Computer Communication Networks

Application

ICEN/ICSI 416 – Fall 2016

Prof. Dola Saha

UNIVERSITY AT ALBANY
State University of New York

Where to find in book?

➢ Materials covered in this section are in Chapter 9 & 7 of "Computer Networks: A Systems Approach", Larry Peterson and Bruce Davie, Elsevier

Problem

➢ Applications need their own protocols.

➢ These applications are part network protocol (in the sense that they exchange messages with their peers on other machines) and part traditional application program (in the sense that they interact with the windowing system, the file system, and ultimately, the user).

➢ We will explore some of the most popular network applications available today.

Outline

➢ Traditional Applications

➢ Multimedia Applications

➢ Infrastructure Services

➢ Overlay Networks

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

➢ network-core devices do not run user applications

➢ applications on end systems allows for rapid app development, propagation
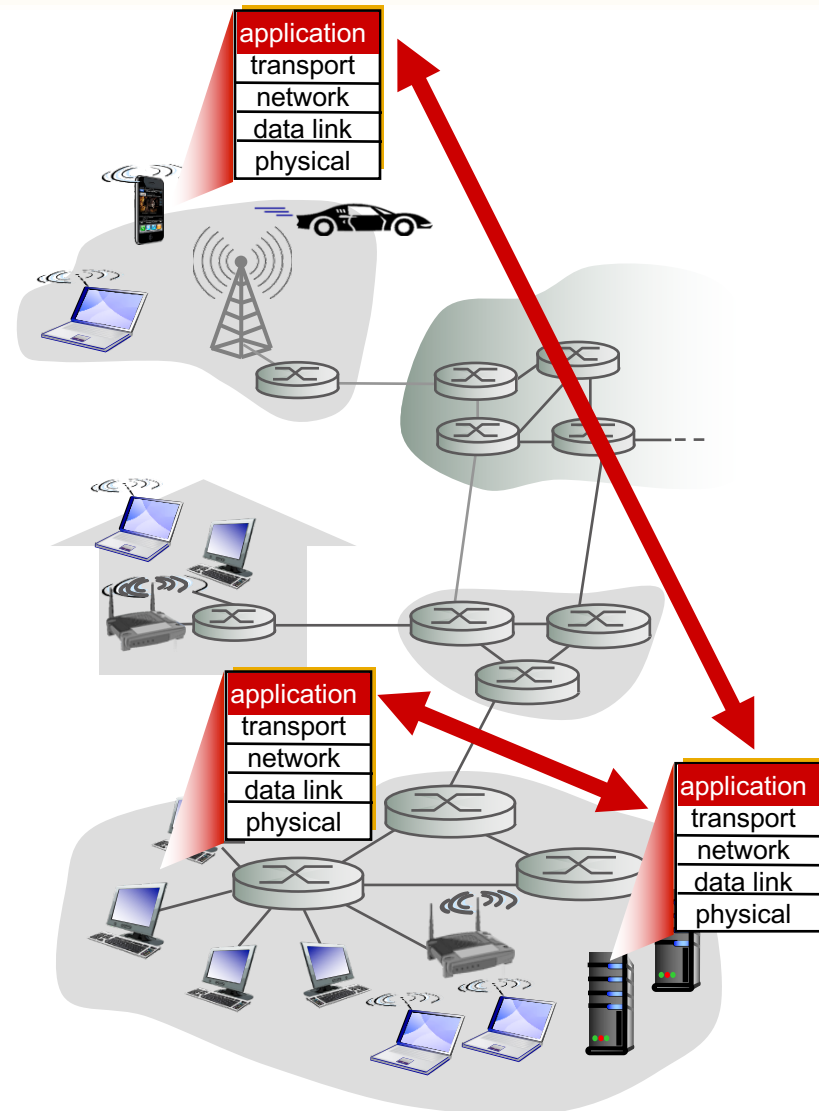
Traditional Applications

➢ Two of the most popular–

  ▪ The World Wide Web and

  ▪ Email.

➢ Broadly speaking, both of these applications use the request/reply paradigm–users send requests to servers, which then respond accordingly.

UNIVERSITY AT ALBANY
State University of New York

Traditional Applications

➢ It is important to distinguish between *application programs* and *application protocols.*

➢ For example, *the HyperText Transport Protocol* (HTTP) is an application protocol that is used to retrieve Web pages from remote servers.

➢ There can be many different application programs—that is, Web clients like Internet Explorer, Chrome, Firefox, and Safari—that provide users with a different look and feel, but all of them use the same HTTP protocol to communicate with Web servers over the Internet.

Application

application	application program	application layer protocol	underlying transport protocol
e-mail	Outlook	SMTP [RFC 2821]	TCP
remote terminal access	Telnet	Telnet [RFC 854]	TCP
Web	Firefox	HTTP [RFC 2616]	TCP
file transfer	FileZilla	FTP [RFC 959]	TCP
streaming multimedia	YouTube Cisco WebEx	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	Google voice Vonage	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Traditional Applications

➢ Two very widely-used, standardized application protocols:

- HTTP: HyperText Transport Protocol is used to communicate between Web browsers and Web servers.
  - RFC 2616 - https://www.ietf.org/rfc/rfc2616.txt

- SMTP: Simple Mail Transfer Protocol is used to exchange electronic mail.
  - RFC 821 - https://tools.ietf.org/rfc/rfc821.txt

Traditional Applications

➢ World Wide Web

- The World Wide Web has been so successful and has made the Internet accessible to so many people that sometimes it seems to be synonymous with the Internet.

- In fact, the design of the system that became the Web started around 1989, long after the Internet had become a widely deployed system.

- The original goal of the Web was to find a way to organize and retrieve information, drawing on ideas about hypertext—interlinked documents—that had been around since at least the 1960s.

Traditional Applications

➢ World Wide Web

- The core idea of hypertext is that one document can link to another document, and the protocol (HTTP) and document language (HTML) were designed to meet that goal.

- One helpful way to think of the Web is as a set of cooperating clients and servers, all of whom speak the same language: HTTP.

- Most people are exposed to the Web through a graphical client program, or Web browser, like Safari, Chrome, Firefox or Internet Explorer.

Traditional Applications

➢ World Wide Web

- Clearly, if you want to organize information into a system of linked documents or objects, you need to be able to retrieve one document to get started.

- Hence, any Web browser has a function that allows the user to obtain an object by "opening a URL."

- URLs (Uniform Resource Locators) are so familiar to most of us by now that it's easy to forget that they haven't been around forever.

- They provide information that allows objects on the Web to be located, and they look like the following:

  o http://www.cs.princeton.edu/index.html

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

   host name ‿‿‿‿‿‿‿‿‿‿‿‿ path name

Traditional Applications

➤ World Wide Web

- If you opened that particular URL, your Web browser would open a TCP connection to the Web server at a machine called www.cs.princeton.edu and immediately retrieve and display the file called index.html.

- Most files on the Web contain images and text and many have other objects such as audio and video clips, pieces of code, etc.

- They also frequently include URLs that point to other files that may be located on other machines, which is the core of the "hypertext" part of HTTP and HTML.

UNIVERSITY AT ALBANY
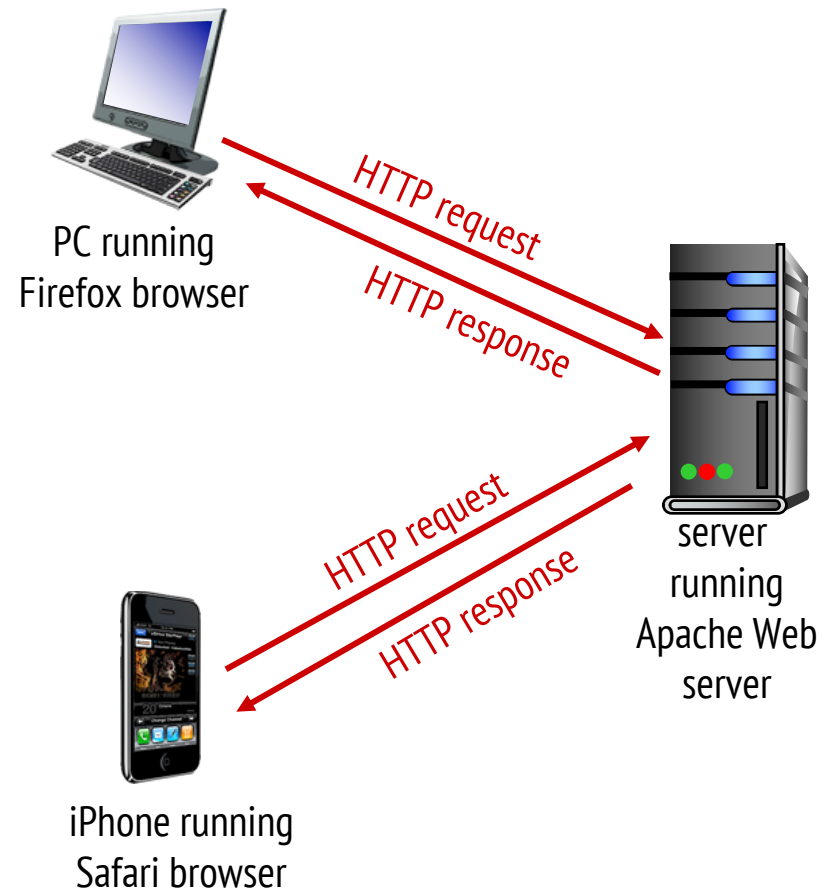State University of New York

Domain Name Space

➢ DNS is hierarchical

➢ Assigned based on affiliation of institution
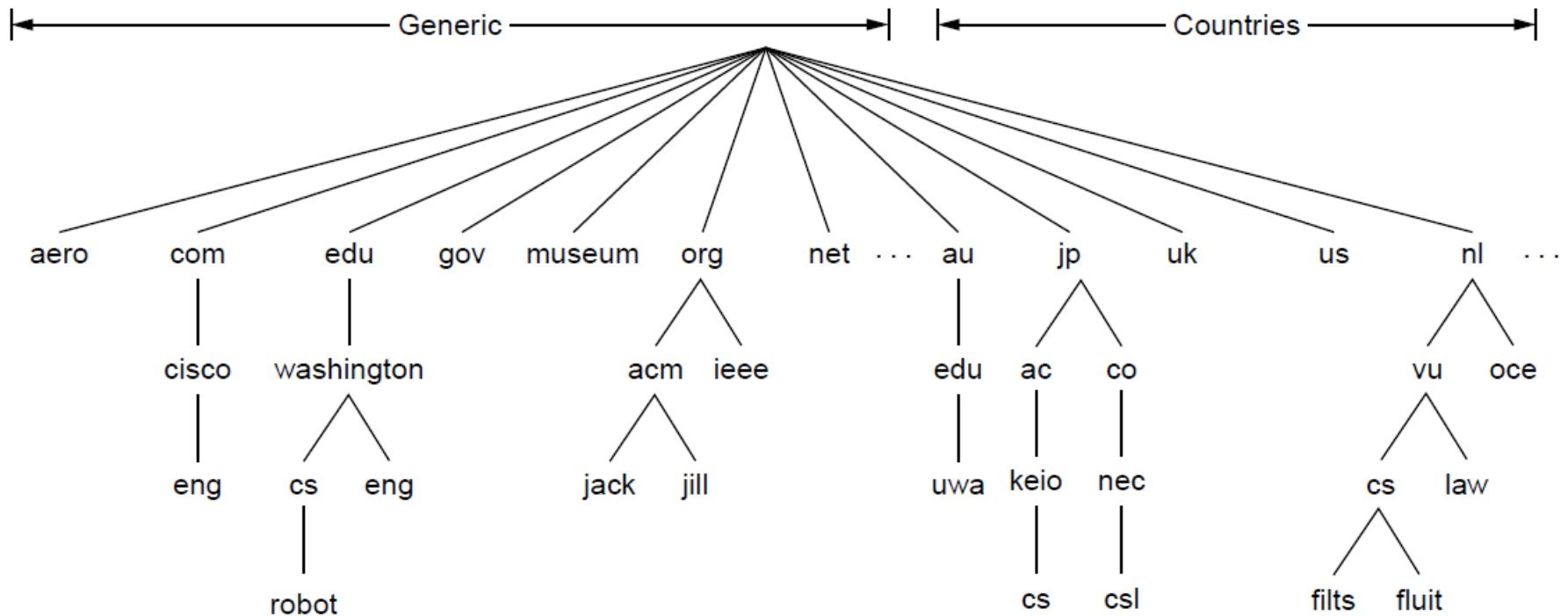
HTTP overview

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

HTTP request

HTTP response

iPhone running
Safari browser

server
running
Apache Web
server

Traditional Applications

➢ World Wide Web

- When you ask your browser to view a page, your browser (the client) fetches the page from the server using HTTP running over TCP.

- HTTP is a text oriented protocol.

- At its core, HTTP is a request/response protocol, where every message has the general form

    START_LINE <CRLF>

    MESSAGE_HEADER <CRLF>

    <CRLF>

    MESSAGE_BODY <CRLF>

- <CRLF> stands for carriage-return-line-feed.

- The first line (START LINE) indicates whether this is a request message or a response message.

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
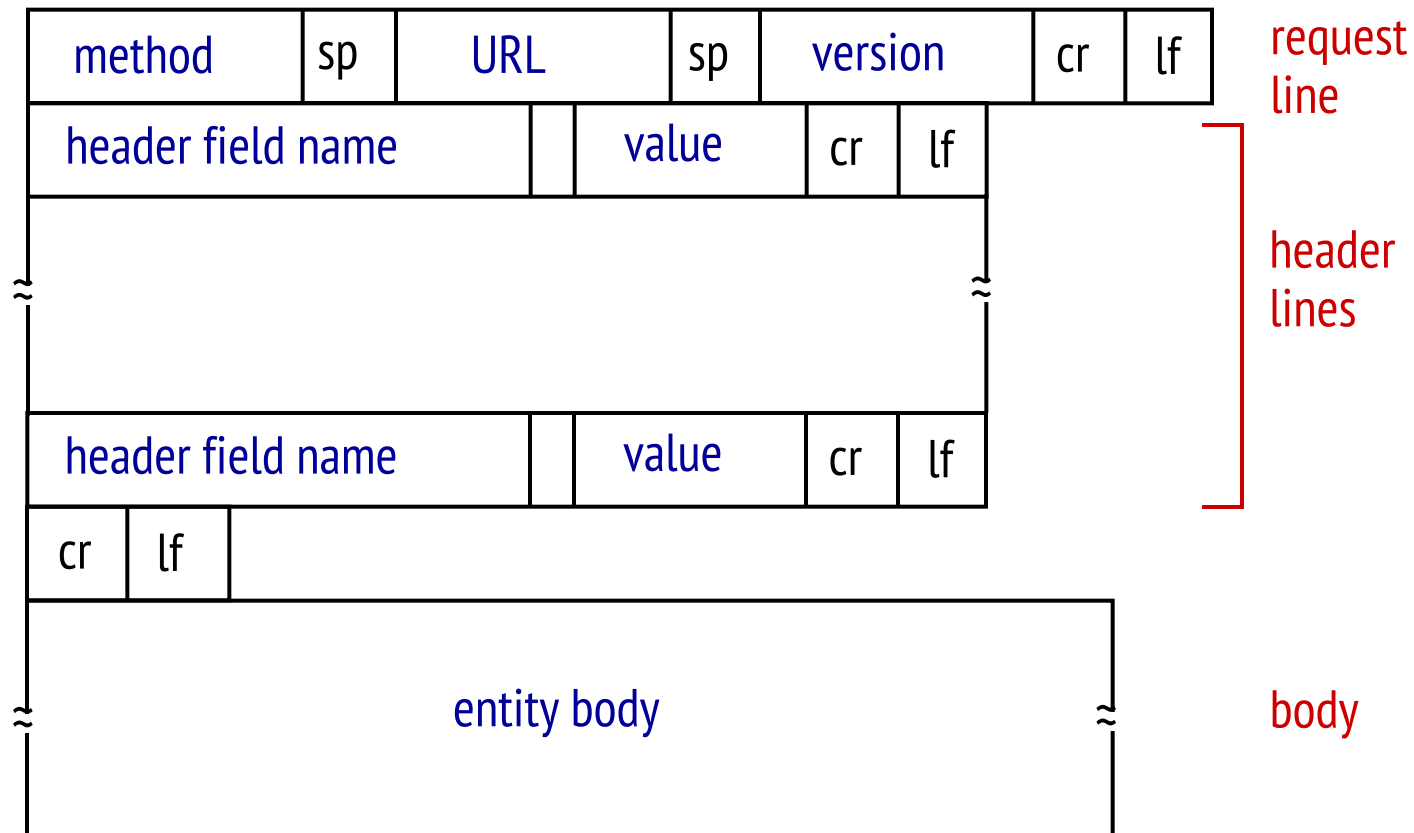
Traditional Applications

- ➢ World Wide Web
  - ▪ Request Messages
    - ○ The first line of an HTTP request message specifies three things:
      - ✓ the operation to be performed,
      - ✓ the Web page the operation should be performed on, and
      - ✓ the version of HTTP being used.

    - ○ Although HTTP defines a wide assortment of possible request operations— including "write" operations that allow a Web page to be posted on a server—the two most common operations are GET (fetch the specified Web page) and HEAD (fetch status information about the specified Web page).

HTTP Request Message

| method | sp | URL | sp | version | cr | lf |

request line

| header field name | | value | cr | lf |

header lines

| header field name | | value | cr | lf |

| cr | lf |

entity body

body

Traditional Applications

➢ World Wide Web

■ Request Messages

Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

HTTP request operations

UNIVERSITY AT ALBANY
State University of New York

Traditional Applications

➢ World Wide Web

- Response Messages
  - Like request messages, response messages begin with a single START LINE.
  - In this case, the line specifies the version of HTTP being used, a three-digit code indicating whether or not the request was successful, and a text string giving the reason for the response.

HTTP Result Codes

- Response Messages

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

Five types of HTTP result codes

Uniform Resource Identifiers

➤ The URLs that HTTP uses as addresses are one type of *Uniform Resource Identifier (URI)*.

➤ A URI is a character string that identifies a resource, where a resource can be anything that has identity, such as a document, an image, or a service.

➤ The format of URIs allows various more-specialized kinds of resource identifiers to be incorporated into the URI space of identifiers.

➤ The first part of a URI is a *scheme* that names a particular way of identifying a certain kind of resource, such as mailto for email addresses or file for file names.

➤ The second part of a URI, separated from the first part by a colon, is the *scheme-specific part*.

UNIVERSITY AT ALBANY
State University of New York

Traditional Applications

➢ World Wide Web

- ▪ TCP Connections
  - ○ The original version of HTTP (1.0) established a separate TCP connection for each data item retrieved from the server.

  - ○ It's not too hard to see how this was a very inefficient mechanism: connection setup and teardown messages had to be exchanged between the client and server even if all the client wanted to do was verify that it had the most recent copy of a page.

  - ○ Thus, retrieving a page that included some text and a dozen icons or other small graphics would result in 13 separate TCP connections being established and closed.
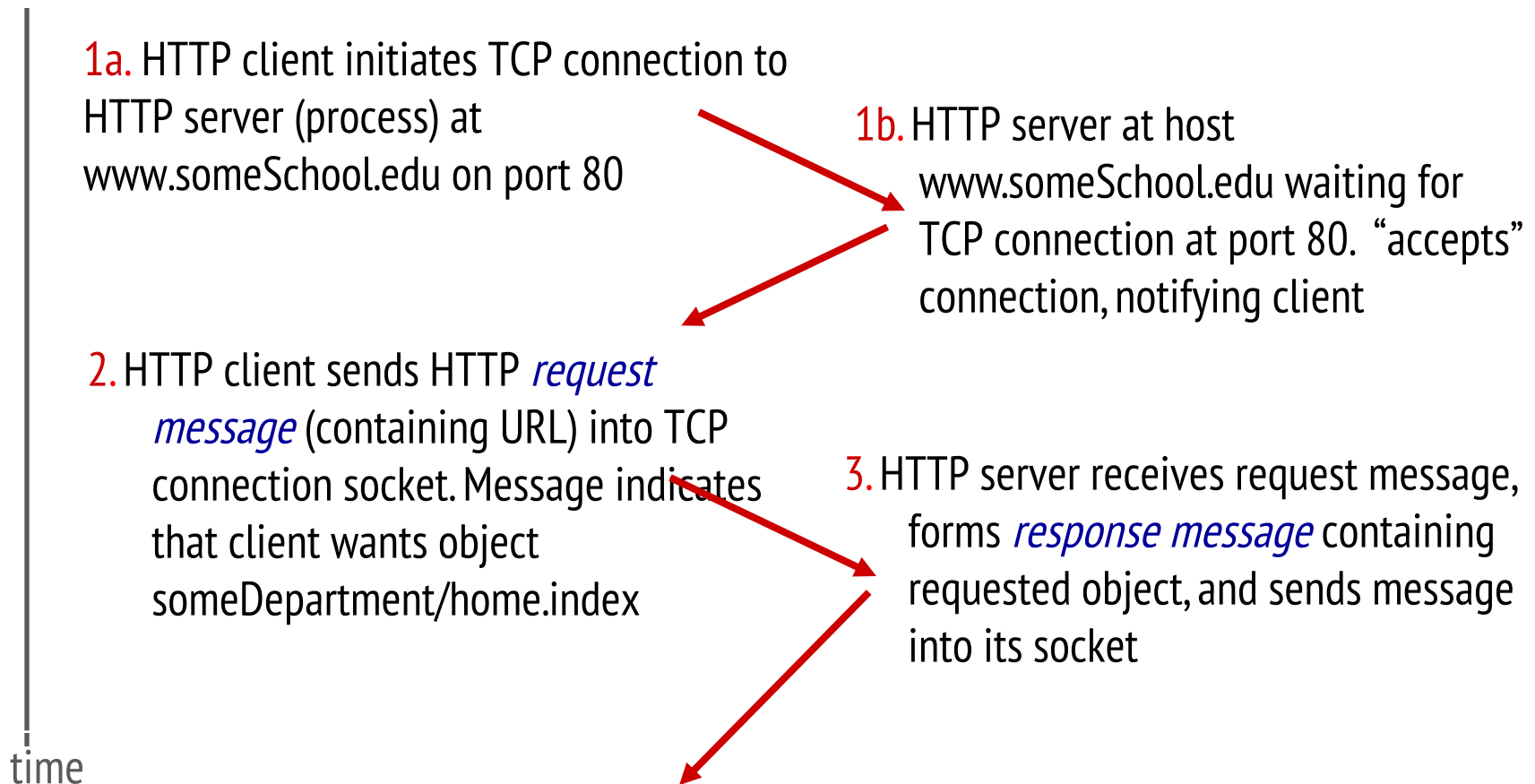
Traditional Applications

➢ World Wide Web

- TCP Connections

  o To overcome this situation, HTTP version 1.1 introduced *persistent connections* – the client and server can exchange multiple request/response messages over the same TCP connection.

  o Persistent connections have many advantages.

