Maintaining user/server state: cookies

Web sites and client browser use *cookies* to maintain some state between transactions

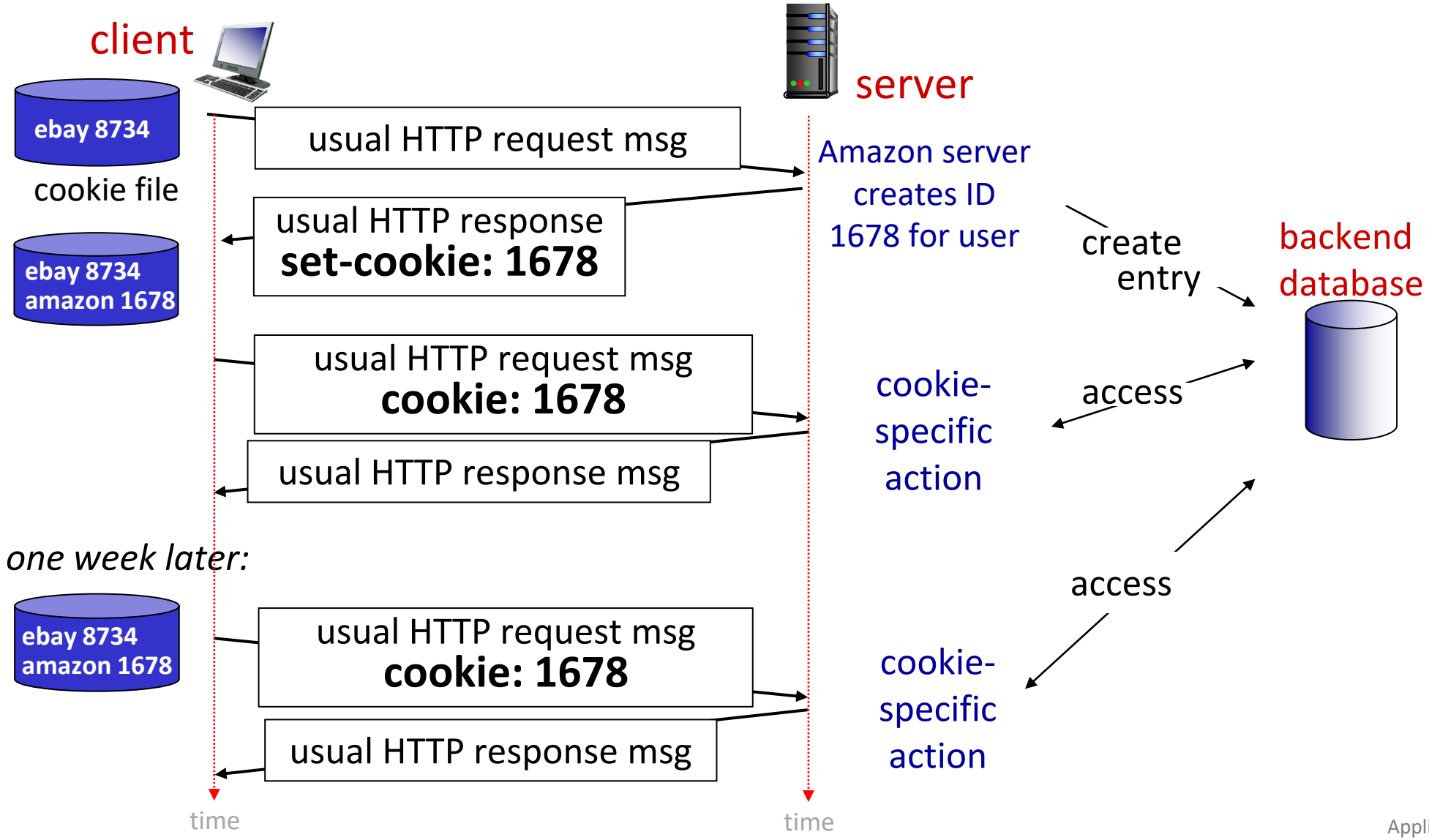
four components:

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 uses browser on laptop, visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID (aka "cookie")
  - entry in backend database for ID
- subsequent HTTP requests from Susan to this site will contain cookie ID value, allowing site to "identify" Susan

Maintaining user/server state: cookies



HTTP cookies: comments

What cookies can be used for:

- shopping carts
- recommendations
- user session state

Conditional GET

Goal: don't send object if cache has up-to-date cached version

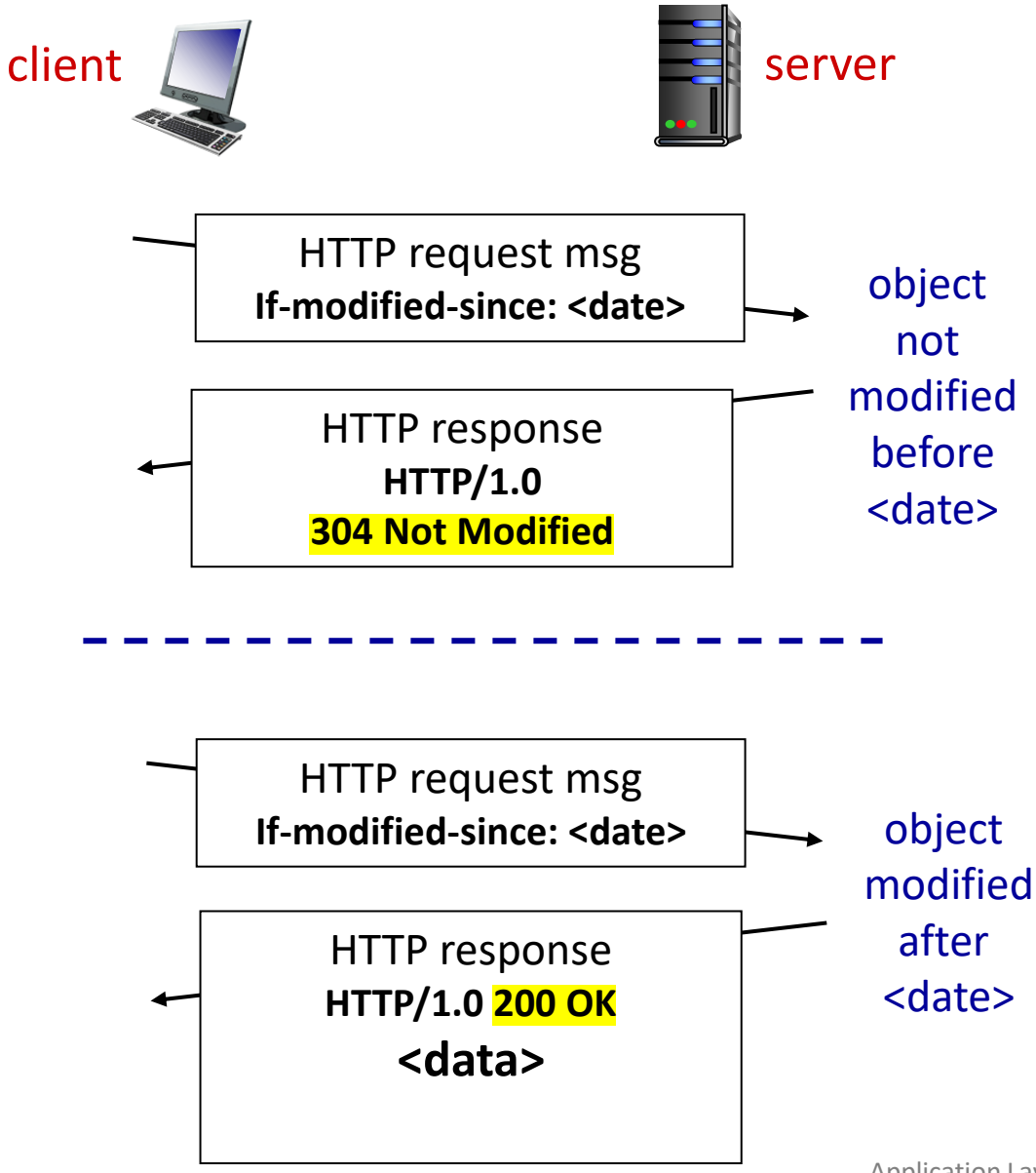
- no object transmission delay
- lower link utilization

