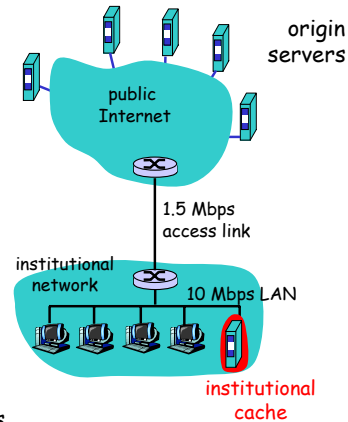
Caching example (cont)

possible solution: install cache

- suppose hit rate is 0.4 - i.e., % of requests served by cache

consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay =
 $\text{Internet delay} + \text{access delay} + \text{LAN delay} = 0.6 \cdot (2.01) \text{ secs} + 0.4 \cdot \text{milliseconds} < 1.4 \text{ secs}$



41

Chapter 2: Summary

our study of network applications:

- application architectures
 - client-server
 - P2P
 - hybrid
- application service requirements:
 - reliability, bandwidth, delay
- Internet transport service model
 - connection-oriented, reliable: TCP
 - unreliable, datagrams: UDP
- specific protocols:
 - HTTP
 - FTP
 - SMTP, POP, IMAP
 - DNS
 - P2P: BitTorrent, Skype
- socket programming

42

Chapter 2: Summary

Most importantly: learning about protocols

- typical request/reply message exchange:
 - client requests info or service
 - server responds with data, status code
- message formats:
 - headers: fields giving info about data
 - data: info being communicated

Important themes:

- control vs. data msgs
 - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable msg transfer
- "complexity at network edge"

43