    ✓ First, they obviously eliminate the connection setup overhead, thereby reducing the load on the server, the load on the network caused by the additional TCP packets, and the delay perceived by the user.

    ✓ Second, because a client can send multiple request messages down a single TCP connection, TCP's congestion window mechanism is able to operate more efficiently.

      ▪ This is because it's not necessary to go through the slow start phase for each page.
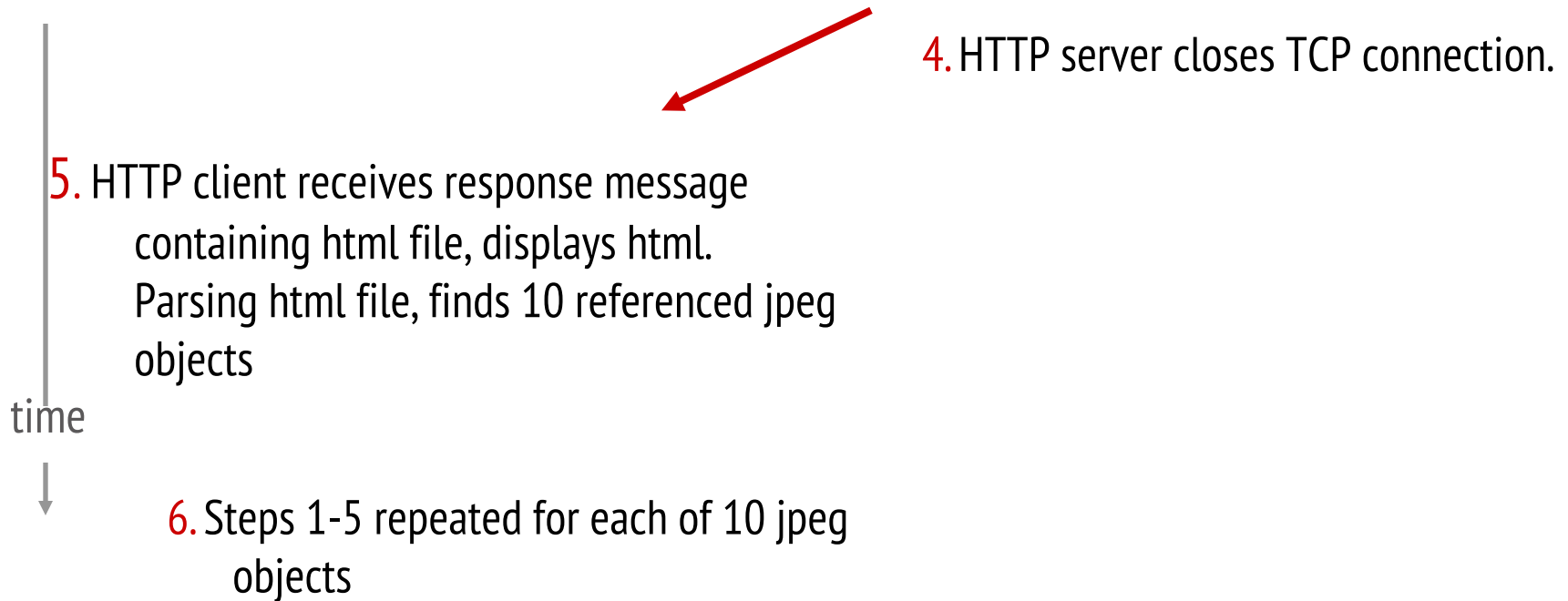
UNIVERSITY AT ALBANY
State University of New York

Non-persistent HTTP

suppose user enters URL:
www.someSchool.edu/someDepartment/home.index

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

Non-persistent HTTP (cont.)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

time

6. Steps 1-5 repeated for each of 10 jpeg objects

Non-persistent HTTP: response time

RTT (definition): time for a small packet to travel from client to server and back

HTTP 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

➢ non-persistent HTTP response time =

  2RTT+ file transmission time

initiate TCP
connection

RTT

request
file

RTT

file
received

time to
transmit
file

time   time
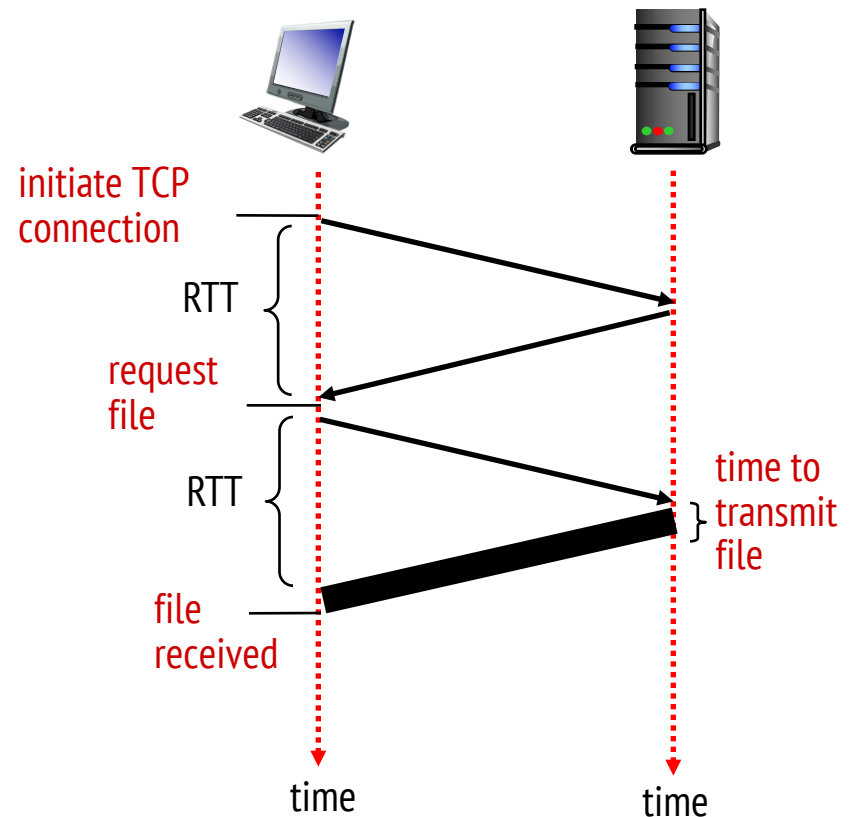
Persistent and non-persistent HTTP

non-persistent HTTP issues:

➢ requires 2 RTTs per object

➢ 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

➢ client sends requests as soon as it encounters a referenced object

➢ as little as one RTT for all the referenced objects

Caching

➢ One of the most active areas of research (and entrepreneurship) in the Internet today is how to effectively cache Web pages.

➢ Caching has benefits.

▪ From the client's perspective, a page that can be retrieved from a nearby cache can be displayed much more quickly than if it has to be fetched from across the world.

▪ From the server's perspective, having a cache intercept and satisfy a request reduces the load on the server.
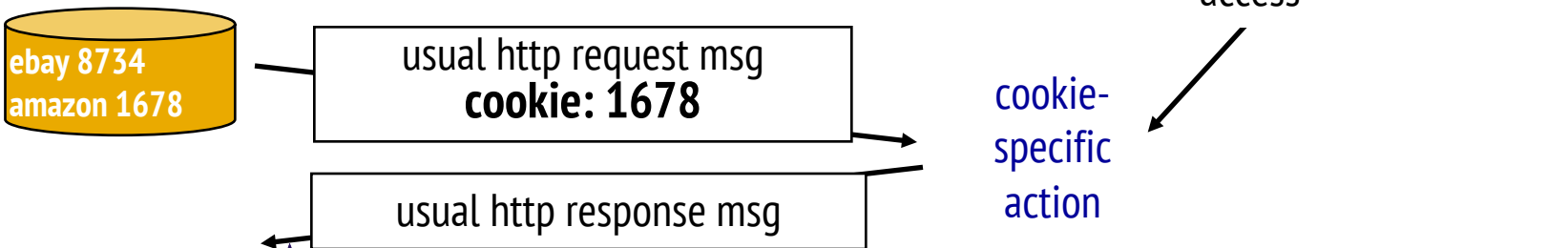
Caching

➢ Caching can be implemented in many different places.

- a user's browser can cache recently accessed pages, and simply display the cached copy if the user visits the same page again.
- a site can support a single site-wide cache.

➢ This allows users to take advantage of pages previously downloaded by other users.

➢ Closer to the middle of the Internet, ISPs can cache pages.

➢ Note that in the second case, the users within the site most likely know what machine is caching pages on behalf of the site, and they configure their browsers to connect directly to the caching host. This node is sometimes called a *proxy*

Cookies: keeping "state" (cont.)

client

server

cookie file

ebay 8734

usual http request msg →

Amazon server
creates ID
1678 for user

create
entry

backend
database

usual http response
set-cookie: 1678

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

Cookies (continued)

what cookies can be used for:

➢ authorization

➢ shopping carts

➢ recommendations

➢ user session state (Web e-mail)

cookies and privacy:

▪ cookies permit sites to learn a lot about you

▪ you may supply name and e-mail to sites

how to keep "state":

▪ protocol endpoints: maintain state at sender/receiver over multiple transactions

▪ cookies: http messages carry state

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
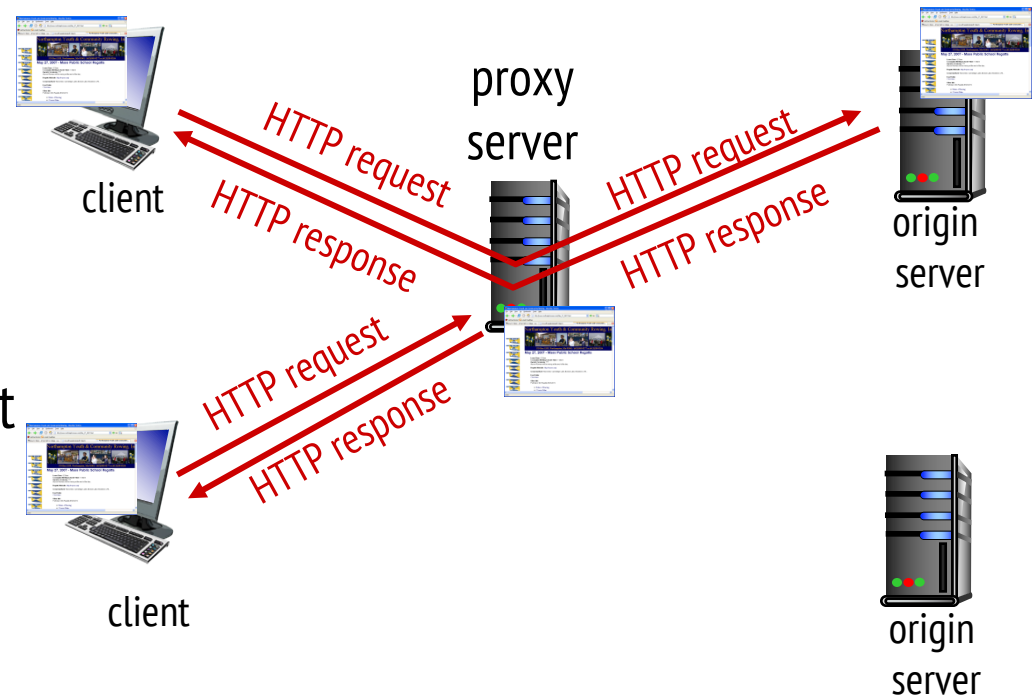
More about Web caching

➢ cache acts as both client and server

  ▪ server for original requesting client
  ▪ client to origin 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
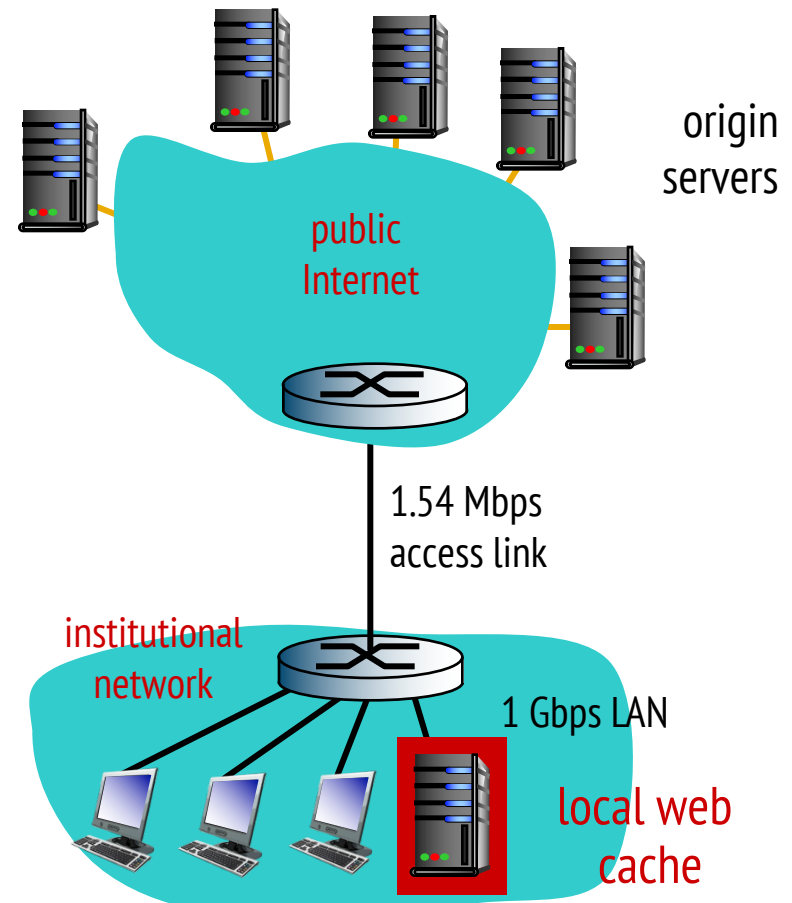
Caching example:

assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers:15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:

- LAN utilization: 15%   *problem!*
- access link utilization = 99%
- total delay   = Internet delay + access delay + LAN delay
  = 2 sec + minutes + usecs

origin servers

public Internet

1.54 Mbps access link

institutional network

1 Gbps LAN
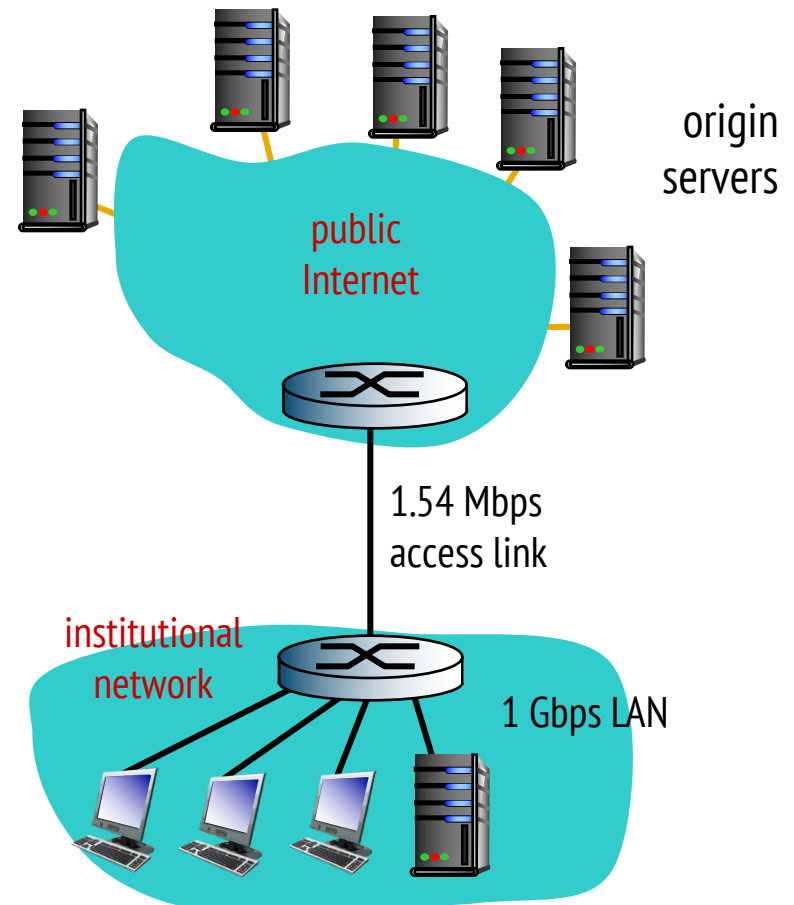
Caching example: fatter access link

assumptions:
- avg object size: 100K bits
- avg request rate from browsers to origin servers:15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps → 154 Mbps

consequences:
- LAN utilization: 15%
- access link utilization = 99% → 9.9%
- total delay = Internet delay + access delay + LAN delay
- = 2 sec + minutes + usecs → msecs
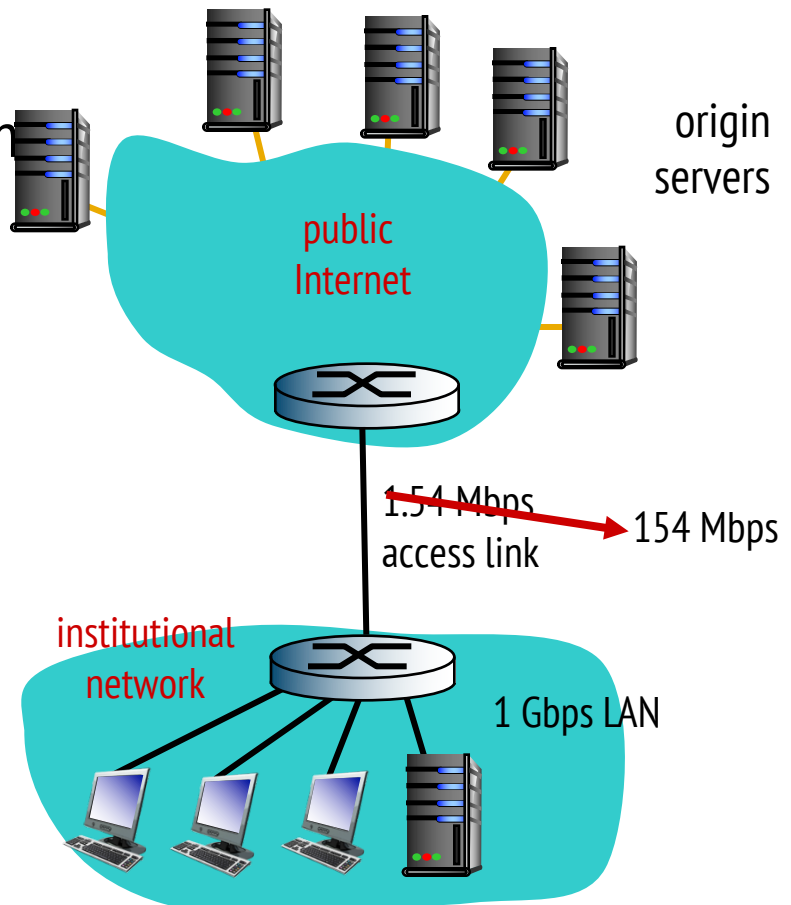
Cost: increased access link speed (not cheap!)

origin servers

public Internet

1.54 Mbps access link → 154 Mbps

institutional network

1 Gbps LAN

Caching example: install local cache

assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers:15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:

- LAN utilization: 15%
- access link utilization = ?
- total delay   = Internet delay + access delay + LAN delay = ?
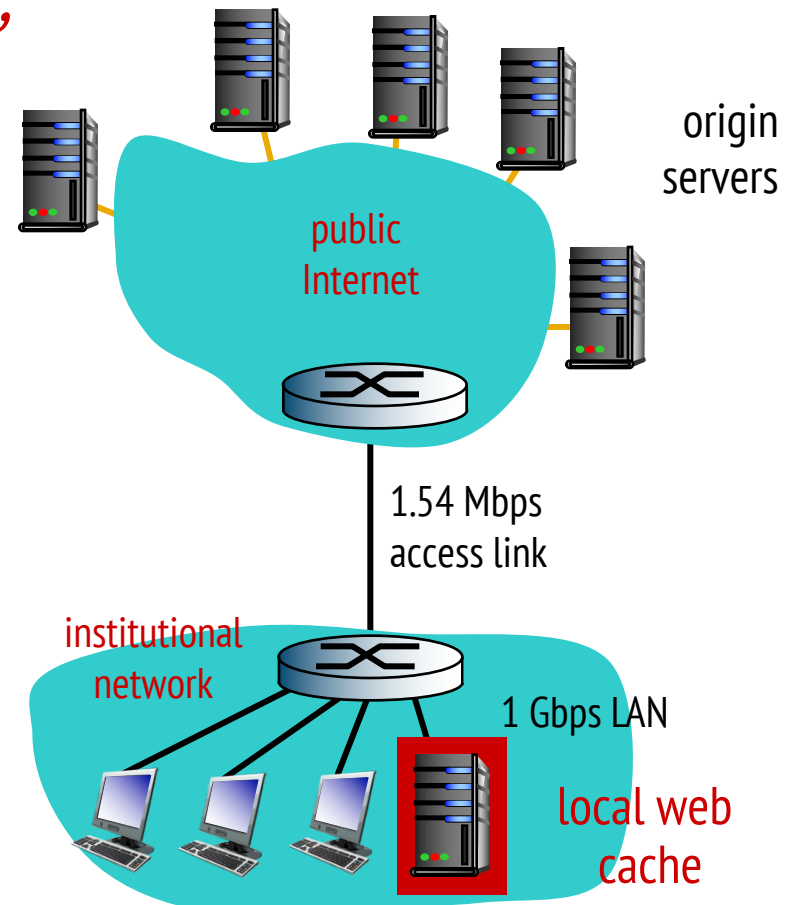
How to compute link utilization, delay?