- *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



HTTP/2

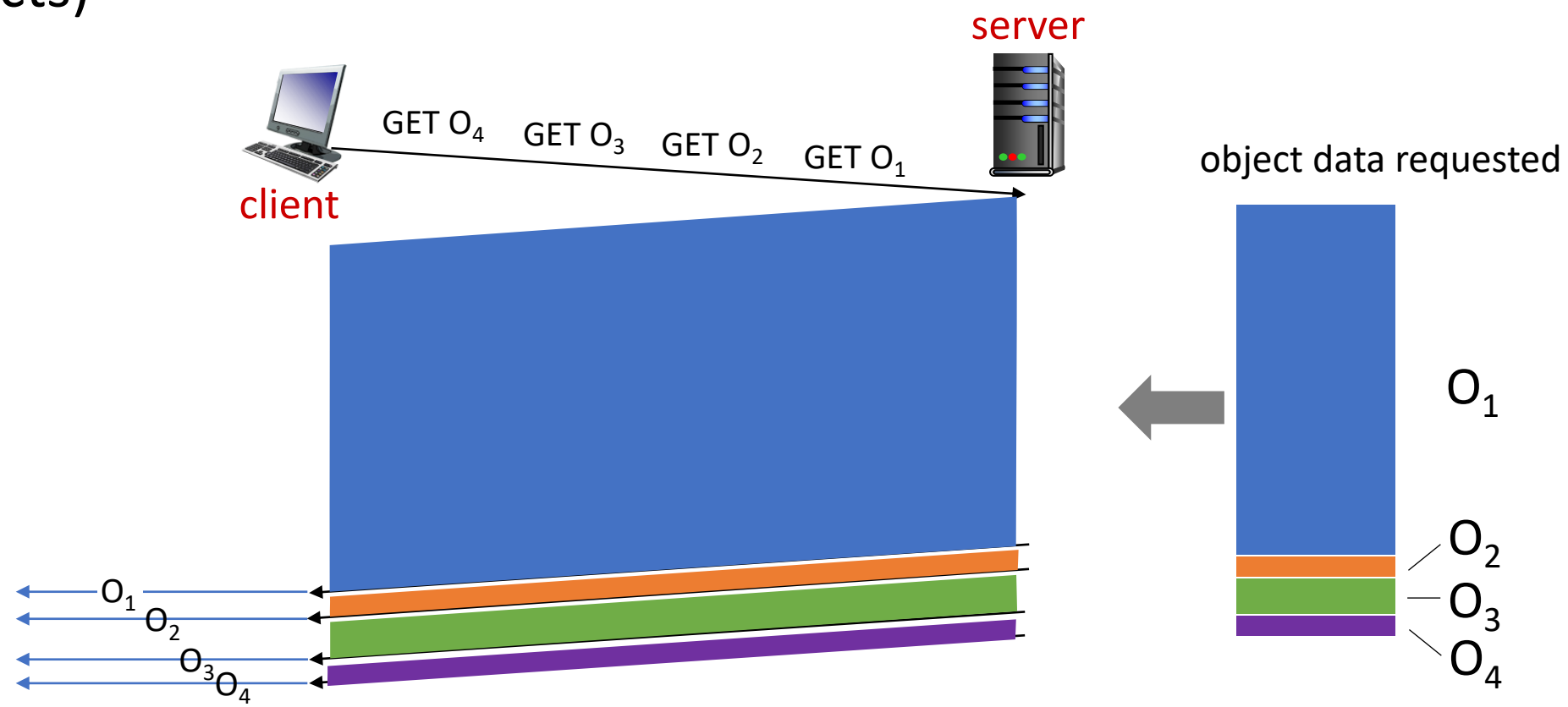
Key goal: decreased delay in multi-object HTTP requests

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- server responds *in-order* (FCFS: first-come-first-served scheduling) to GET requests
- with FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

