Chapter 2
Application Layer

App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response
- Message format:
  - Syntax :what fields in messages & how fields are delineated
  - Semantics: meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

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

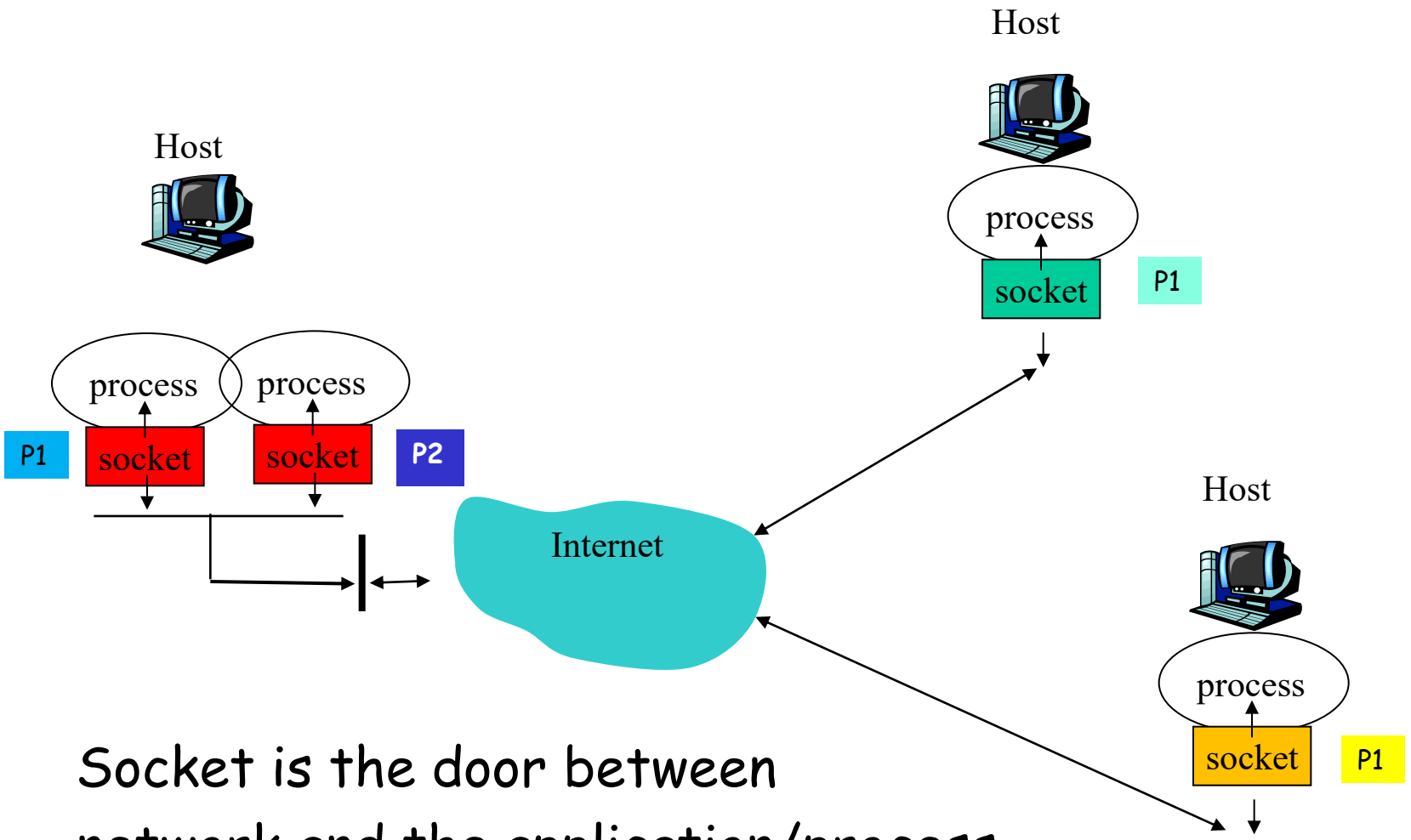
Proprietary protocols:

- e.g., KaZaA, real audio

Network application

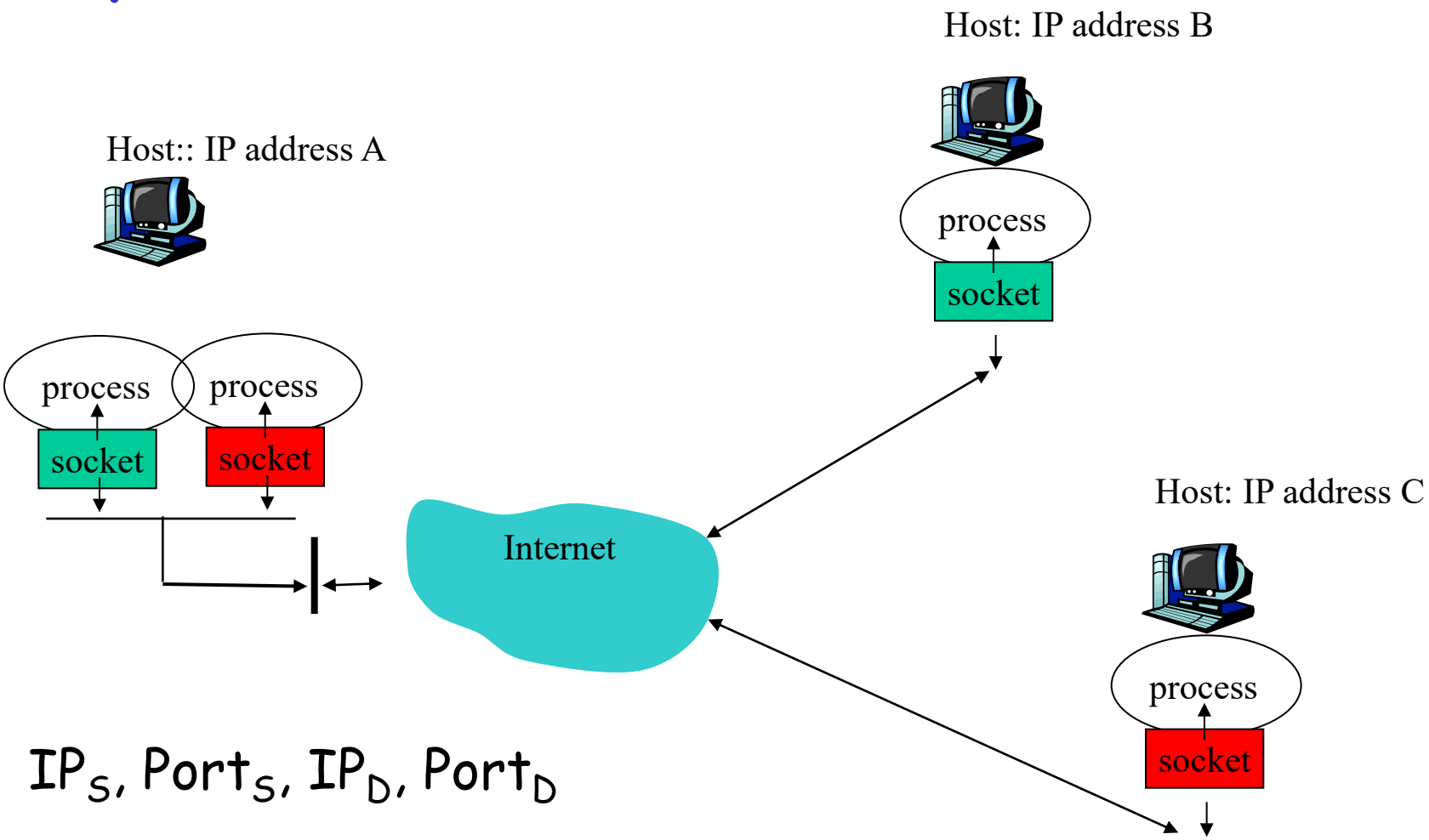
- Non Network application: An application that executes on a single host.
- Now Consider: Two applications on 2 different hosts connected by a network
- In order to communicate, need to identify the parties
- Phone network: phone number (10 digits)
- Computer network: IP address
  - IPv4 (32 bits) 128.6.24.78
  - IPv6 (128 bits) 2001:4000:A000:C000:6000:B001:412A:8000
- In addition to host address, we need one more.
- More than one program executing on a host
- Which Program to talk to ?
- We need another identity : port #

IP address & port number



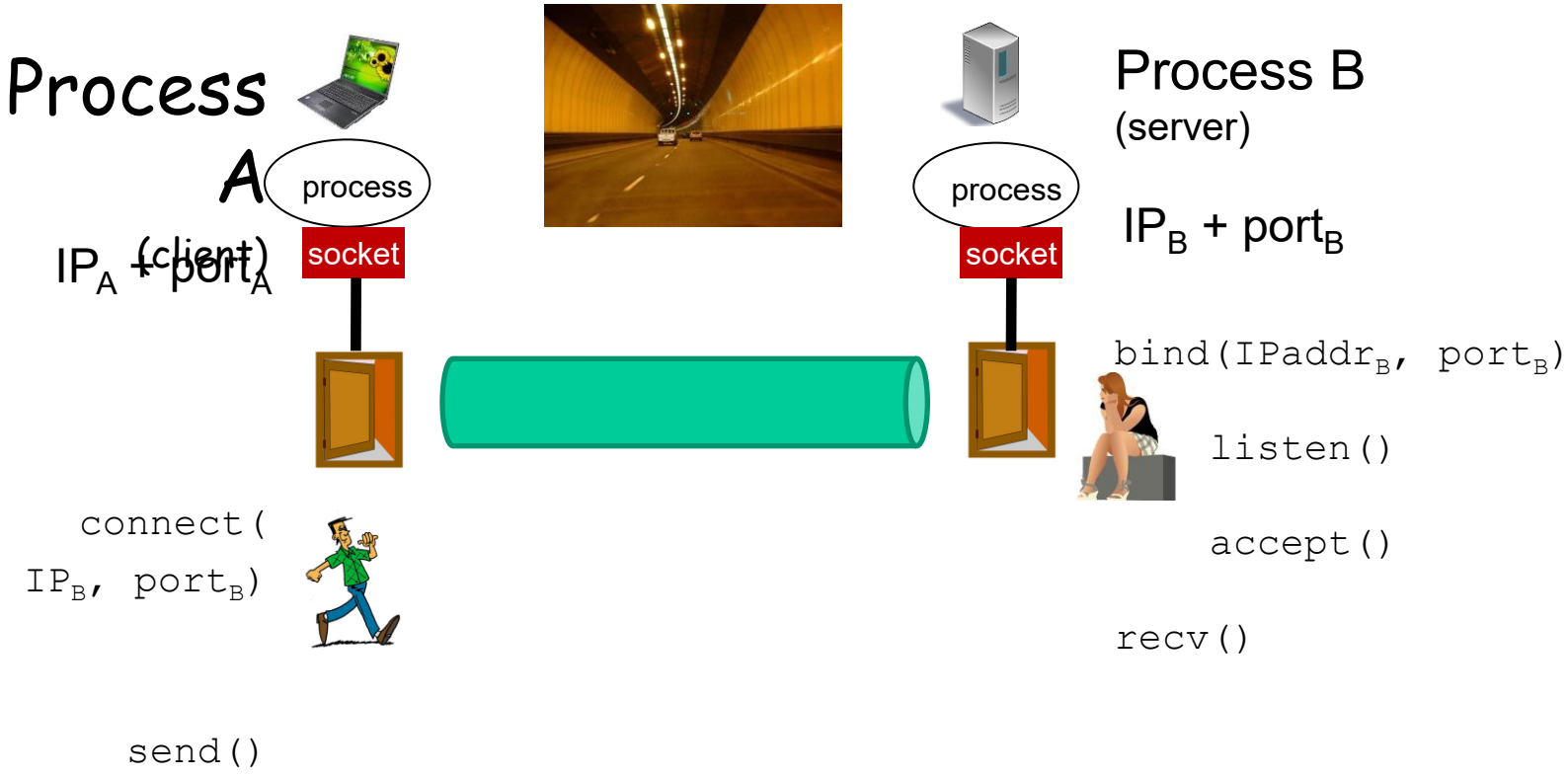
Socket is the door between network and the application/process

A network connection is a 4-tuple

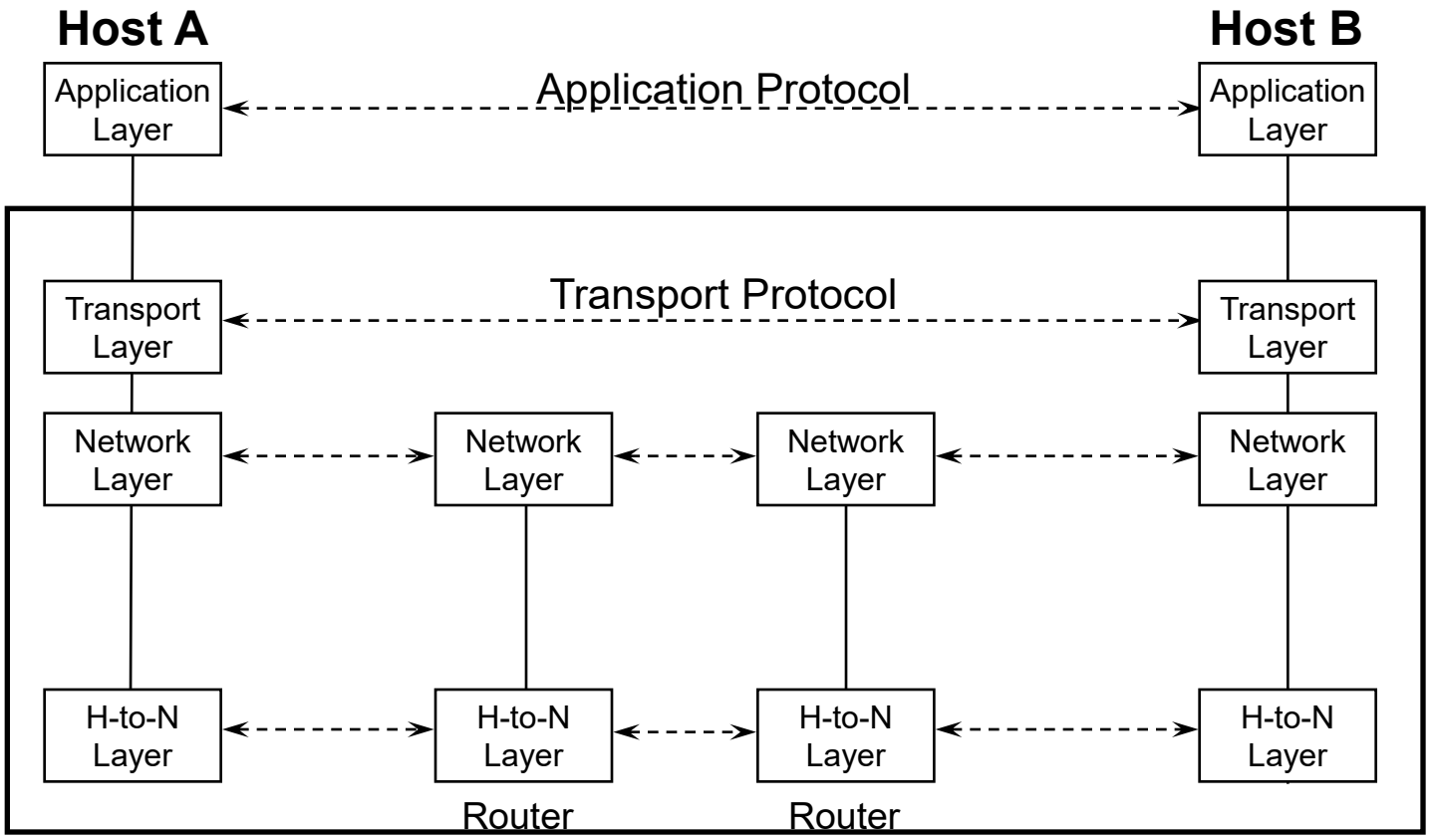


$IP_S, Port_S, IP_D, Port_D$

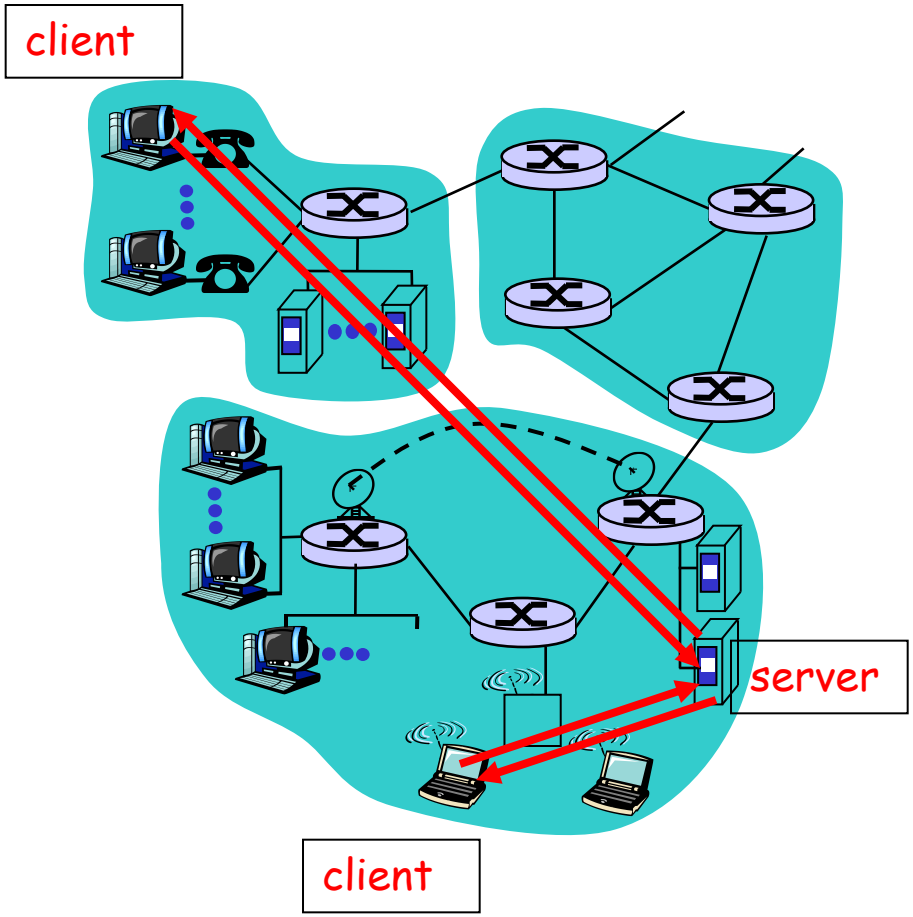
Socket system calls



Recall:Services provided by lower layers



Client-server architecture



server:

- always-on host
- permanent IP address
- server farms for scaling

clients:

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

Why?

