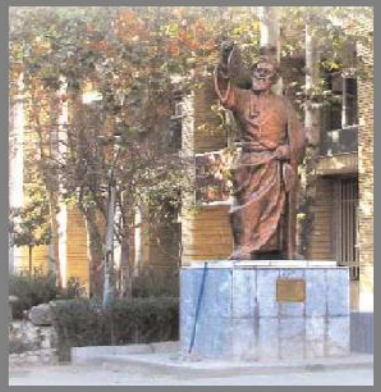
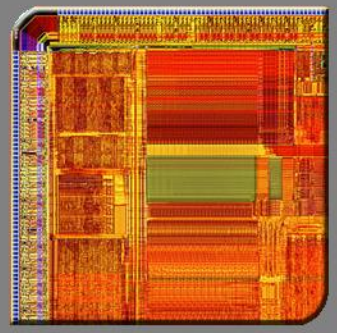
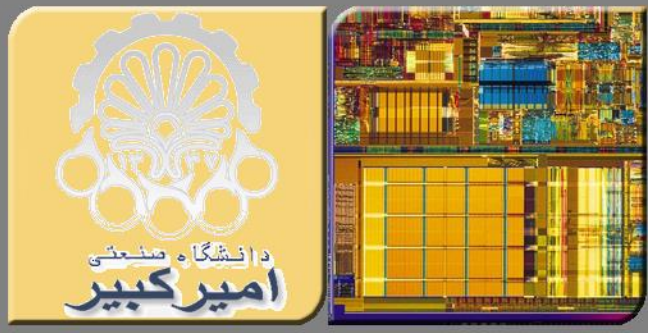
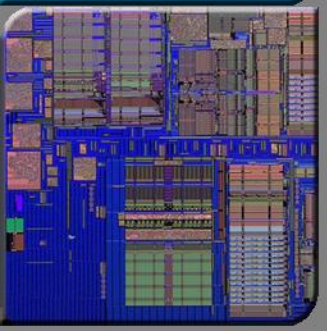






شبکه های کامپیوتری

فصل دوم: لایه کاربرد



مدرس: دکتر محمد حسن شیرعلی شهرضا
دانشگاه صنعتی امیرکبیر

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

بخش های فصل دوم: لایه کاربرد (نیمه اول فصل)

۲-۱ اصول برنامه های کاربردی شبکه

2.1 principles of network applications

۲-۲ وب و HTTP

2.2 Web and HTTP

۲-۳ انتقال فایل - FTP

2.3 FTP

۲-۴ پست الکترونیک در اینترنت

2.4 electronic mail

- SMTP, POP3, IMAP

فصل دوم: لایه کاربرد (اهداف فصل)

اهداف این فصل (our goals):

- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- creating network applications
  - socket API

چند اپلیکیشن کاربردی شبکه

- 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
- …
- …

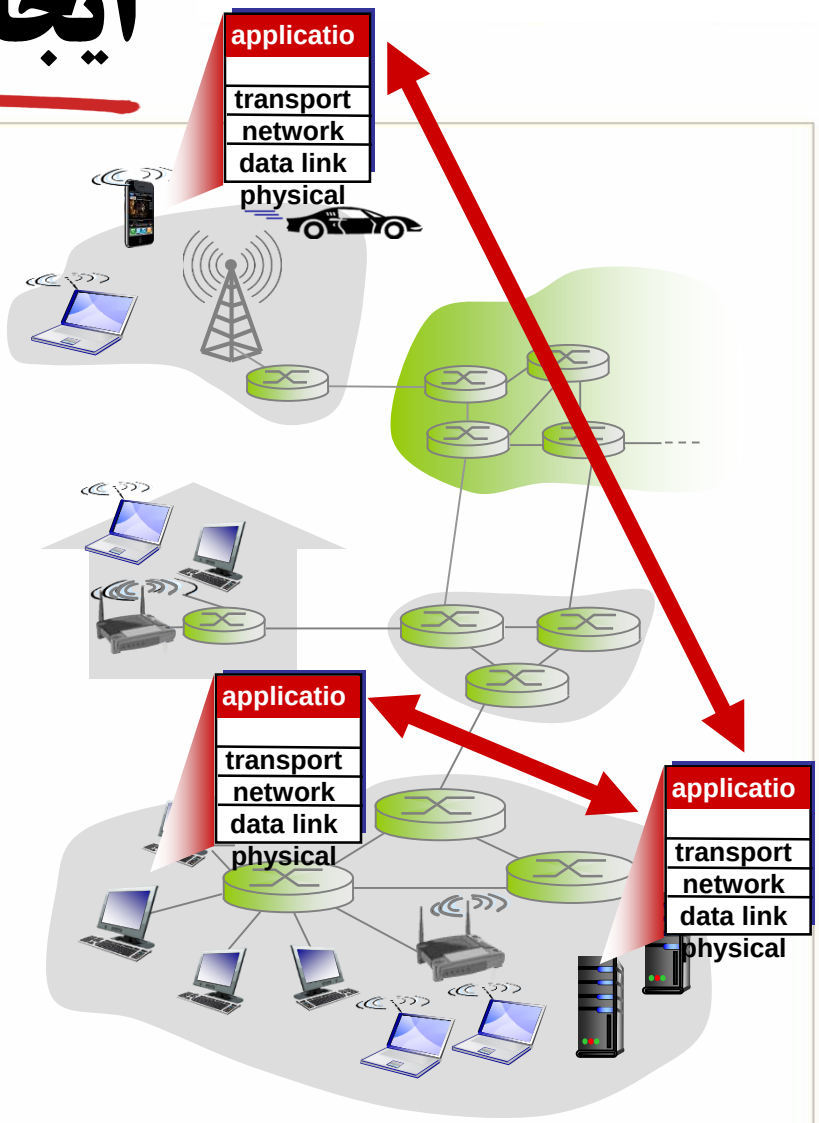
ایجاد یک اپلیکیشن کاربردی

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



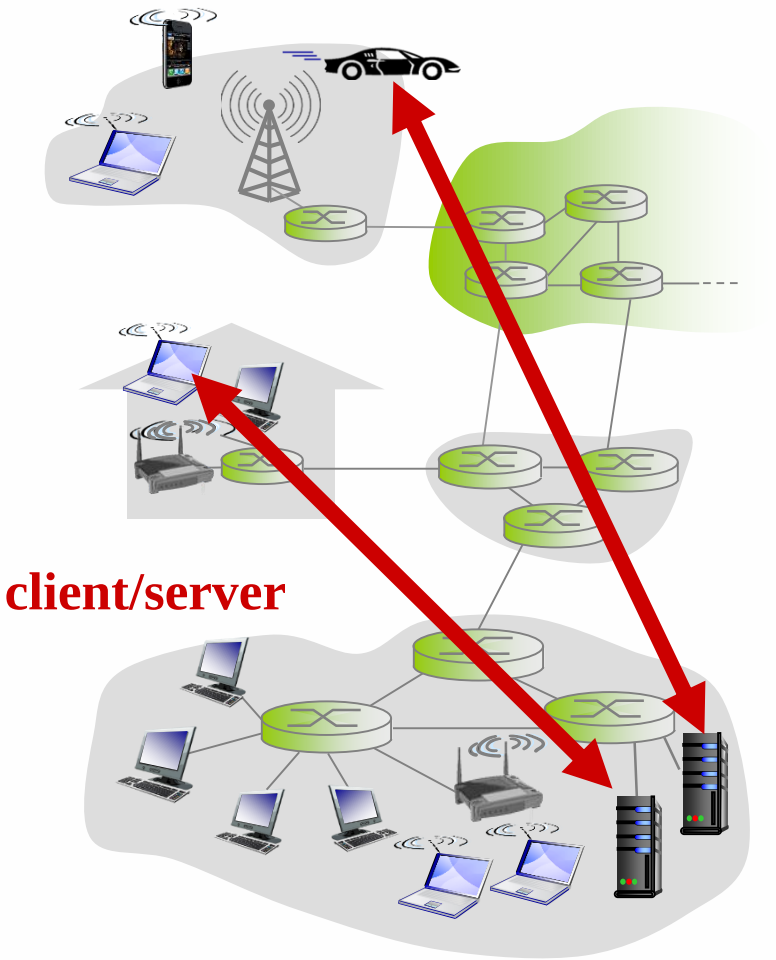
معماری اپلیکیشن (برنامه کاربردی)

ساختارهای ممکن برای اپلیکیشن:

possible structure of applications:

- روش مشتری- سرویس دهنده client-server
- روش نقطه به نقطه peer-to-peer (P2P)

معماری مشتری– سرویس دهنده



سرویس دهنده (server):