HTTP/2: mitigating HOL blocking

HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)



objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

HTTP/2

Key goal: decreased delay in multi-object HTTP requests

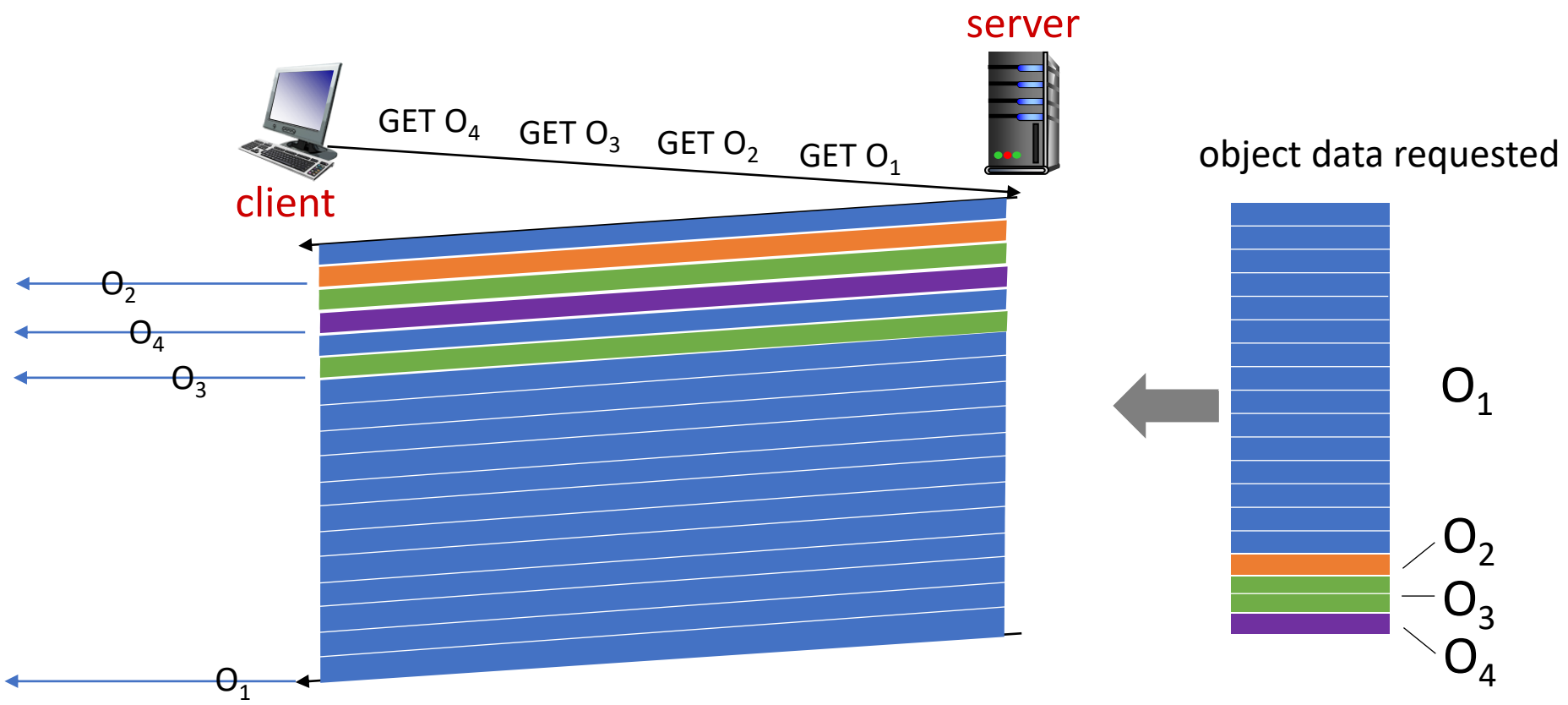
HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1
- transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- divide objects into frames, schedule frames to mitigate HOL blocking

HTTP/3: adds security , per object error- and congestion-control (more pipelining) over UDP **[Upcoming major version of HTTP]**