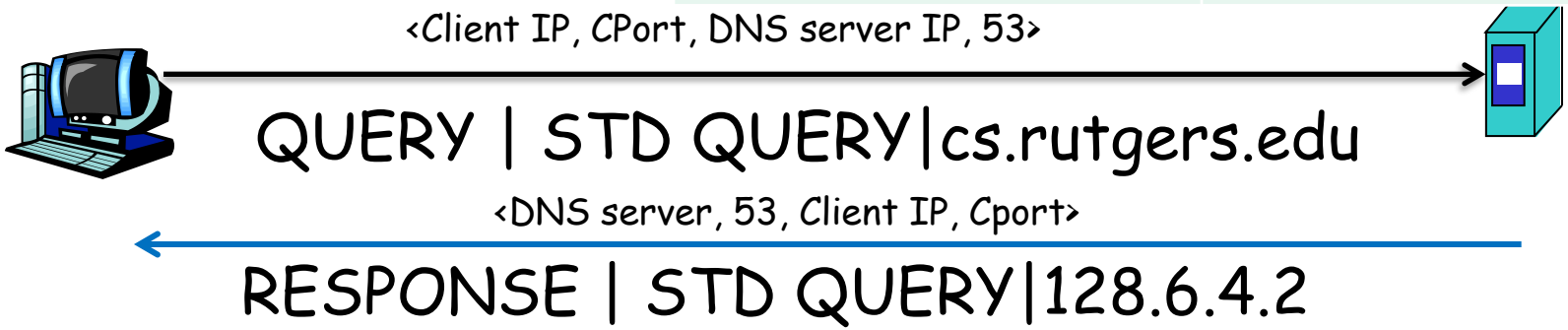
For any networked application we need to know the IP address of a host given its name

Domain Name System (DNS)

- Problem statement:
  - Average brain can easily remember 7 digits for a few names
  - On average, IP addresses have 12 digits
  - We need an easier way to remember IP addresses
- Solution:
  - Use alphanumeric names to refer to hosts
  - Just as a contacts or telephone directory (white pages)
  - Add a service (called DNS) to map between alphanumeric host names and binary IP addresses
  - We call this *Address Resolution*

Simple DNS

DOMAIN NAME	IP ADDRESS
chat.openai.COM	172.64.150.28
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



- Simple but does not scale
- Every new host needs to be entered in this table
- Performance? Failure?

DNS

Centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

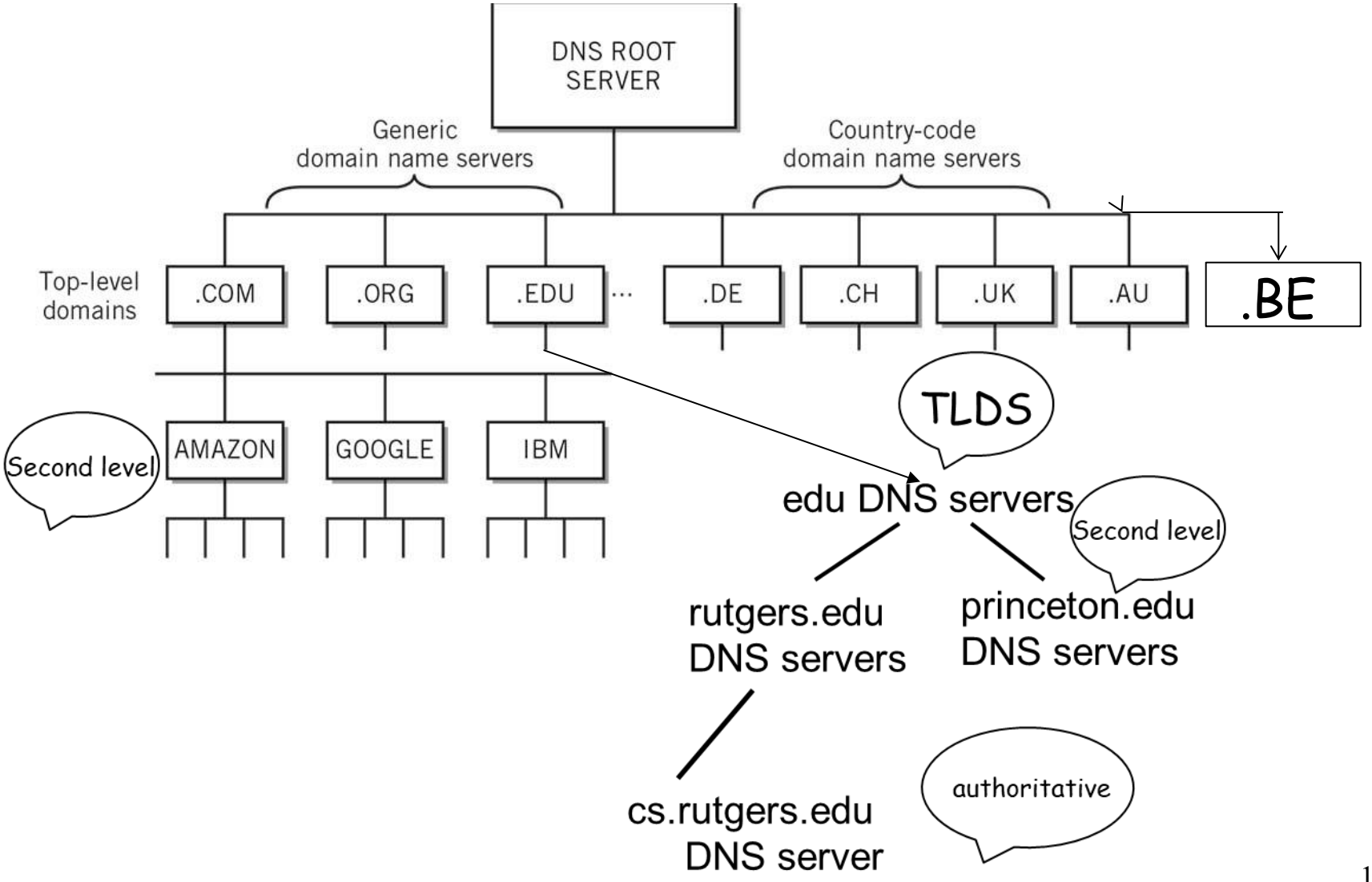
Distributed /Hierarchical

Hierarchy

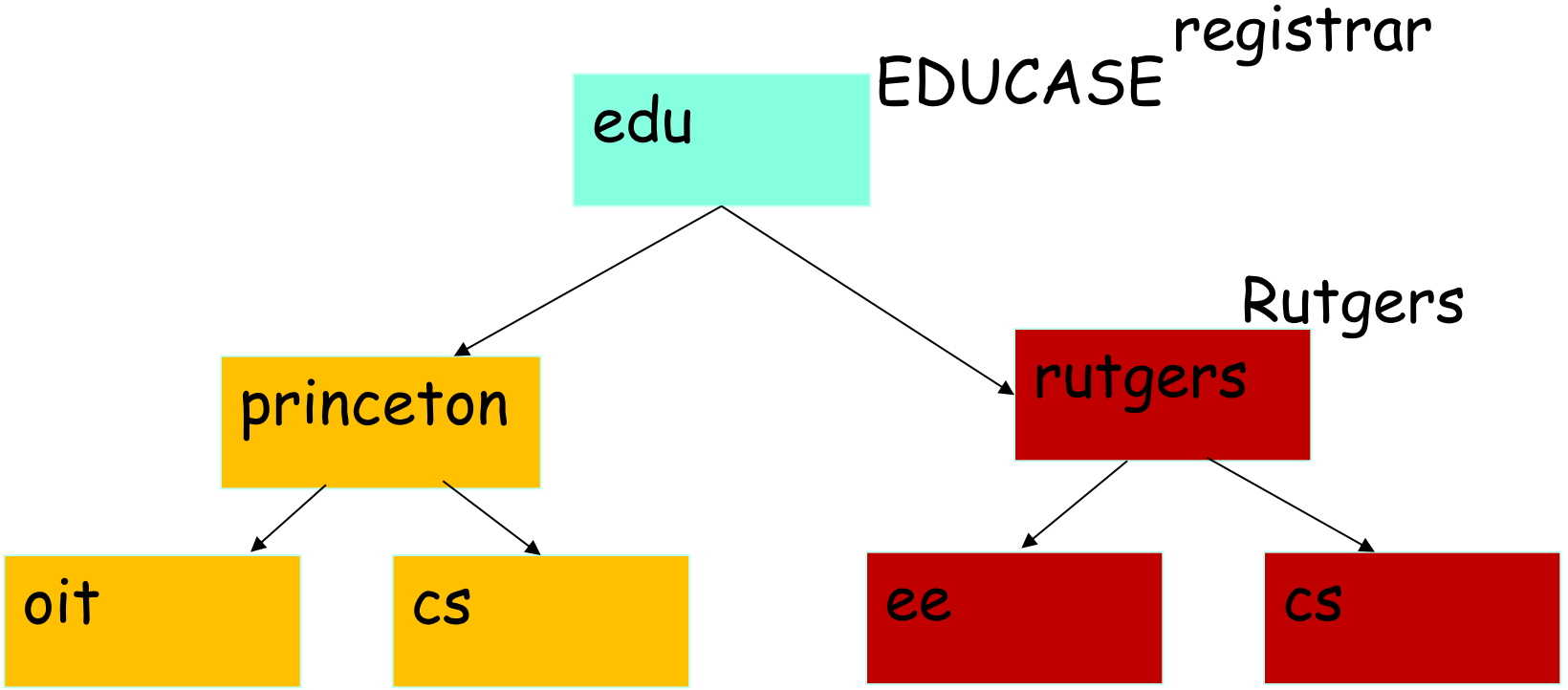
- Names are hierarchical
- Authority is hierarchical
  - distant centralized database
- Infrastructure is hierarchical

scales well!

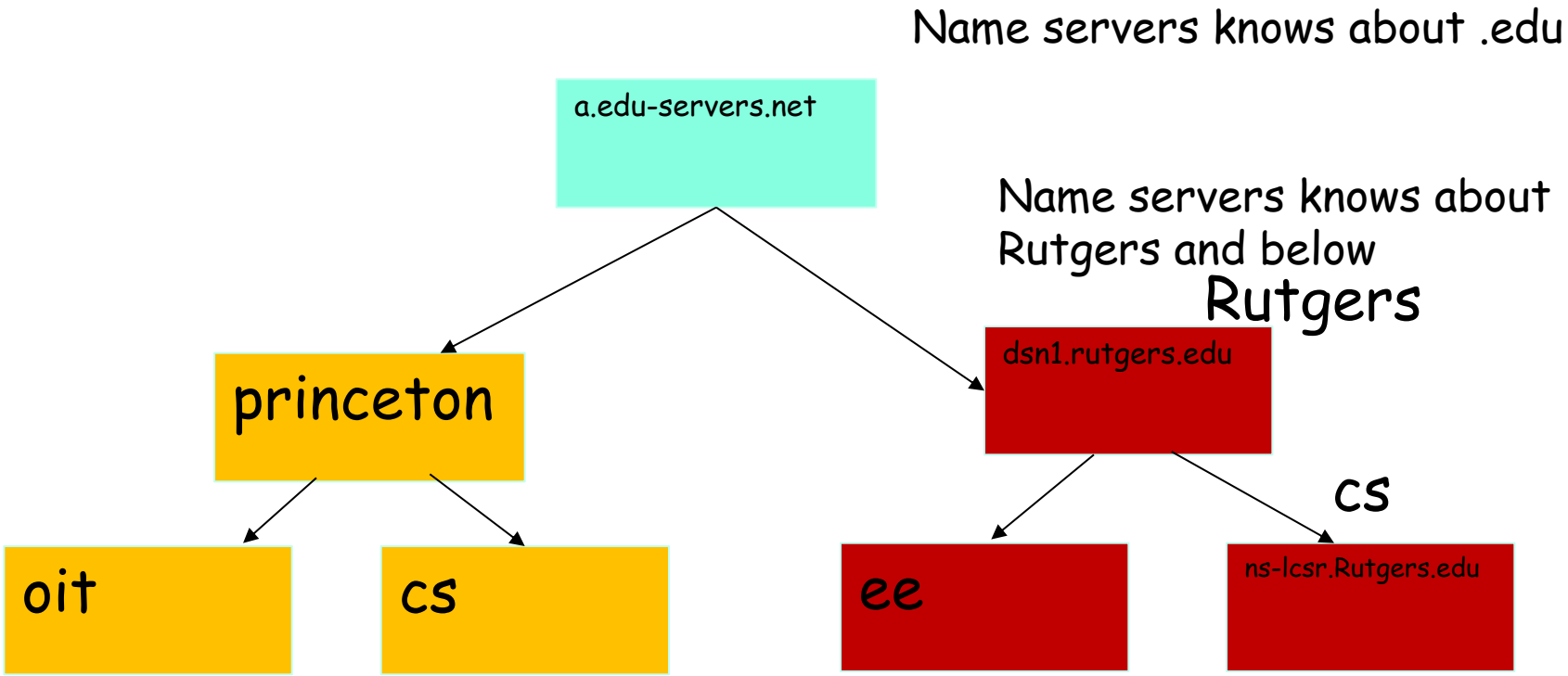
DNS naming hierarchy



DNS Authority hierarchy



DNS Infrastructure hierarchy

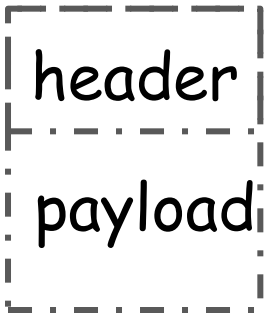


Root DNS servers

- 13 root servers a.root-servers.net,.. m.root-servers.net
- A root server 198.41.0.4, 2001:503:ba3e::2:30
- C root server 192.33.4.12, 2001:500:2::c
- IP addresses are well-known and published
- Physically many more than 13 located around the world
- Load balancing and reliability

DNS Protocol

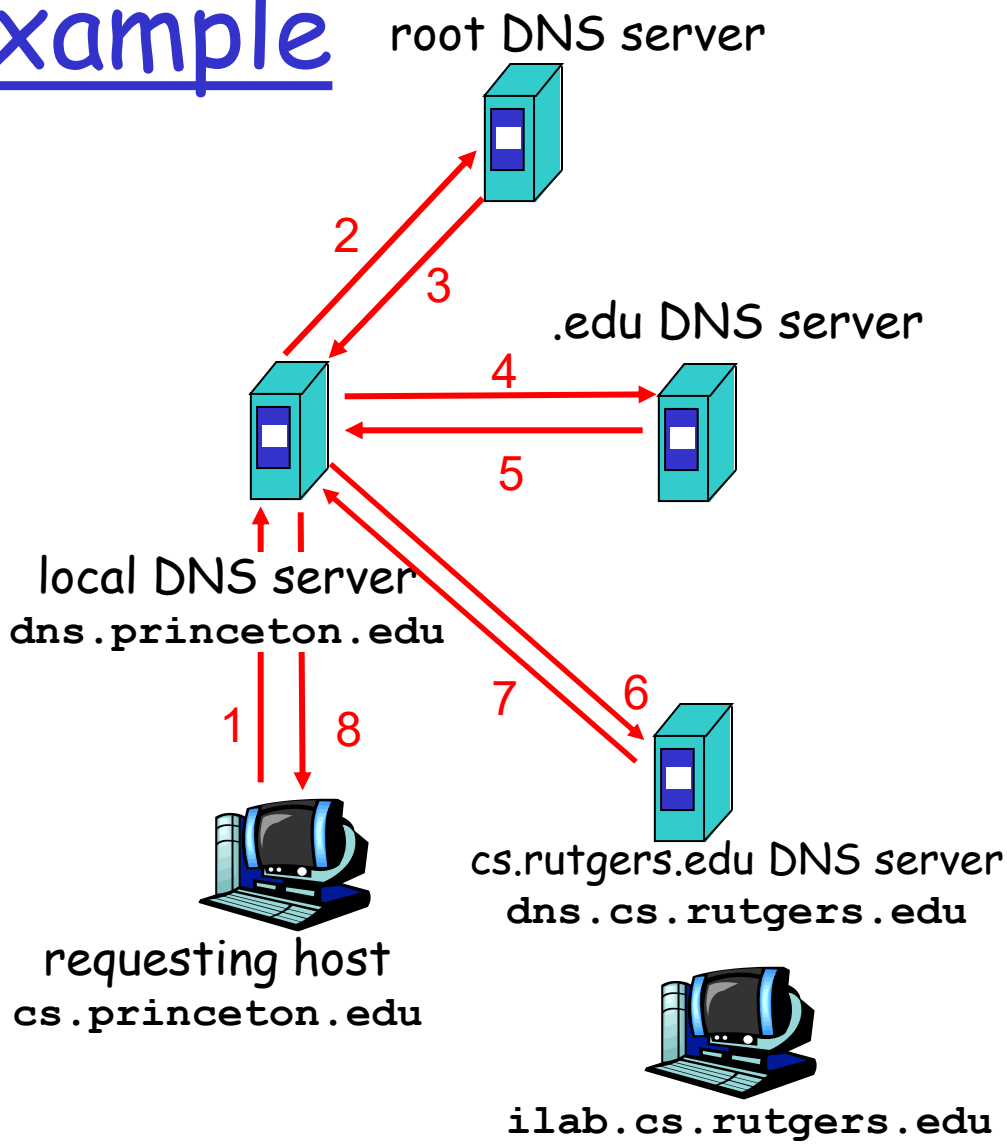
- Client and Server
- Client connects to Port 53
- DNS server address should be known
  - Either manually configured or automatically
- Two types of messages
  - Queries
  - Responses
- Type of Query (OPCODE) methods
  - Standard query (0x0)
    - Request domain name for a given IP address
  - Status (0x2)
  - Updates (0x5)
    - Provide a binding of IP address to domain name
- Each type has a common message format that follows the header



DNS Protocol

- When client wants to know an IP address for a host name
  - Client sends a DNS query to the primary name server in its zone
  - If name server contains the mapping, it returns the IP address to the client
  - Otherwise, the name server forwards the request to the root name server
  - The request works its way down the tree toward the host until it reaches a name server with the correct mapping