Cost: web cache (cheap!)

origin servers

public Internet

1.54 Mbps access link

institutional network

1 Gbps LAN

local web cache

Caching example: install local cache

Calculating access link utilization, delay with cache:

➤ suppose cache hit rate is 0.4

- ■ 40% requests satisfied at cache, 60% requests satisfied at origin

➤ access link utilization:
- ■ 60% of requests use access link

➤ data rate to browsers over access link

= 0.6*1.50 Mbps = .9 Mbps
- ■ utilization = 0.9/1.54 = .58

➤ total delay
- ■ = 0.6 * (delay from origin servers) +0.4 * (delay when satisfied at cache)
- ■ = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs
- ■ less than with 154 Mbps link (and cheaper too!)

origin servers

public Internet

1.54 Mbps access link

institutional network

1 Gbps LAN

local web cache

Conditional GET

- *Goal:* don't send object if cache has up-to-date cached version
  - no object transmission delay
  - lower link utilization
- *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

client   server

| HTTP request msg |
| **If-modified-since: <date>** |

object not modified before <date>

| HTTP response |
| **HTTP/1.0** |
| **304 Not Modified** |

| HTTP request msg |
| **If-modified-since: <date>** |

object modified after <date>

| HTTP response |
| **HTTP/1.0 200 OK** |
| **<data>** |
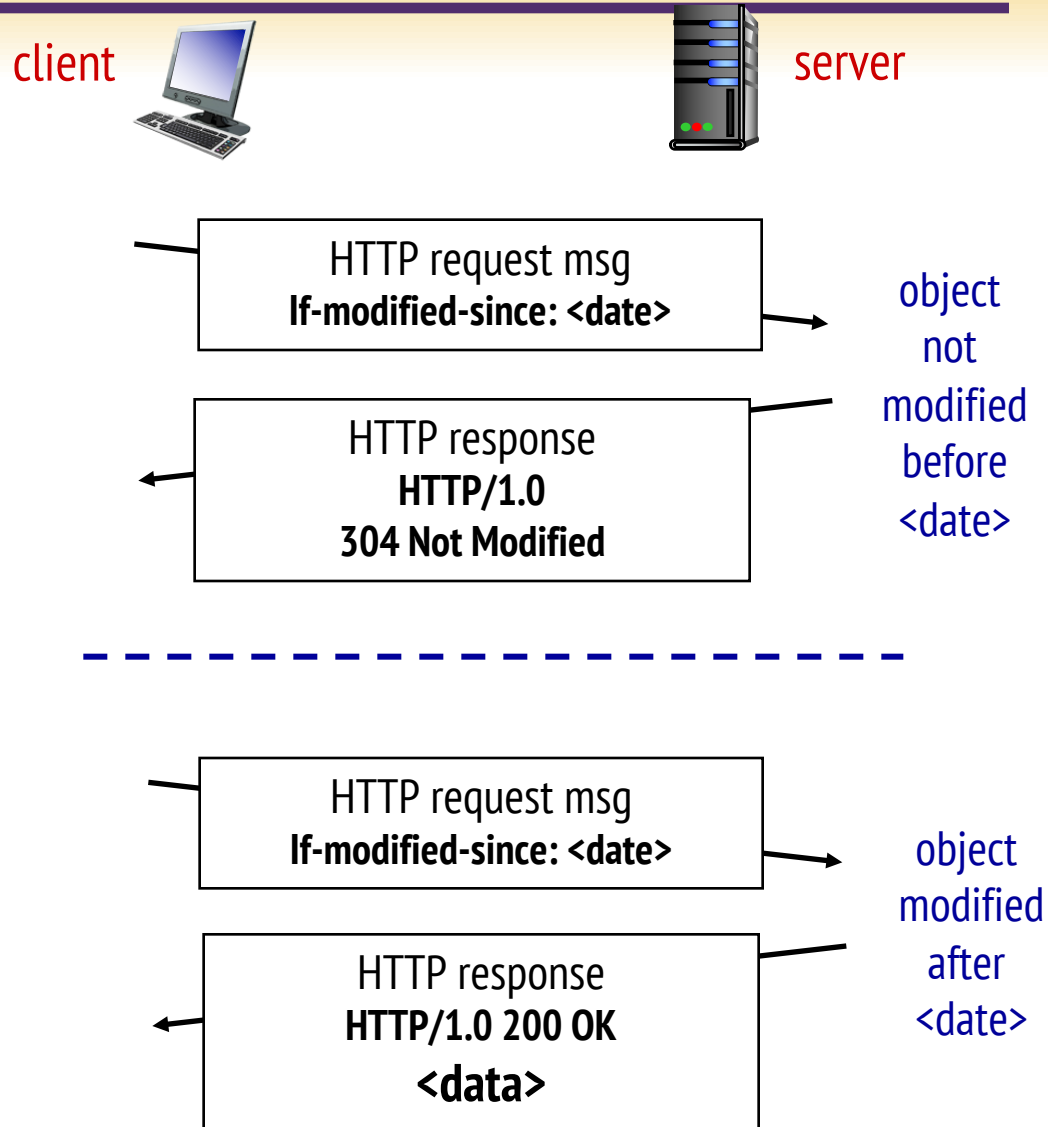
Electronic Mail

➢ SMTP, MIME, IMAP

- Email is one of the oldest network applications

- It is important
  - (1) to distinguish the user interface (i.e., your mail reader) from the underlying message transfer protocols (such as SMTP or IMAP), and
  - (2) to distinguish between this transfer protocol and a companion protocol (RFC 822 and MIME) that defines the format of the messages being exchanged
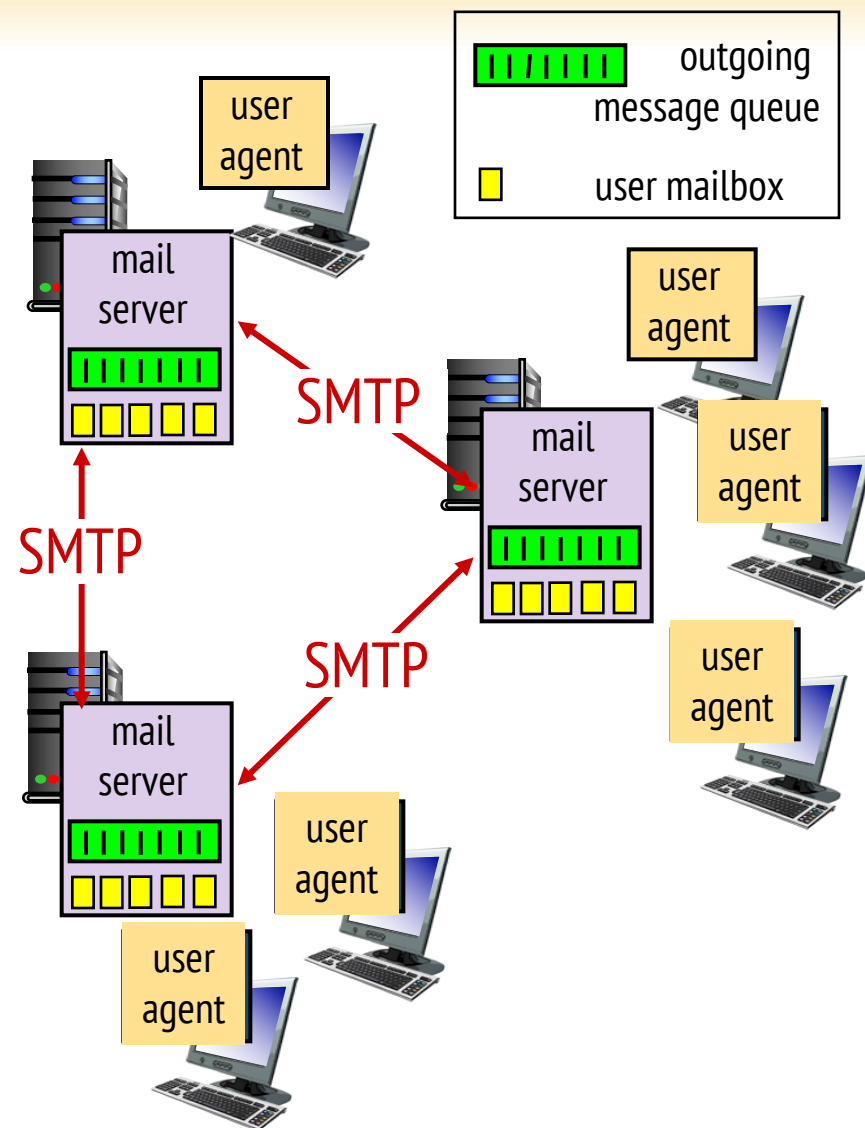
Electronic Mail

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: servers

mail servers:

➢ *mailbox* contains incoming messages for user

➢ *message queue* of outgoing (to be sent) mail messages

➢ *SMTP protocol* between mail servers to send email messages

- client: sending mail server
- "server": receiving mail server
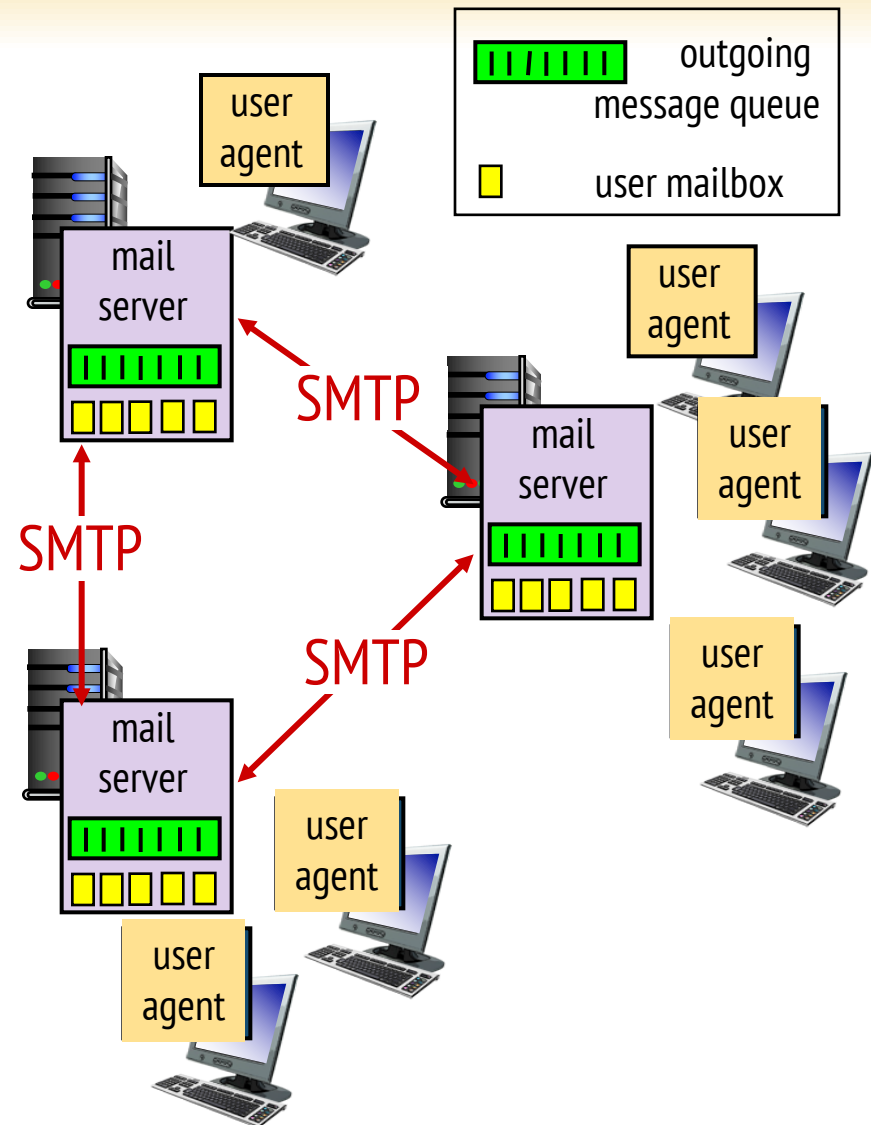
Electronic Mail: SMTP [RFC 2821]

➢ 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

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message "to" bob@someschool.edu

2) Alice's UA sends message to her mail server; message placed in message queue

3) client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message

Alice's mail server
(rutgers.edu)

Bob's mail server
(albany.edu)
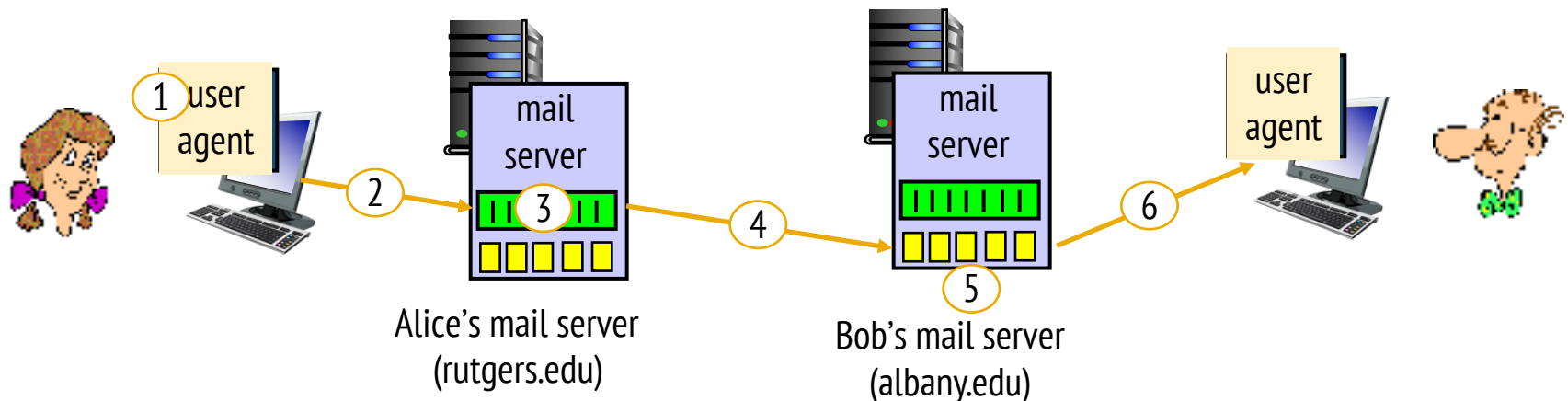
Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

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:

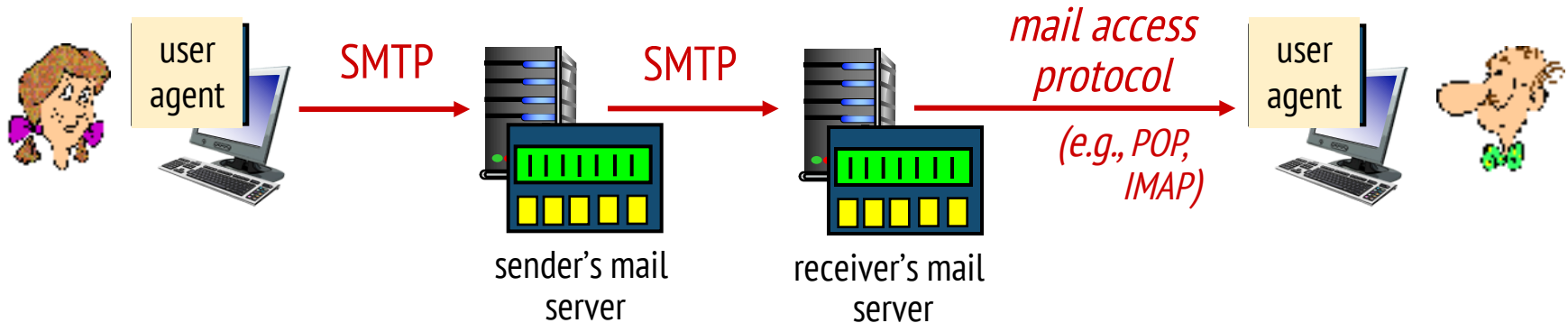   *different from* SMTP MAIL FROM, RCPT TO: commands!

➢ Body: the "message"

   ▪ ASCII characters only

header
body

blank line
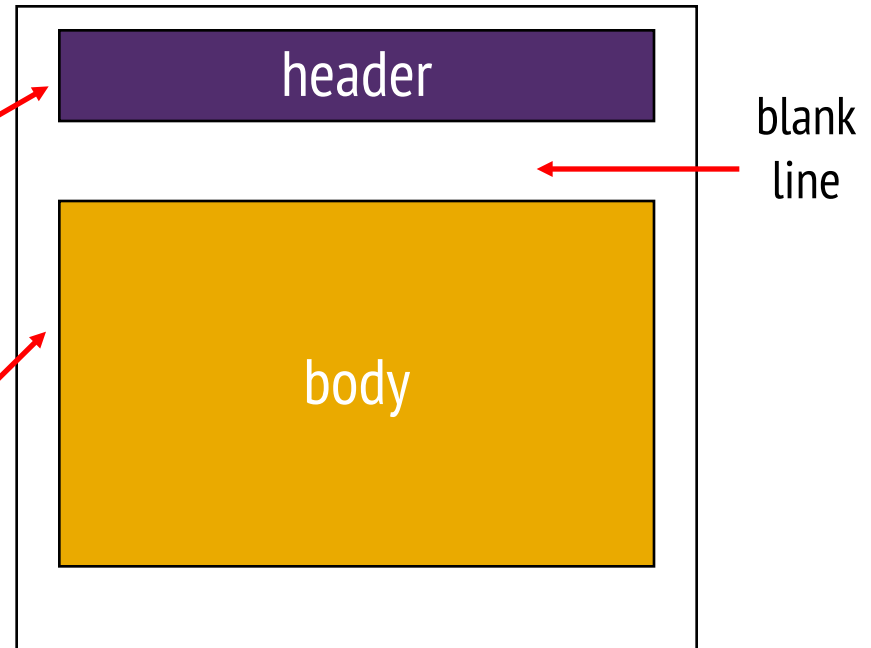
➢ MIME – Multipurpose Internet Mail Extensions

   ▪ Supports non-text attachments

UNIVERSITY AT ALBANY
State University of New York

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.

POP3 protocol

authorization phase

➢ client commands:

■ `user`: declare username

■ `pass`: password

➢ server responses

■ `+OK`

■ `-ERR`

transaction phase, client:

➢ `list`: list message numbers

➢ `retr`: retrieve message by number

➢ `dele`: delete

➢ `quit`

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more) and IMAP

more about POP3

➢ previous example uses POP3 "download and delete" mode

  ▪ Bob cannot re-read e-mail if he changes client

➢ POP3 "download-and-keep": copies of messages on different clients

➢ POP3 is stateless across sessions

IMAP

➢ keeps all messages in one place: at server

➢ allows user to organize messages in folders

➢ keeps user state across sessions:

➢ names of folders and mappings between message IDs and folder name

Multimedia Applications

➢ Audio and video have become key types of traffic, e.g., voice over IP, and video streaming.

- Digital audio
- Digital video
- Streaming stored media
- Streaming live media
- Real-time conferencing

Digital Audio (1)

➢ ADC (Analog-to-Digital Converter) produces digital audio from a microphone

- Telephone: 8000 8-bit samples/second (64 Kbps); computer audio is usually better quality (e.g., 16 bit)

ADC →

Continuous audio
(sine wave)

Digital audio
(sampled, 4-bit quantized)
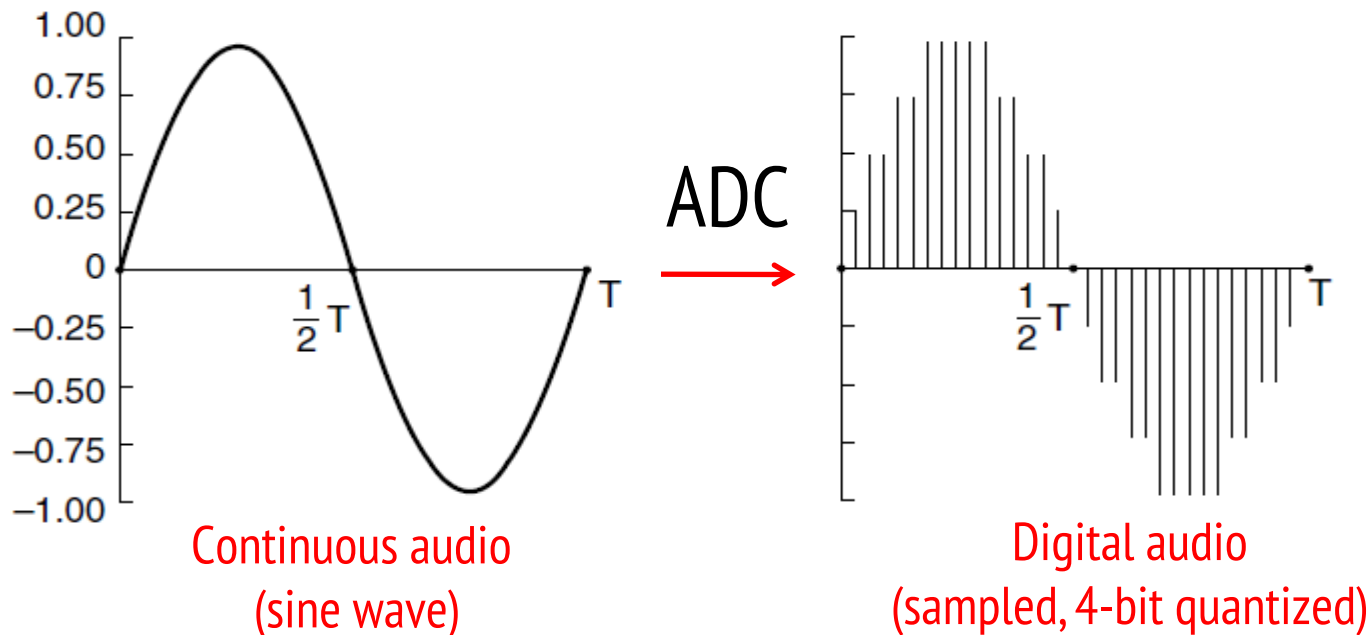
Digital Audio (2)

➢ Digital audio is typically compressed before it is sent

- Lossy encoders (like Advanced Audio Coding - AAC) exploit human perception

- It is not necessary to encode signals whose power falls below Threshold of audibility.