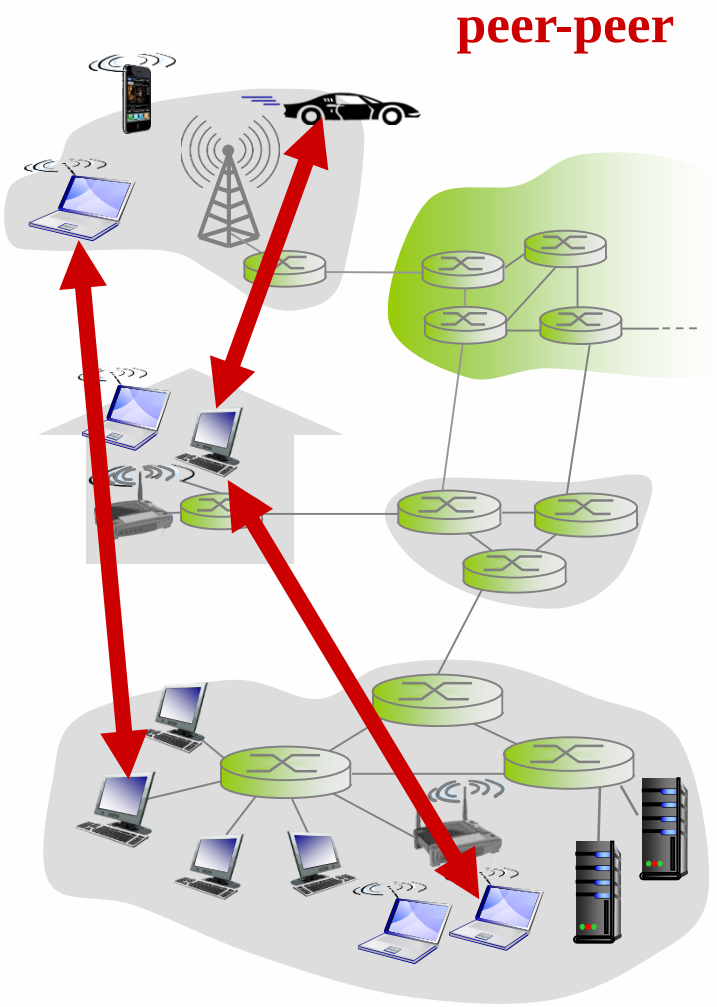
- always-on host
- permanent IP address
- data centers for scaling

مشتری یا کلاینت (clients):

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

معماری نقطه به نقطه

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management



ارتباط بین فرآیندها

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
- processes in different hosts communicate by exchanging messages

clients, servers

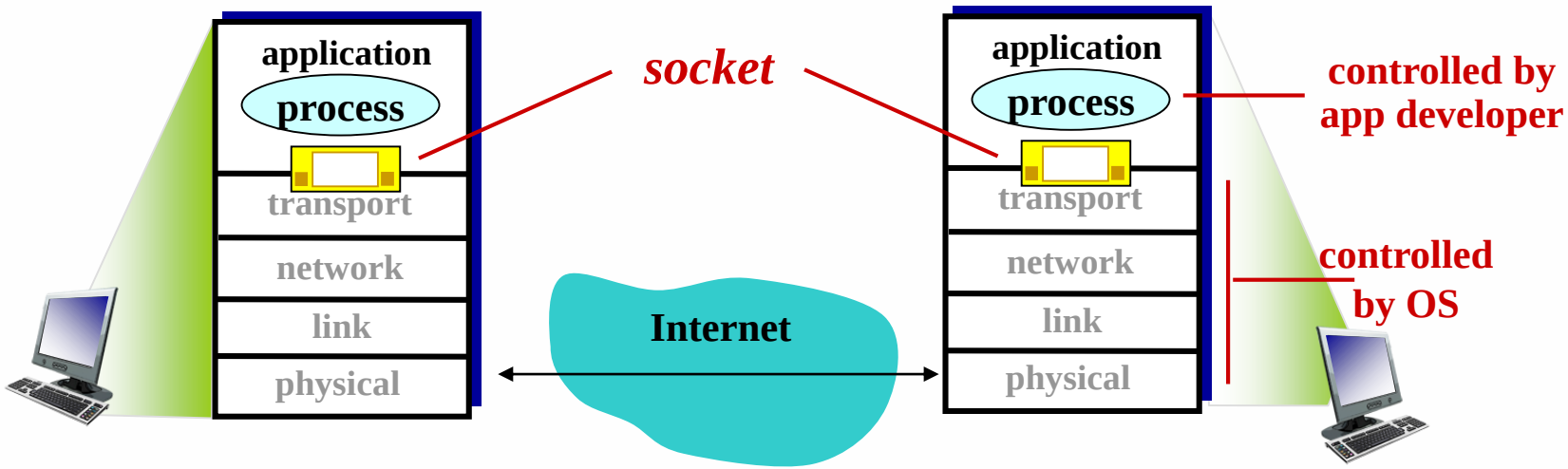
client process: process that initiates communication

server process: process that waits to be contacted

- **aside: applications with P2P architectures have client processes & server processes**

سوکت ها

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



آدرس دهی فرآیندها

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
  - ***A:*** **no, *many* processes can be running on same host**
- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80
- more shortly…

تعریف پروتکل برای لایه کاربرد

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

open protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

یک برنامه کاربردی چه سرویس‌هایی از لایه انتقال نیاز دارد؟

یکپارچگی یا تمامیت داده data integrity

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

زمانبندی timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

بازده بالا throughput

- **some apps (e.g., multimedia) require minimum amount of throughput to be “effective”**
- **other apps (“elastic apps”) make use of whatever throughput they get**

امنیت security

- **encryption, data integrity, …**

سرویس‌های لایه انتقال برای چند برنامه کاربردی مرسوم

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100’s msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100’s msec
text messaging	no loss	elastic	yes and no

سرویس‌های لایه انتقال در پروتکل‌های اینترنت

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, orconnection setup,
- Q: why bother? Why is there a UDP?

تعدادی از برنامه‌های کاربردی اینترنت: پروتکل برنامه کاربردی و پروتکل لایه انتقال آن

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

امن سازی TCP

TCP & UDP

- no encryption
- cleartext passwds sent into socket traverse Internet in cleartext

SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- Apps use SSL libraries, which “talk” to TCP

SSL socket API

- cleartext passwds sent into socket traverse Internet encrypted

بخش های فصل دوم: لایه کاربرد

۲-۱ اصول برنامه های کاربردی شبکه

2.1 principles of network applications

۲-۲ وب و HTTP

2.2 Web and HTTP

۲-۳ انتقال فایل - FTP

2.3 FTP

۲-۴ پست الکترونیک در اینترنت

2.4 electronic mail
- SMTP, POP3, IMAP

۲-۵ سامانه نام دامنه

2.5 DNS

۲-۶ کاربردهای نقطه به نقطه

2.6 P2P applications

۲-۷ برنامه نویسی سوکت

2.7 socket programming with UDP and TCP

وب و HTTP

First, a review…

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

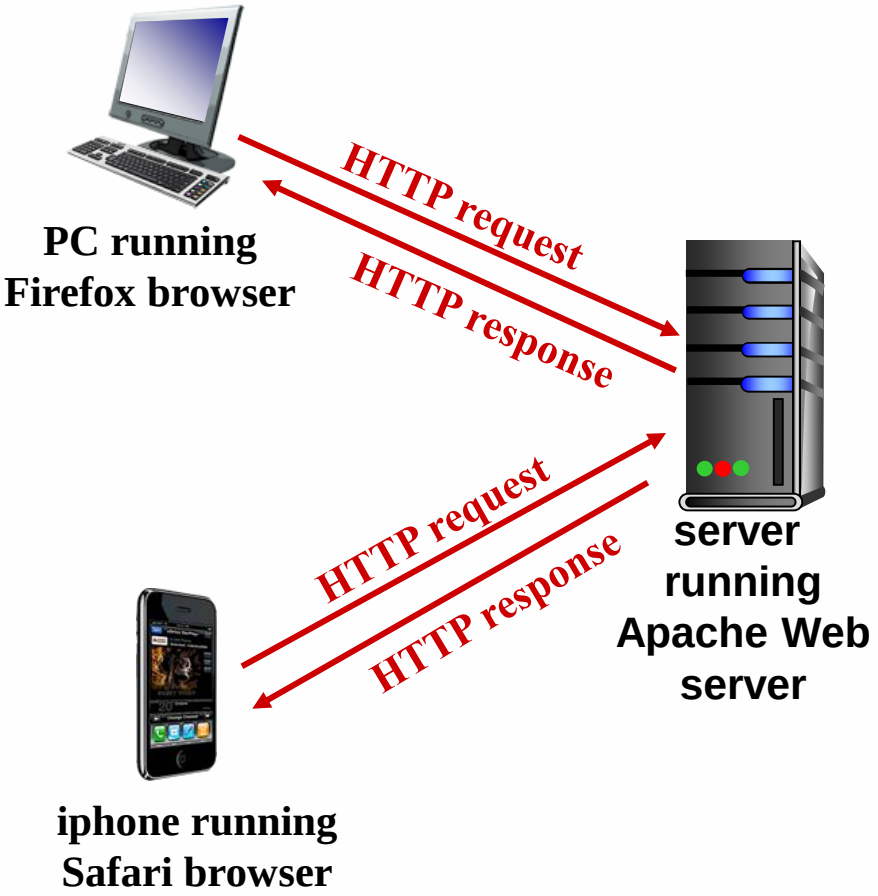
`www.someschool.edu/someDept/pic.gif`

host name **path name**

مرور کلی HTTP

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



مرور کلی HTTP (ادامه)