Example

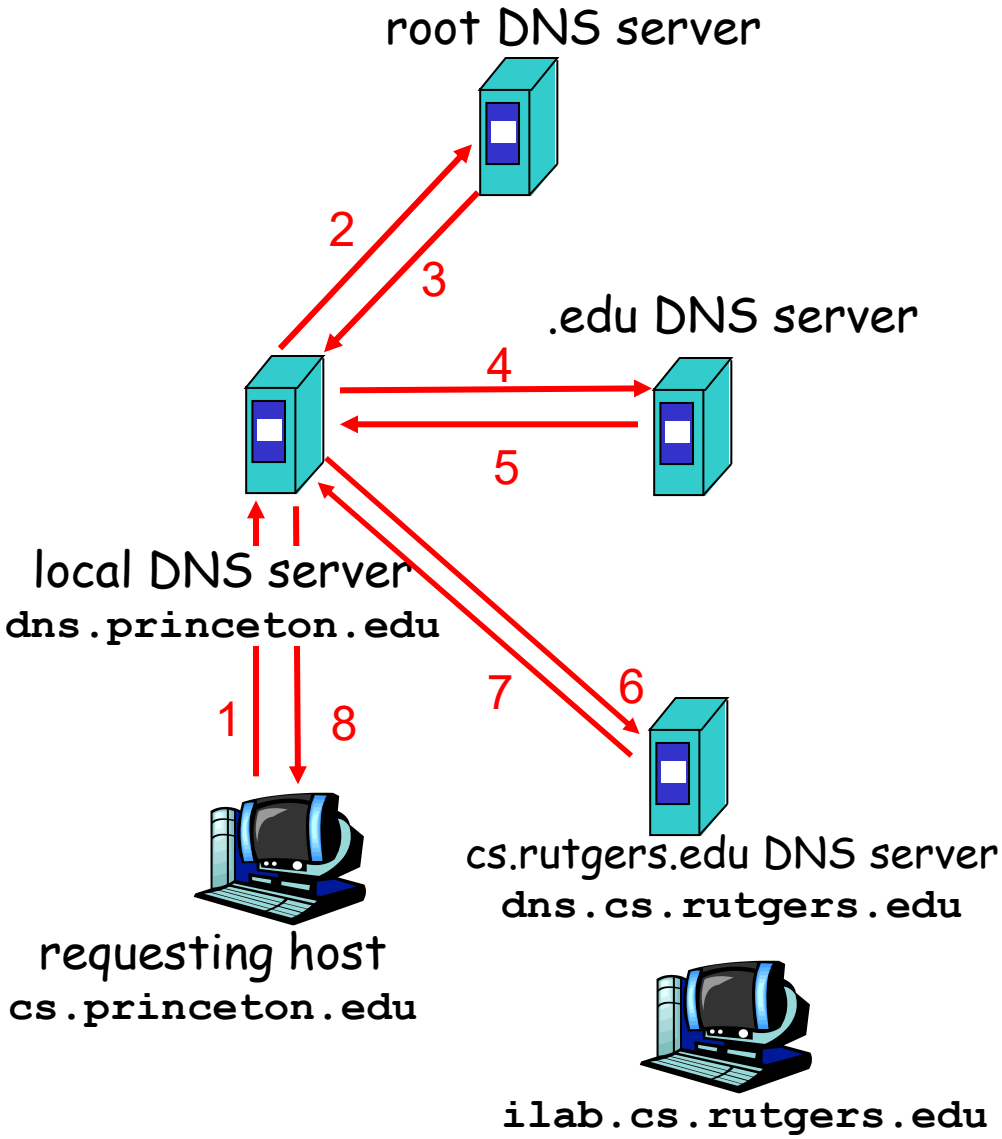


- Host at cs.princeton.edu wants IP address for ilab.cs.princeton.edu

Query type

iterated query:

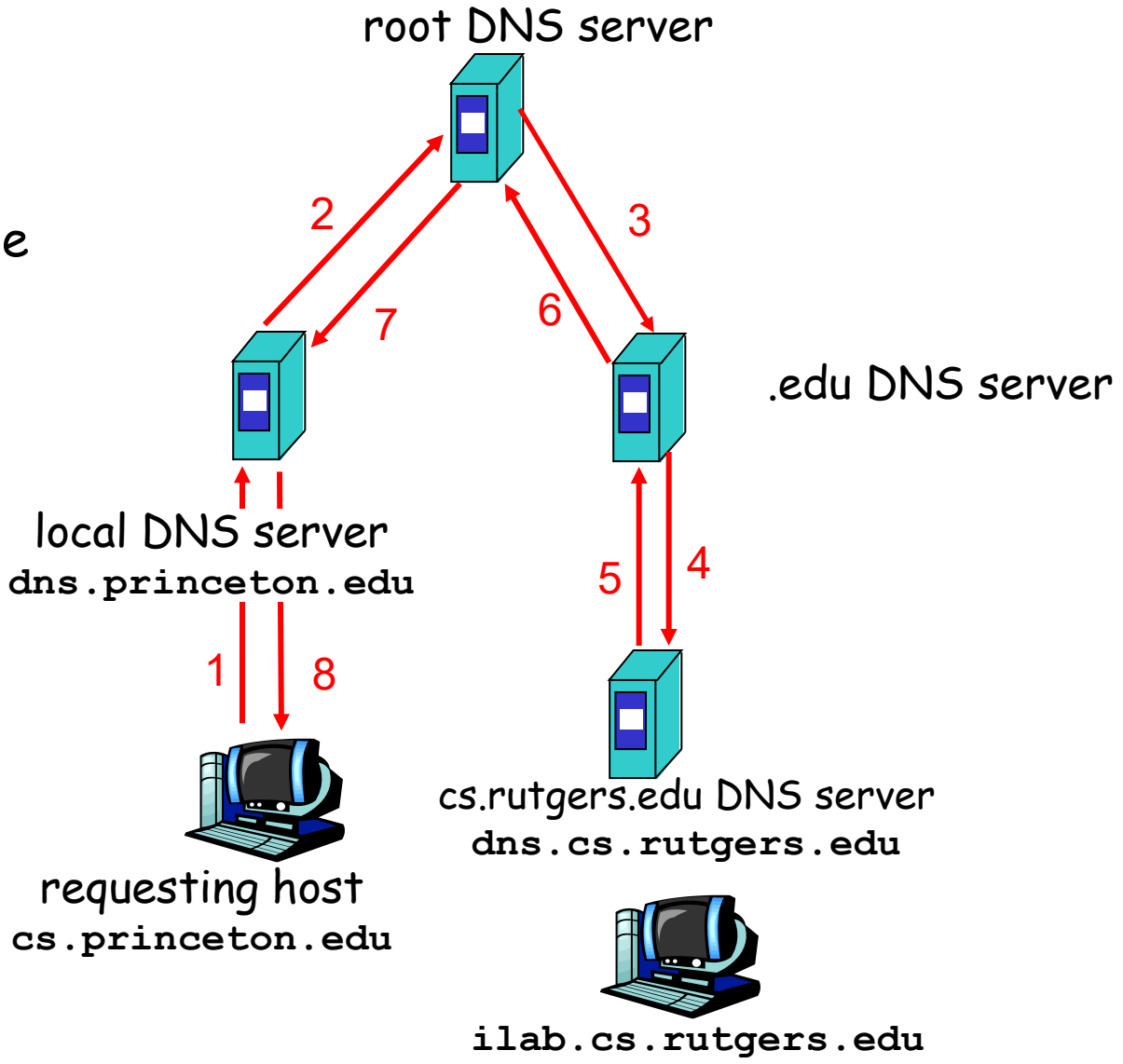
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Query type

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?



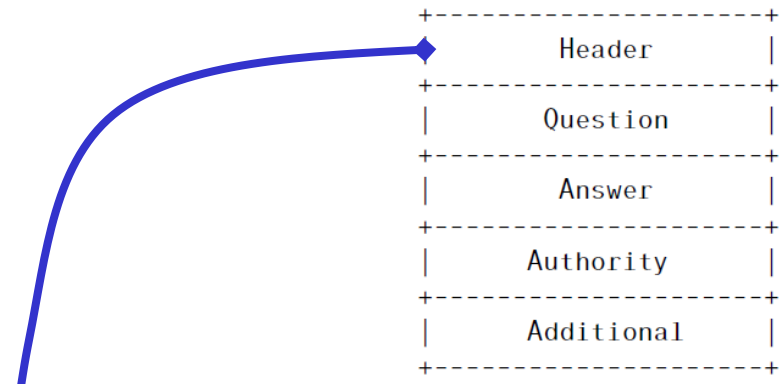
DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited

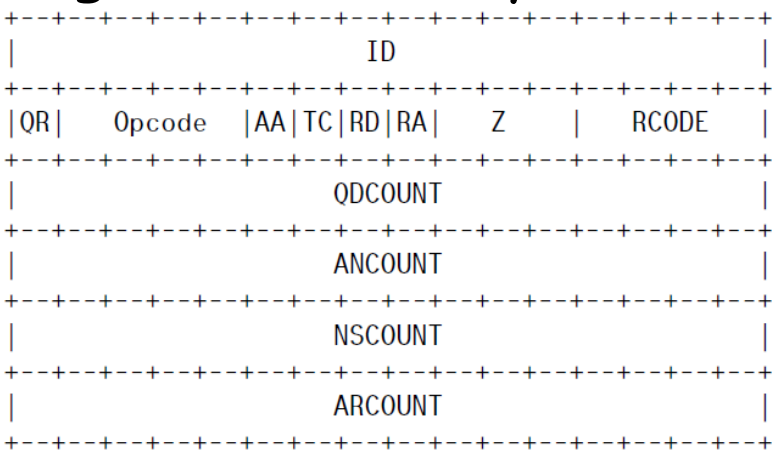
DNS protocol, messages

DNS protocol : *query* and *reply* messages, both with same *message format*

- QR = 0 for Query, 1 for Response
- Opcode= 0 standard
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - Authoritative Answer
  - Truncation
  - recursion desired
  - recursion available
  - Response code



Msg header (12 bytes)



DNS protocol, messages

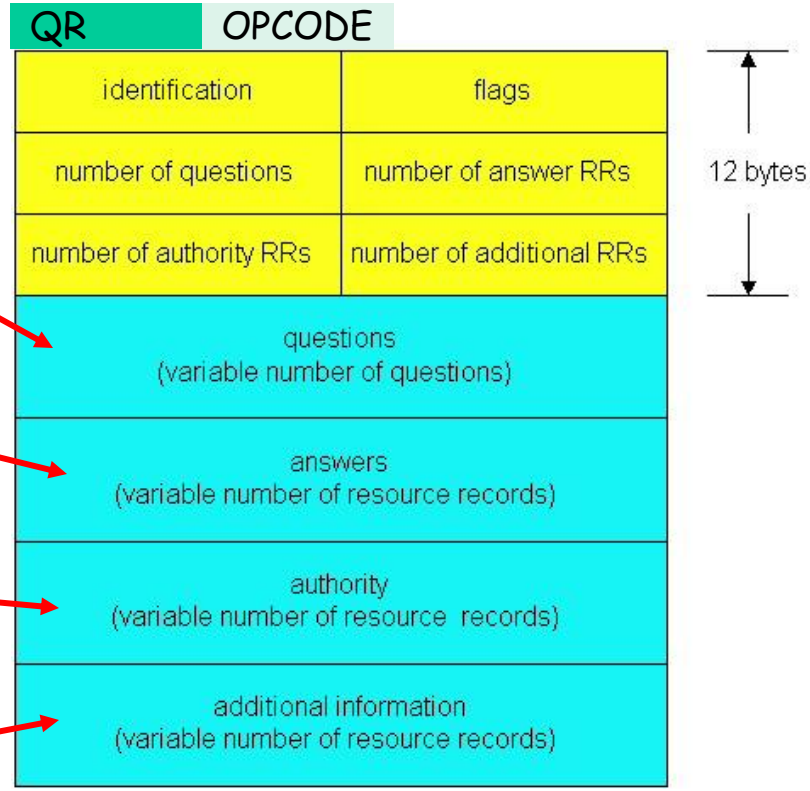
Name, type fields for a query

RRs in response to query

records for authoritative servers

Information about nameserver

additional "helpful" info that may be used



In a query, only number of questions will be >0

DNS Query

DNS: distributed db storing resource records (RR)

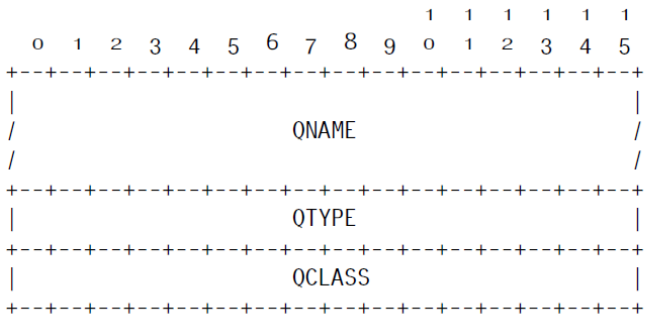
Answer format: **(name, type, class,ttl,RDATA=addr)**

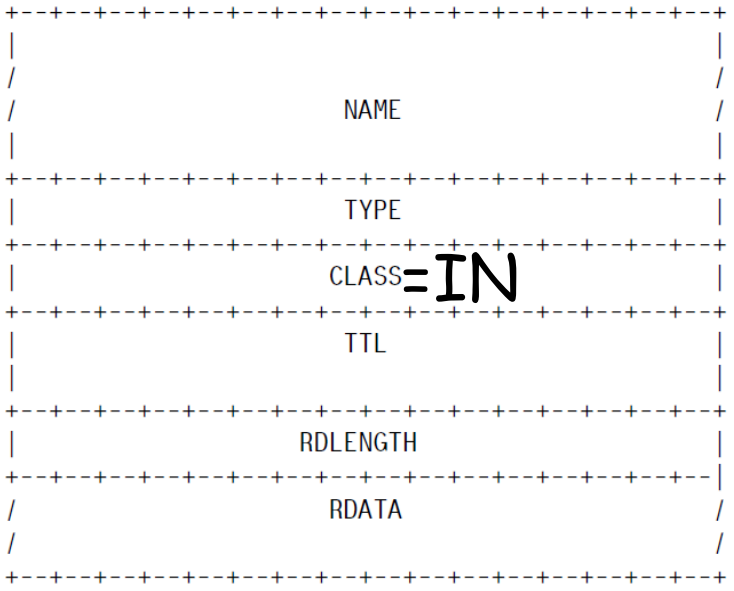
- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=AAAA
  - **name** is hostname
  - **value** is IPv6 address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=CNAME
  - **name** is alias name for some "canonical" (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**

DNS Question/Answer





TTL: Time to Live: how long is the entry valid

RDATA: IP address

DNS Record example

RRs in response to query

NAME	ilab.cs.rutgers.edu
TYPE	A
CLASS	IN
TTL	3600
ADDRESS	"128.6.4.101"

records for authoritative servers

Information about nameserver

NAME	cs.rutgers.edu
TYPE	NS
CLASS	IN
TTL	3600
NSDNAME	ns1.rutgers.edu

RRs in additional section

NAME	ns1.rutgers.edu
TYPE	A
CLASS	IN
TTL	3600
ADDRESS	216.239.32.10 if AAAA 2001:4860:4802:32::a

DNS

DNS services

- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names
- Mail server aliasing
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

Bootstrapping DNS

- How does a host contact the name server if all it has is the name and no IP address?
- IP address of at least 1 nameserver must be given a priori
  - or with another protocol (DHCP, bootp)

  - File `/etc/resolv.conf` in unix

  - `Start -> settings-> control panel-> network ->TCP/IP -> properties` in windows

Themes

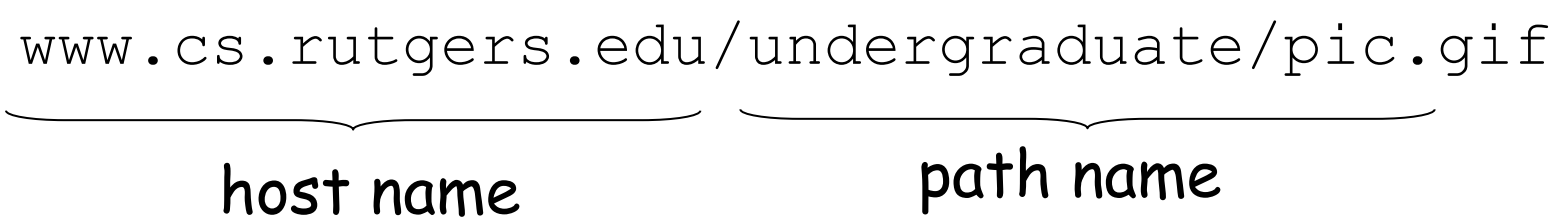
- Request/response nature of these protocols
- How Messages are structured
  - HTTP, SMTP, FTP - simple ASCII protocols
- Caching
- Name Lookup
  - Division of concerns (e.g. zones)
  - Hierarchy structure


http://www.