- Large compression ratios (can be >10X)
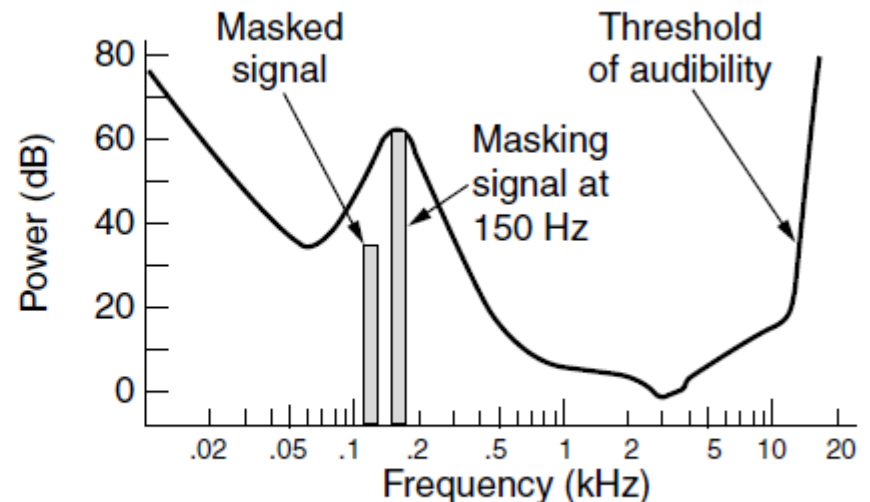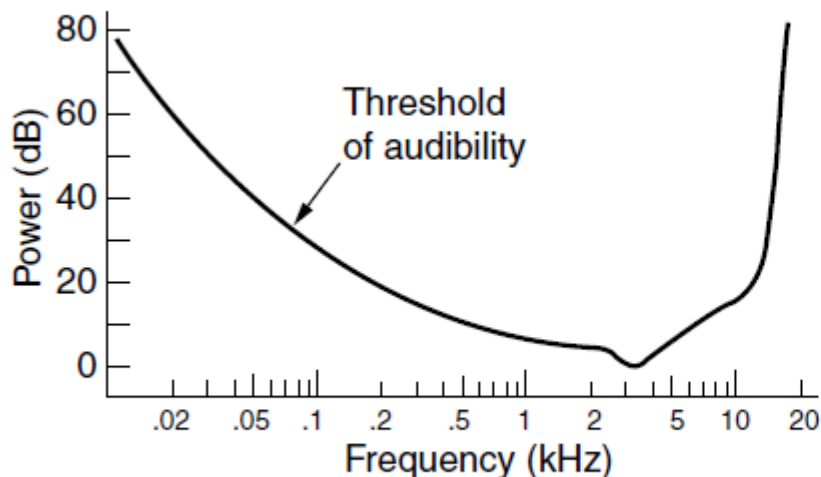
Sensitivity of the ear varies with frequency
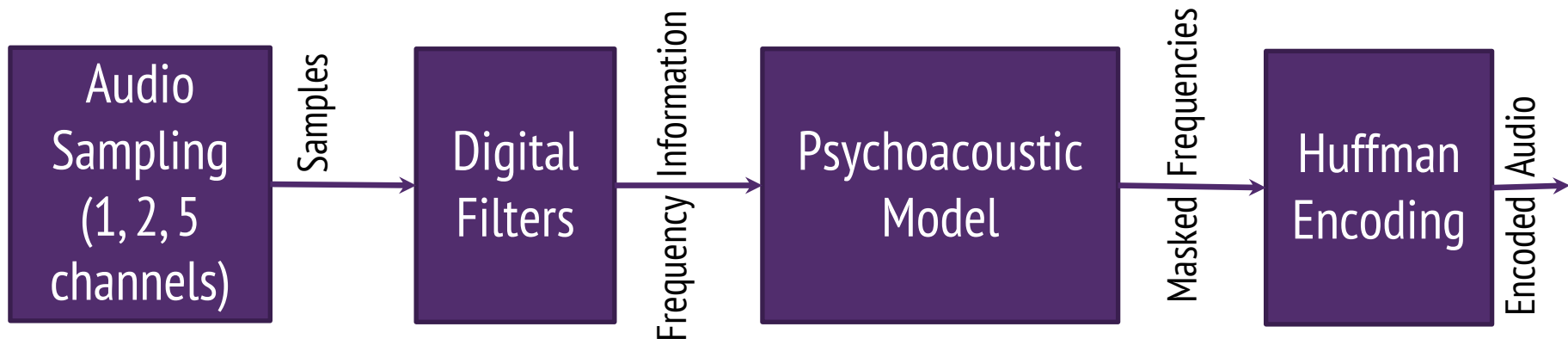
A loud tone can mask nearby tones

Digital Audio (3)

➢ ## Perceptual Coding

- Based on science of psychoacoustics (way humans perceive sound)
- Frequency Masking
- Temporal Masking
- Examples: MP3, AAC

➢ ## Huffman Encoding

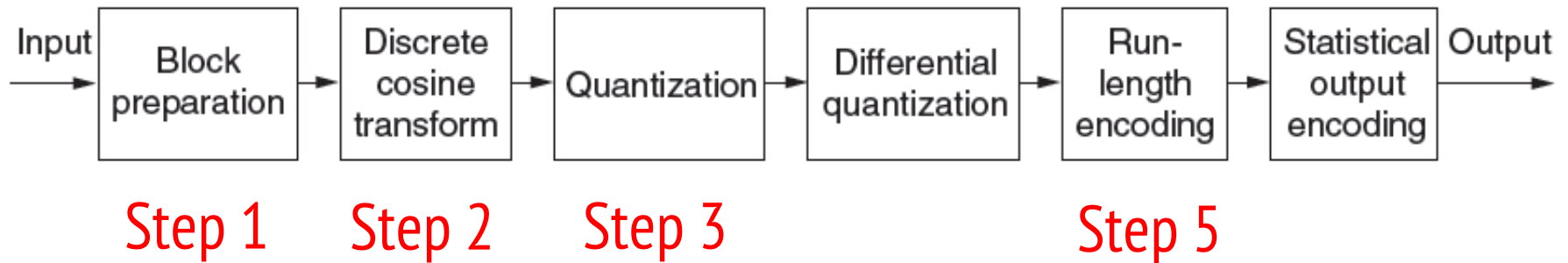- Assigns short codes to numbers that appear frequently

| Audio Sampling (1, 2, 5 channels) | → Samples → | Digital Filters | → Frequency Information → | Psychoacoustic Model | → Masked Frequencies → | Huffman Encoding | → Encoded Audio → |

Digital Video (1)

- ➢ Video is digitized as pixels (sampled, quantized)
  - ▪ SD TV quality: 640x480 pixels, 24-bit color, 30 times/sec
  - ▪ HD TV quality: 1280x720 pixels
  - ▪ Player refreshes 500-100 times to minimize flicker
- ➢ Video is sent compressed due to its large bandwidth
  - ▪ Lossy compression exploits human perception
    - o E.g., JPEG for still images, MPEG, H.264 for video
  - ▪ Large compression ratios (often 50X for video)
  - ▪ Video is normally > 1 Mbps, versus >10 kbps for speech and >100 kbps for music

UNIVERSITY AT ALBANY
State University of New York

Digital Video (2)

➢ JPEG lossy compression sequence for one image:

Input → | Block preparation | → | Discrete cosine transform | → | Quantization | → | Differential quantization | → | Run-length encoding | → | Statistical output encoding | → Output

Step 1   Step 2   Step 3   Step 5

Digital Video (3)

➢ Step 1: Pixels are mapped to luminance (brightness)/chrominance (YCbCr) color space

- Luma signal (Y), Chroma signal: 2 components (Cb and Cr)
- Chrominance is sub-sampled, the eye is less sensitive to chrominance
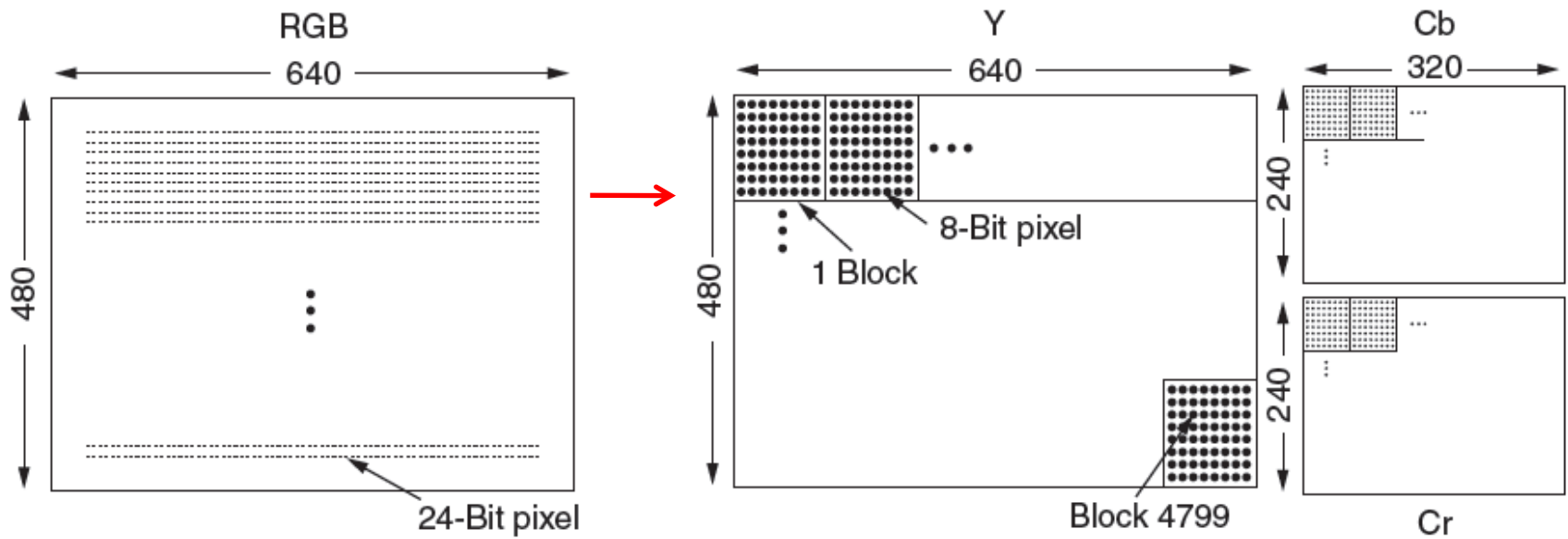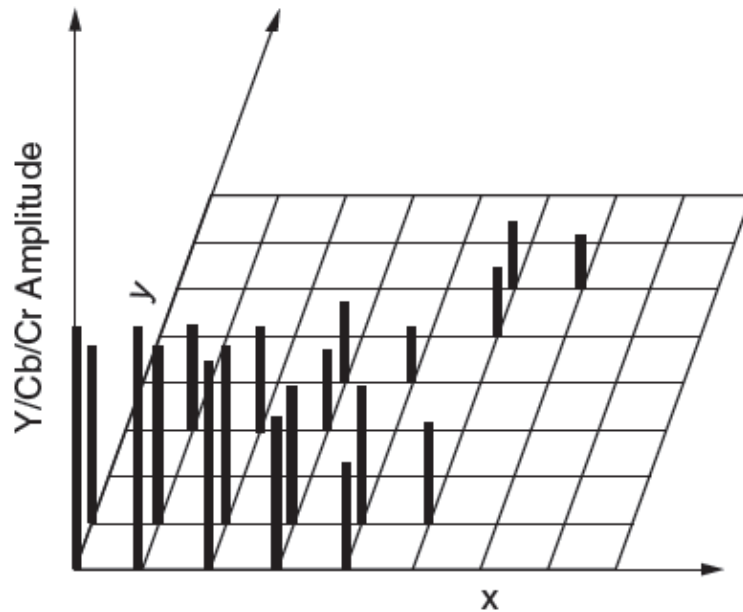

Input 24-bit RGB pixels

8-bit luminance pixels

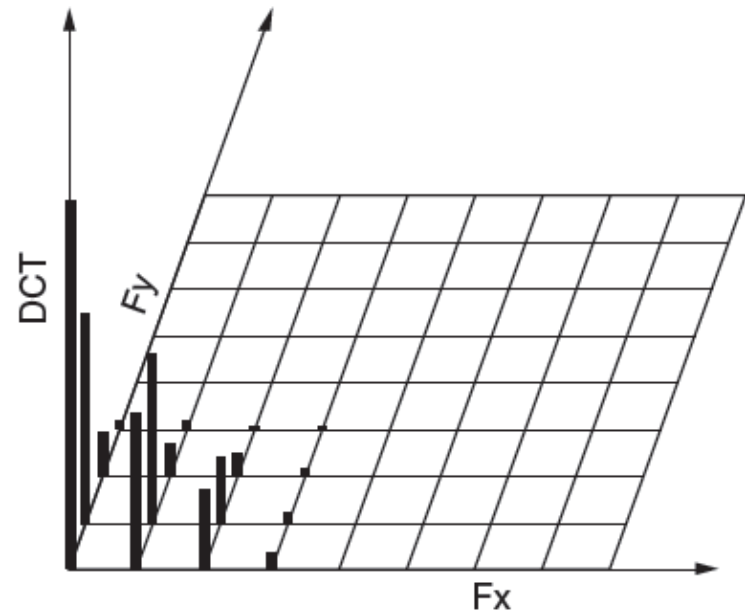8-bit chrominances for every 4 pixels

Digital Video (4)

➢ Step 2: Each component block is transformed to spatial frequencies with DCT (Discrete Cosine Transformation)

■ Captures the key image features

One component block   Transformed block

Digital Video (5)

➢ Step 3: DCT coefficients are quantized by dividing by thresholds; reduces bits in higher spatial frequencies

- Top left element is differenced over blocks (Step 4)

DCT coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Input   /   Thresholds   =   Output

Digital Video (6)

➢ Step 5: The block is run-length encoded in a zig-zag order. Then it is Huffman coded before sending (Step 6)

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Order in which the block coefficients are sent

Digital Video (7)

➢ MPEG compresses over a sequence of frames, further using motion tracking to remove temporal redundancy

- MPEG Output
- I (Intra-coded) frames are self-contained
- P (Predictive) frames use block motion predictions
- B (Bidirectional) frames may base prediction on future frame
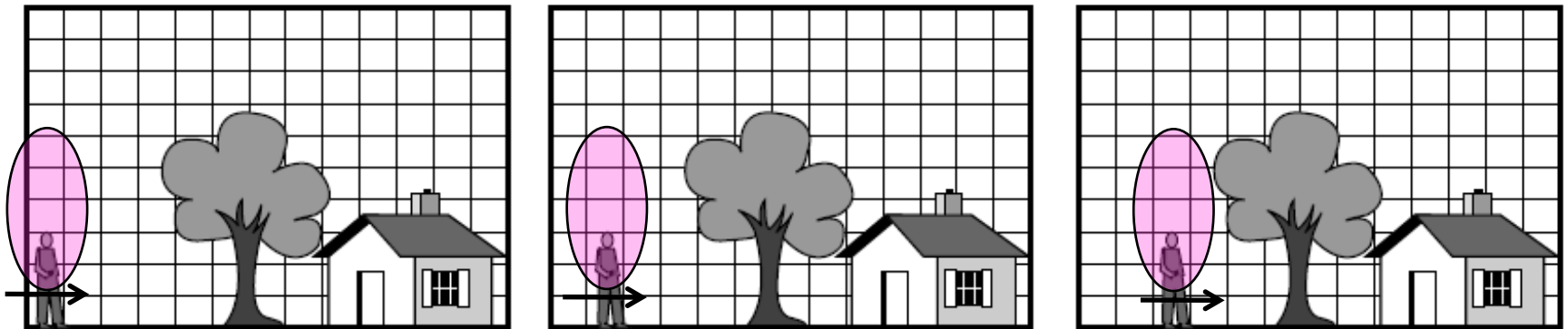
Three consecutive frames with stationary and moving components
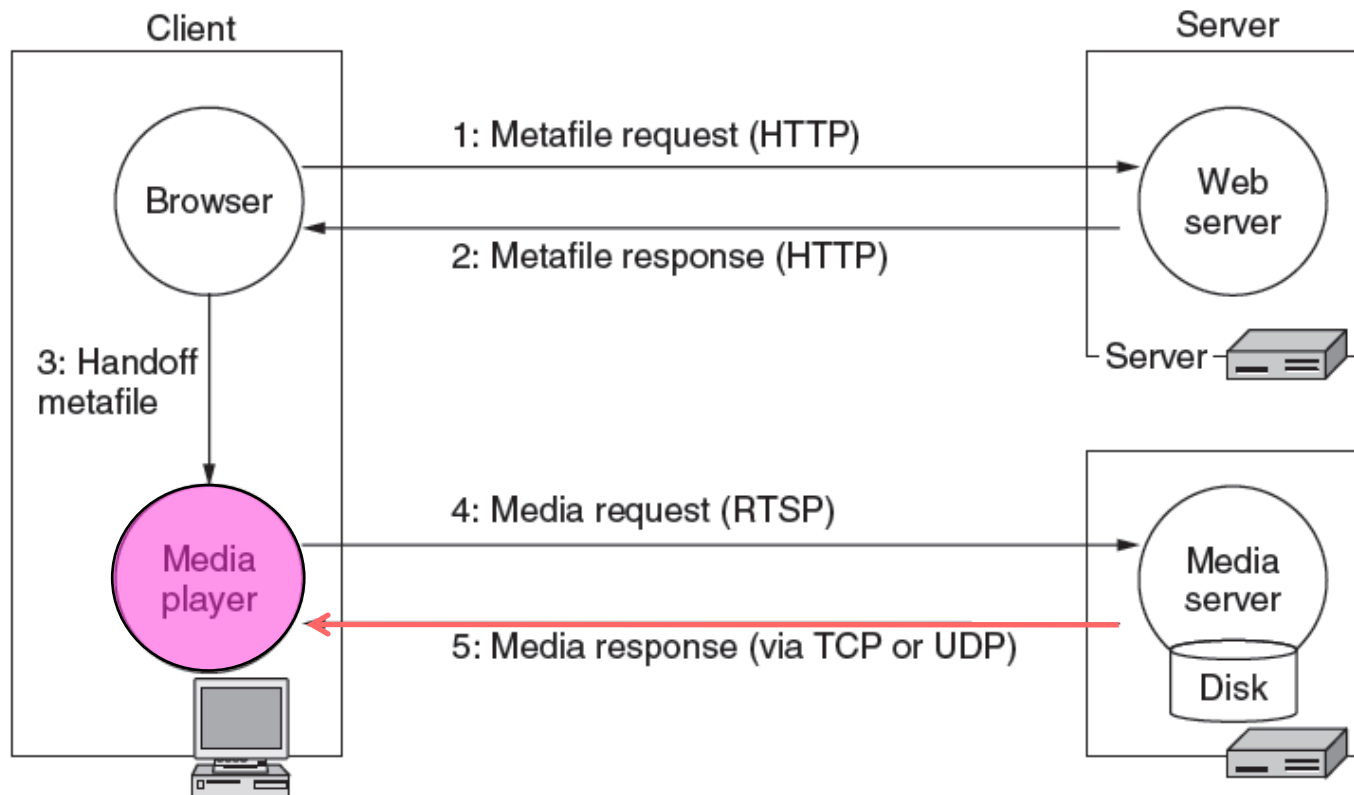
Streaming Stored Media (1)

➢ A simple method to stream stored media, e.g., for video on demand, is to fetch the video as a file download

- But has large startup delay, except for short files
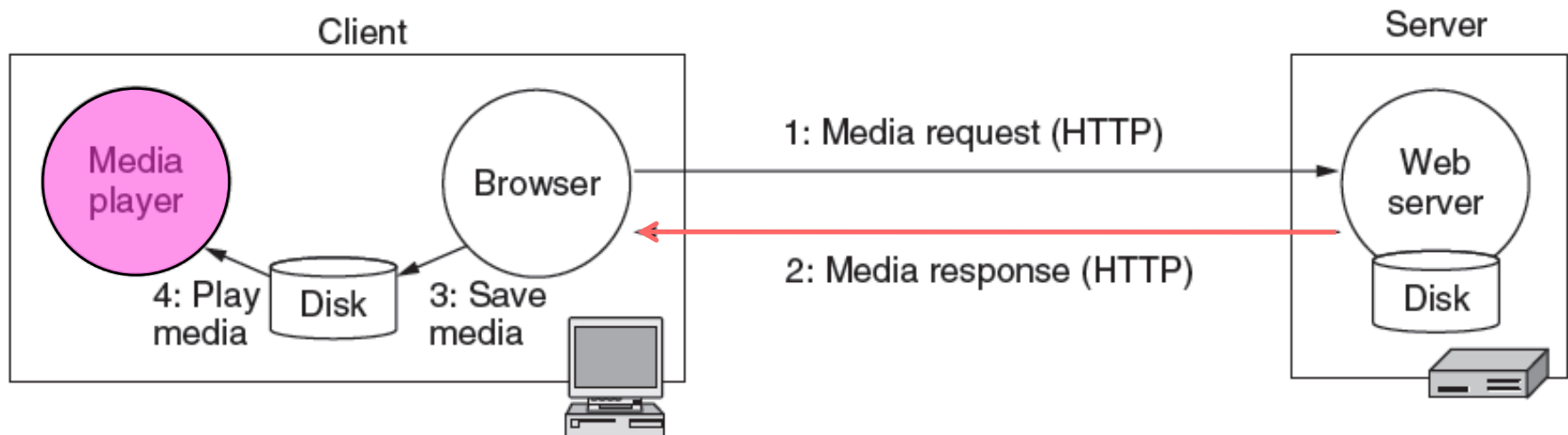
Streaming Stored Media (2)

➢ Effective streaming starts the playout during transport
- With RTSP (Real-Time Streaming Protocol)
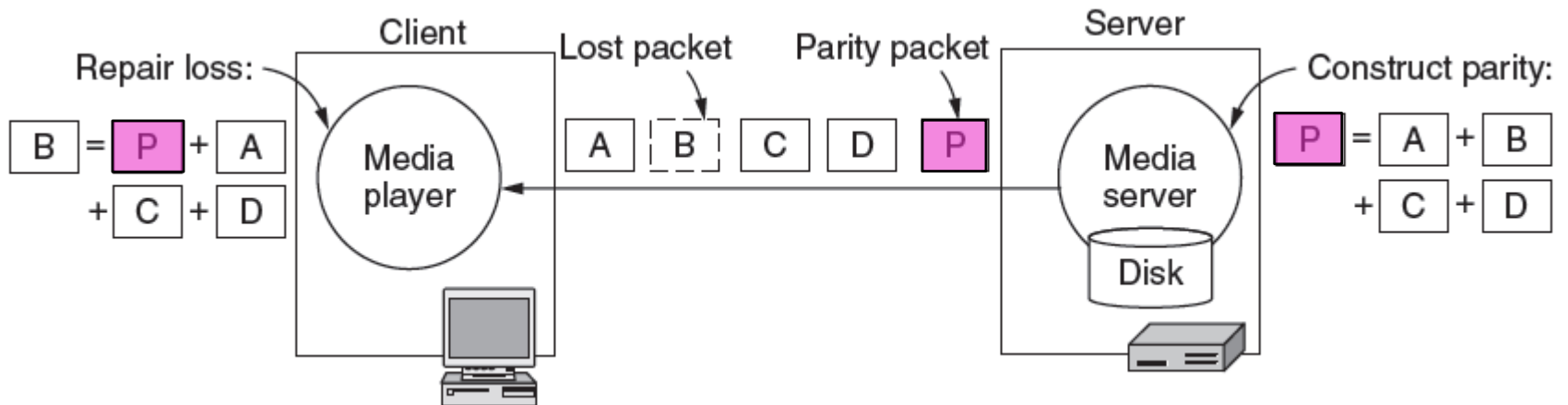
Streaming Stored Media (3)

➢ Key problem: how to handle transmission errors

| Strategy	Advantage	Disadvantage
Use reliable transport (TCP)	Repairs all errors	Increases jitter significantly
Add FEC (e.g., parity)	Repairs most errors	Increases overhead, decoding complexity and jitter
Interleave media	Masks most errors	Slightly increases decoding complexity and jitter

Streaming Stored Media (4)