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

توضیح

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**

اتصالات HTTP

HTTP ناپایا (non-persistent)

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

HTTP پایا (persistent)

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

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

HTTP ناپایا (ادامه)

time

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

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

HTTP ناپایا: زمان پاسخ دهی

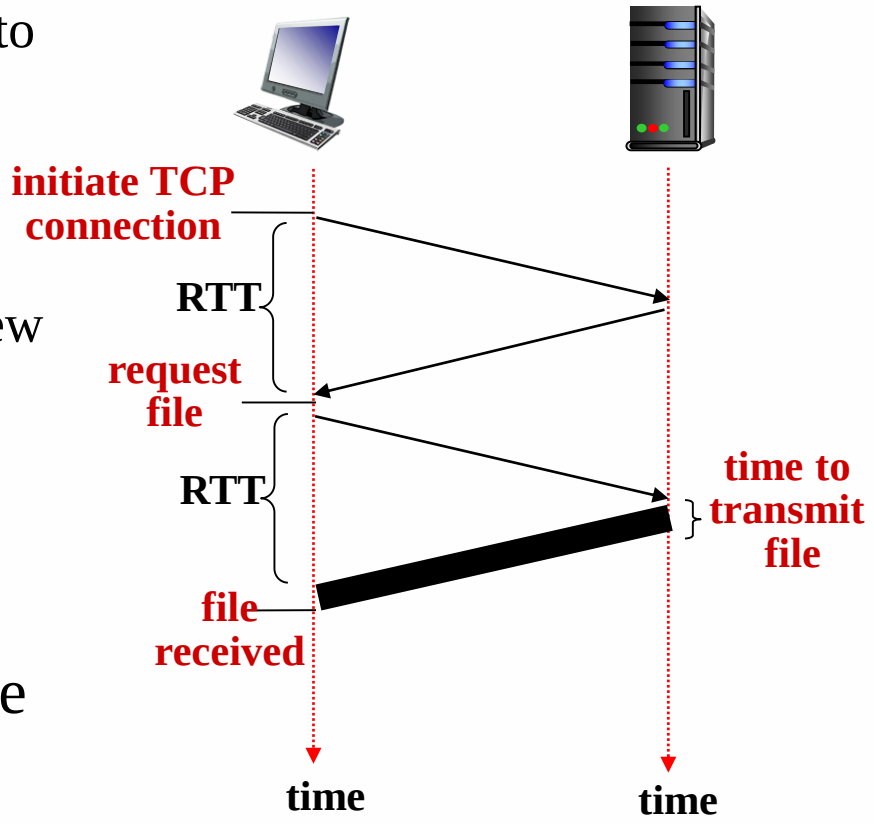
RTT = Round Trip Time زمان رفت و برگشت

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



HTTP پایا

مشکلات HTTP ناپایا

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

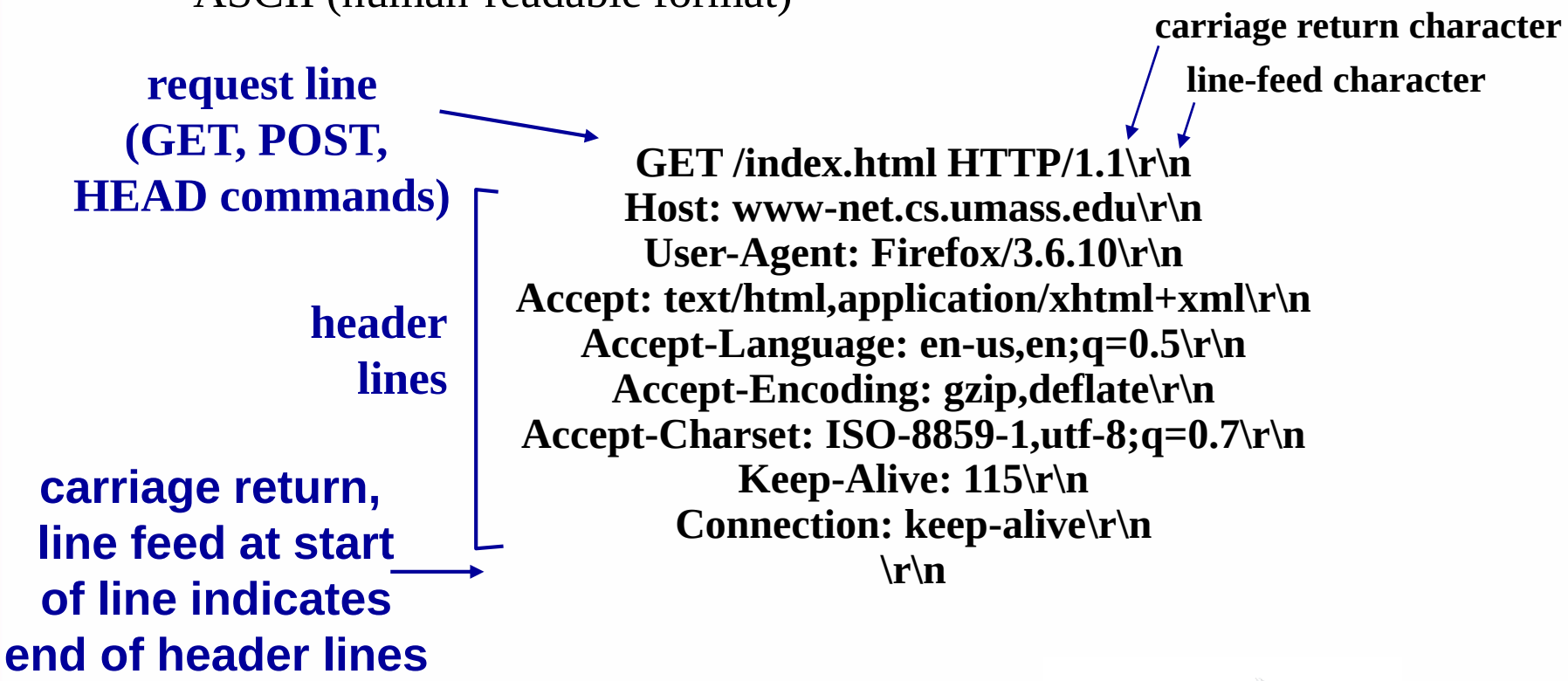
HTTP پایا

persistent HTTP:

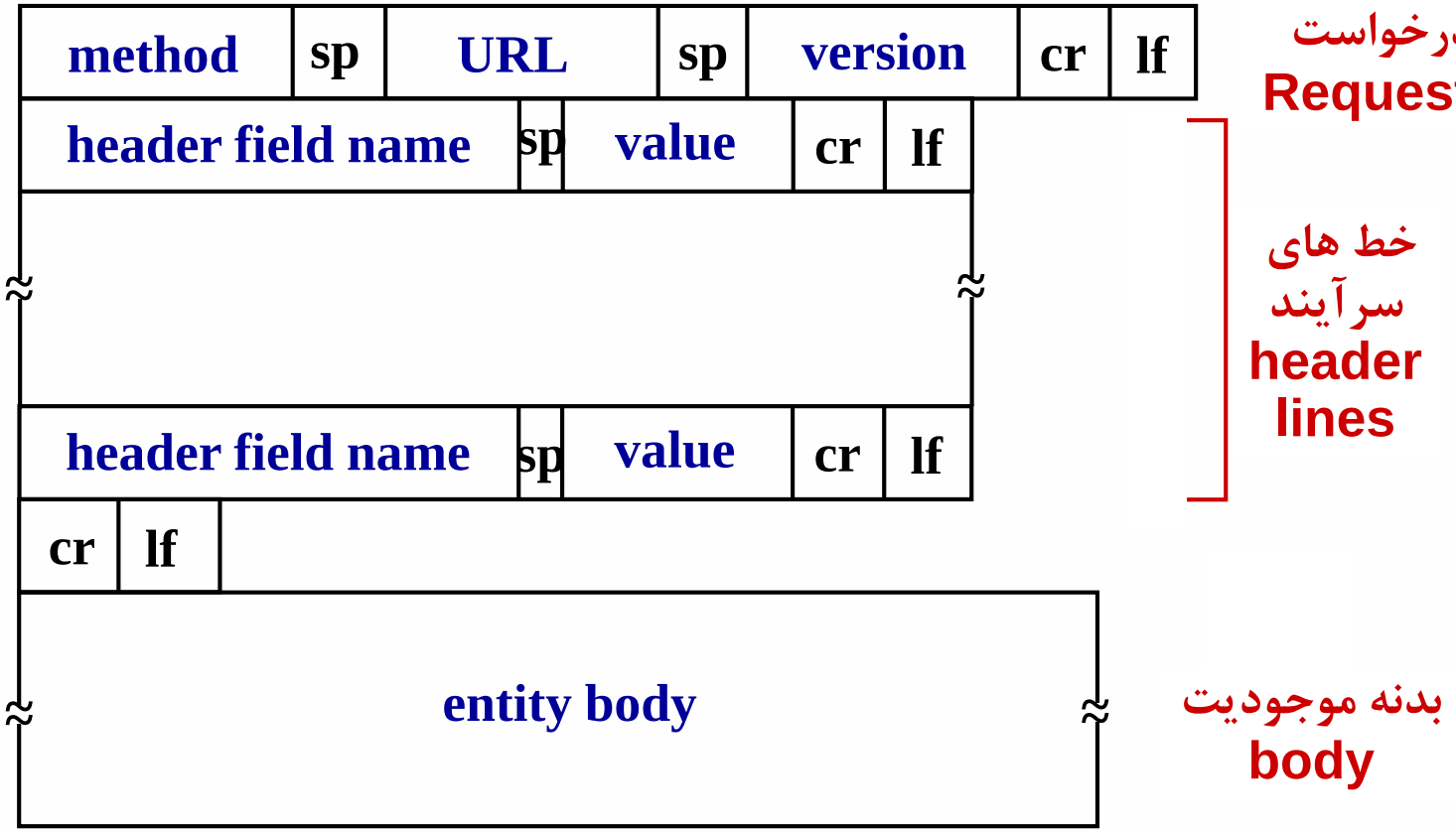
- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

پیام درخواست در HTTP

- در HTTP دو نوع پیام وجود دارد: پیام تقاضا یا درخواست و پیام پاسخ
- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)



قالب کلی یک پیام درخواست در HTTP



Uploading form input انواع بارگذاری فرم ورودی

روش استفاده از دستور POST

POST method:

- web page often includes form input
- input is uploaded to server in entity body

روش استفاده از URL

URL method:

- uses GET method
- input is uploaded in URL field of request line:

```
www.somesite.com/animalsearch?monkeys&banana
```

Method types انواع متدها

HTTP/1.0:

- GET
- POST
- HEAD
  - مشابه GET است و برای عیب‌یابی استفاده می‌شود و سرور در پاسخ آن، یک پیام پاسخ بدون شی می‌فرستد
  - asks server to leave requested object out of response

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - برای آپلود یا بارگذاری فایل
  - uploads file in entity body to path specified in URL field
- DELETE
  - برای حذف فایل
  - deletes file specified in the URL field

پیام پاسخ در HTTP (response message)

خط وضعیت

status line (protocol status code status phrase)

خطهای سرآیند

header lines

داده

data, e.g., requested HTML file

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

response status codes کدهای وضعیت در پیام پاسخ

- status code appears in 1st line in server-to-client response message.
- some sample codes:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

- requested object moved, new location specified later in this msg (Location:)

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

تست HTTPتوسط خودتان (سمت کلاینت)

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

by typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. look at response message sent by HTTP server!

یا می توانید از نرم‌افزار وایرشارک استفاده نمایید

(or use Wireshark to look at captured HTTP request/response)

User-server state: cookies کوکی: نگهداری وضعیت

many Web sites use cookies

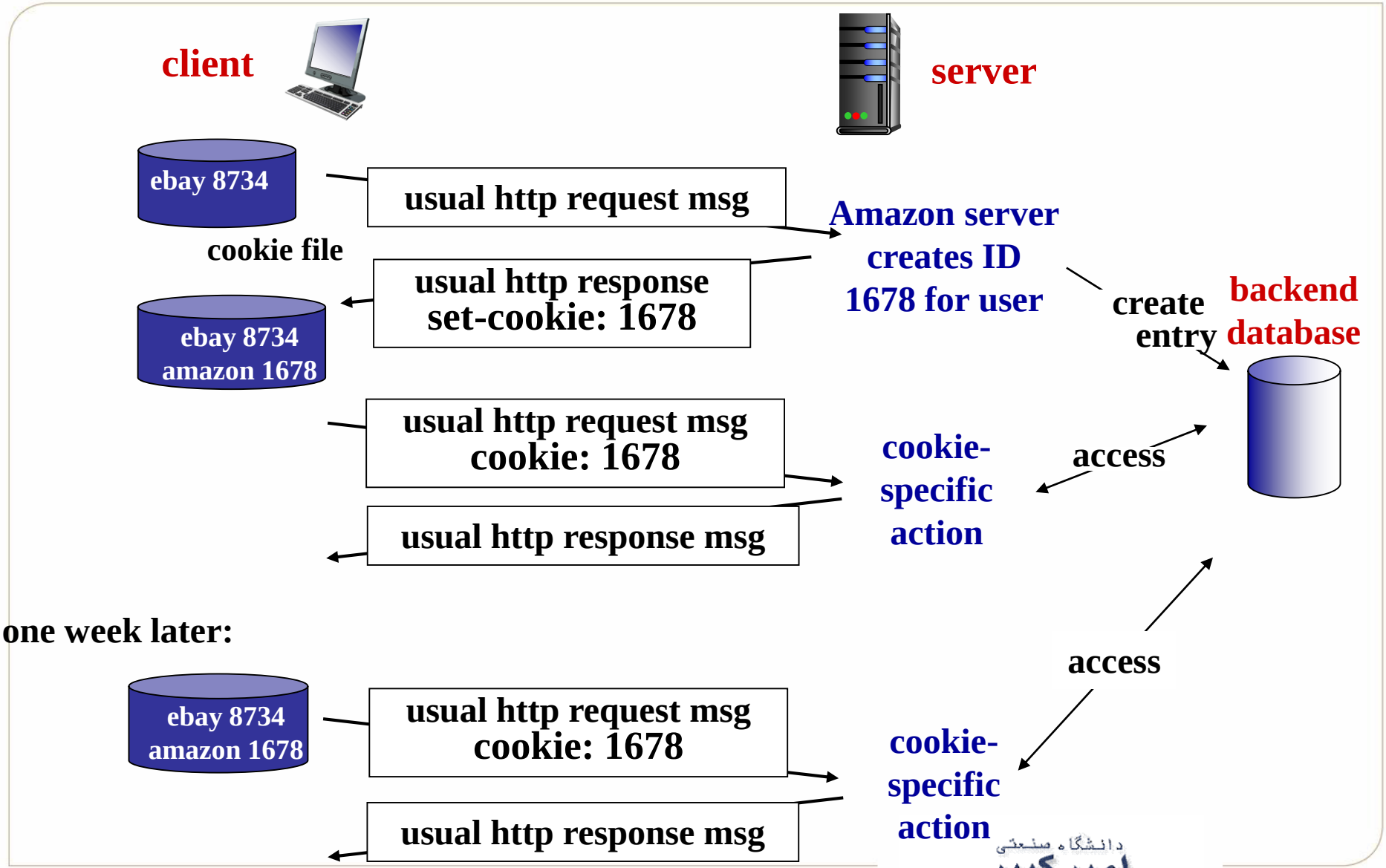
four components:

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

example:

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

کوکی: نگهداری وضعیت (ادامه)



کوکی (ادامه)

what cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

توضیح

cookies and privacy:

- **cookies permit sites to learn a lot about you**
- **you may supply name and e-mail to sites**