Web and HTTP

First some jargon

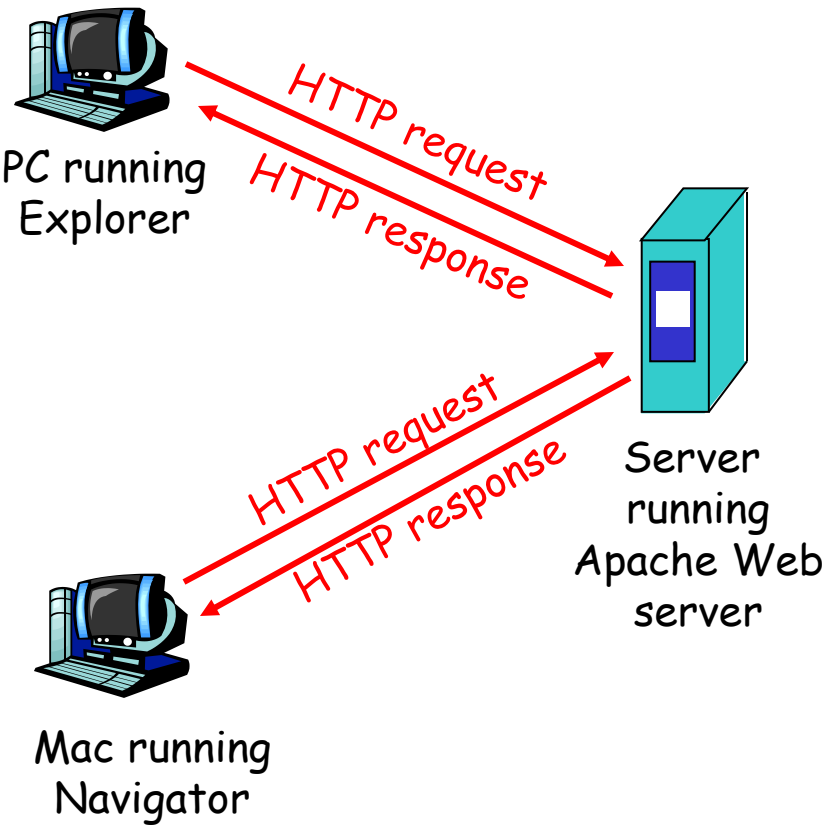
- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:



HTTP overview

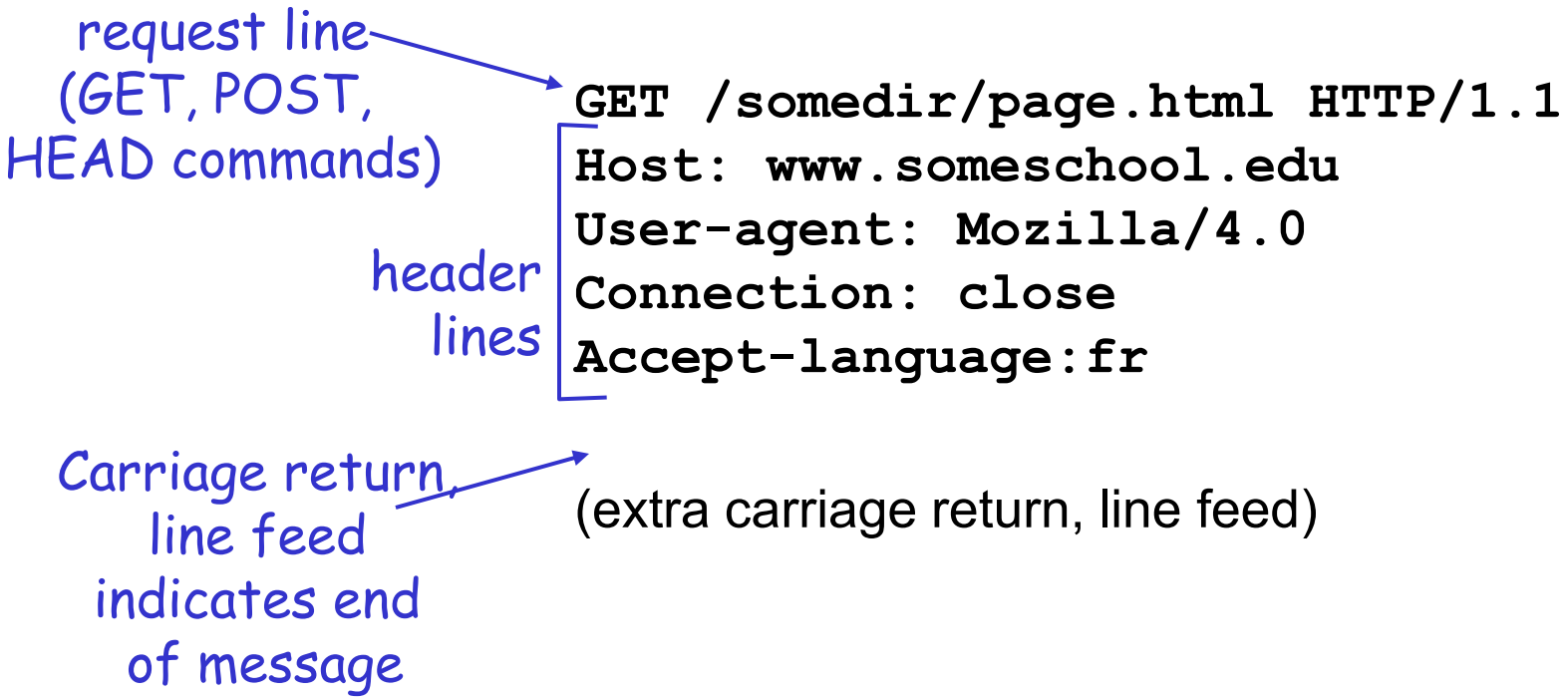
HTTP: hypertext transfer protocol

- 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

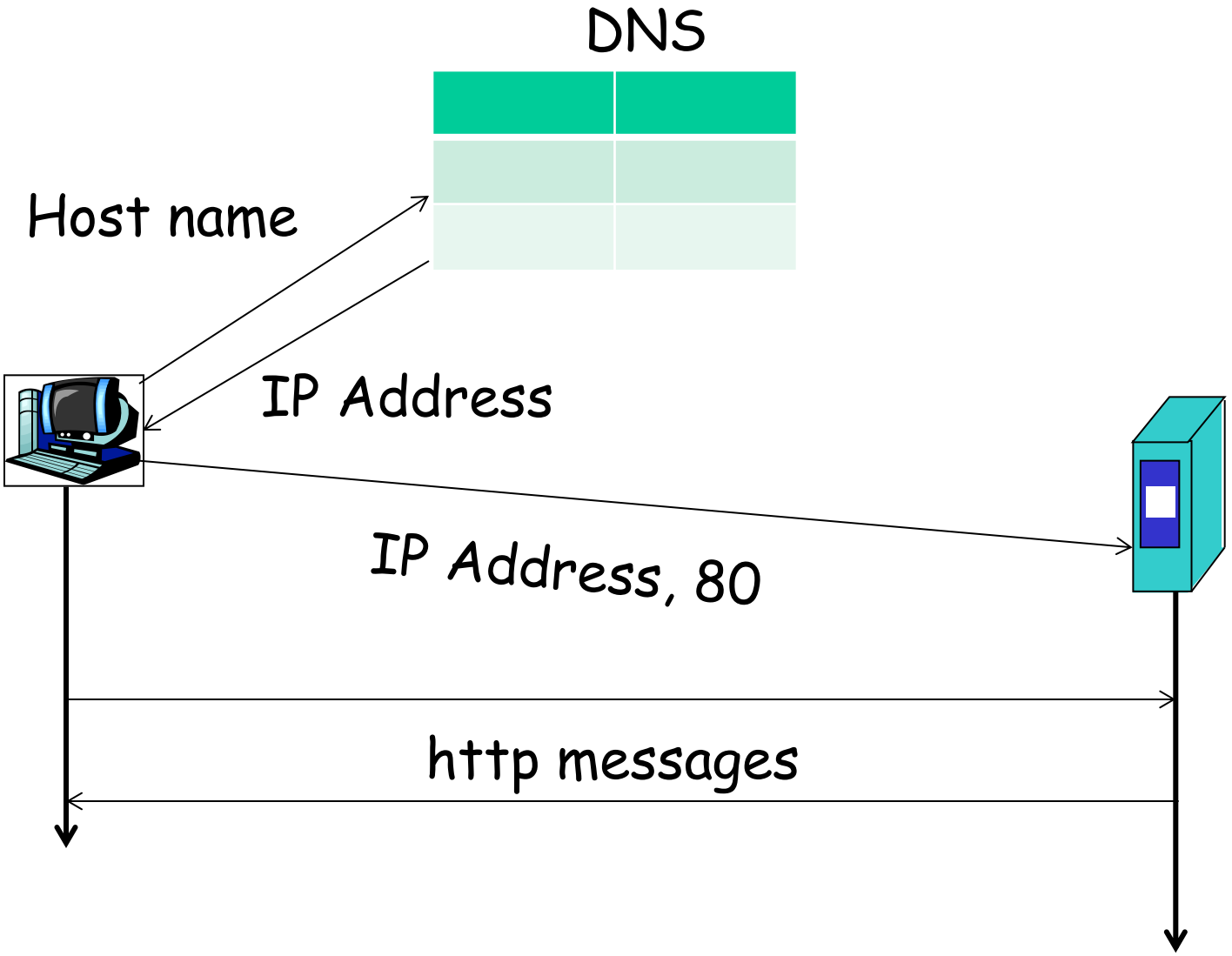


HTTP messages: request message

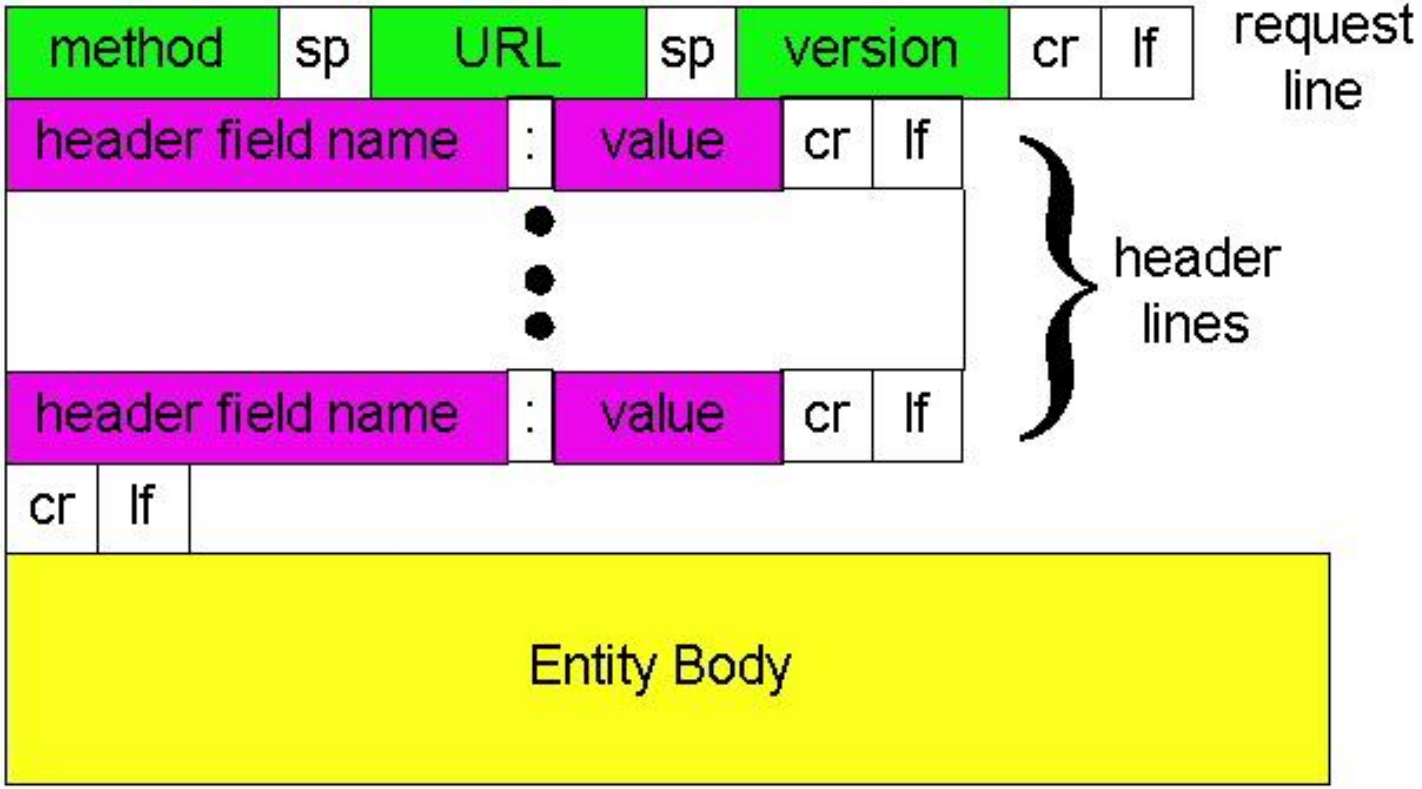
- HTTP request message:
  - ASCII (human-readable format)



Client server connection



HTTP request message: general format



http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14

Method types

- GET
  - Get the file specified in the path URL field in entity body
- POST
  - accept the entity enclosed in the entity body as a new subordinate of the resource identified by the URL field
- HEAD
  - asks server to leave requested object out of response
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

Post method – Upload form input

Post method:

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

URL method:

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

Example: Client POST request

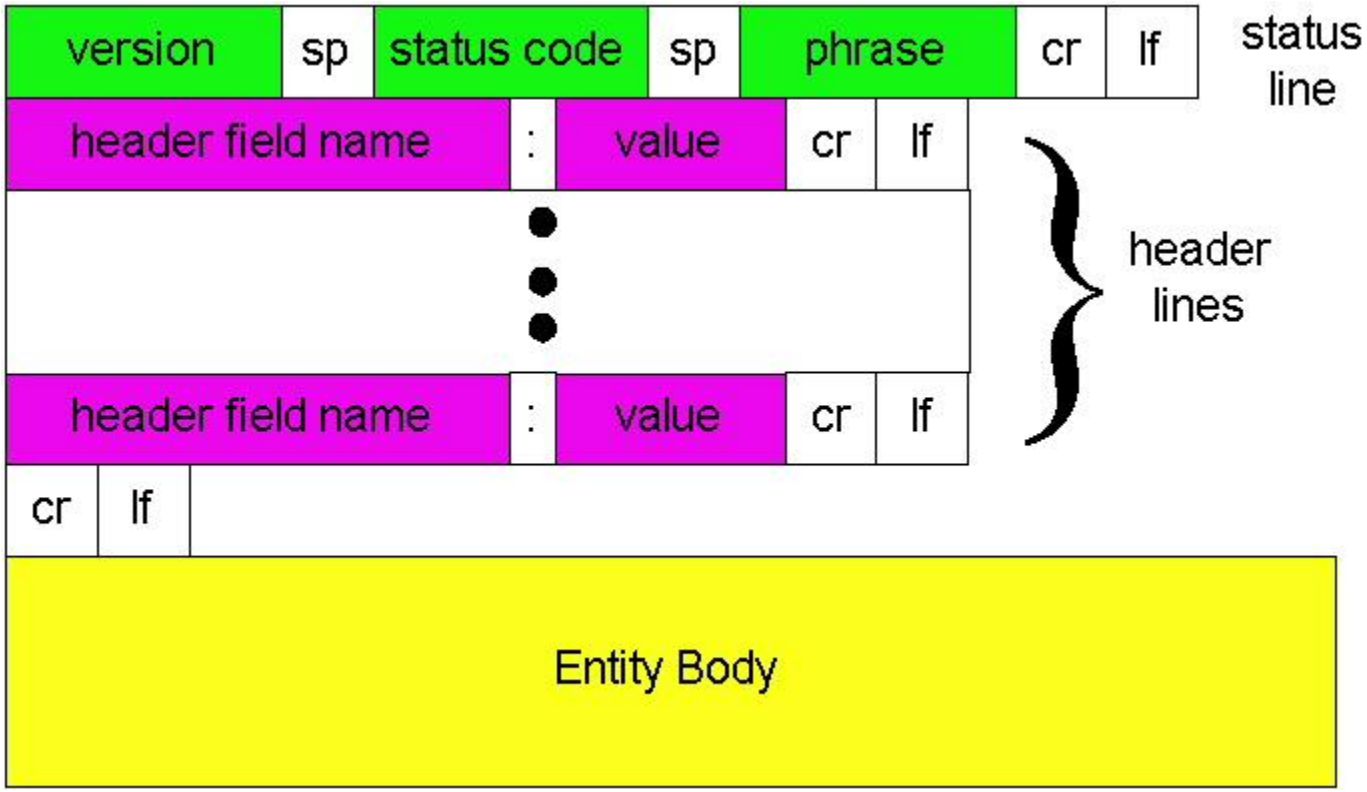
```
POST /cgi-bin/rats.cgi HTTP/1.0
Referer: http://nes:8192/cgi-bin/rats.cgi
Connection: Keep-Alive
User-Agent: Mozilla/4.73 [en] (X11; U; Linux 2.2.12-20 i686)
Host: nes:8192
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
   image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Content-type: application/x-www-form-urlencoded
Content-length: 93

Account=cs111fall&First=Alice&Last=White&SSN=123456789&Bday=01
   011980&State=CreateAccount
```

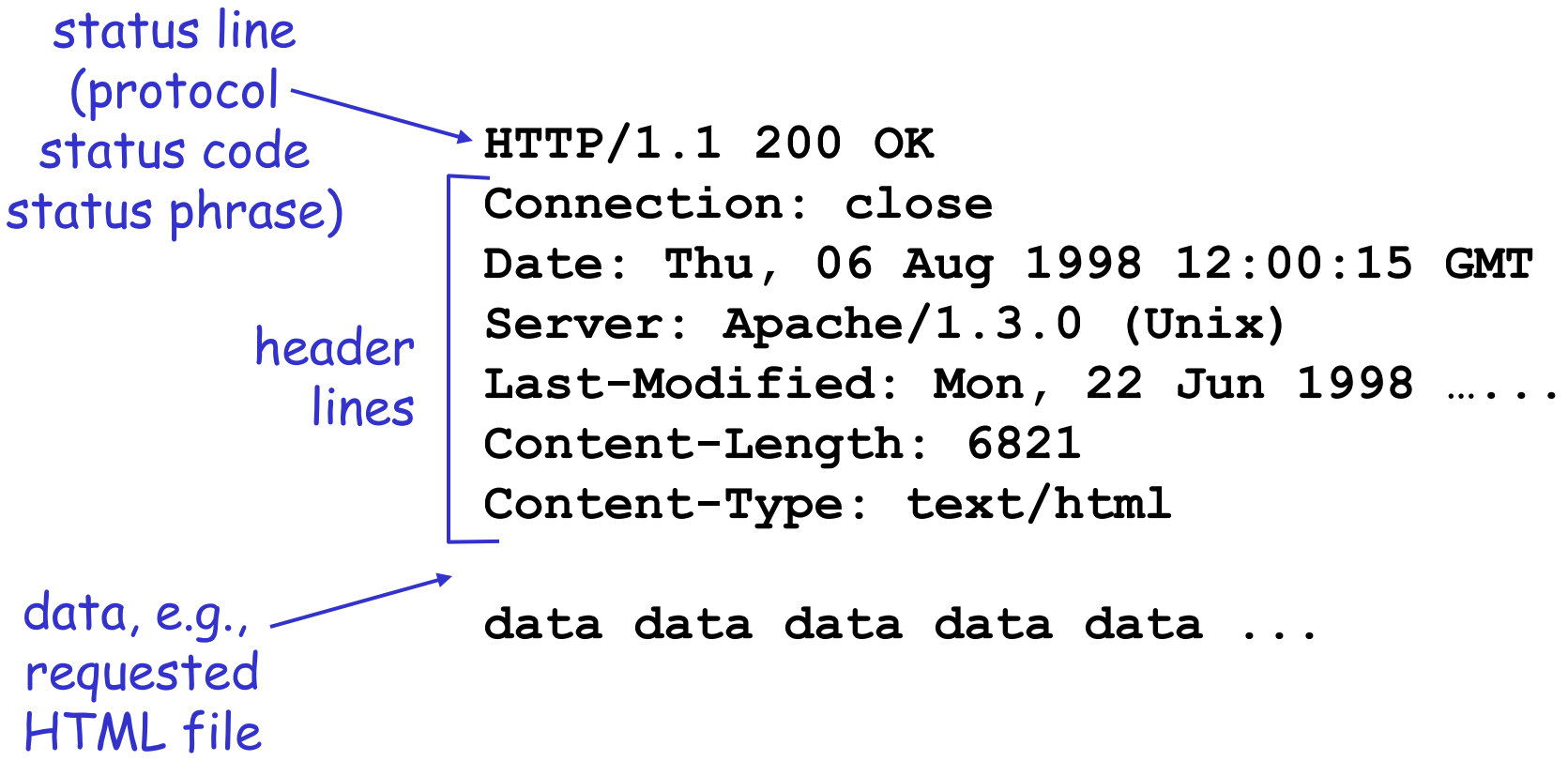
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14

http response message: general format

Unlike http request, No method name



HTTP message: response message



HTTP response status codes

In first line in server->client response message.