➢ Parity packet can repair one lost packet in a group of N
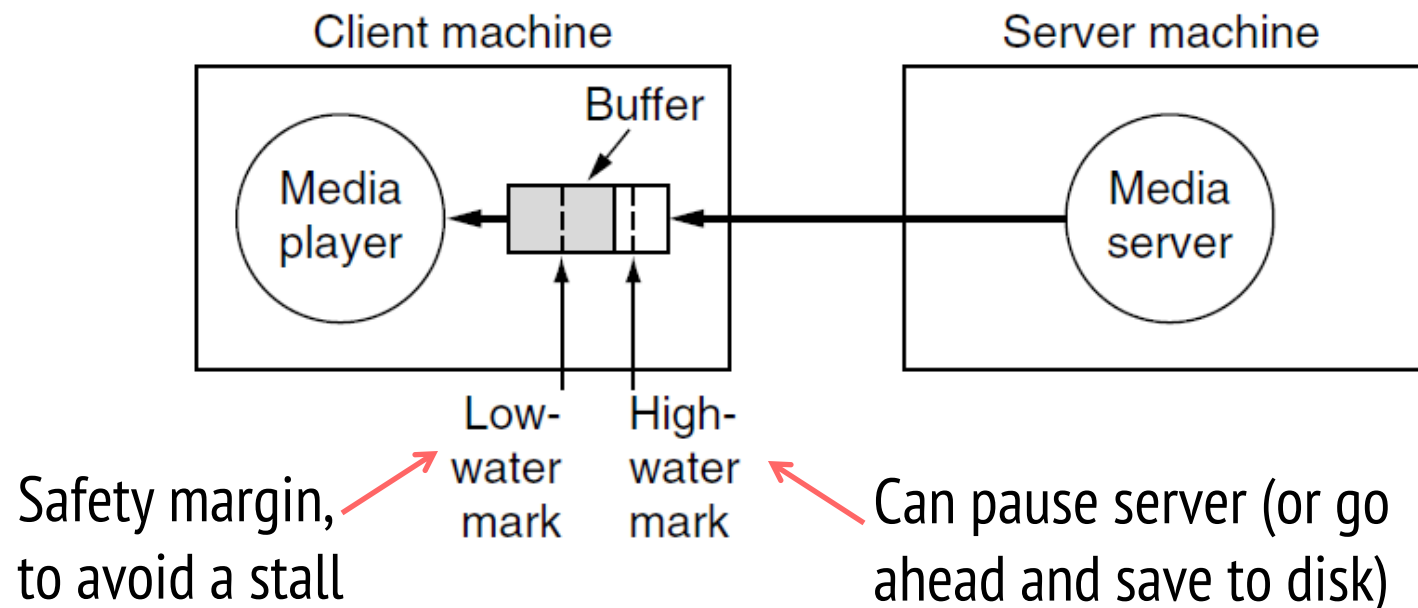 ▪ Decoding is delayed for N packets

Streaming Stored Media (5)

➢ Interleaving spreads nearby media samples over different transmissions to reduce the impact of loss
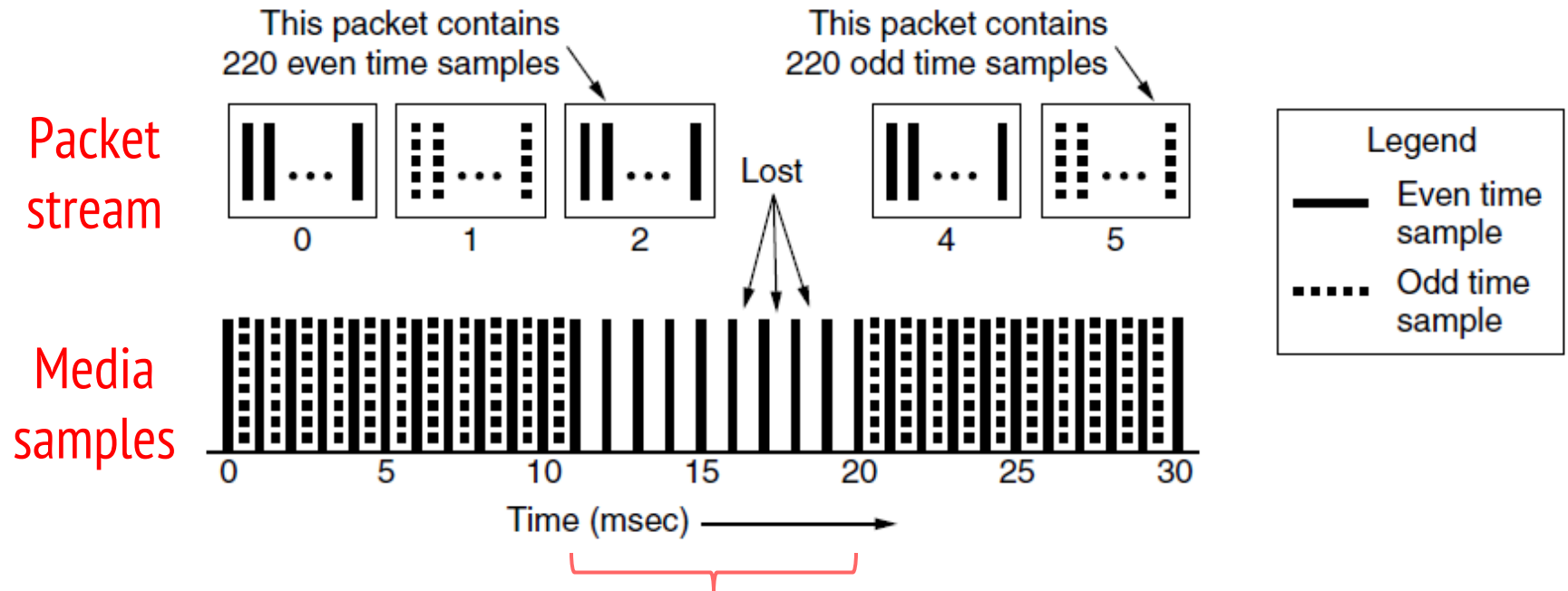
Loss reduces temporal resolution; doesn't leave a gap

Streaming Stored Media (6)

➢ Key problem: media may not arrive in time for playout due to variable bandwidth and loss/retransmissions

- Client buffers media to absorb jitter; we still need to pick an achievable media rate

Client machine

Server machine

Buffer

Media player

Media server

Low-water mark

High-water mark

Safety margin, to avoid a stall

Can pause server (or go ahead and save to disk)

Streaming Stored Media (7)

➢ RTSP commands
- Sent from player to server to adjust streaming

| Command	Server action
DESCRIBE	List media parameters
SETUP	Establish a logical channel between the player and the server
PLAY	Start sending data to the client
RECORD	Start accepting data from the client
PAUSE	Temporarily stop sending data
TEARDOWN	Release the logical channel

Streaming Live Media (1)

➤ Streaming live media is similar to the stored case plus:

- Can't stream faster than "live rate" to get ahead
  - Usually need larger buffer to absorb jitter

- Often have many users viewing at the same time
  - UDP with multicast greatly improves efficiency. It is rarely available, so many TCP connections are used.
  - For very many users, content distribution is used [later]
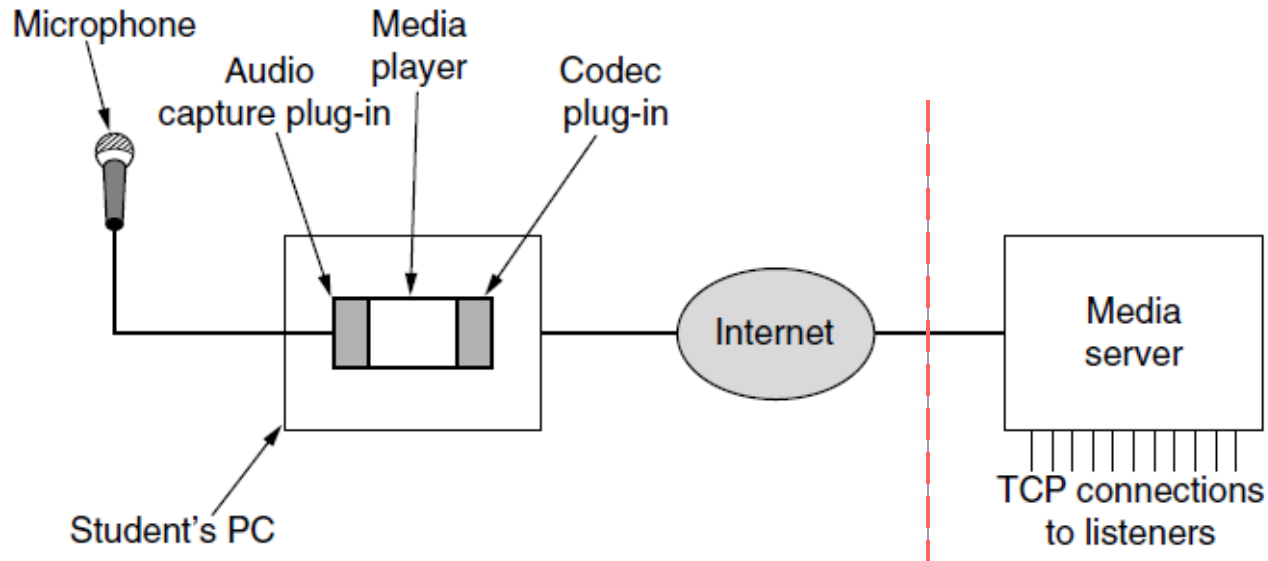
Streaming Live Media (2)

➢ With multicast streaming media, parity is effective
  ▪ Clients can each lose a different packet and recover
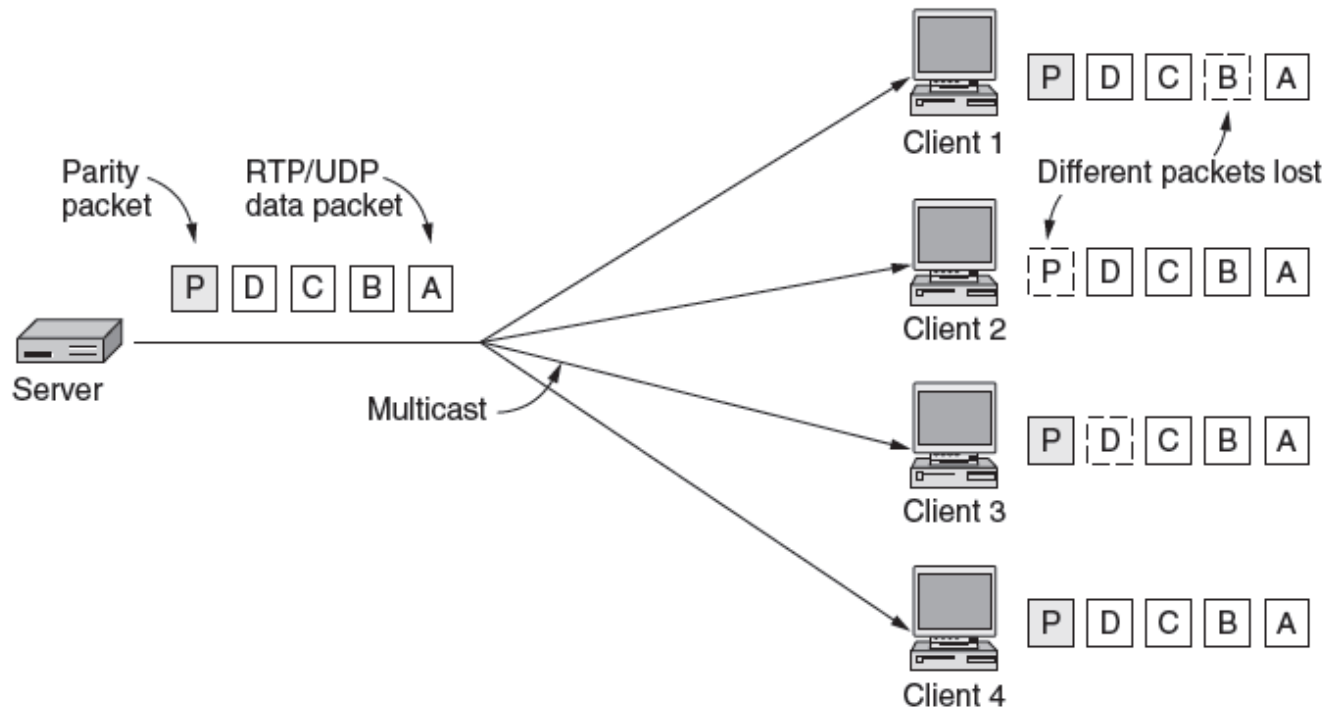
Streaming Live Media (2)

➢ Production side of a student radio station.

Real-Time Conferencing (1)

➢ Real-time conferencing has two or more connected live media streams, e.g., voice over IP, Skype video call

➢ Key issue over live streaming is low (interactive) latency
- Want to reduce delay to near propagation
- Benefits from network support, e.g., QoS
- Or, benefits from ample bandwidth (no congestion)
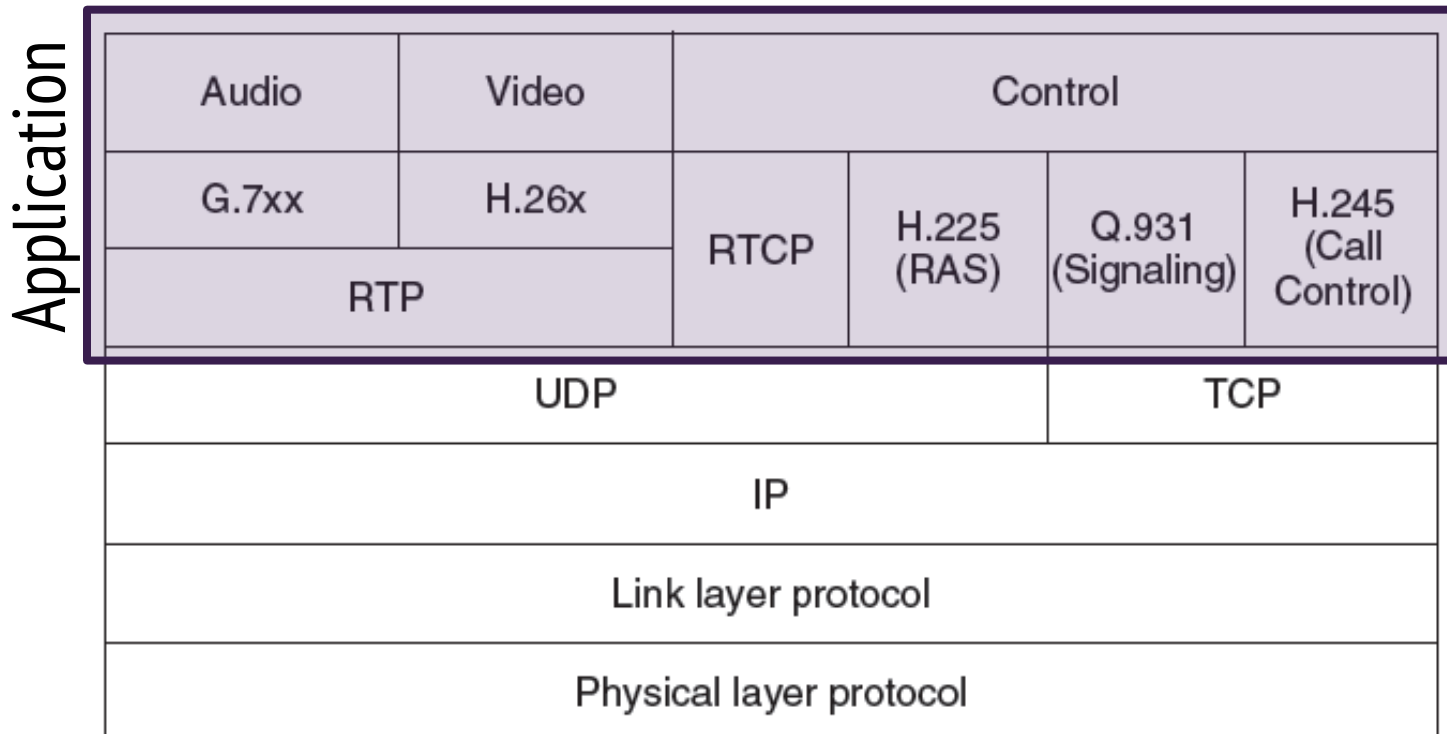
Real-Time Conferencing (2)

➢ H.323 architecture for Internet telephony supports calls between Internet computers and PSTN phones.
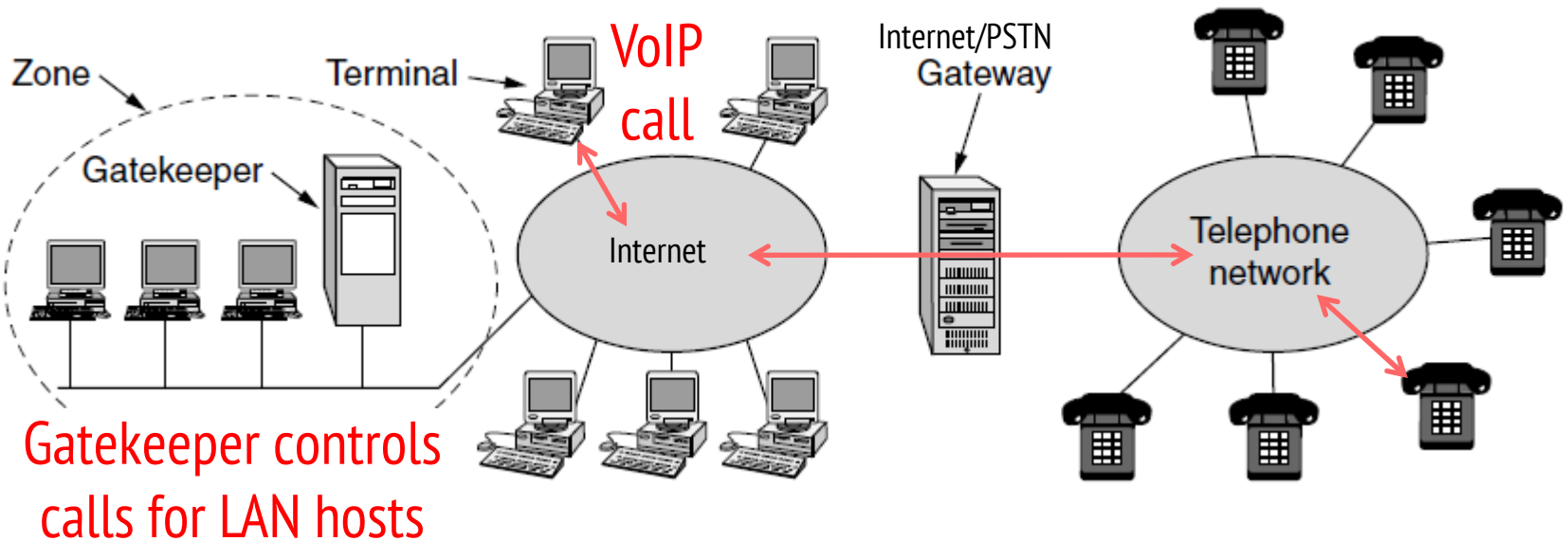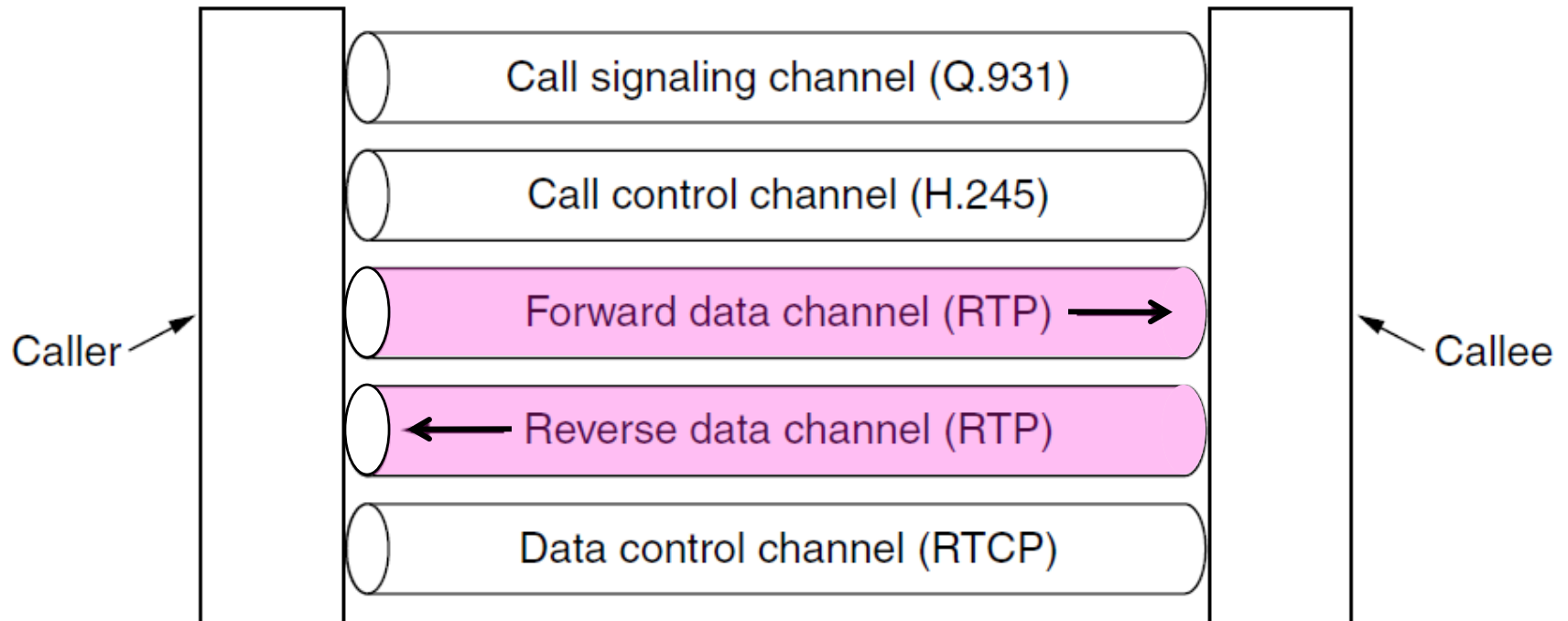
Real-Time Conferencing (3)

- ➤ H.323 protocol stack:
  - ▪ Call is digital audio/video over RTP/UDP/IP
  - ▪ Call setup is handled by other protocols (Q.931 etc.)