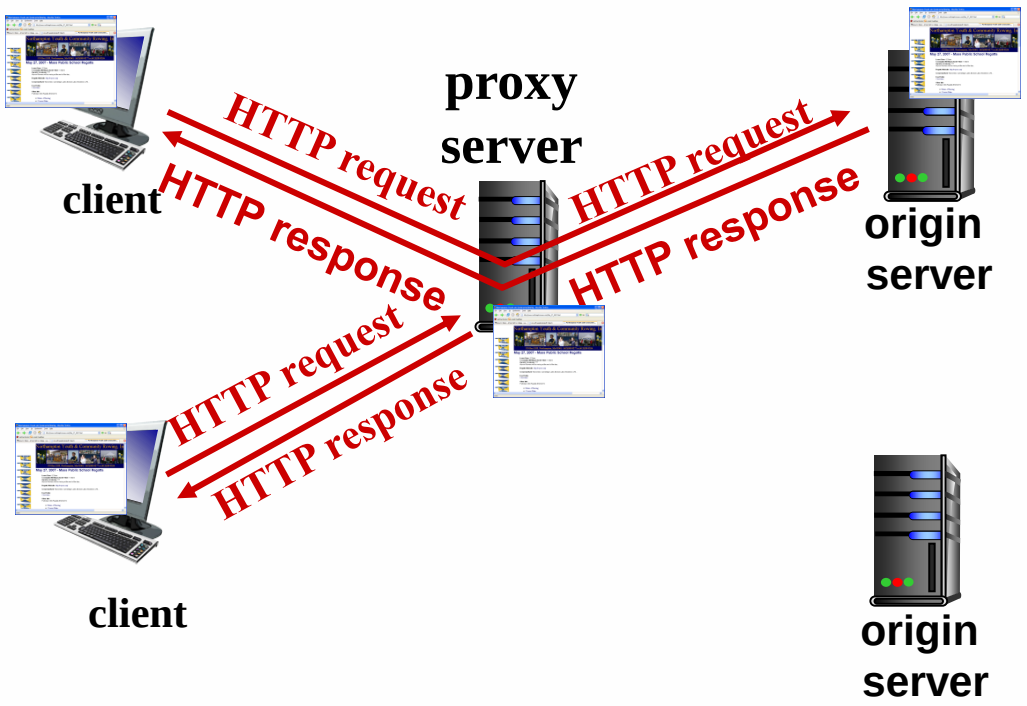
how to keep "state":

- **protocol endpoints: maintain state at sender/receiver over multiple transactions**
- **cookies: http messages carry state**

حافظه نهان برای وب (سرور پراکسی)

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



توضیح بیشتر برای حافظه نهان وب

- 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 (so too does P2P file sharing)

مثال برای حافظه نهان وب

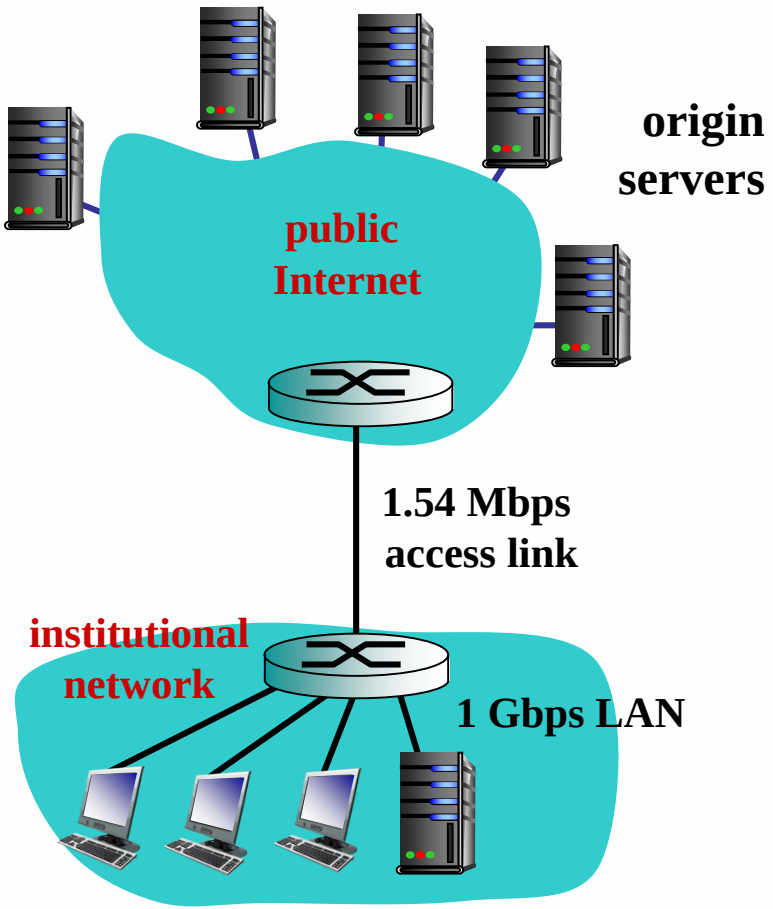
assumptions:

- 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:

problem!

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



مثال برای حافظه نهان وب: استفاده بیشتر از لینک

assumptions:

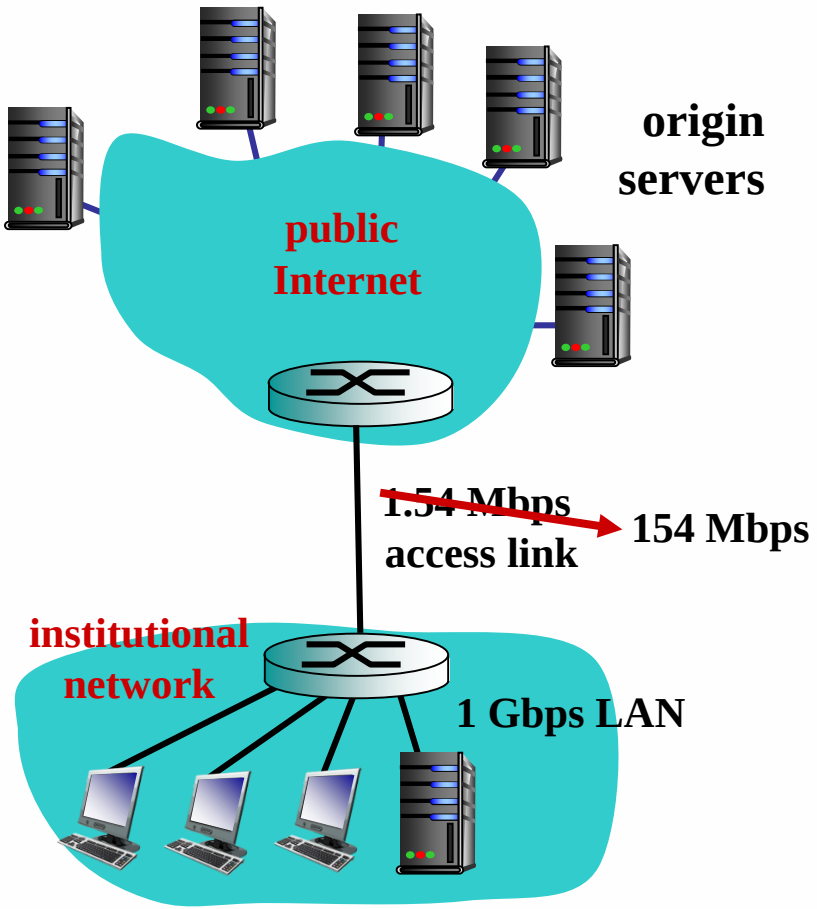
- avg object size: 100K bits
- avg request rate from browsers to origin servers:15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 *Mbps*

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!)

مثال برای حافظه نهان وب: حافظه نهان محلی

assumptions:

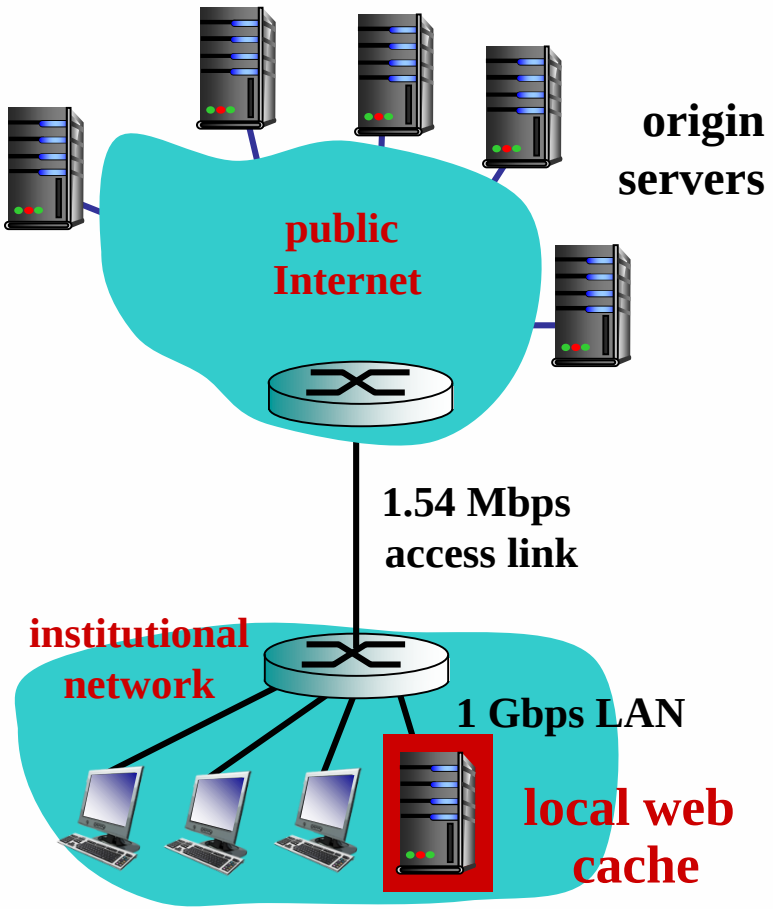
- avg object size: 100K bits
- avg request rate from browsers to origin servers:15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

consequences:

- LAN utilization: 15%
- access link utilization = 100%
- total delay = Internet delay + access delay + LAN delay
  = 2 sec + minutes + usecs

How to compute link utilization, delay?