A few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

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

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet www.cs.rutgers.edu 80
```

Opens connection to port 80 (default HTTP server port). Anything typed in sent to port 80 at www.eden.rutgers.edu

2. Type in a GET HTTP request:

```
GET /~badri/index.php
Host: www.eden.rutgers.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!

Additional about HTTP

- Persistent vs. Nonpersistent HTTP connections
- Cookies (User-server state)
- Web caches

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

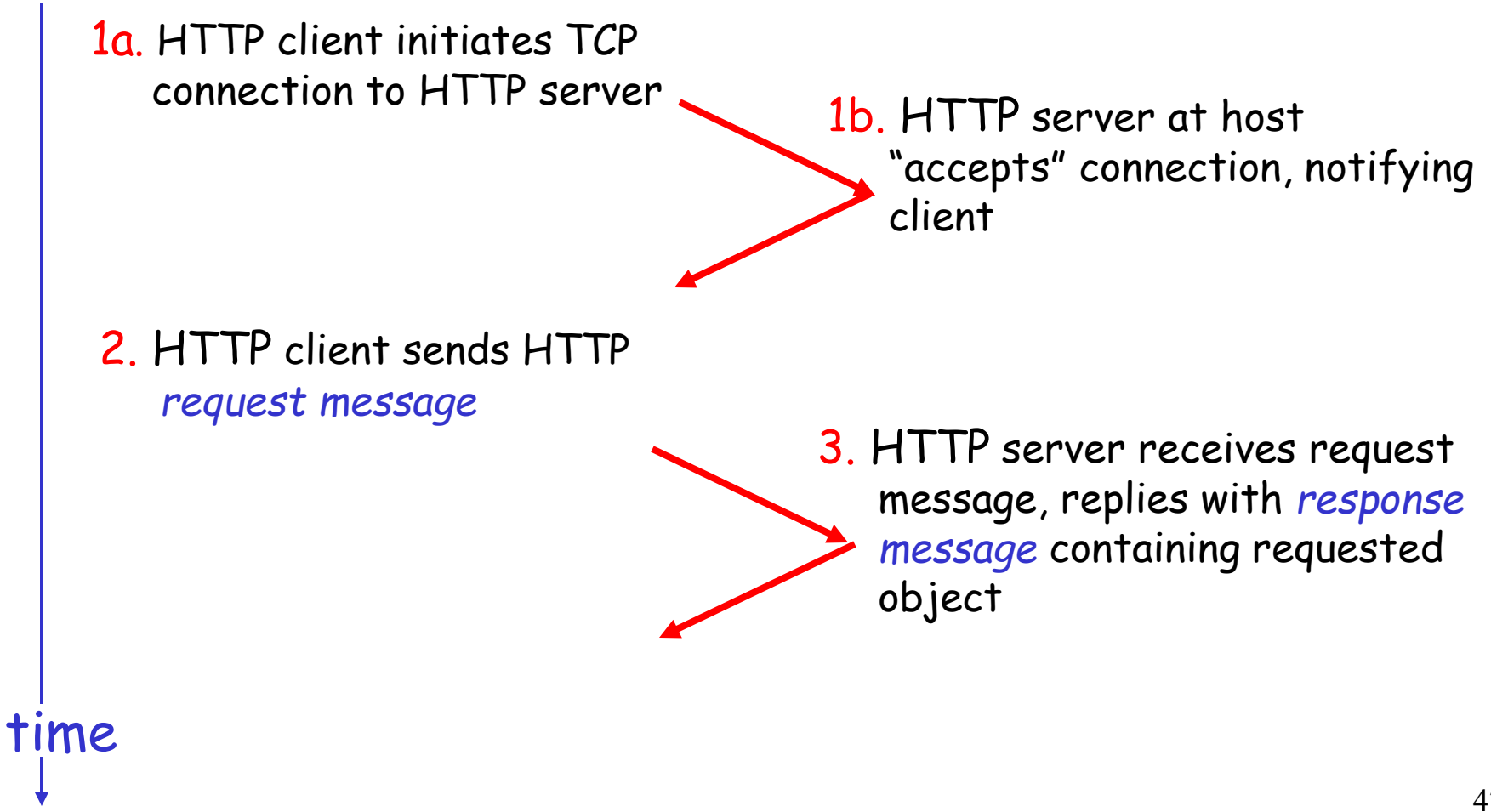
TCP is a kind of communication service provided by the transport layer. It requires the connection to be set up before data communication.

Nonpersistent HTTP

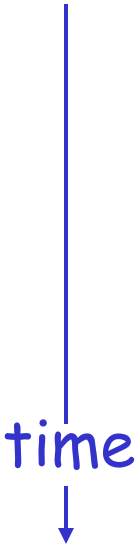
Suppose user enters URL

`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



Nonpersistent 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

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

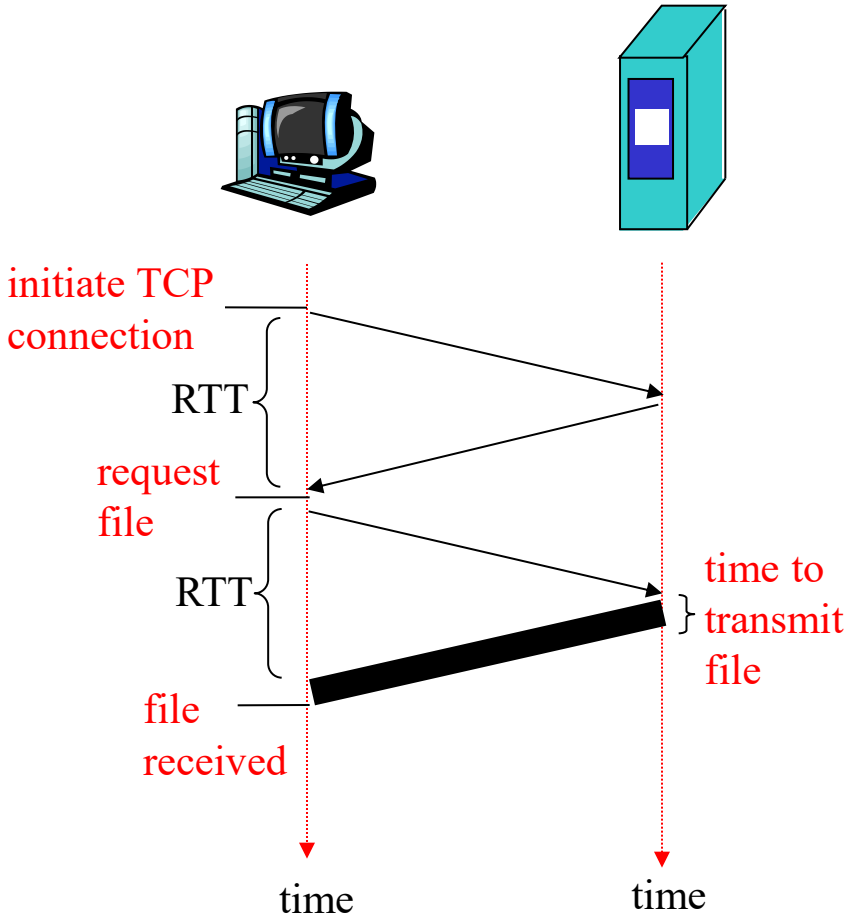
HTTP: Response time

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



Persistent vs. Nonpersistent

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- Browsers can 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

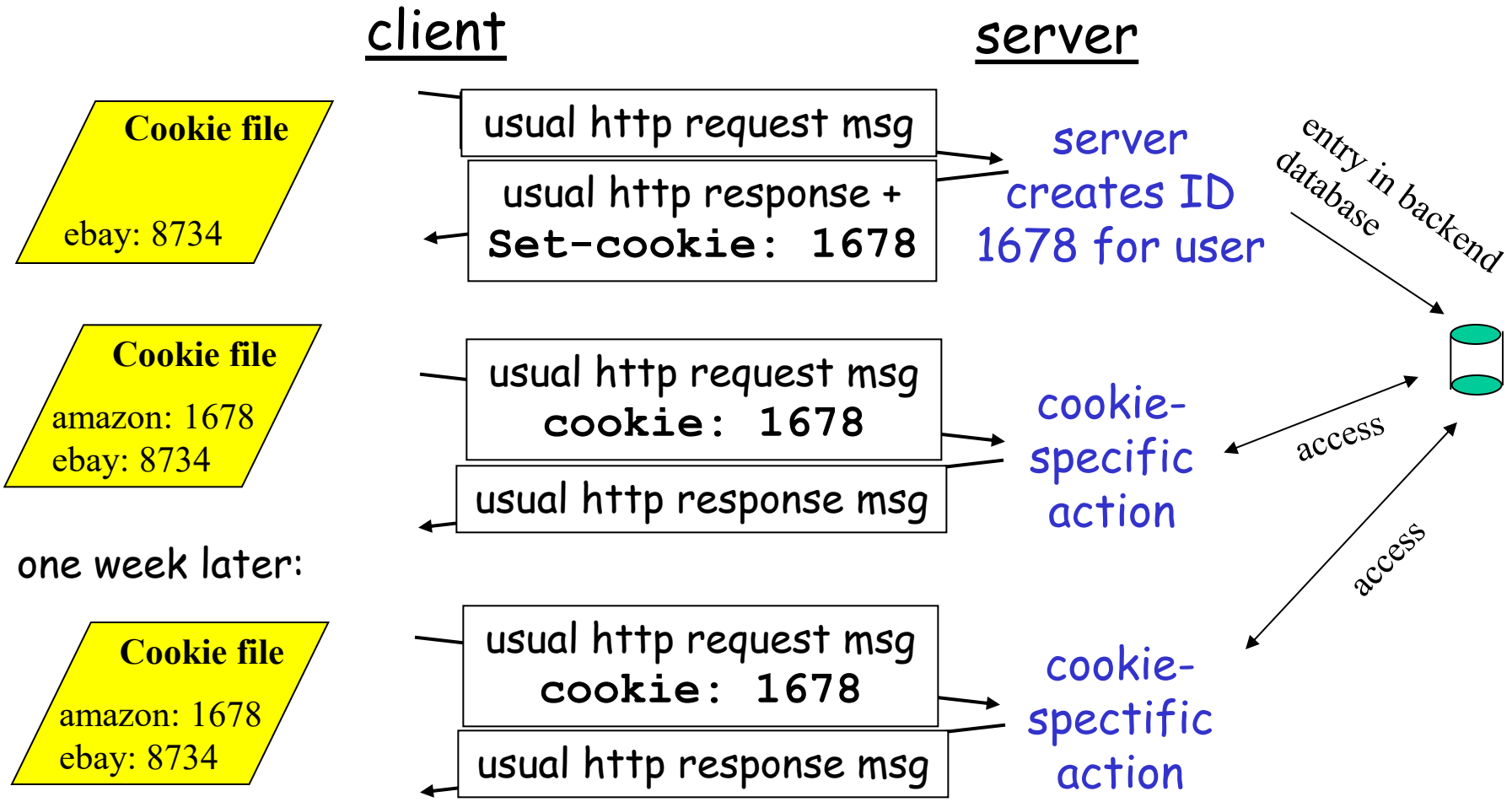
HTTP: user-server state

HTTP is "stateless"

- server maintains no information about past client requests

- What state can bring:
  - authorization
  - shopping carts
  - recommendations
  - user session state

Cookies: keeping "state"



Cookies (continued)

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

Cookies (continued)

aside

Cookies and privacy:

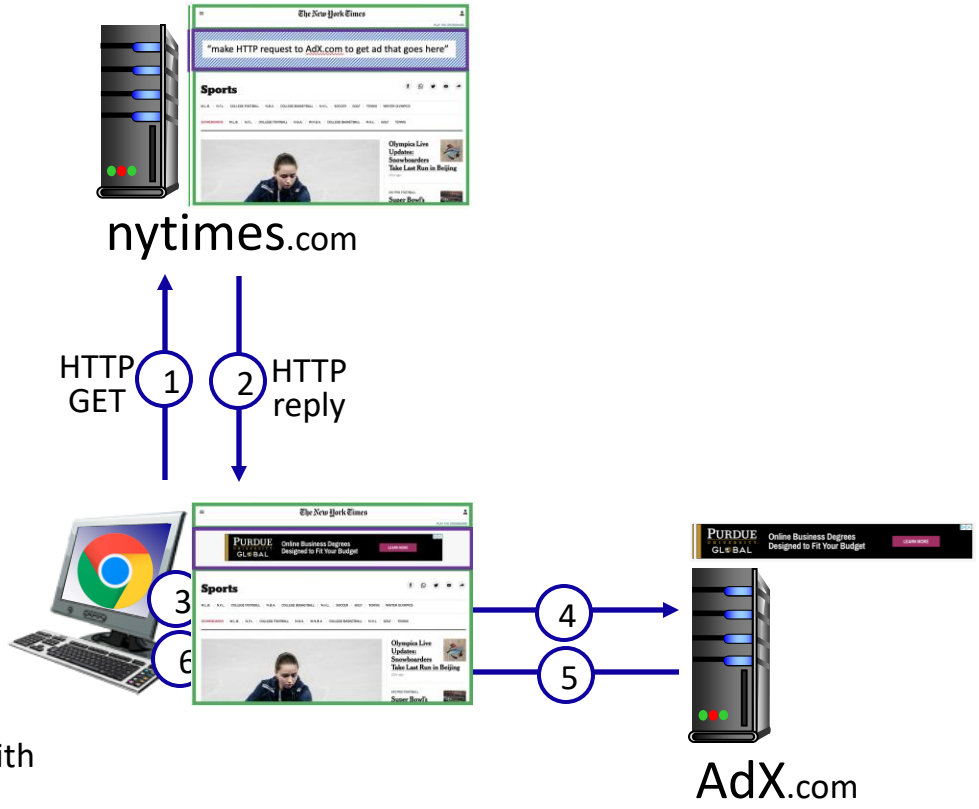
- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

Example: displaying a NY Times web page

1 2 GET base html file from nytimes.com

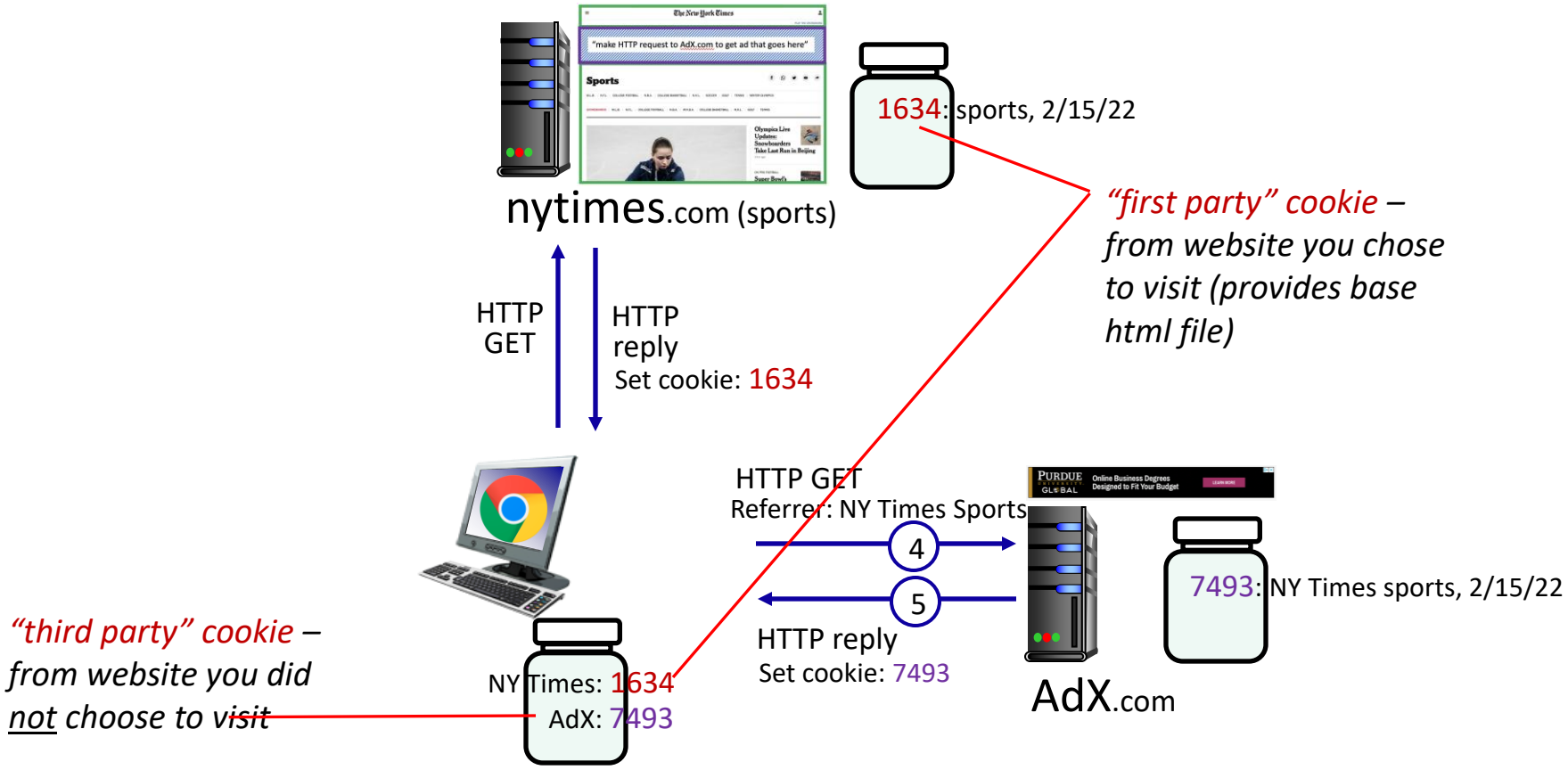
4 5 fetch ad from AdX.com

7 display composed page

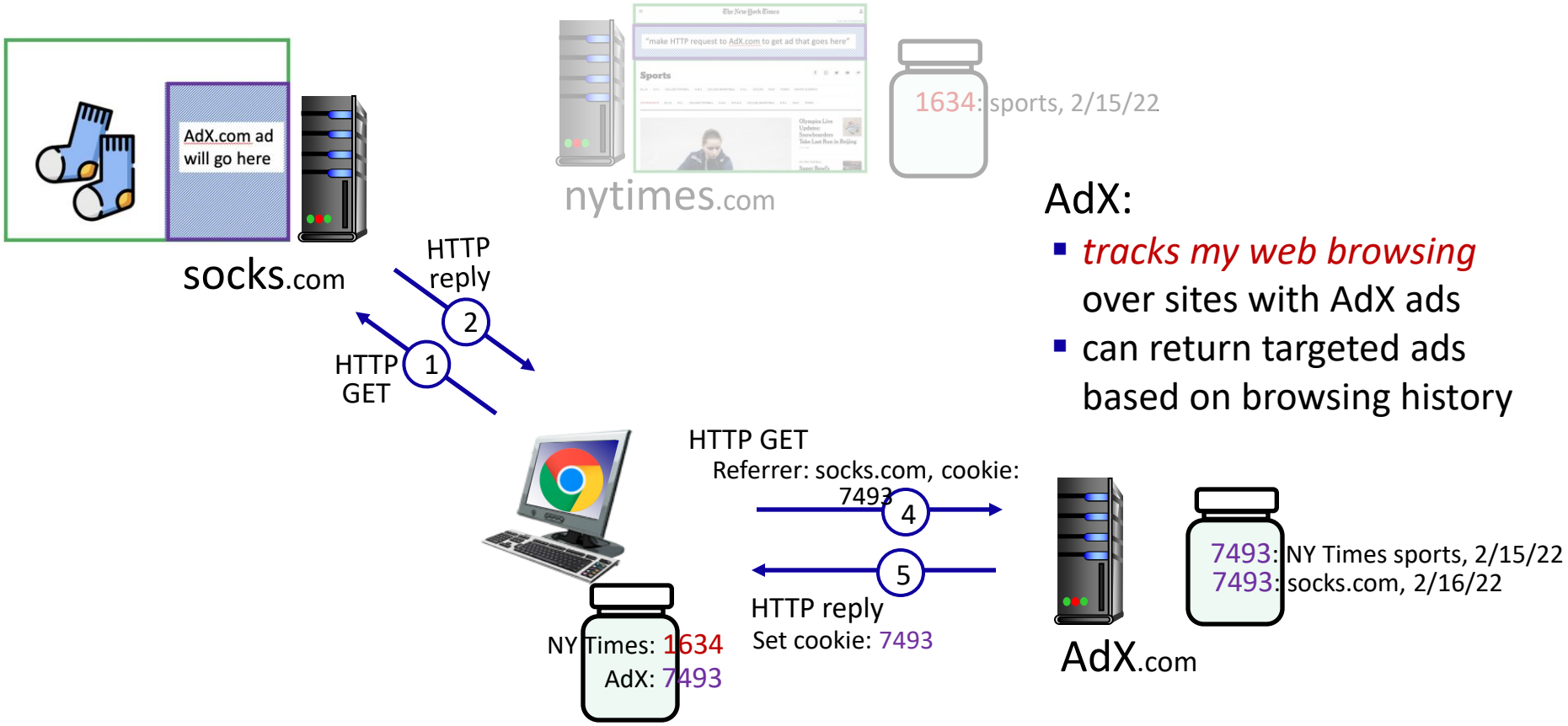


NY times page with embedded ad displayed

Cookies: tracking a user's browsing behavior



Cookies: tracking a user's browsing behavior



AdX:

- *tracks my web browsing* over sites with AdX ads
- can return targeted ads based on browsing history

Cookies: tracking a user's browsing behavior (one day later)

1634: sports, 2/15/22
1634: arts, 2/17/22

HTTP GET
cookie: 1634

HTTP reply
Set cookie: 1634

HTTP GET
Referrer: nytimes.com, cookie: 7493

4

5

HTTP reply
Set cookie: 7493
Returned ad for socks!

NY Times: 1634
AdX: 7493

AdX.com

7493: NY Times sports, 2/15/22
7493: socks.com, 2/16/22
7493: NY Times arts, 2/15/22

Http 1.1 vs http/2 (since 2015)

http 1.1

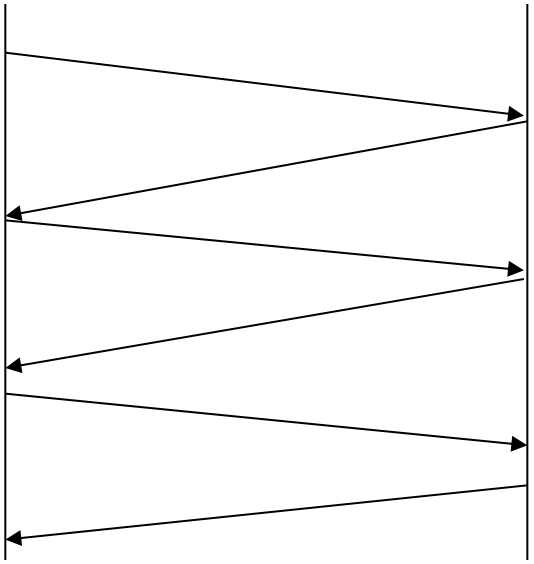
- All request go over a single connection, sequentially
- http requests are sequential
- With multiple objects per page, a large request will create a backlog for other requests and response time suffers

http/2

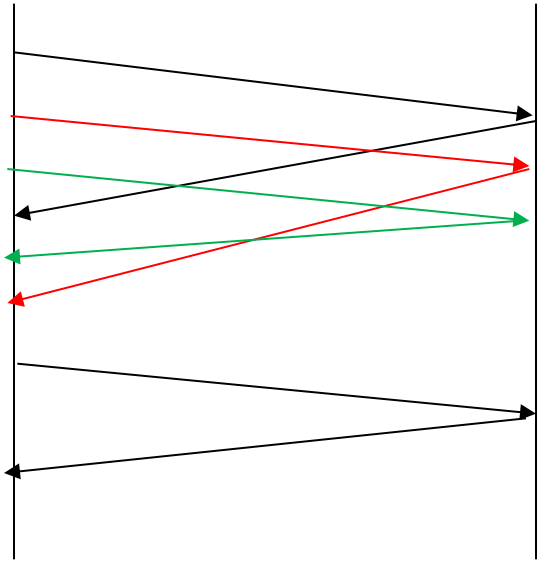
- All requests still use one connection but are multiplexed
- http requests are concurrent
- Responses can be received in any order
- http payload separated into header frames and data frames
- Stream, messages, frames

Http 1.1 vs http/2

http 1.1



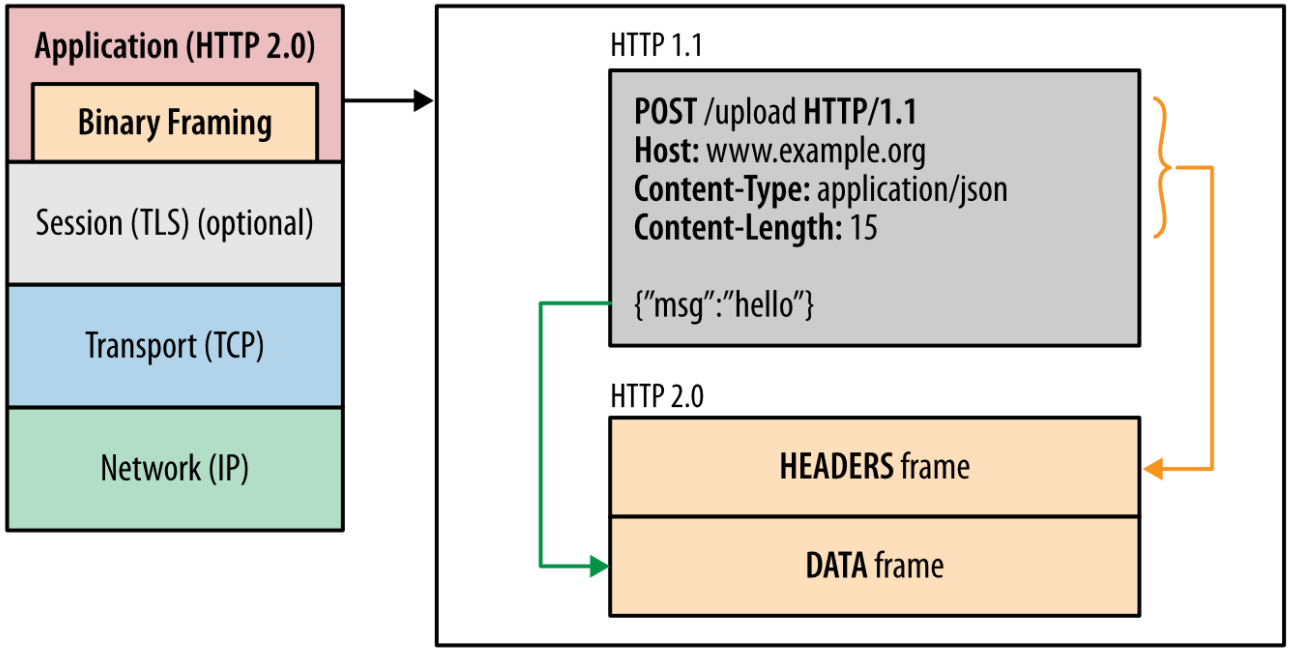
http/2

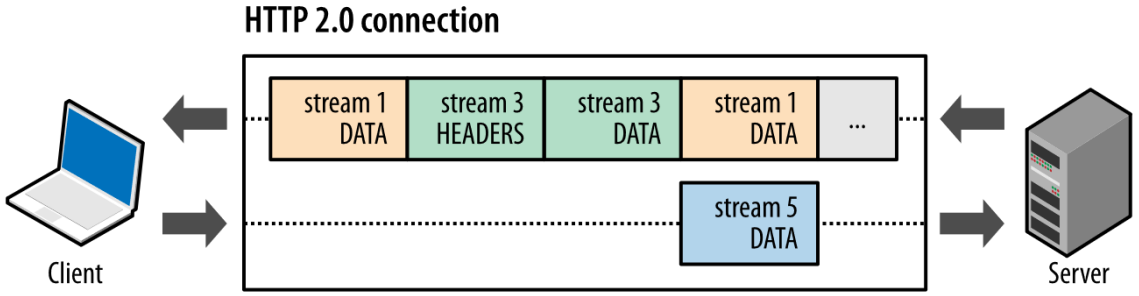


HTTP/2 goals (2015)

- Reduce latency by multiplexing multiple request response
- Minimize overhead by allowing header compression
- Request prioritization
- Server push
- Now (2019) http/3 uses UDP QUIC

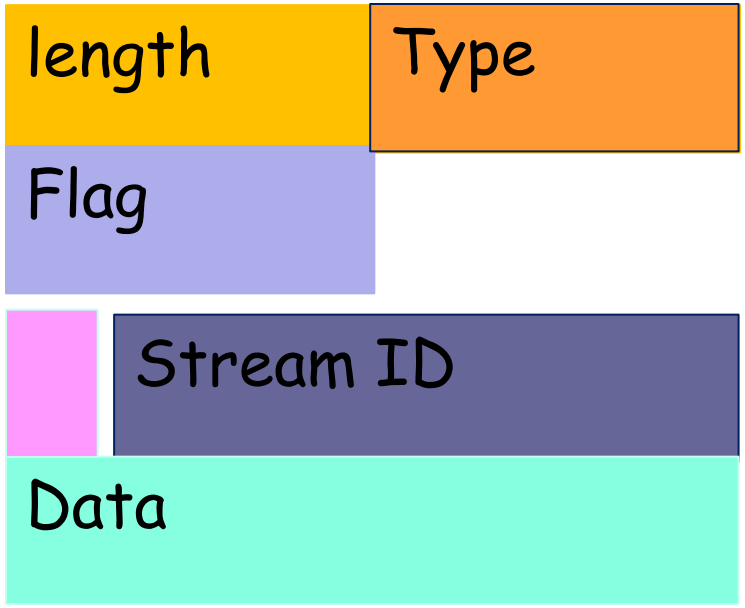
Separate payload into headers and data





https://developers.google.com/web/fundamentals/performance/http2

Frame Structure

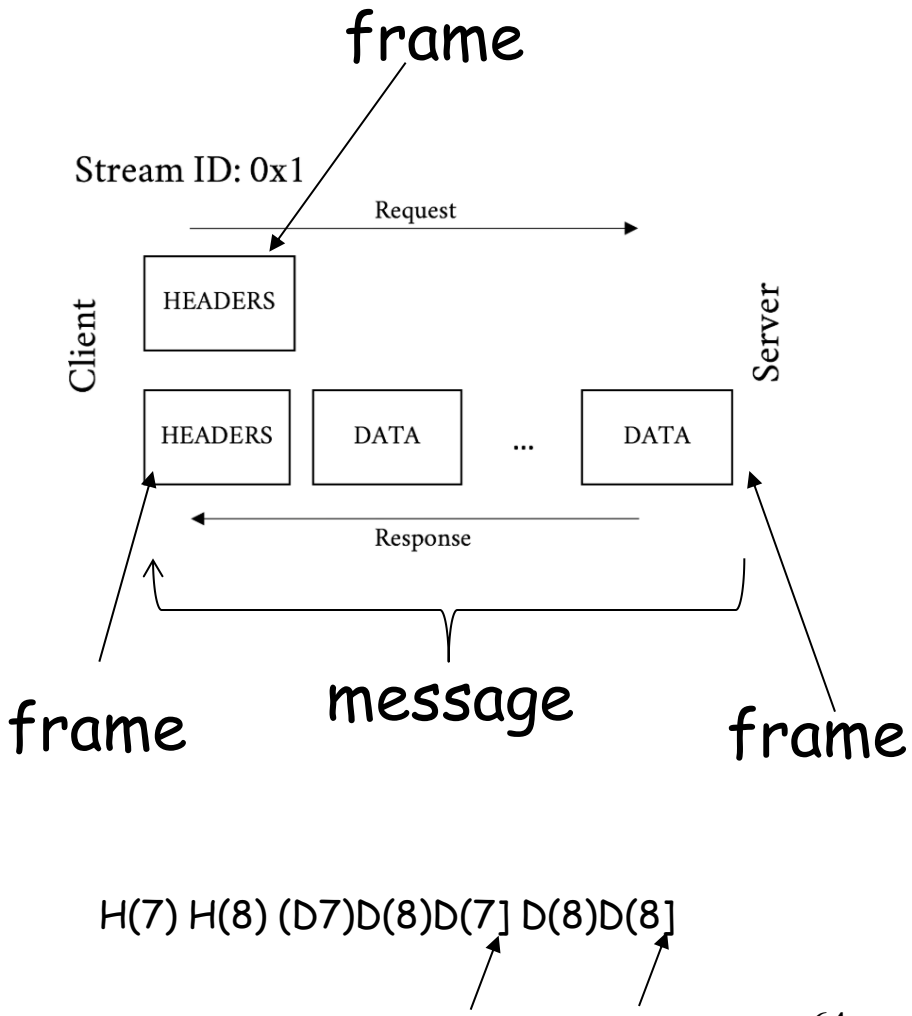


- Length: indicates the length of the frame. A frame is usually less than 2^14 bits but can be up to 2^24 bits.
  - Therefore, its size usually is less than 16K.
- Type indicated the frame's type, such as data frames (HEADERS frame, DATA frame) and flow-control frames (SETTINGS frame, PRIORITY frame, etc.).
- HTTP/2 defines 10 types, but it can be up to 2^8 types. You can define custom types when needed.
- Flag field: for simple flow control, such as END_HEADERS indicating the end of the headers data.
- Stream ID marks the stream identity. The identifier can be up to 2^31, and its most significant bit is reserved..
- Data: is the frame payload.

https://developers.google.com/web/fundamentals/performance/http2

Streams, messages and frames

- Stream: a bidirectional flow of bytes between client and server or vice versa
- Message: A complete sequence of frames(header frame plus zero or more data frames) that make up a request or response
- Frame: smallest unit of communication with at least a header identifying the stream # (32 bits)



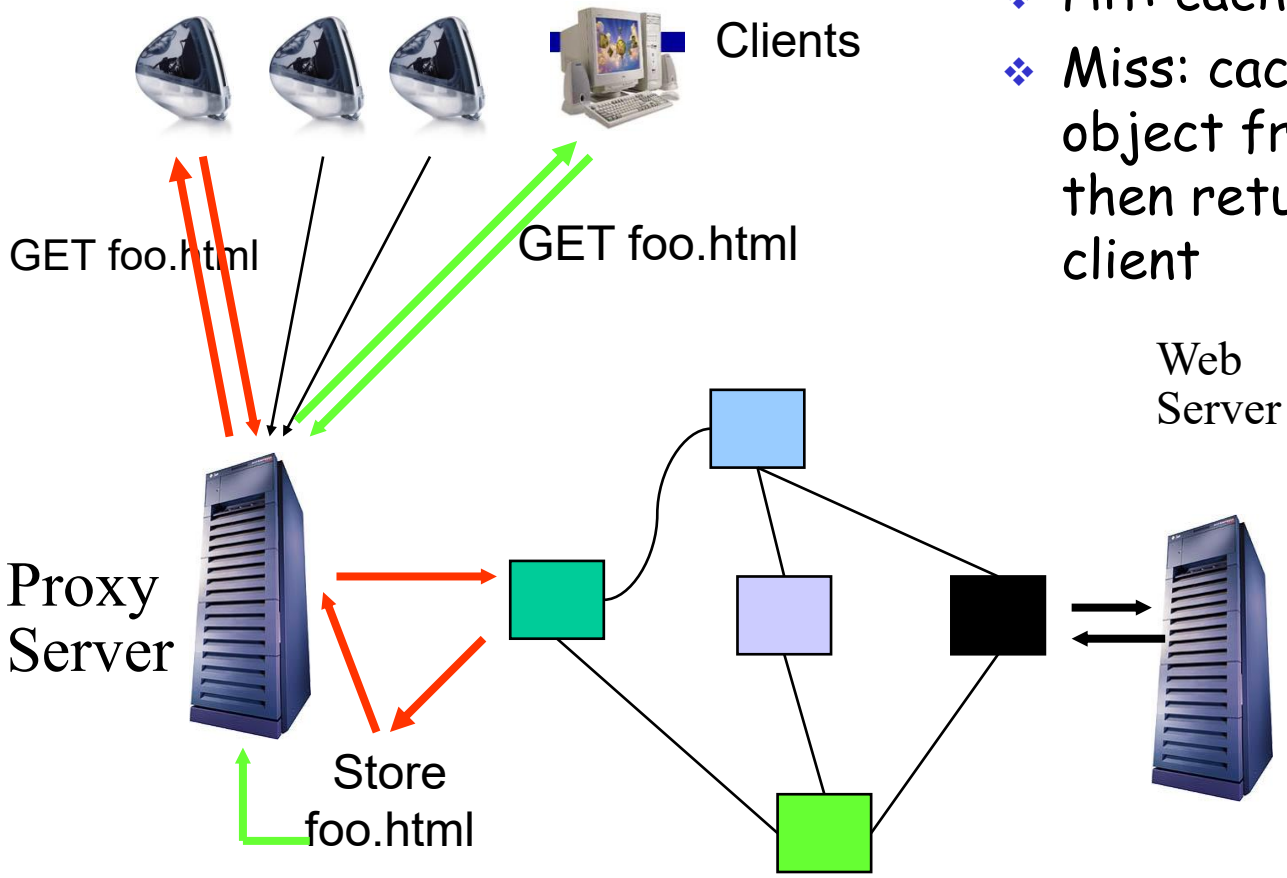
Web caches (proxy server)

Why?