Real-Time Conferencing (4)

➢ Logical channels that make up an H.323 call

Real-Time Conferencing (5)

➢ SIP (Session Initiation Protocol) is an alternative to H.323 to set up real-time calls

- Simple, text-based protocol with URLs for addresses
- Data is carried with RTP / RTCP as before

Method	Description
INVITE	Request initiation of a session
ACK	Confirm that a session has been initiated
BYE	Request termination of a session
OPTIONS	Query a host about its capabilities
CANCEL	Cancel a pending request
REGISTER	Inform a redirection server about the user's current location

Real-Time Conferencing (6)

➢ Setting up a call with the SIP three-way handshake

- Proxy server lets a remote callee be connected
- Call data flows directly between caller/callee
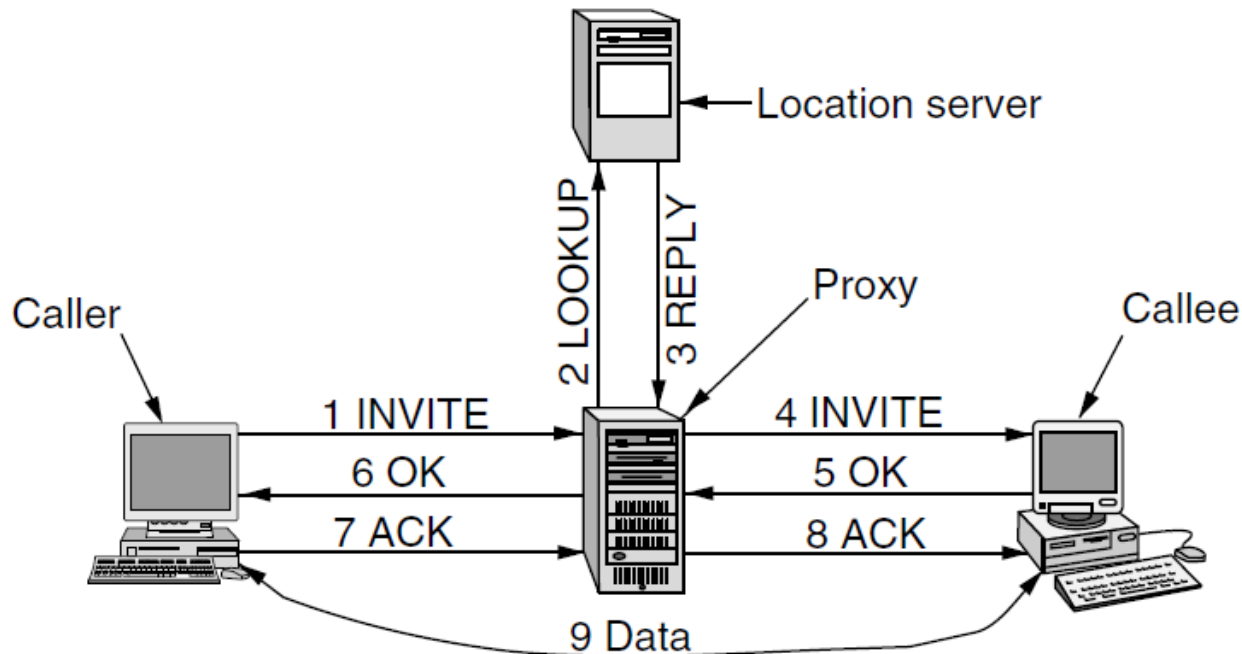
Real-Time Conferencing (7)

Item	H.323	SIP
Designed by	ITU	IETF
Compatibility with PSTN	Yes	Largely
Compatibility with Internet	Yes, over time	Yes
Architecture	Monolithic	Modular
Completeness	Full protocol stack	SIP just handles setup
Parameter negotiation	Yes	Yes
Call signaling	Q.931 over TCP	SIP over TCP or UDP
Message format	Binary	ASCII
Media transport	RTP/RTCP	RTP/RTCP
Multiparty calls	Yes	Yes
Multimedia conferences	Yes	No
Addressing	URL or phone number	URL
Call termination	Explicit or TCP release	Explicit or timeout
Instant messaging	No	Yes
Encryption	Yes	Yes
Size of standards	1400 pages	250 pages
Implementation	Large and complex	Moderate, but issues
Status	Widespread, esp. video	Alternative, esp. voice

Comparison of H.323 and SIP.

DNS: domain name system

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., www.yahoo.com - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:

- *distributed database* implemented in hierarchy of many name servers
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)

note: core Internet function, implemented as application-layer protocol complexity at network's "edge"

DNS: services, structure

DNS services
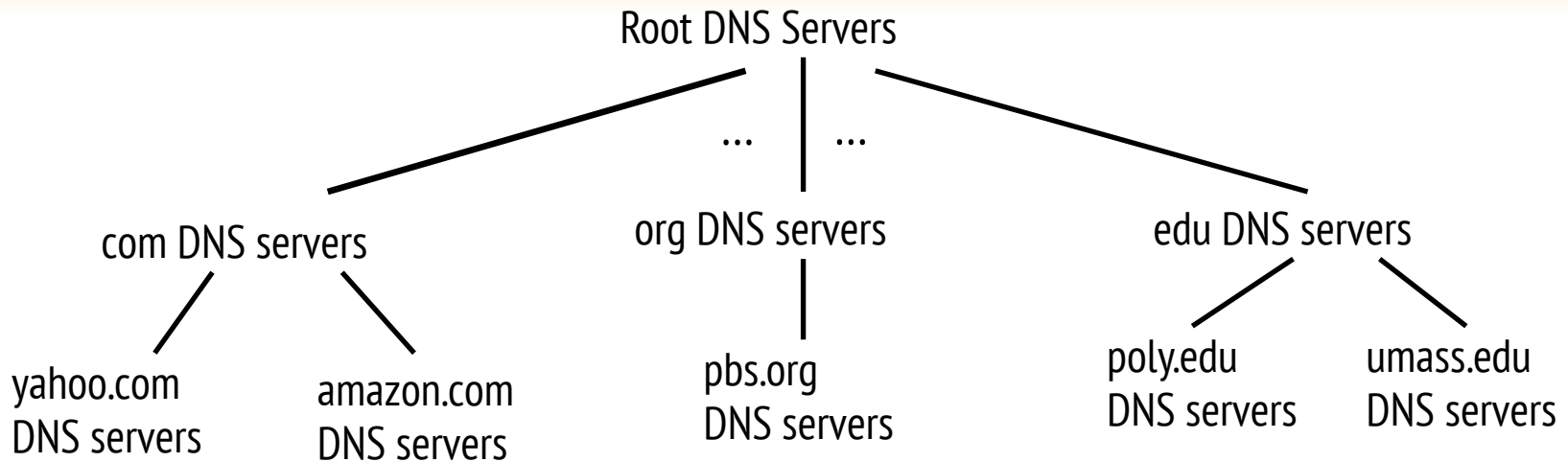
➢ hostname to IP address translation

➢ host aliasing
  ▪ canonical, alias names

➢ mail server aliasing

➢ load distribution
  ▪ replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

➢ single point of failure

➢ traffic volume

➢ distant centralized database

➢ maintenance

A: doesn't scale!

UNIVERSITY AT ALBANY
State University of New York

DNS: a distributed, hierarchical database

Root DNS Servers

... ...

com DNS servers
org DNS servers
edu DNS servers

yahoo.com
DNS servers

amazon.com
DNS servers

pbs.org
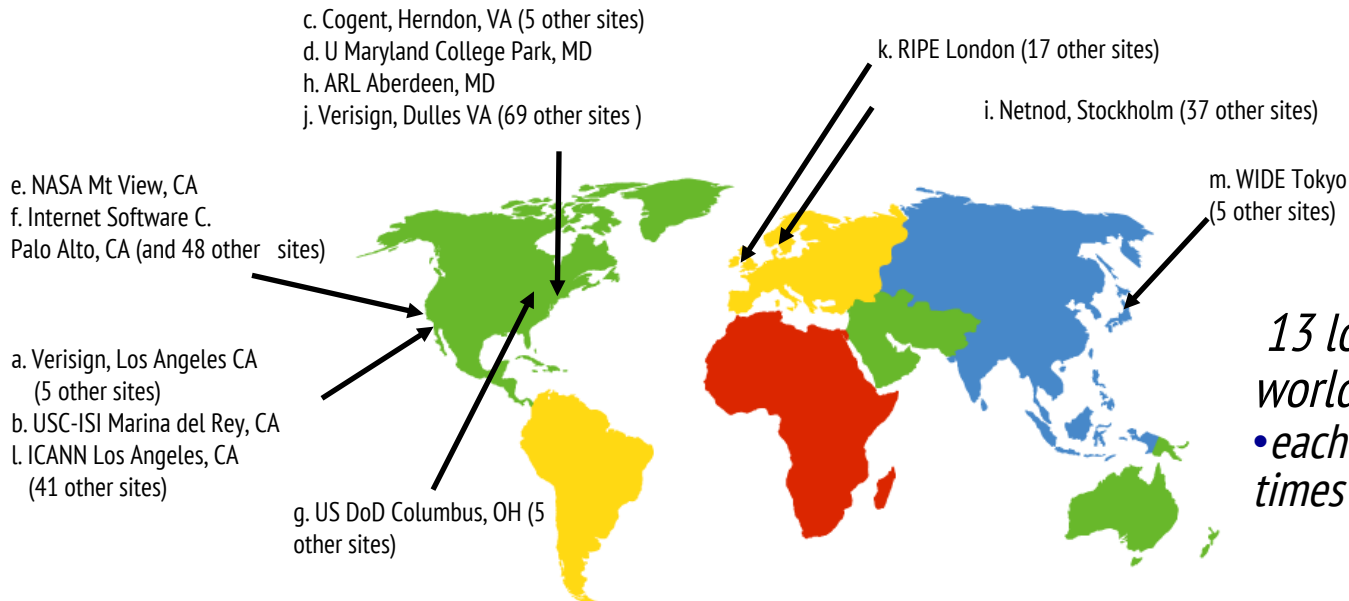DNS servers

poly.edu
DNS servers

umass.edu
DNS servers

client wants IP for www.amazon.com; 1st approximation:

➢ client queries root server to find com DNS server

➢ client queries .com DNS server to get amazon.com DNS server

➢ client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: root name servers

➢ contacted by local name server that can not resolve name

➢ root name server:

   ▪ contacts authoritative name server if name mapping not known

   ▪ gets mapping

   ▪ returns mapping to local name server

c. Cogent, Herndon, VA (5 other sites)
d. U Maryland College Park, MD
h. ARL Aberdeen, MD
j. Verisign, Dulles VA (69 other sites)

k. RIPE London (17 other sites)

i. Netnod, Stockholm (37 other sites)

e. NASA Mt View, CA
f. Internet Software C.
Palo Alto, CA (and 48 other   sites)

m. WIDE Tokyo
(5 other sites)

a. Verisign, Los Angeles CA
   (5 other sites)
b. USC-ISI Marina del Rey, CA
l. ICANN Los Angeles, CA
   (41 other sites)

g. US DoD Columbus, OH (5 other sites)

13 logical root name "servers" worldwide
• *each "server" replicated many times*

UNIVERSITY AT ALBANY
State University of New York

84

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

UNIVERSITY AT ALBANY
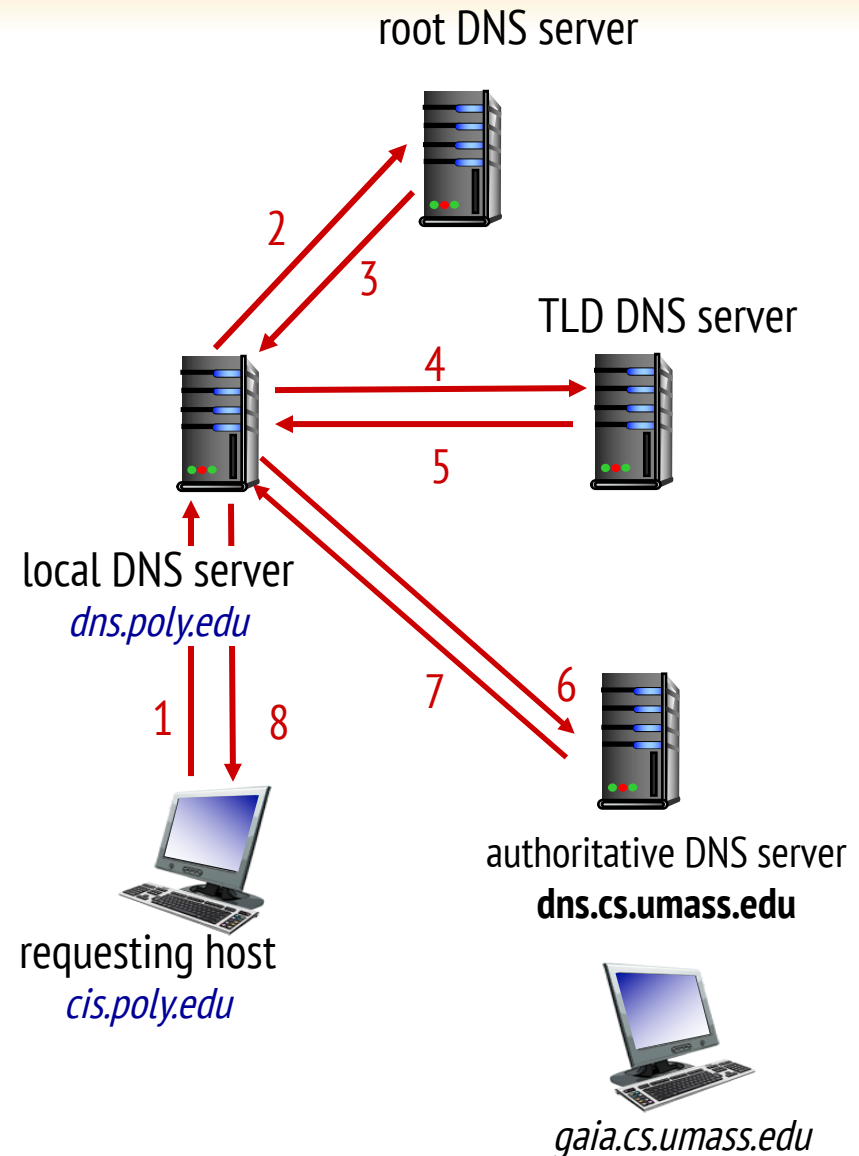State University of New York

Local DNS name server

➢ does not strictly belong to hierarchy

➢ each ISP (residential ISP, company, university) has one
  - also called "default name server"

➢ when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

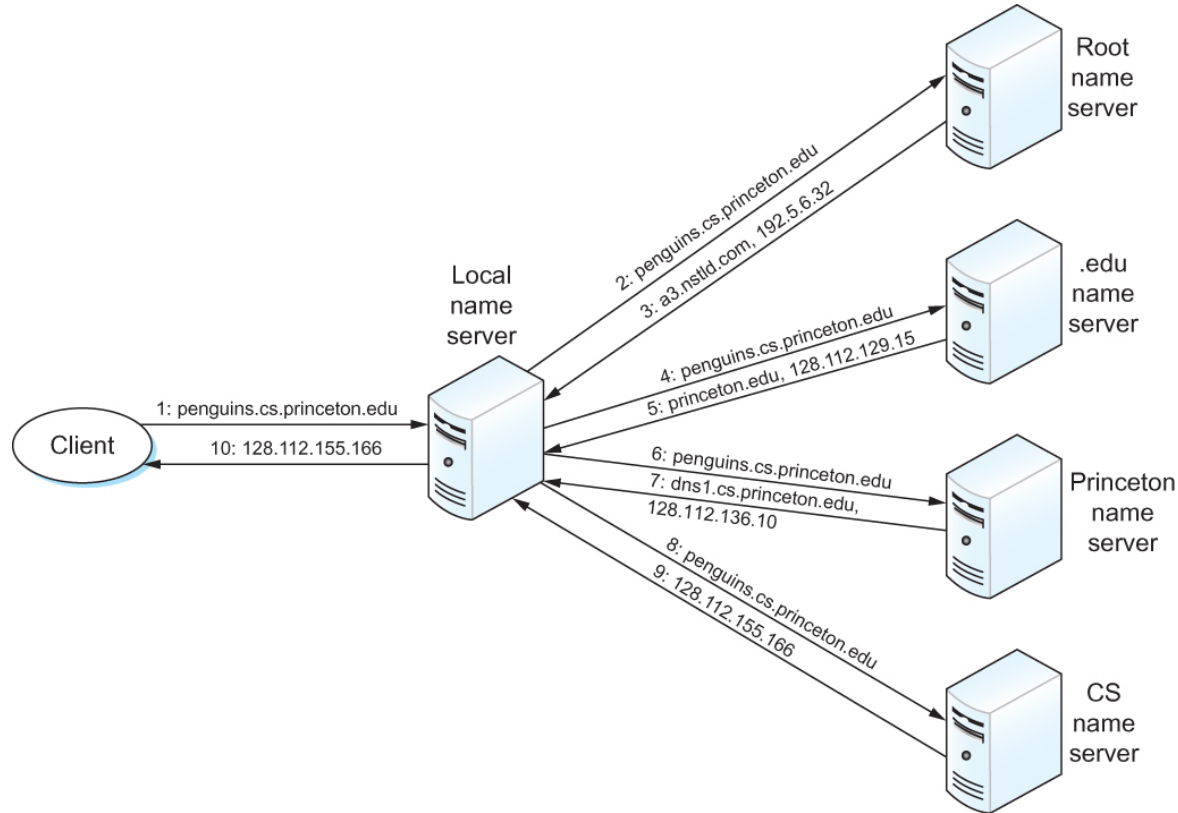➤ host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

root DNS server

TLD DNS server

2

3

4

5

local DNS server
dns.poly.edu

1 8

7 6

authoritative DNS server
dns.cs.umass.edu

requesting host
cis.poly.edu

gaia.cs.umass.edu

Infrastructure Services

➢ Name Resolution
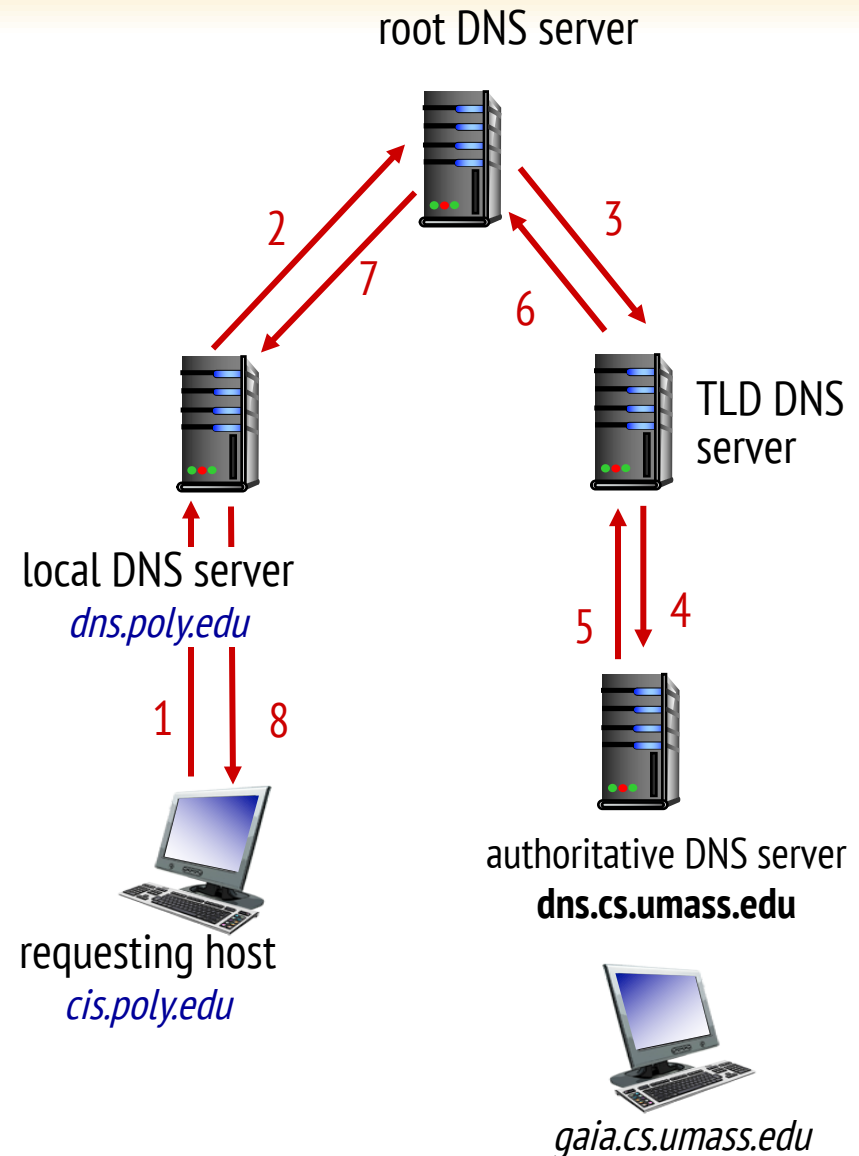

Name resolution in practice, where the numbers 1–10 show the sequence of steps in the process.

DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?

root DNS server

2
7
3
6

local DNS server
dns.poly.edu

TLD DNS server

5   4

1   8

authoritative DNS server
dns.cs.umass.edu

requesting host
cis.poly.edu

gaia.cs.umass.edu

UNIVERSITY AT ALBANY
State University of New York

DNS: caching, updating records

➢ once (any) name server learns mapping, it *caches* mapping

- cache entries timeout (disappear) after some time (TTL)
- TLD servers typically cached in local name servers
  - thus root name servers not often visited

➢ cached entries may be *out-of-date* (best effort name-to-address translation!)