Cost: **web cache (cheap!)**



مثال برای حافظه نهان وب: حافظه نهان محلی

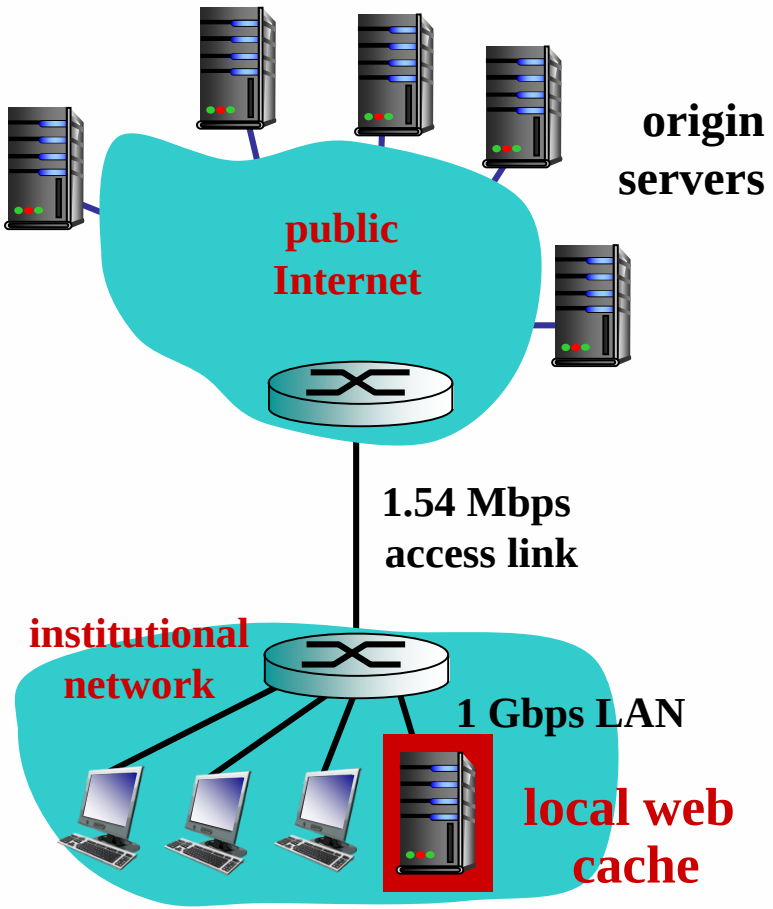
Calculating access link utilization, delay with cache:

- 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!)**

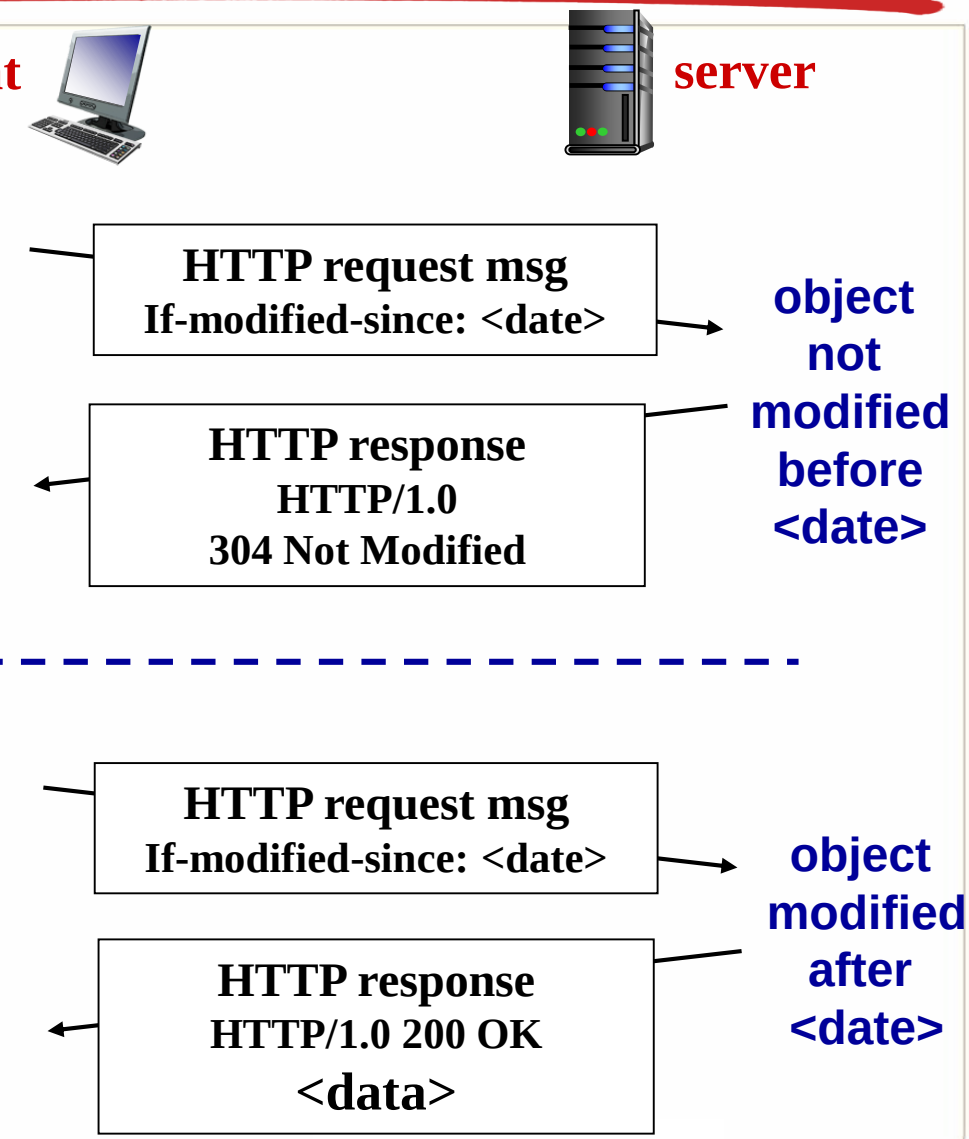


Conditional GET پیام دریافت شرطی

- *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



بخش های فصل دوم: لایه کاربرد

۲-۱ اصول برنامه های کاربردی شبکه

2.1 principles of network applications

۲-۲ وب و HTTP

2.2 Web and HTTP

۲-۳ انتقال فایل - FTP

2.3 FTP

۲-۴ پست الکترونیک در اینترنت

2.4 electronic mail
- SMTP, POP3, IMAP

۲-۵ سامانه نام دامنه

2.5 DNS

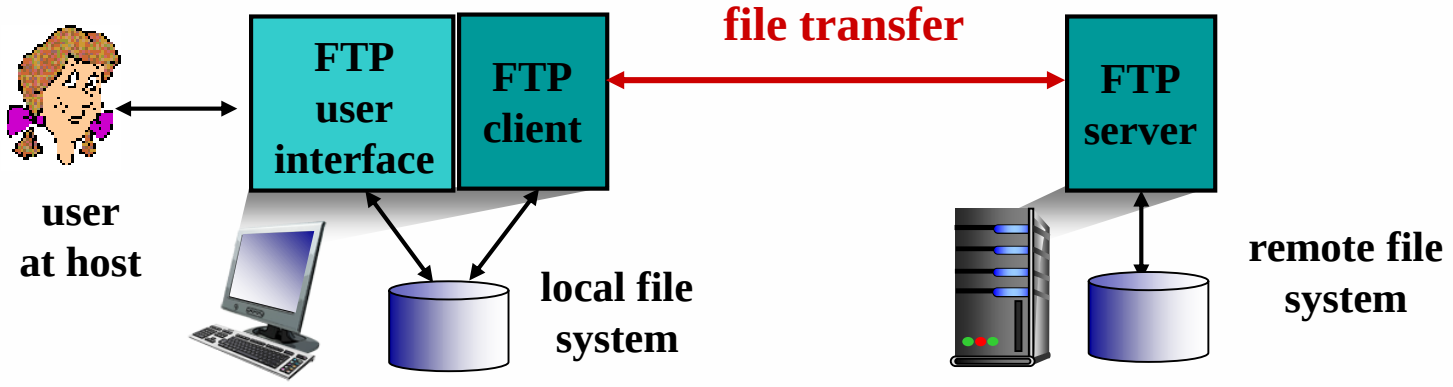
۲-۶ کاربردهای نقطه به نقطه

2.6 P2P applications

۲-۷ برنامه نویسی سوکت

2.7 socket programming with UDP and TCP

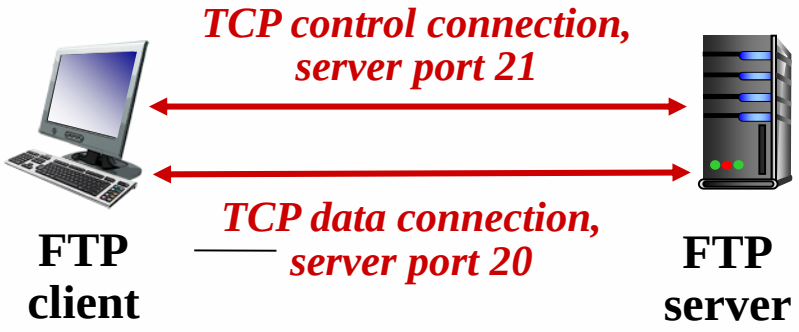
پروتکل انتقال فایل (FTP)



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 20 and port 21

دو اتصال مجزا برای داده و کنترل در پروتکل انتقال فایل

- FTP client contacts FTP server at port 21, using TCP
- client authorized over control connection
- client browses remote directory, sends commands over control connection
- when server receives file transfer command, *server* opens *2nd* TCP data connection (for file) *to* client
- after transferring one file, server closes data connection



- **server opens another TCP data connection to transfer another file**
- **control connection: *"out of band"***
- **FTP server maintains "state": current directory, earlier authentication**

دستورات و پاسخ‌ها در پروتکل انتقال فایل

sample commands:

- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores (puts) file onto remote host

sample return codes

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can’t open data connection**
- **452 Error writing file**

بخش های فصل دوم: لایه کاربرد

۲-۱ اصول برنامه های کاربردی شبکه

2.1 principles of network applications

۲-۲ وب و HTTP

2.2 Web and HTTP

۲-۳ انتقال فایل - FTP

2.3 FTP

۲-۴ پست الکترونیک در اینترنت

2.4 electronic mail

– SMTP, POP3, IMAP

۲-۵ سامانه نام دامنه

2.5 DNS

۲-۶ کاربردهای نقطه به نقطه

2.6 P2P applications

۲-۷ برنامه نویسی سوکت

2.7 socket programming with UDP and TCP

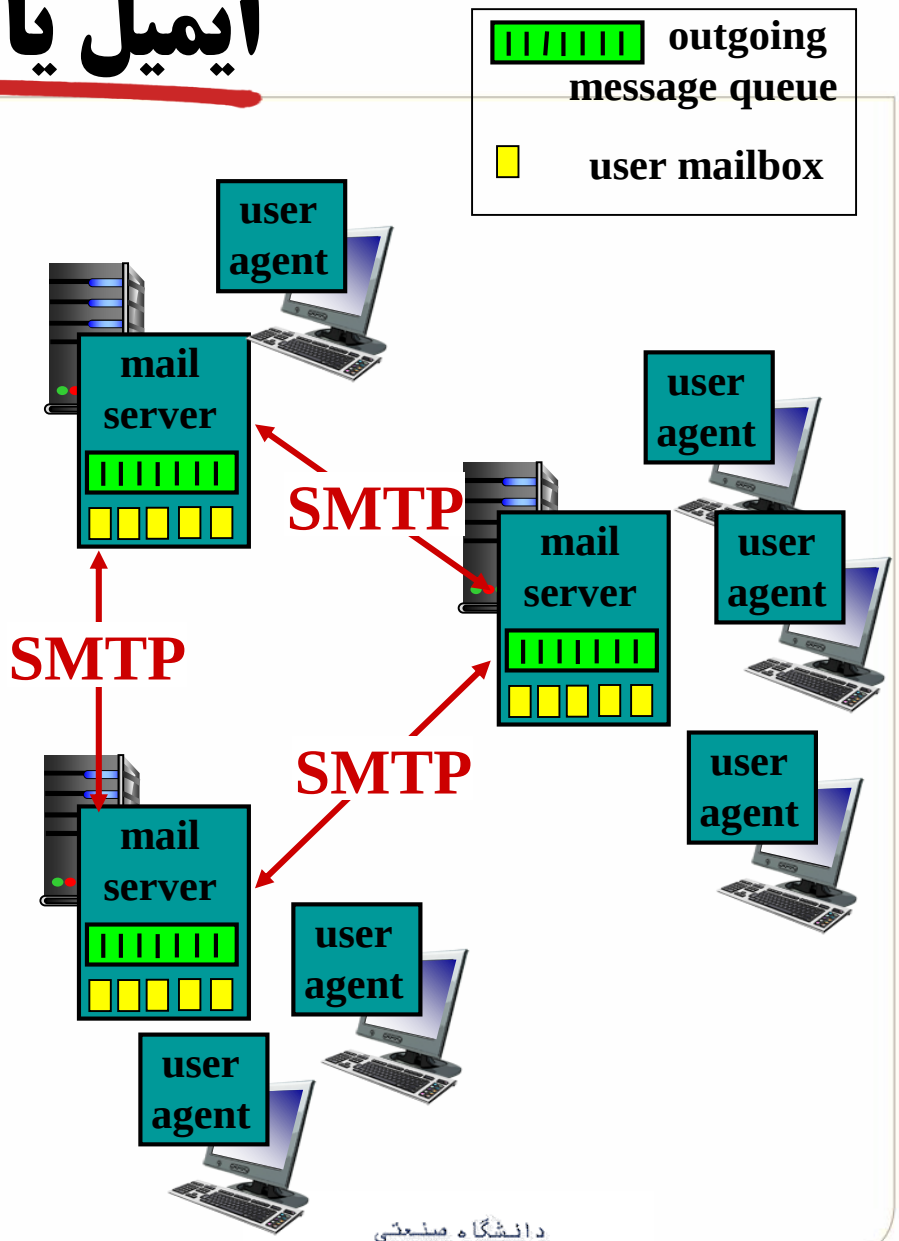
Electronic mail ایمیل یا رایانامه

Three major components:

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

User Agent

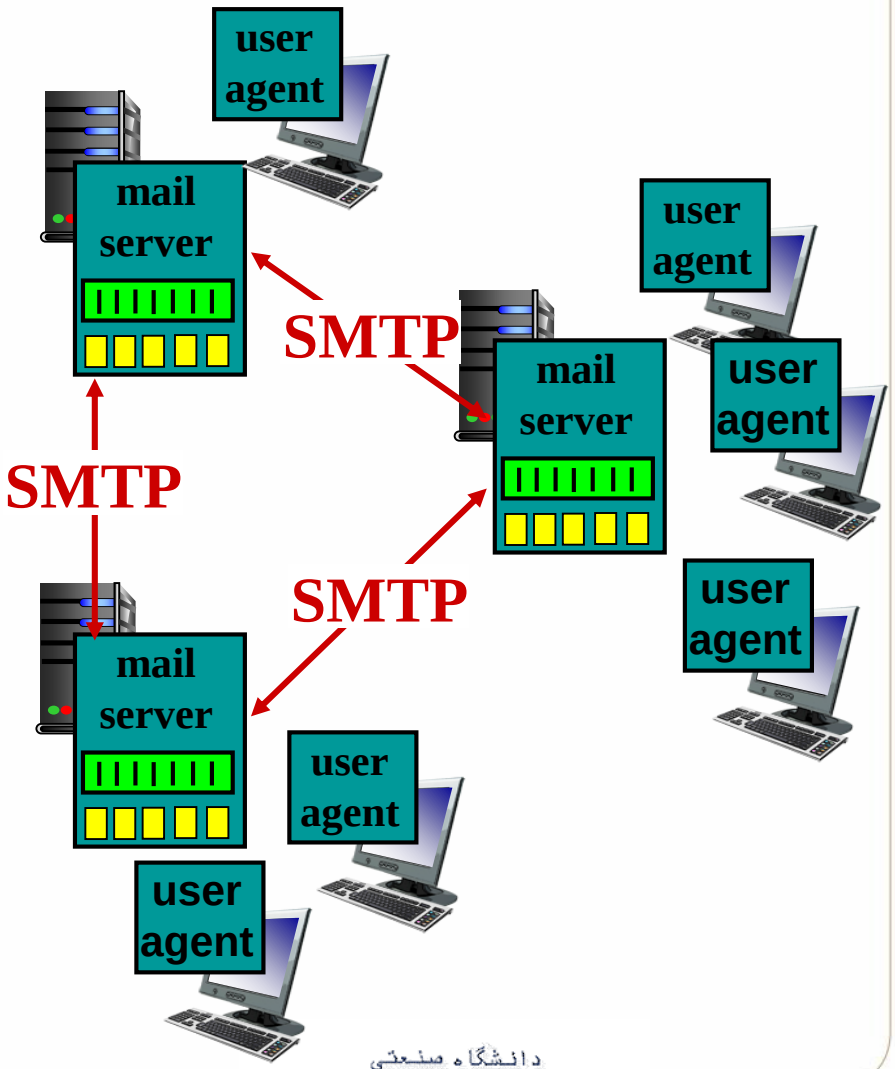
- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server