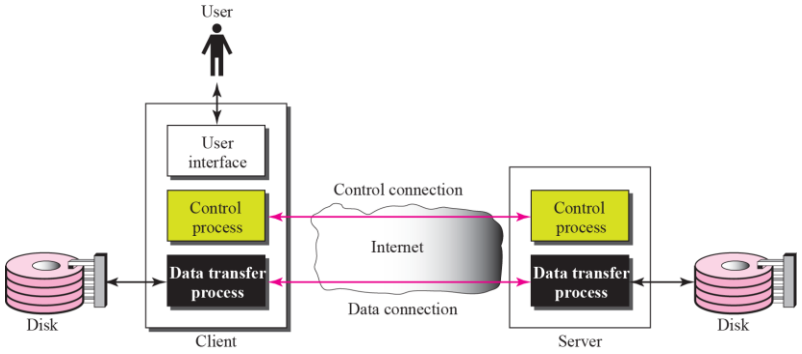
HTTP/2: mitigating HOL blocking

HTTP/2: objects divided into frames, frame transmission interleaved



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

File Transfer Protocol (FTP)



- It is like HTTP.
- You connect to a FTP server and can "get" or "put" files, after which data transfer happens.

- Some differences compared to HTTP:

(1) FTP sends control [TCP,21] and data [TCP,20] through separate TCP connections.

- Control connection: user identification, password, commands to change remote directory, commands to put and get files.
- Data connection: It is used to actually send the file.

- FTP is said to send control information out of band unlike HTTP.
- Control connection remains open throughout the duration of user session, but a new data connection is created for each file transferred within a session. (i.e., data connection are non-persistent)

(2) FTP keeps state over a session, e.g., user's current working directory etc.

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- The Domain Name System DNS
- E-mail, SMTP, IMAP
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

DNS: Domain Name System

people: many identifiers:

- name, passport #, Aadhar #

Internet hosts, routers:

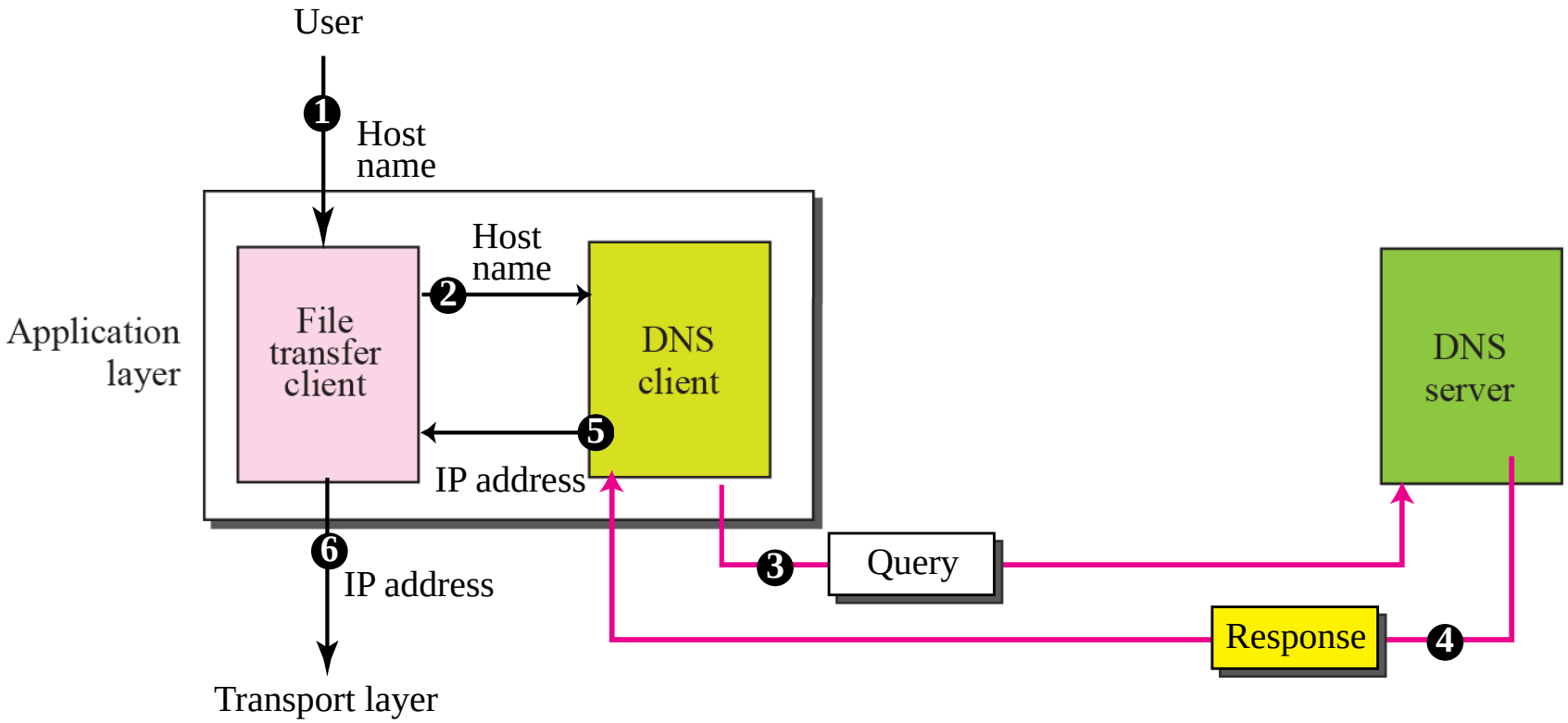
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., cs.umass.edu - used by humans