- if name host changes IP address, may not be known Internet-wide until all TTLs expire

➢ update/notify mechanisms proposed IETF standard

- RFC 2136

DNS records

DNS: distributed database storing resource records (RR)

> RR format: `(name, value, type, ttl)`

type=A
- **name** is hostname
- **value** is IP address

type=NS
name is domain (e.g., foo.com)

value is hostname of authoritative name server for this domain

type=CNAME
- **name** is alias name for some "canonical" (the real) name
- **www.ibm.com** is really **servereast.backup2.ibm.com**
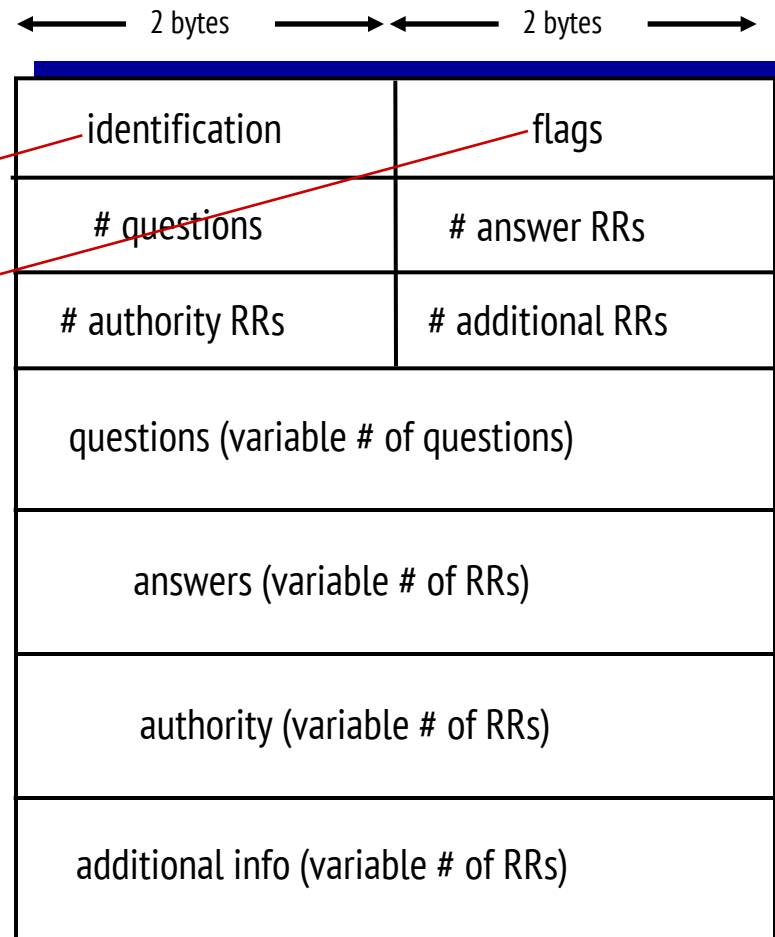- **value** is canonical name

type=MX
- **value** is name of mailserver associated with **name**

UNIVERSITY AT ALBANY
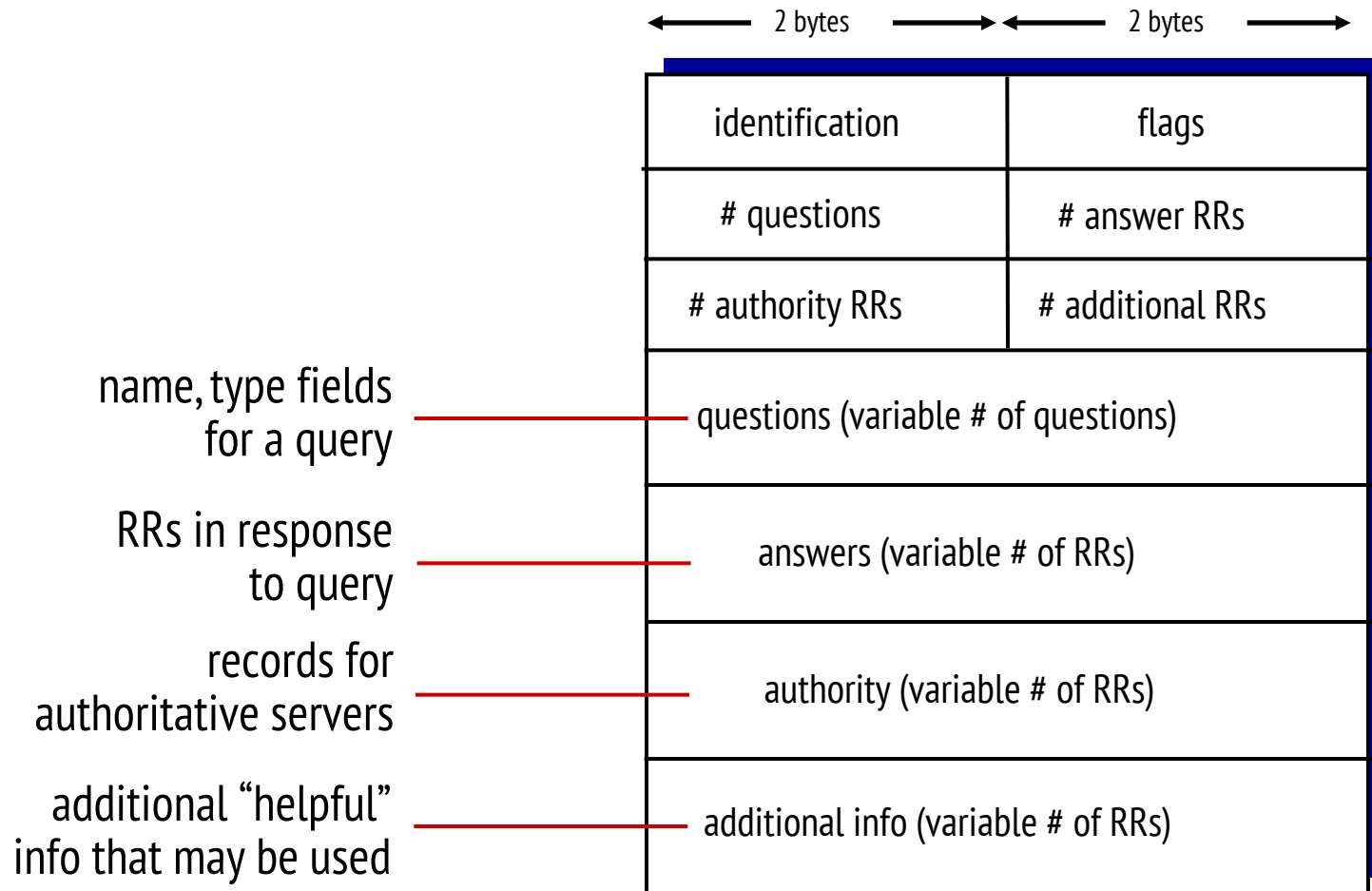State University of New York

DNS protocol, messages

➤ *query* and *reply* messages, both with same *message format*

message header

- identification: 16 bit # for query, reply to query uses same #

- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

← 2 bytes →	← 2 bytes →
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

DNS Protocol, messages

2 bytes ← → 2 bytes

identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs

name, type fields for a query ──── questions (variable # of questions)

RRs in response to query ──── answers (variable # of RRs)

records for authoritative servers ──── authority (variable # of RRs)

additional "helpful" info that may be used ──── additional info (variable # of RRs)

Inserting records into DNS

➤ example: new startup "Network Utopia"

➤ register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)

- ▪ provide names, IP addresses of authoritative name server (primary and secondary)
- ▪ registrar inserts two RRs into .com TLD server:

  (networkutopia.com, dns1.networkutopia.com, NS)

  (dns1.networkutopia.com, 212.212.212.1, A)

➤ create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com

UNIVERSITY AT ALBANY
State University of New York

Pure P2P architecture

➢ *no* always-on server

➢ arbitrary end systems directly communicate

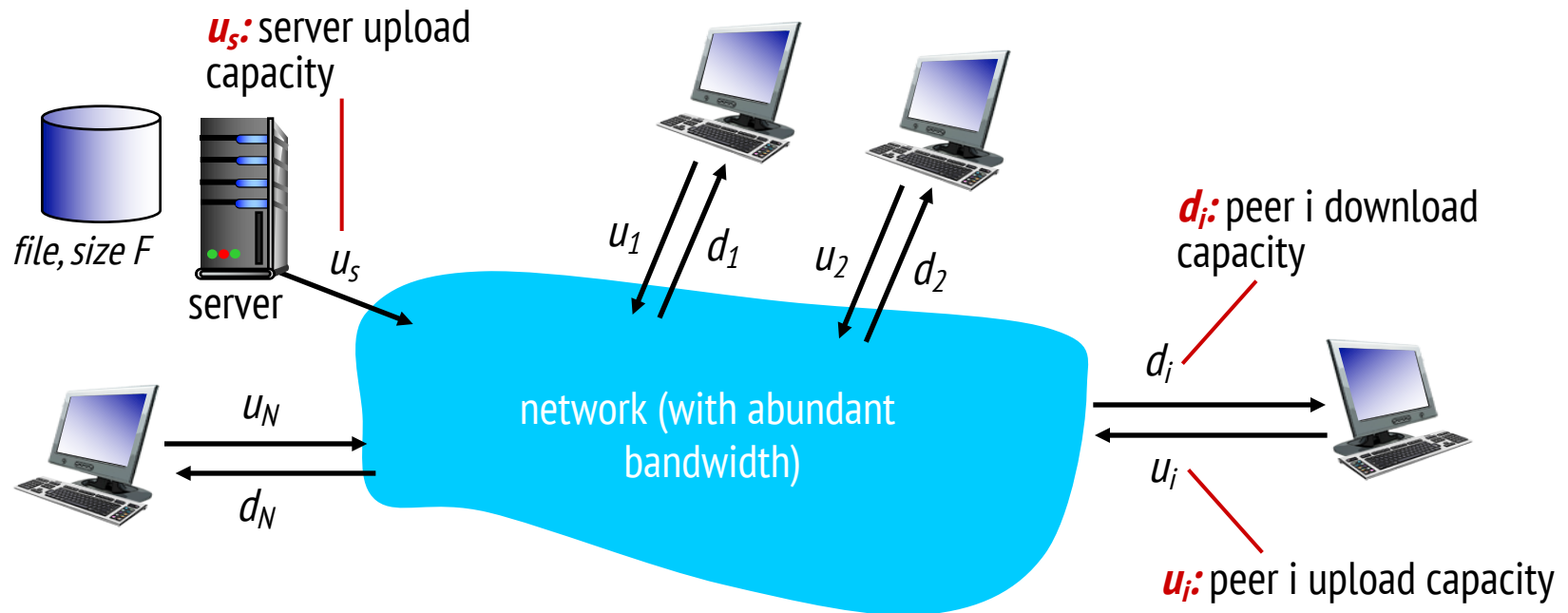➢ peers are intermittently connected and change IP addresses

examples:

- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)
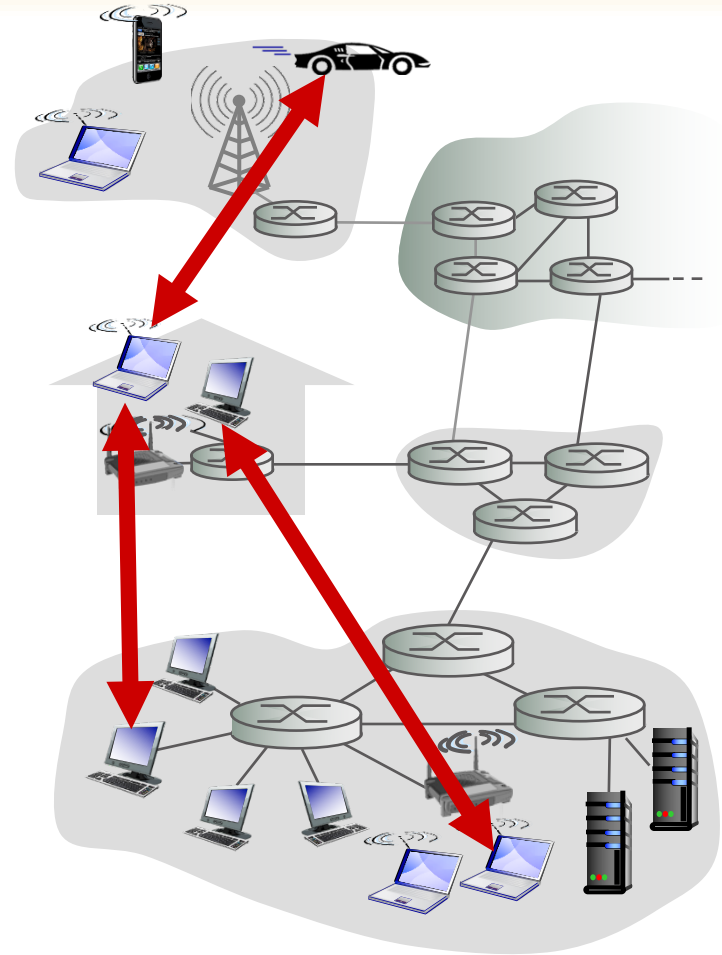
File distribution: client-server vs P2P

Question: how much time to distribute file (size *F*) from one server to *N* *peers*?

- peer upload/download capacity is limited resource

u_s: server upload capacity

file, size F

server

u_s

u_1 d_1 u_2 d_2

d_i: peer i download capacity

d_i

network (with abundant bandwidth)

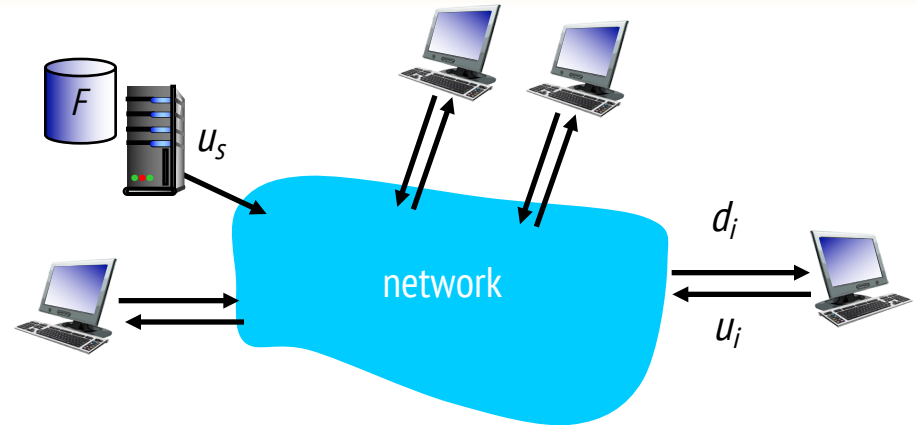u_N

d_N

u_i

u_i: peer i upload capacity

File distribution time: client-server

➤ *server transmission:* must sequentially send (upload) N file copies:

- time to send one copy: F/u_s
- time to send N copies: NF/u_s

➤ *client:* each client must download file copy

- ❑ d_{min} = min client download rate
- ❑ min client download time: F/d_{min}

time to distribute F to N clients using client-server approach

$$D_{c-s} > \overline{max}\{NF/u_s, F/d_{min}\}$$

increases linearly in N

File distribution time: P2P

- ➤ **server transmission**: must upload at least one copy
  - time to send one copy: F/u_s

- ➤ **client**: each client must download file copy
  - min client download time: F/d_{min}

- ➤ **clients**: as aggregate must download NF bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$
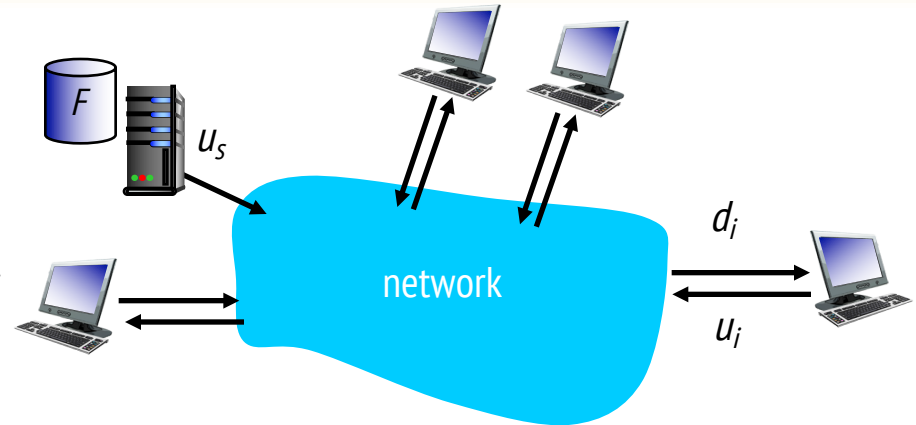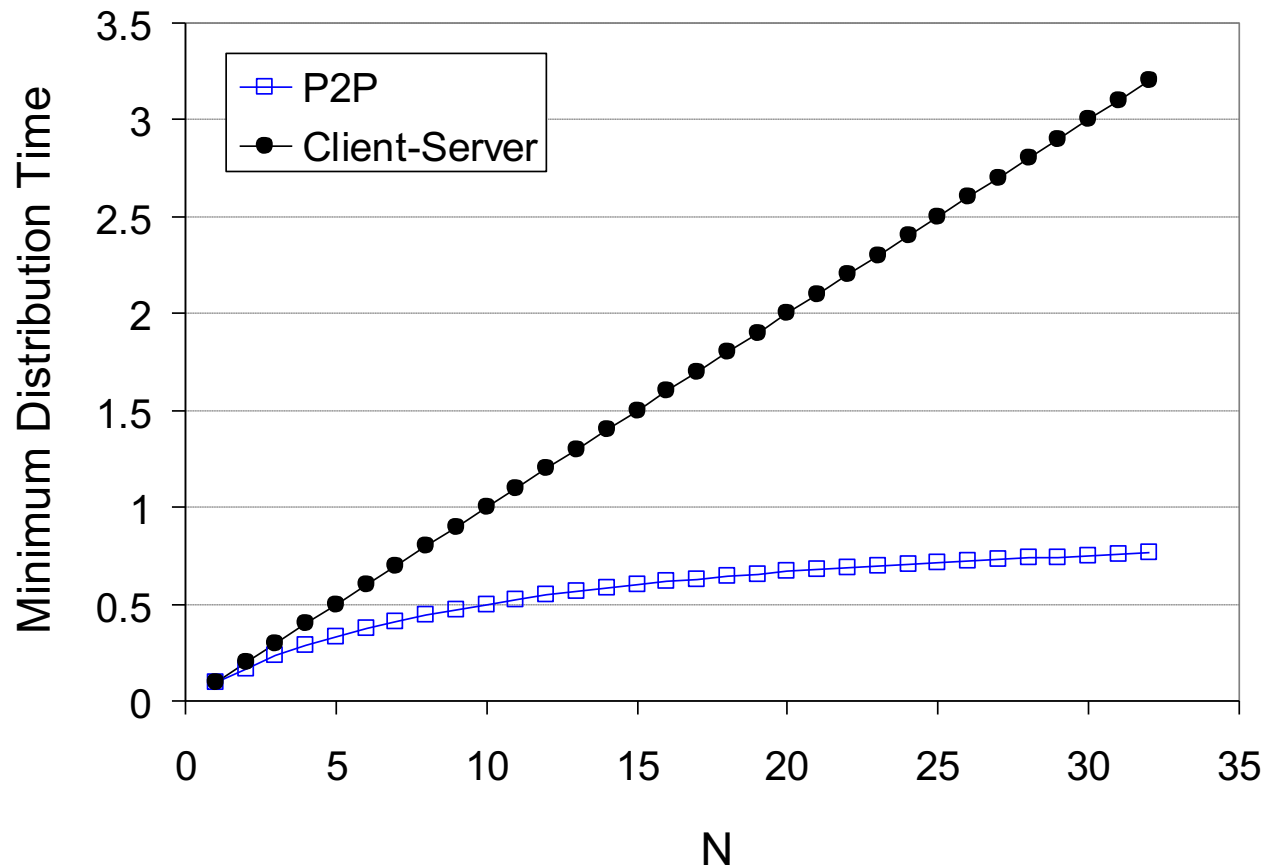

time to distribute F to N clients using P2P approach

$$D_{P2P} >= max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geqslant u_s$
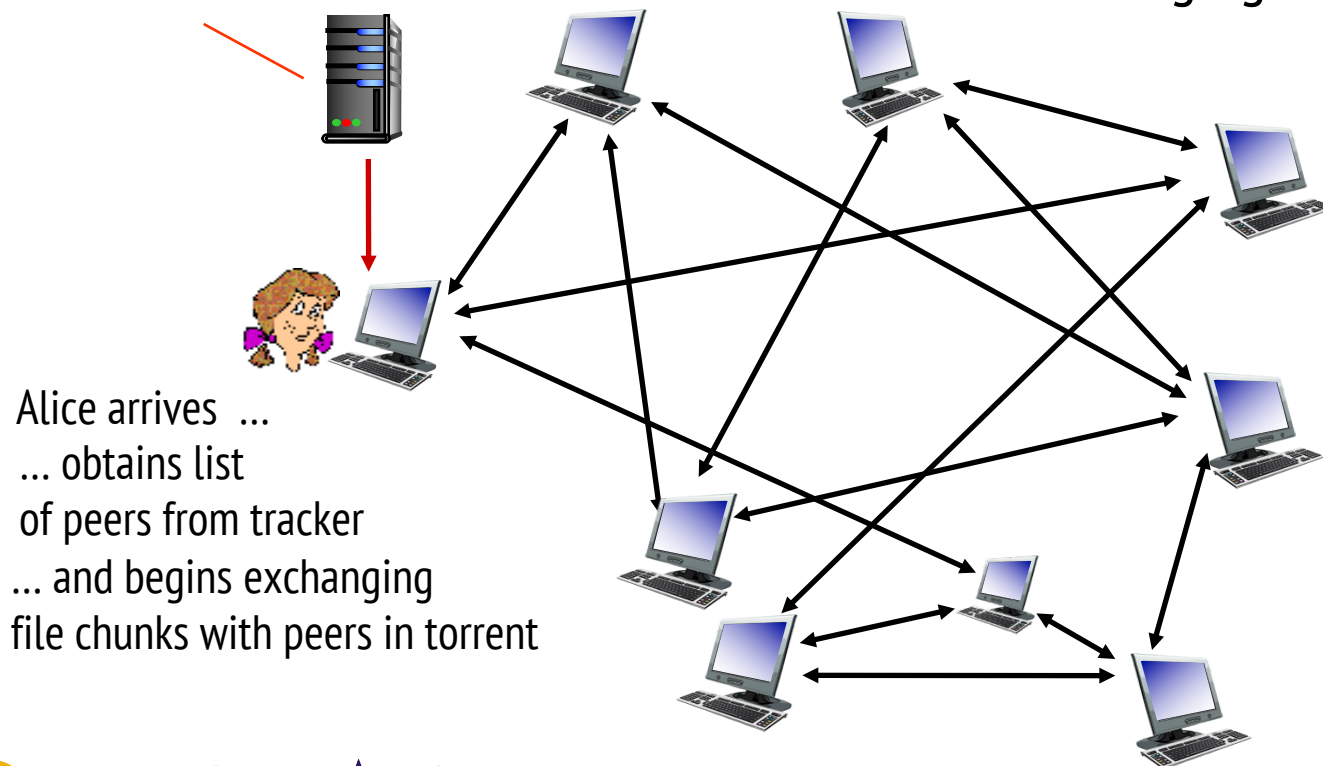
P2P file distribution: BitTorrent protocol

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file

Alice arrives ...
... obtains list
of peers from tracker
... and begins exchanging
file chunks with peers in torrent

P2P file distribution: BitTorrent

➢ peer joining torrent:
   ▪ has no chunks, but will accumulate them over time from other peers
   ▪ registers with tracker to get list of peers, connects to subset of peers ("neighbors")