ایمیل یا رایانامه: سرور نامه‌ها

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

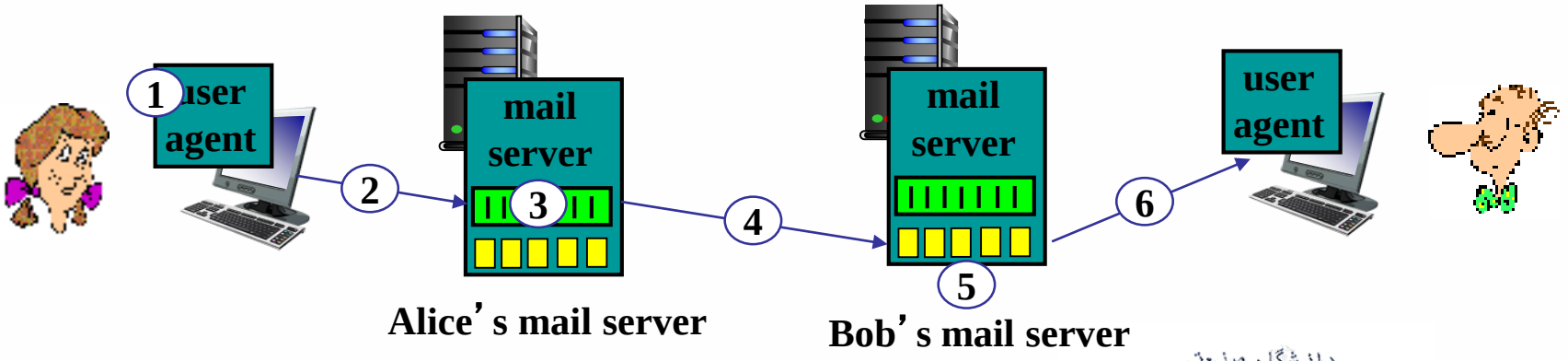


ایمیل یا رایانامه: پروتکل SMTP

- 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, FTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCI

سناریو ارسال یک ایمیل یا رایانامه

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



مثالی برای تعاملات SMTP

```
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
```

بررسی تعاملات SMTP توسط خودتان

- **`telnet servername 25`**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

چند نکته در رابطه با SMTP

- 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

مقایسه SMTP با HTTP

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

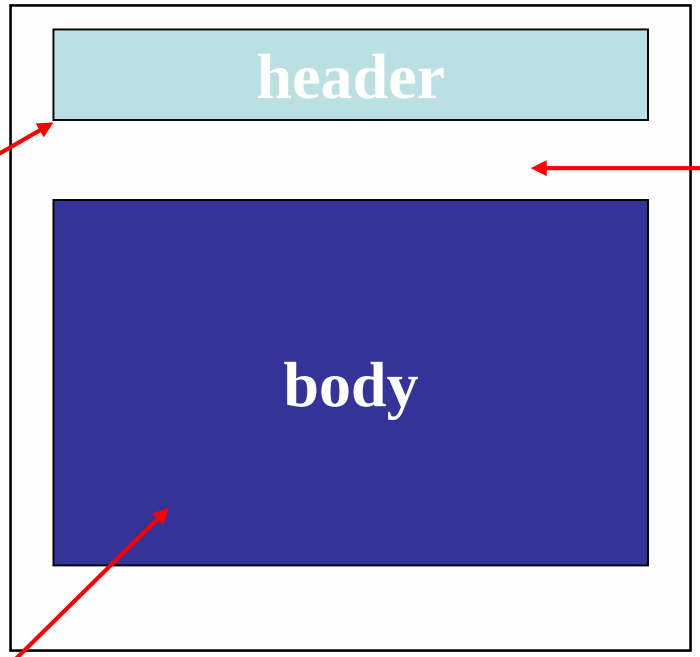
قالب پیام‌ها در ایمیل

SMTP: protocol for exchanging email msgs

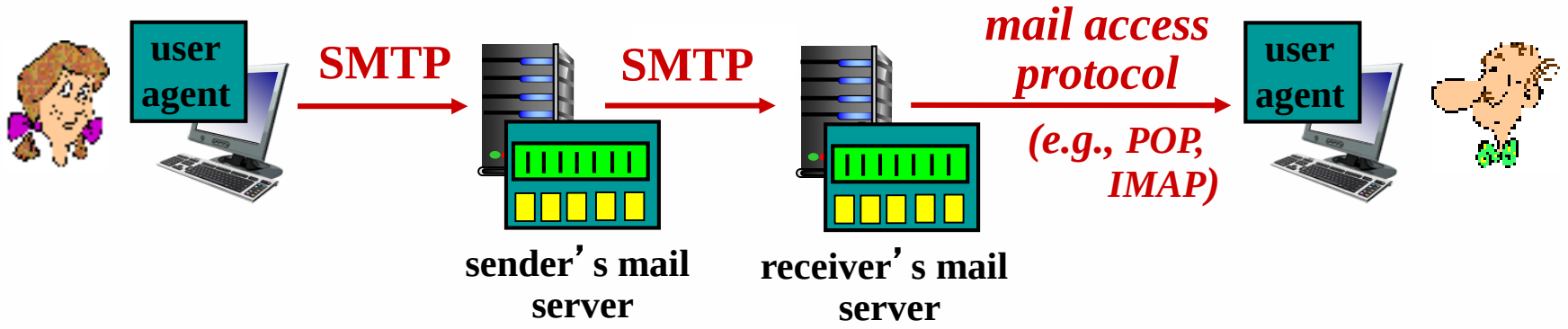
RFC 822: standard for text message format:

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

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



پروتکل‌های دسترسی به ایمیل



- 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 msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

پروتکل POP3

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
```

پروتکل POP3 (ادامه)

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

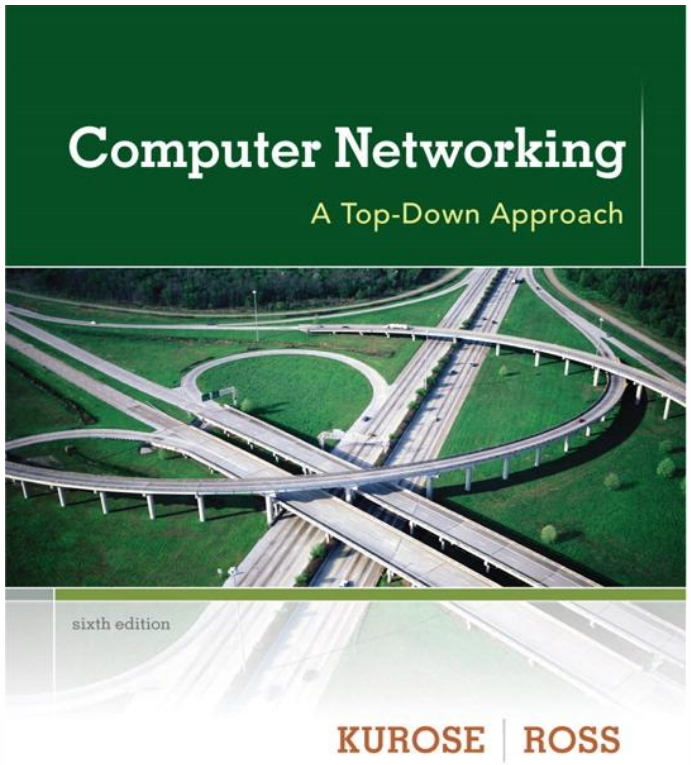
پروتکل IMAP

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

منابع

این پاورپوینت از روی پاورپوینت مربوط به فصل ۲
ویرایش ششم کتاب تهیه شده است



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012