- Reduce response time for client request.
- Reduce traffic on an institution's access link.

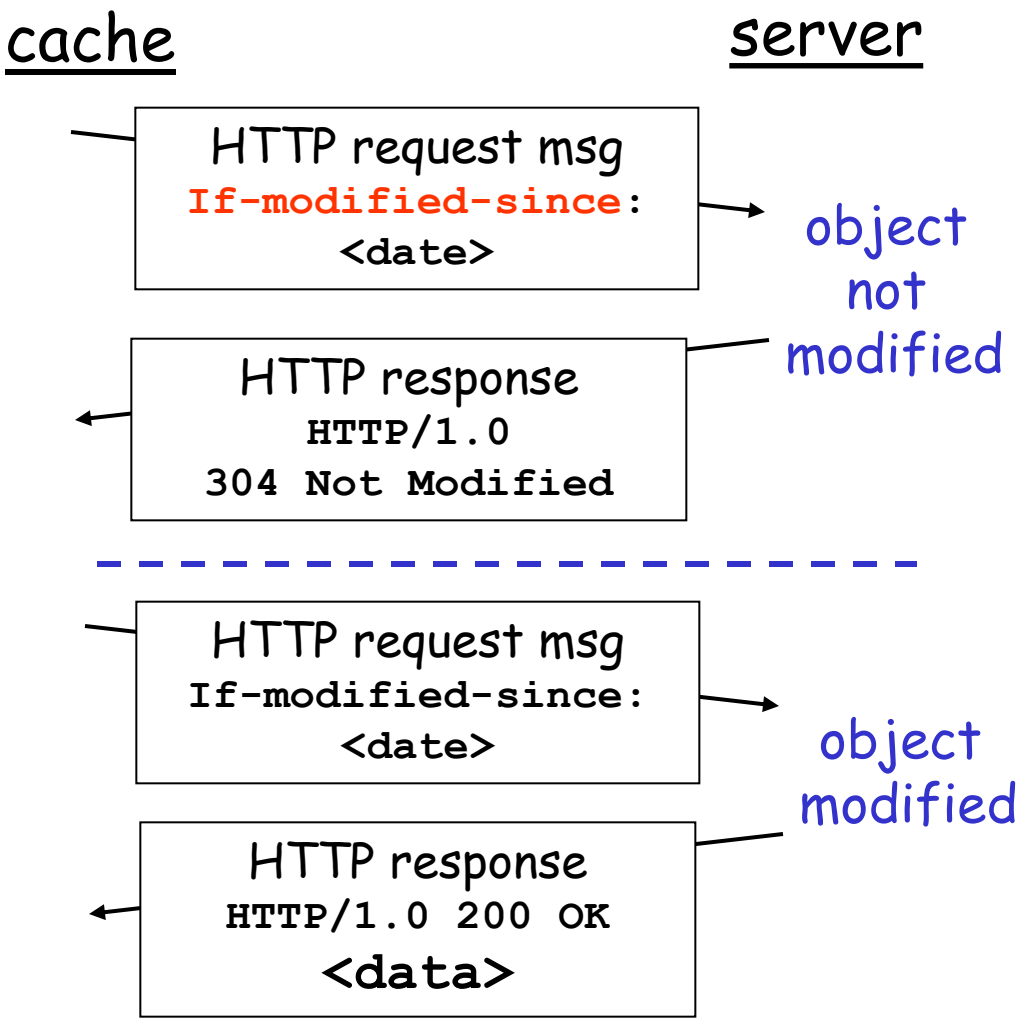
Web caches (proxy server)

- browser sends all HTTP requests to cache
  - Hit: cache returns object
  - Miss: cache requests object from origin server, then returns object to client



Web caches: implementation

- Conditional Get guarantees cache content is up-to-date while still saves traffic and response time whenever posssible



Content Distribution Networks (CDN)

Why?

- Reduce bandwidth requirements of content provider
- Reduce $$ of maintaining Servers
- Cache for server content
- Reduce traffic on the link to the content provider.
- Improve response time to user

Content distribution networks (CDNs)

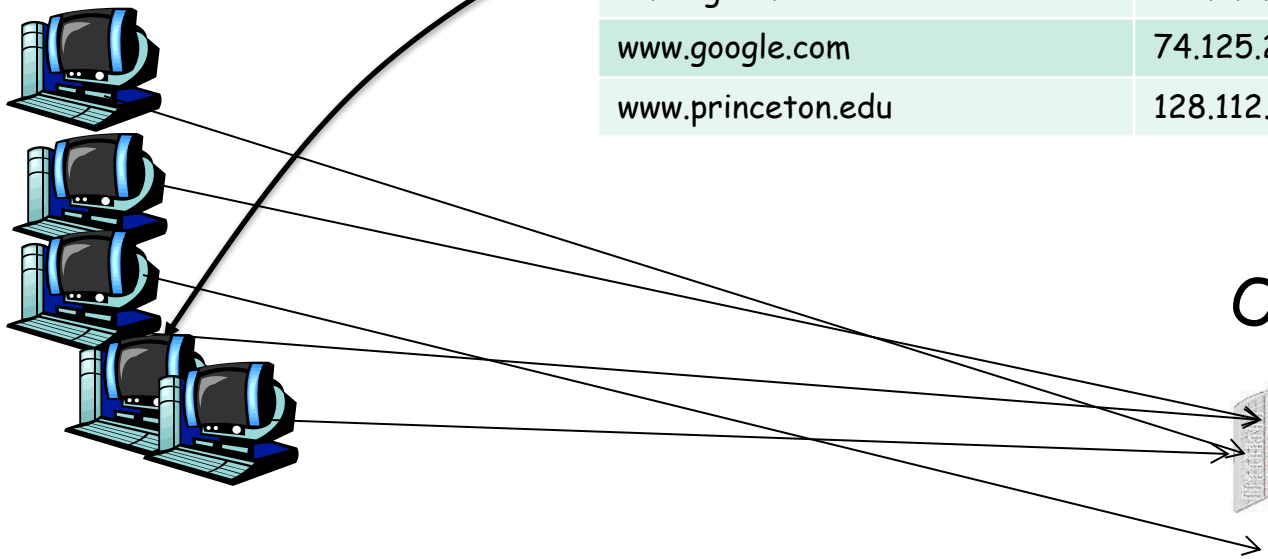
challenge: how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long (and possibly congested) path to distant clients

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

Without CDN

DOMAIN NAME	IP ADDRESS
WWW.instagram.COM	31.13.71.124
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



Origin Server

31.13.71.124

- Huge B/W requirements
- Does not scale
- So, Distribute content to geographically distributed servers
- Use DNS to redirect request to copies user content

Content distribution networks (CDNs)

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
    - close to users
    - Akamai: 240,000 servers deployed in > 120 countries (2015)
  - *bring home:* smaller number (10's) of larger clusters in POPs near access nets
    - used by Limelight

CDN terms

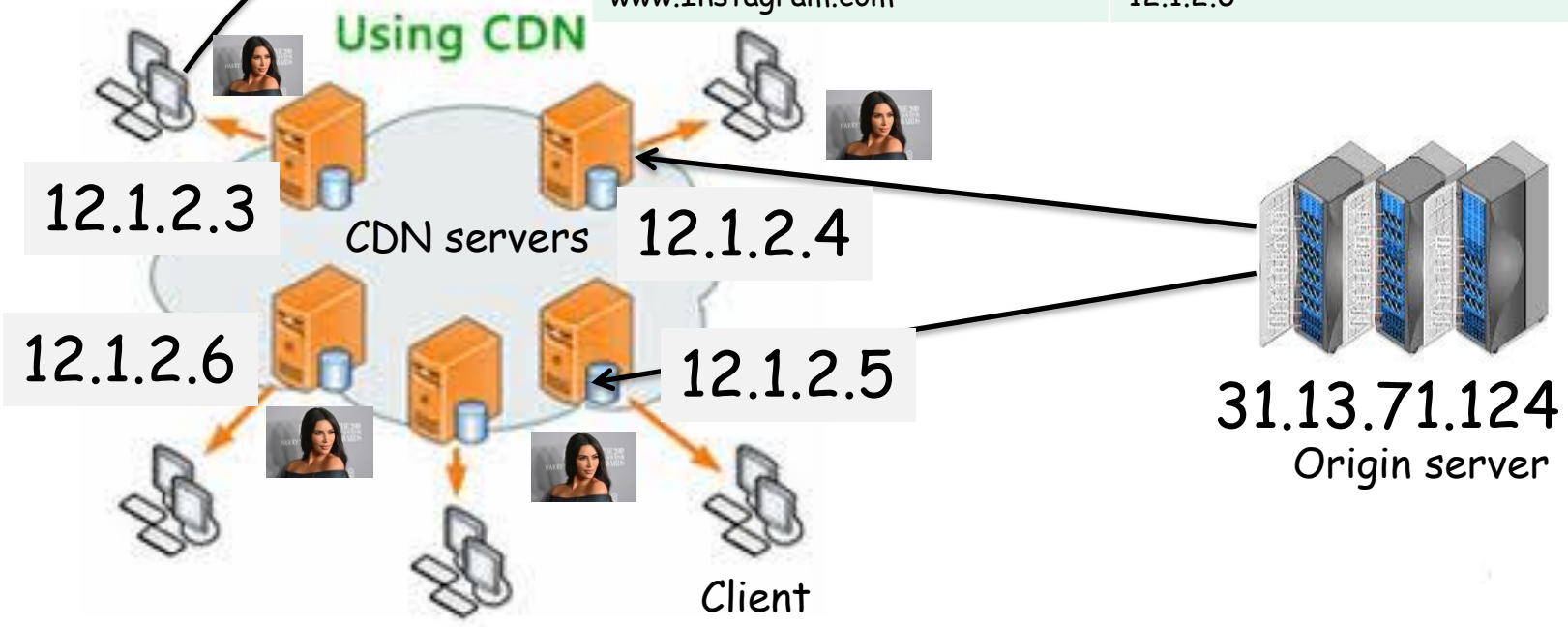
- Origin server
  - Server that holds the authoritative copy of the content
- CDN server
  - A replica server owned by the CDN provider
- CDN name server
  - A DNS like name server used for redirection
- Client

With CDN

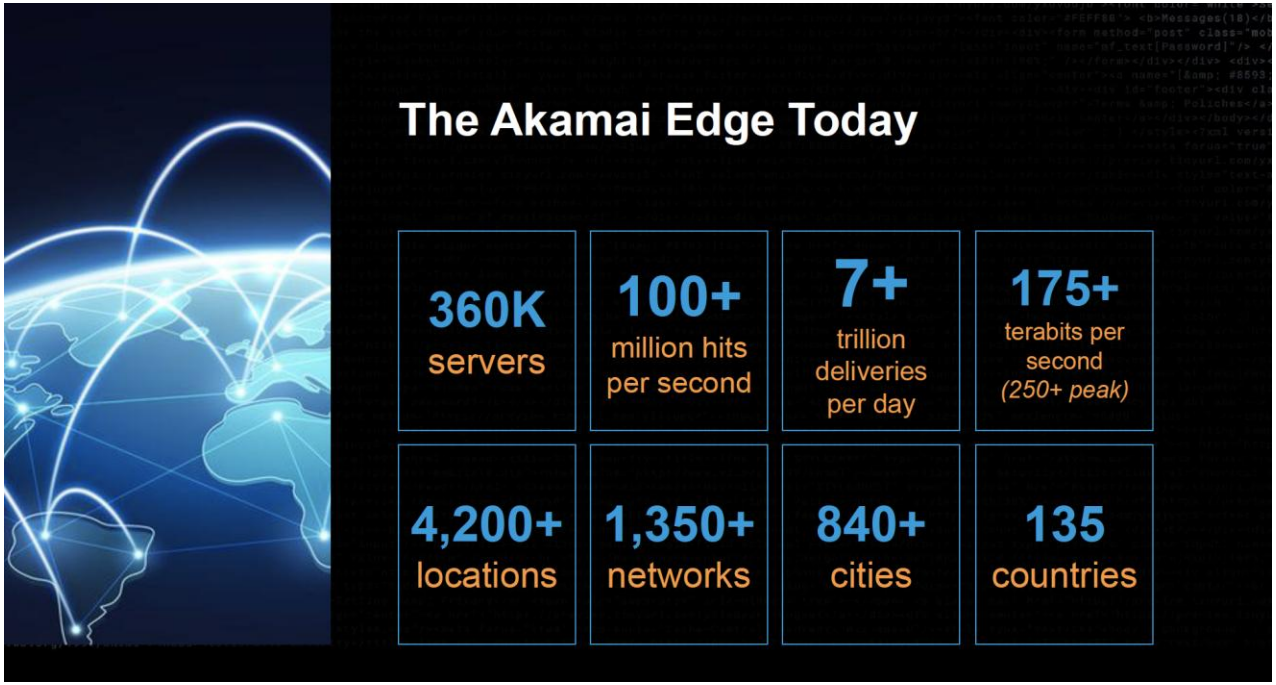
DOMAIN NAME	IP ADDRESS
WWW.instgram.COM	124.8.9.8 (NS of CDN)
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86

CDN Name Server (124.8.9.8)

DOMAIN NAME	IP ADDRESS
WWW.instram.COM	12.1.2.3
www.instagram.com	12.1.2.4
www.instgram.com	12.1.2.5
www.Instagram.com	12.1.2.6



Akamai today:



Source: https://networkingchannel.eu/living-on-the-edge-for-a-quarter-century-an-akamai-retrospective-downloc

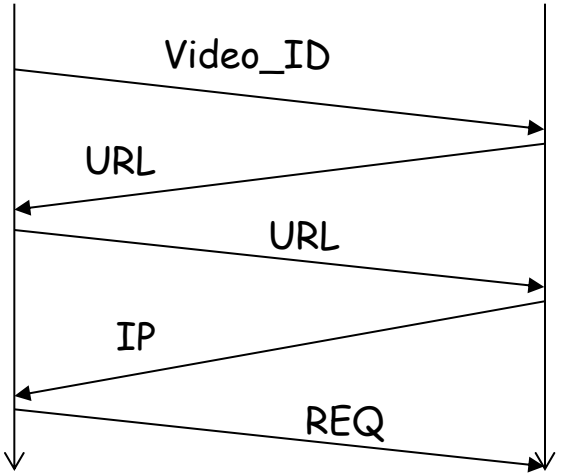
- YouTube video ID namespace

- Each video identifies by 11 character string
- {a-Z,0-9, , -} giving 64^{11} video ID space
  - E.g., UIREuv2UcAN
  - https://youtu.be/H0rCKyc7__U
- Video names are uniformly distributed over the ID space
- Each Video ID mapped to 192 names/URLs

DNS namespace

- YouTube

- Each ID is mapped to URL: v[1-24].lscache.[1-8].c.youtube.com
- E.g., UC552HeREOK → v4.lscache1.c.youtube.com
- ID mapped to 192 DNS host names (may have changed now)
- Each host name returns IP address of CDN server
- CDN resolves each DNS request to multiple IP addresses

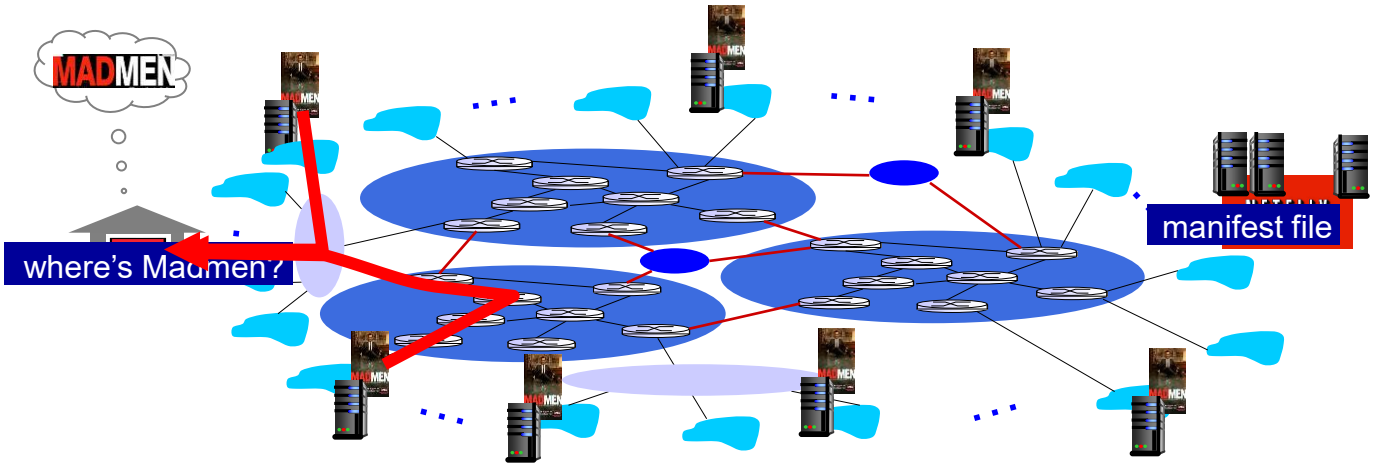


DNS:URL→ IP of (CDN)

IP address of cache server for the video

How does Netflix work?