➢ while downloading, peer uploads chunks to other peers

➢ peer may change peers with whom it exchanges chunks

➢ churn: peers may come and go

➢ once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent
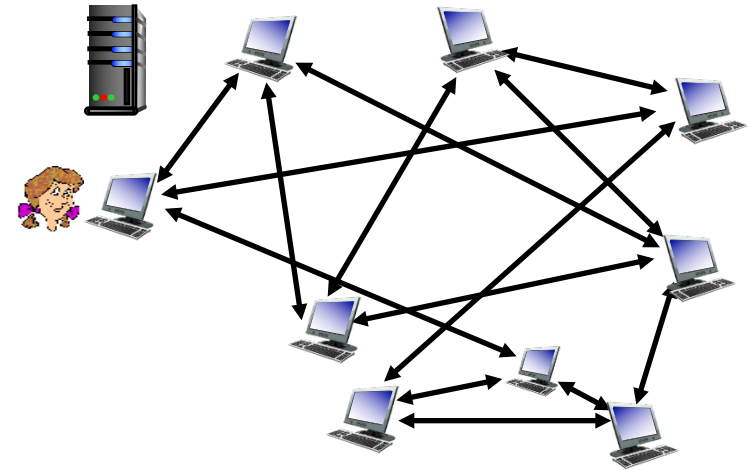
# BitTorrent: requesting, sending file chunks

requesting chunks:

- at any given time, different peers have different subsets of file chunks

- periodically, Alice asks each peer for list of chunks that they have

- Alice requests missing chunks from peers, rarest first

sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every 10 secs

- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4
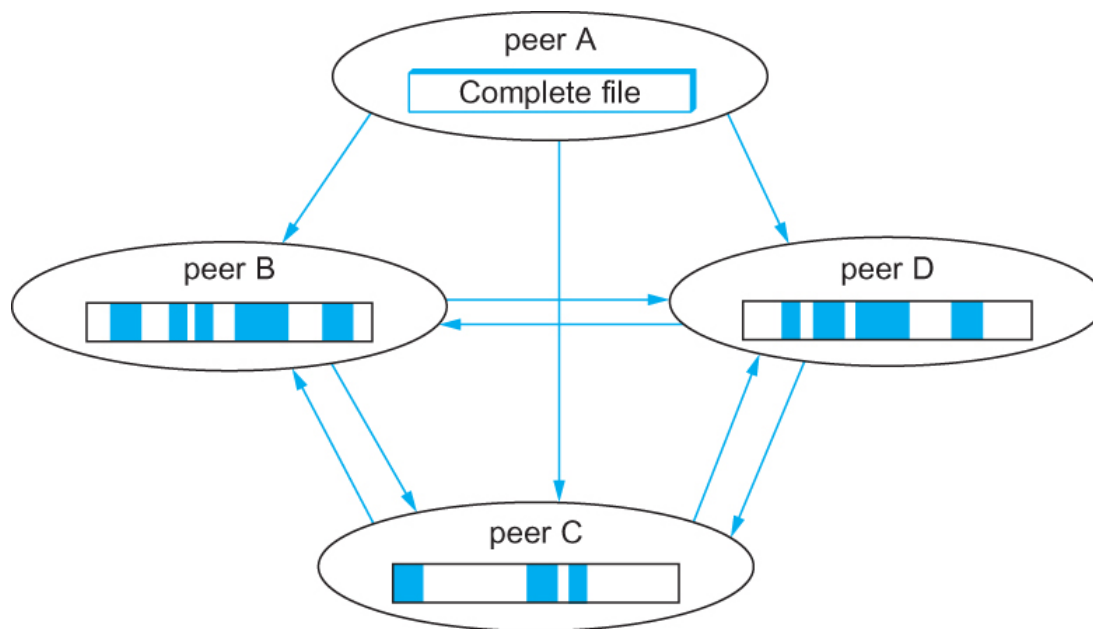
BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers

higher upload rate: find better trading partners, get file faster !
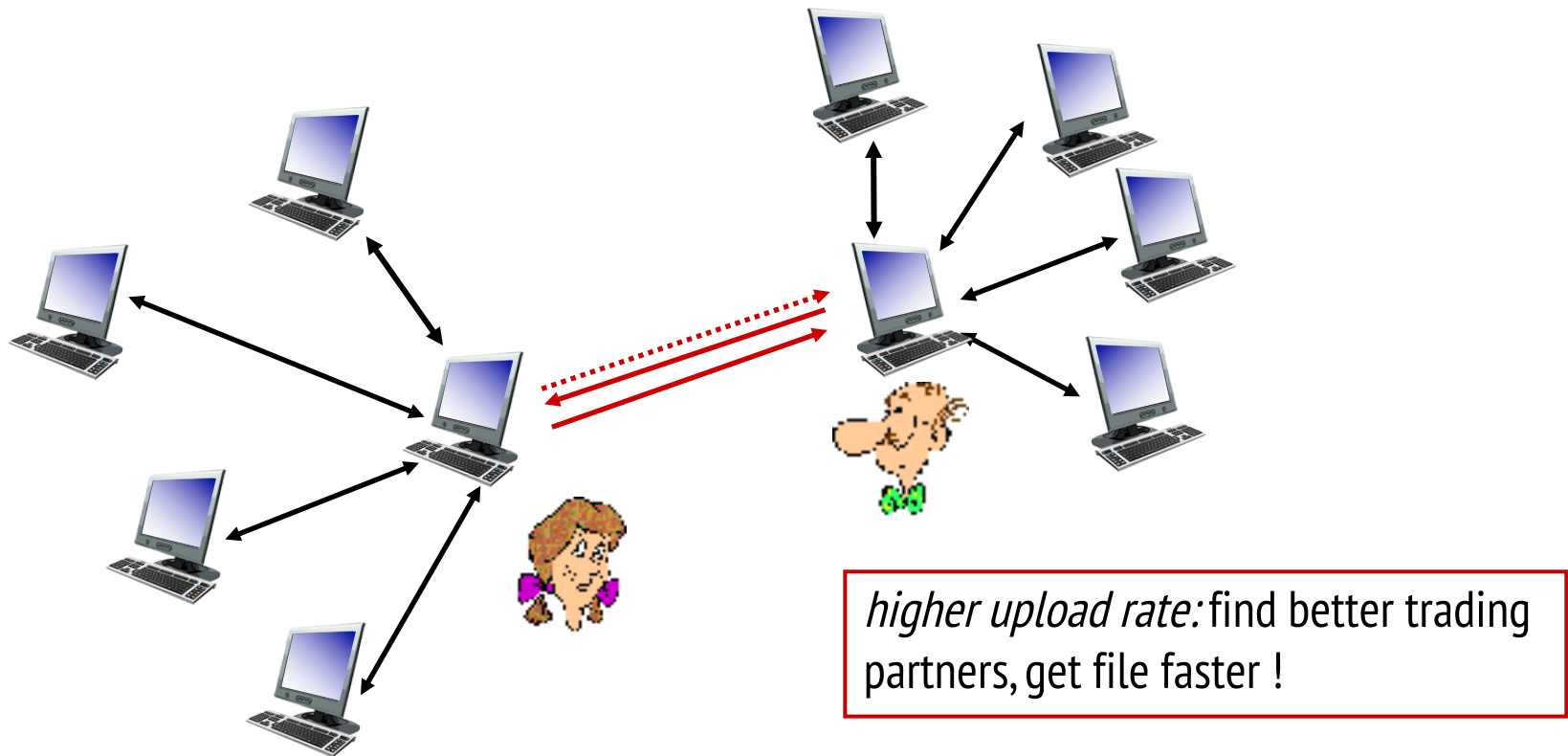
BitTorrent: another aspect

Peers in a BitTorrent swarm download from other peers that may not yet have the complete file

Video Streaming and CDNs: context

➢ video traffic: major consumer of Internet bandwidth

  ▪ Netflix, YouTube: 37%, 16% of downstream residential traffic

  ▪ ~1B YouTube users, ~75M Netflix users

➢ challenge: scale - how to reach ~1B users?

  ▪ single mega-video server won't work (why?)

➢ challenge: heterogeneity

  ▪ different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)

➢ *solution*: distributed, application-level infrastructure

Streaming stored video:

simple scenario:

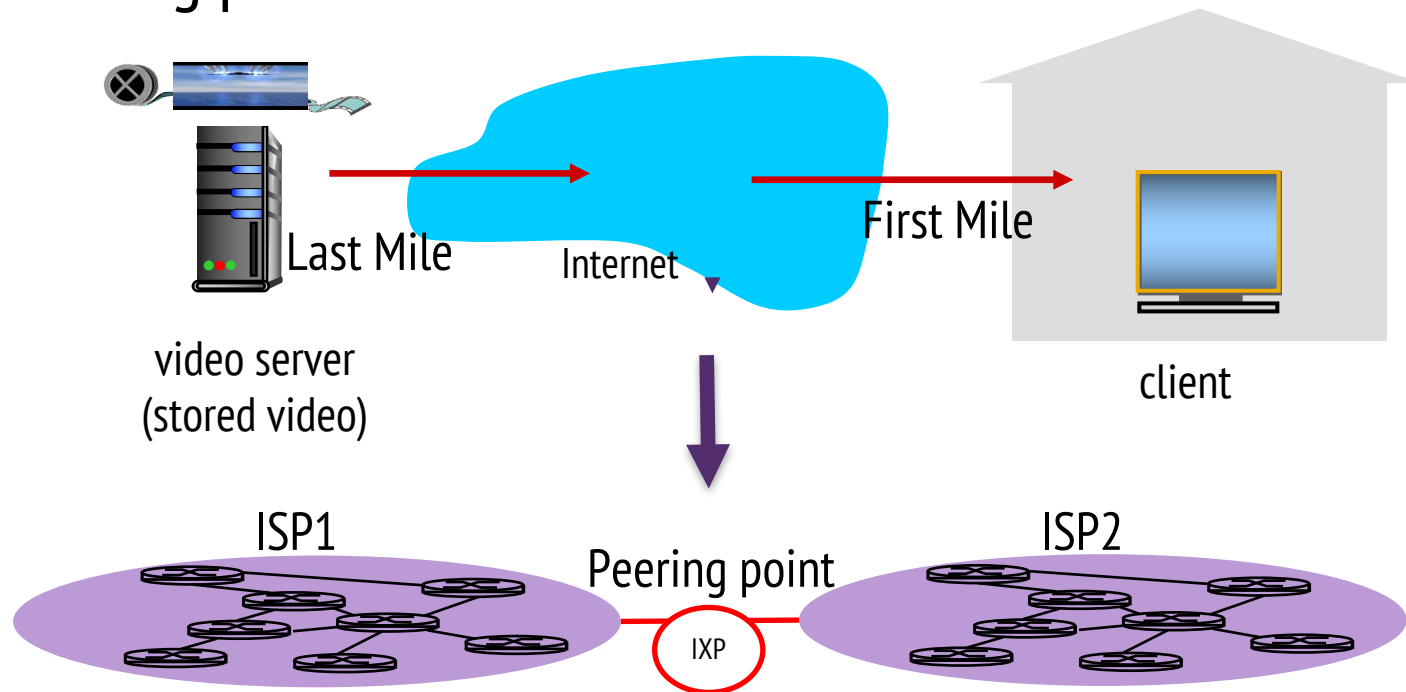video server
(stored video)

Internet

client
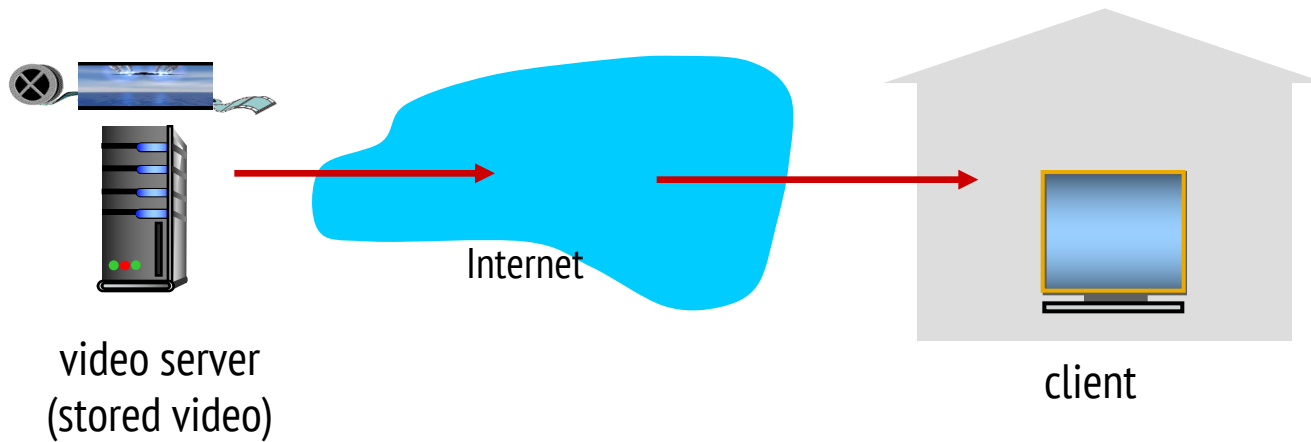
Bottlenecks in the system

➢ The first mile

➢ The last mile

➢ The server itself

➢ Peering points

Last Mile

Internet

First Mile

video server
(stored video)

client

ISP1

Peering point

IXP

ISP2

Streaming multimedia: DASH

➤ *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP

➤ *server:*

- divides video file into multiple chunks
- each chunk stored, encoded at different rates
- *manifest file:* provides URLs for different chunks

➤ *client:*

- periodically measures server-to-client bandwidth
- consulting manifest, requests one chunk at a time
  - ○ chooses maximum coding rate sustainable given current bandwidth
  - ○ can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

➢ *DASH: Dynamic, Adaptive Streaming over HTTP*

➢ *"intelligence"* at client: client determines

- *when* to request chunk (so that buffer starvation, or overflow does not occur)

- *what encoding rate* to request (higher quality when more bandwidth available)

- *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

Content distribution networks

➢ *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

➢ *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*

- *enter deep:* push CDN servers deep into many access networks
  - o close to users
  - o used by Akamai, 1700 locations
- *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
  - o used by Limelight

UNIVERSITY AT ALBANY
State University of New York

Content Distribution Networks (CDNs)

➢ subscriber requests content from CDN

➢ CDN: stores copies of content at CDN nodes

- ▪ e.g. Netflix stores copies of MadMen
- ▪ directed to nearby copy, retrieves content
- ▪ may choose different copy if network path congested
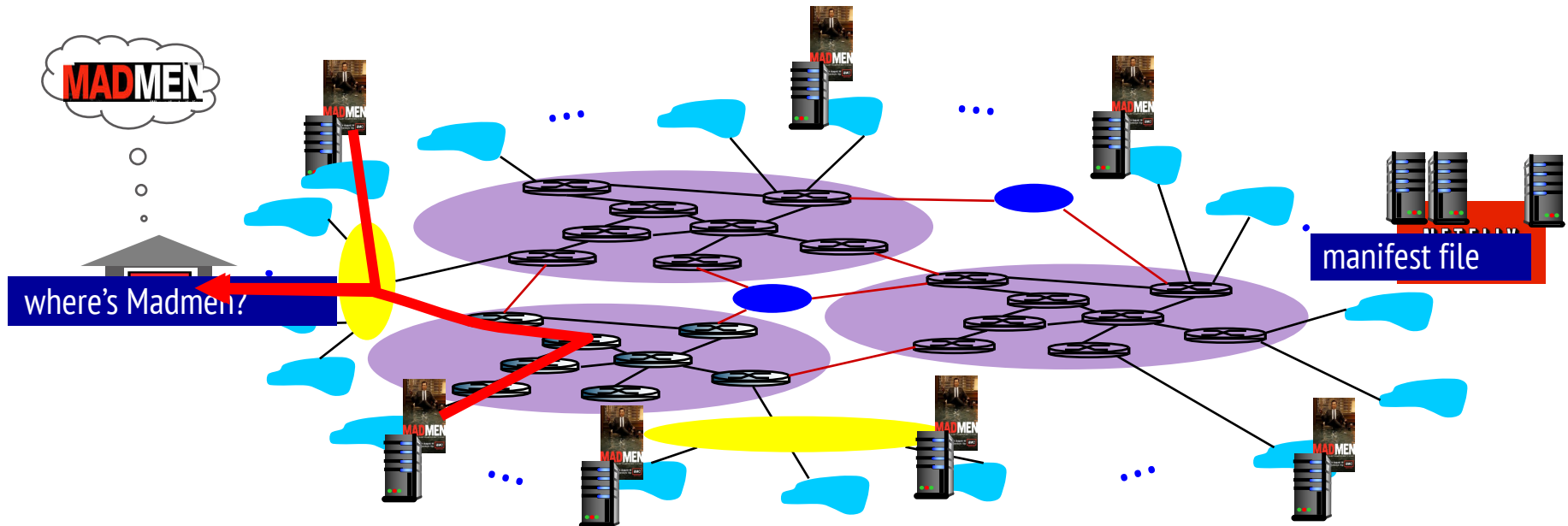
CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B23V
- video stored in CDN at http://KingCDN.com/NetC6y&B23V

1. Bob gets URL for video
http://netcinema.com/6Y7B23V
from netcinema.com web page

2. resolve http://netcinema.com/6Y7B23V
via Bob's local DNS

6. request video from
KINGCDN server,
streamed via HTTP

Bob's
local DNS
server

netcinema.com

3. netcinema's DNS returns URL
http://KingCDN.com/NetC6y&B23V

4&5. Resolve
http://KingCDN.com/NetC6y&B23
via KingCDN's authoritative DNS,
which returns IP address of KingCDN
server with video

netcinema's
authoratative DNS

KingCDN.com

KingCDN
authoritative DNS

Case study: Netflix

Amazon cloud

upload copies of multiple versions of video to CDN servers

CDN server

CDN server

CDN server

Netflix registration, accounting servers

3. Manifest file returned for requested video

2. Bob browses Netflix video

2

3

1

1. Bob manages Netflix account

4. DASH streaming

UNIVERSITY AT ALBANY
State University of New York

CDN: In short

- ➢ The idea of a CDN is to geographically distribute a collection of *server surrogates* that cache pages normally maintained in some set of *backend servers*
    - ■ Akamai operates what is probably the best-known CDN.

- ➢ Thus, rather than have millions of users wait forever to contact www.cnn.com when a big news story breaks—such a situation is known as a *flash crowd—it is possible to spread this load* across many servers.

- ➢ Moreover, rather than having to traverse multiple ISPs to reach www.cnn.com, if these surrogate servers happen to be spread across all the backbone ISPs, then it should be possible to reach one without having to cross a peering point.

CDN Components
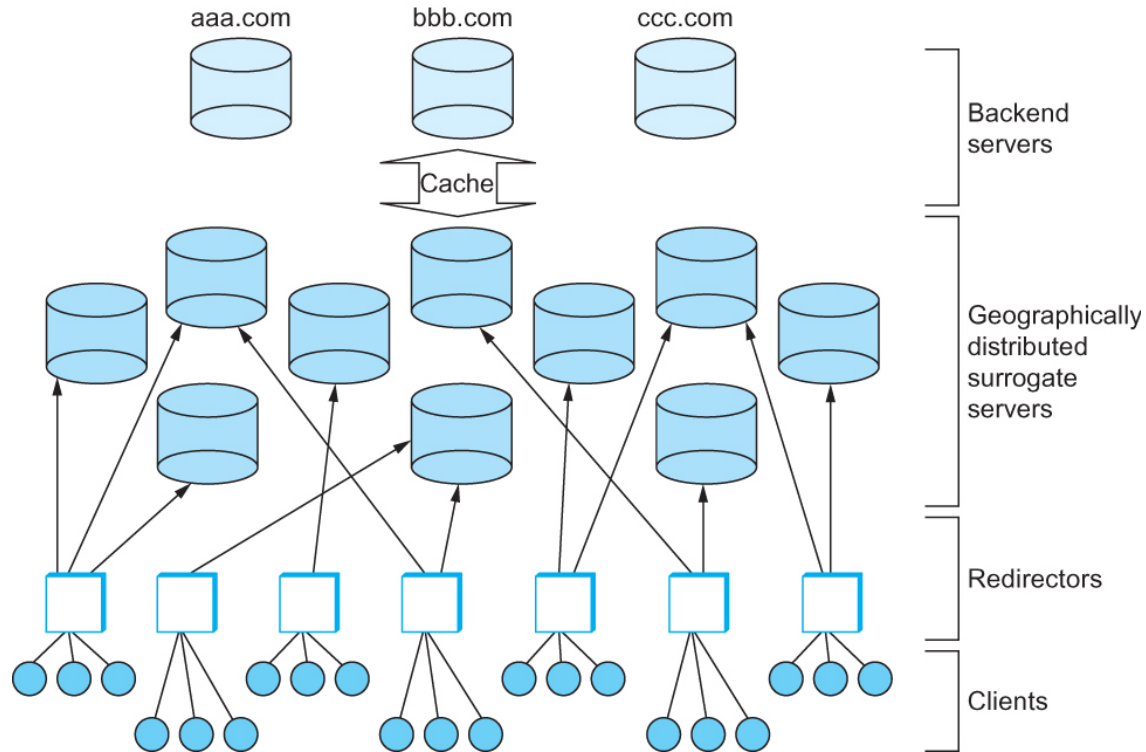
Components in a Content Distribution Network (CDN).

Summary

➤ We have discussed some of the popular applications in the Internet
- Electronic mail, World Wide Web

➤ We have discussed multimedia applications

➤ We have discussed infrastructure services
- Domain Name Services (DNS)

➤ We have discussed overlay networks

➤ We have discussed content distribution networks

Install

➢ Opnet Network Simulator

➢ Welcome to bring laptop to class

What are the restrictions for Riverbed Modeler Academic Edition 17.5?

Categories: Riverbed Modeler (OPNET Modeler), Riverbed Modeler (OPNET Modeler Wireless Suite), Riverbed Modeler (OPNET Modeler Wireless Suite for Defense)
Solution Number: S24443

ISSUE

Is there any difference between Riverbed Modeler Academic Edition and Riverbed Modeler? What are the feature limitations in the Academic Version of the software?

SOLUTION

Riverbed Modeler Academic Edition 17.5 is a stripped-down version of Modeler intended for universities using simulation software for networking courses. This academic release features the following highlights:

- Free for all users (registration required)
- Replaces IT Guru Academic Edition 9.1
- Based on Riverbed Modeler version 17.5 PL6
- Built-in Wireless capability
- 6-month renewable license
- Community support for users
- Supported on Windows only

Register with albany.edu address

riverbed® support

f   🐦   g+   [Search]

SOFTWARE & DOCUMENTATION   **KNOWLEDGE BASE**   **LICENSES**   **MY RIVERBED**

Academic Edition Account Signup

Modeler
Academic Edition

Welcome to the Riverbed Modeler Academic Edition site. Using the form below, please register to download the Riverbed Modeler Academic Edition software.

■ Required fields

Field	
First Name ■	
Last Name ■	
Address 1	
Address 2	
City	
State	
Country ■	
Postal Code	
Email Address ■	
Re-enter Email ■	
Phone	
Fax	

Please enter the name of your employer,
or type N/A if not applicable:

Employer ■

Please tell us about the academic program you are enrolled in,
or type N/A if not applicable:

Name of Academic ■