- Netflix: stores copies of content (e.g., MADMEN) at its (worldwide) OpenConnect CDN nodes
- subscriber requests content, service provider returns manifest
  - using manifest, client retrieves content at highest supportable rate
  - may choose different rate or copy if network path congested

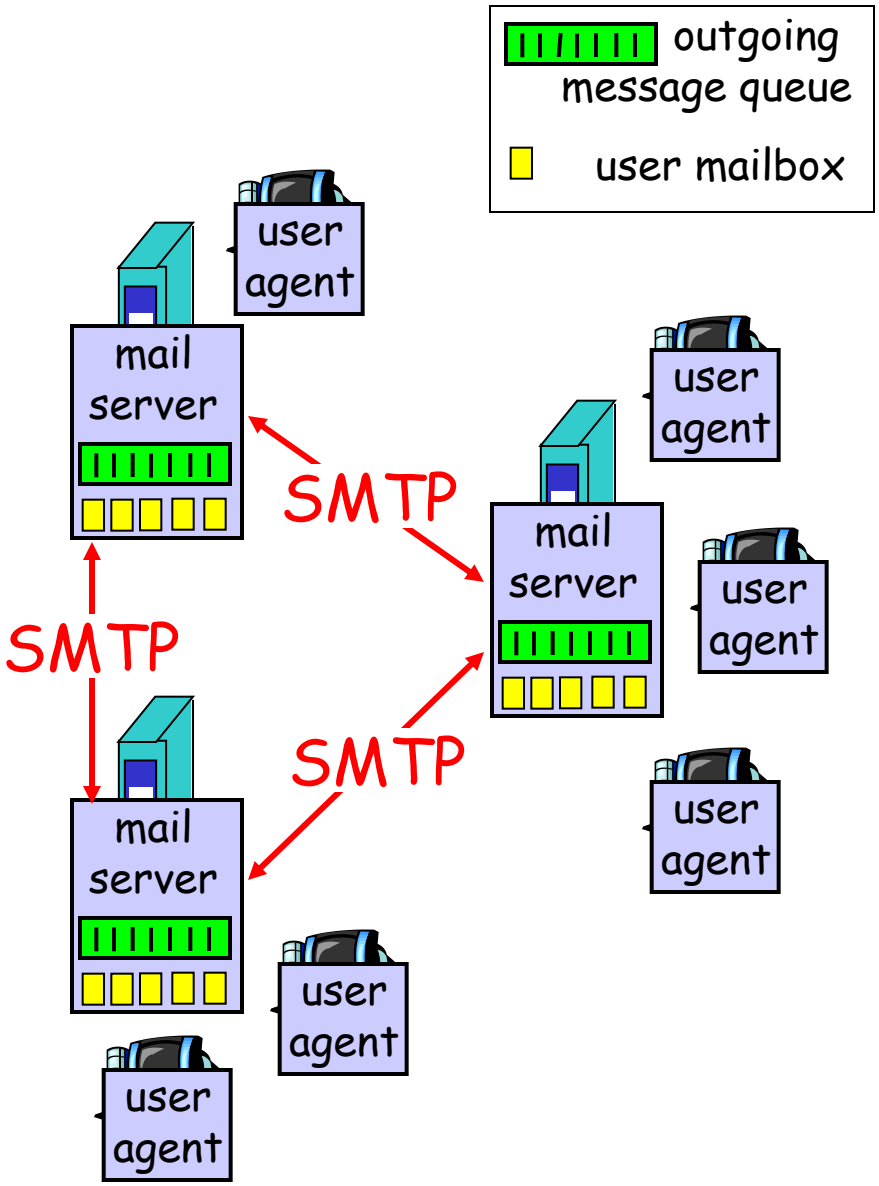



SMTP

Electronic Mail

Three major components:

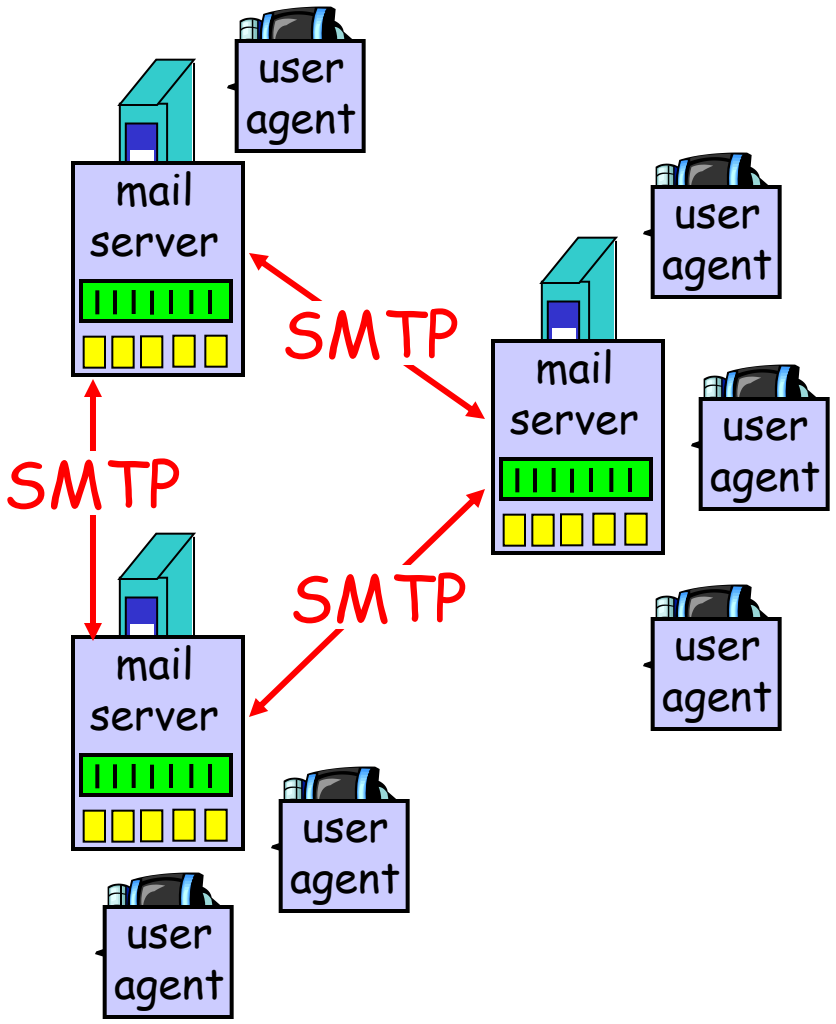
1. user agents
   - a.k.a. "mail reader"
   - e.g., gmail, Outlook, yahoo



Electronic Mail: mail servers

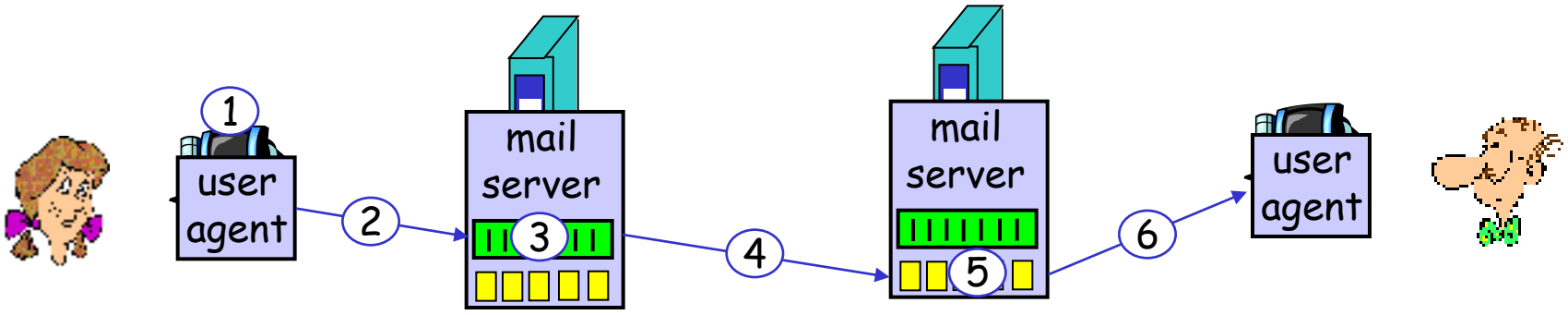
2. Mail Servers
   - mailbox contains incoming messages for user
   - message queue of outgoing (to be sent) mail messages
- Sender mail server makes connection to Receiver mail server
  - IP address, port 25

3. SMTP protocol
   - Used to send messages
   - Client: sending user agent or sending mail server
   - server: receiving mail server



Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "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



Sample SMTP interaction

220 hill.com SMTP service ready

HELO town.com

250 hill.com Hello town.com, pleased to meet you

MAIL FROM: <jack@town.com>

250 <jack@town.com>… Sender ok

RCPT TO: <jill@hill.com>

250 <jill@hill.com>… Recipient ok

DATA

354 Enter mail, end with "." on a line by itself

Jill, I'm not feeling up to hiking today. Will you please fetch me a pail of water?

.

250 message accepted

QUIT

221 hill.com closing connection

MAIL

Table 23.2 *Responses*

Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

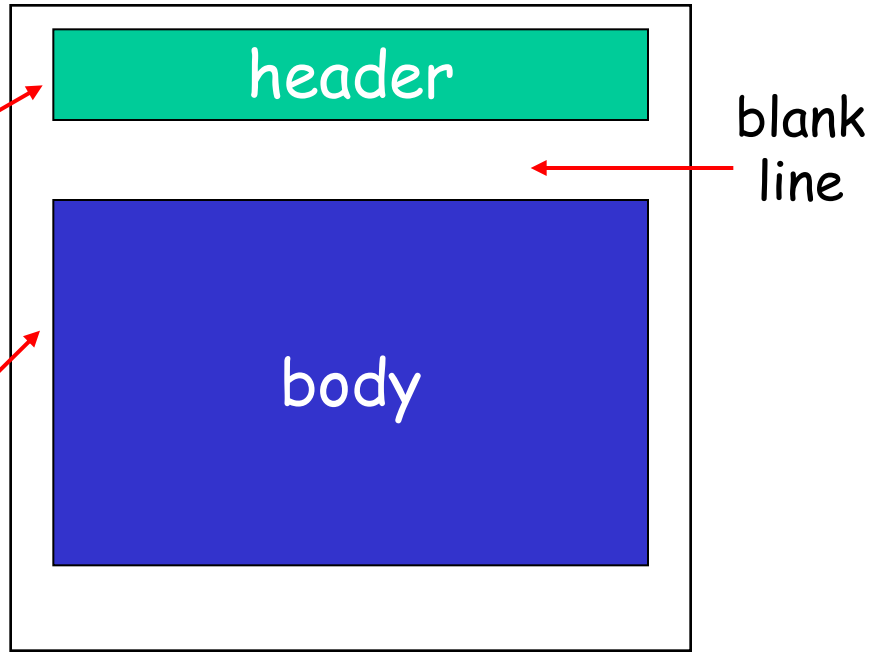
Mail message (stored on server) format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

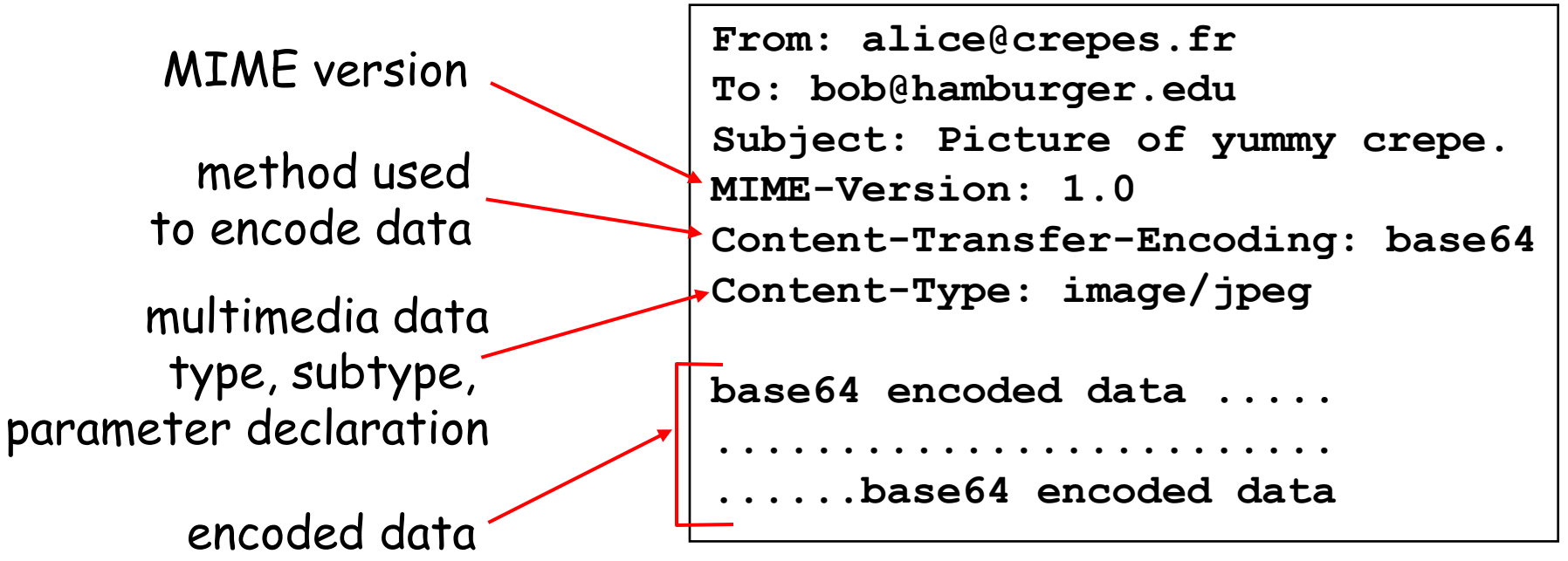
- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - the "message", ASCII characters only

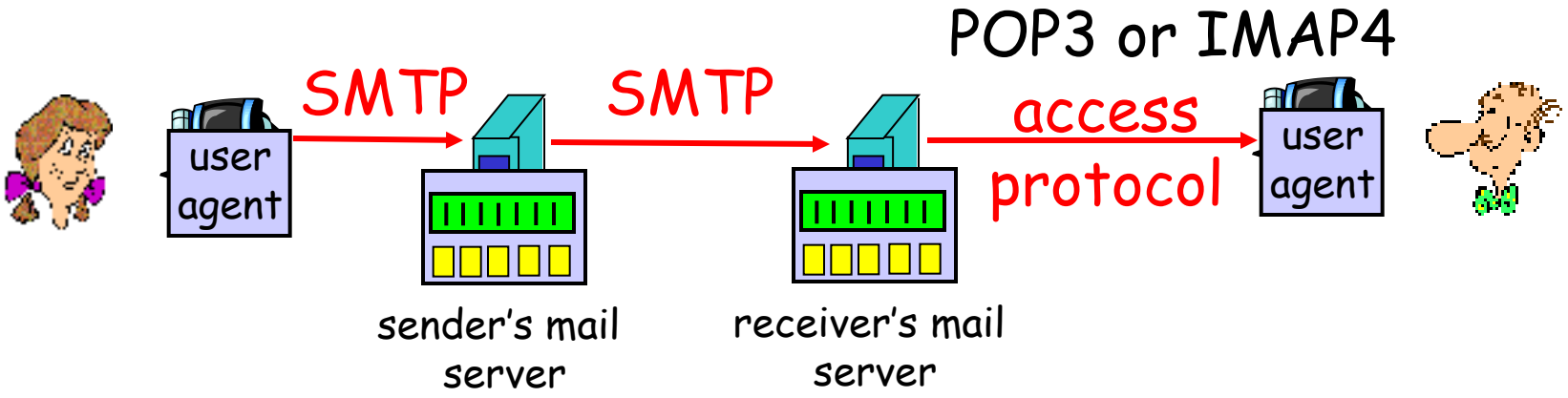


Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - Client connects to POP3 server on TCP port 110 (secure, 995)
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Client connects to TCP port 143 (secure, 993)
  - HTTP: Hotmail , Yahoo! Mail, gmail,etc.

POP vs IMAP

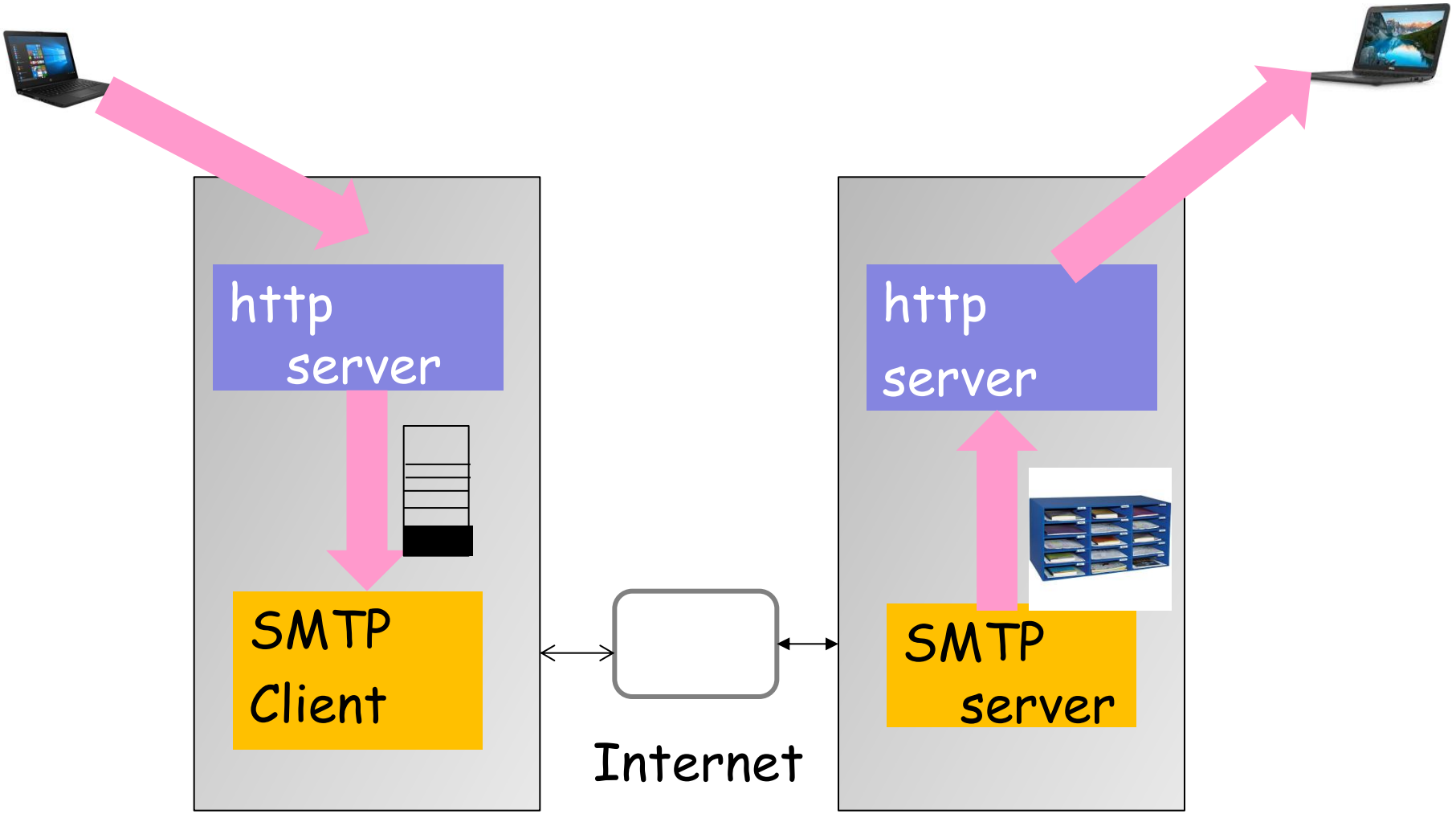
- POP3
- Offline processing
- Client retrieves email from server, then deleted from server(delete mode, keep mode)
- Latest changes are at the client

- IMAPv4
- Online processing
- Client can have a copy of specific emails
- Deletes done at the server
- Latest changes at server (synched)

Web based email

- Connect to email servers via web browser
- Browsers talk http
- Email servers talk SMTP
- Need a bridge to retrieve email using http
- E.g., gmail, yahoo mail, hot mail etc

Web based email



SMTP: final words

Comparison with 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