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*

Purpose of DNS



DNS: services, structure

DNS services

- hostname to IP address translation
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

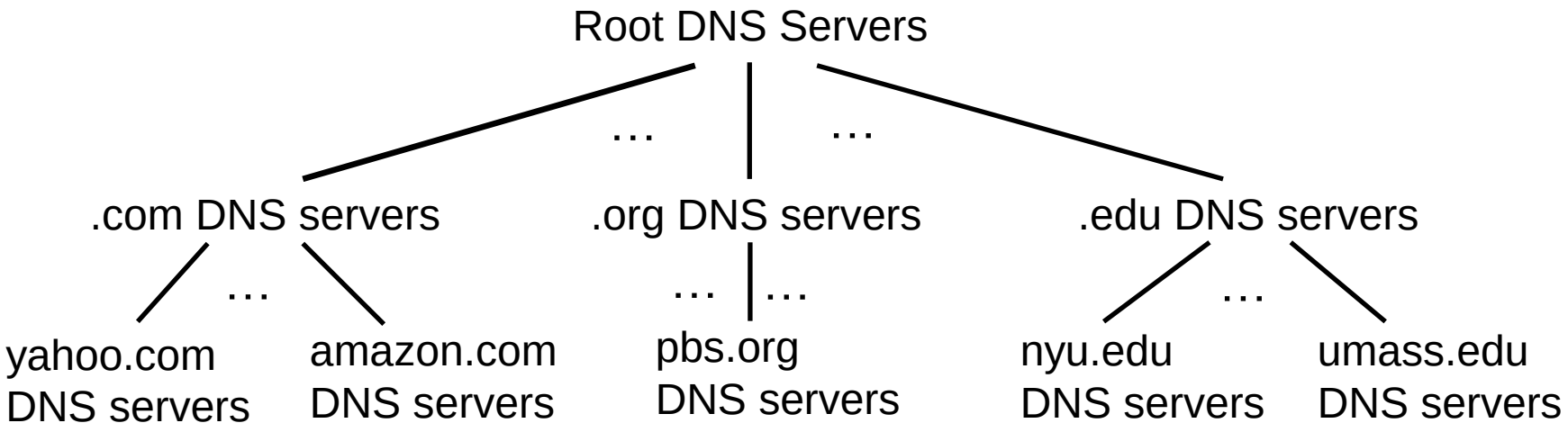
Q: Why not centralize DNS?

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

A: doesn't scale!

- Comcast DNS servers alone: 600B DNS queries per day

DNS: a distributed, hierarchical database



Root

Top Level Domain

Authoritative

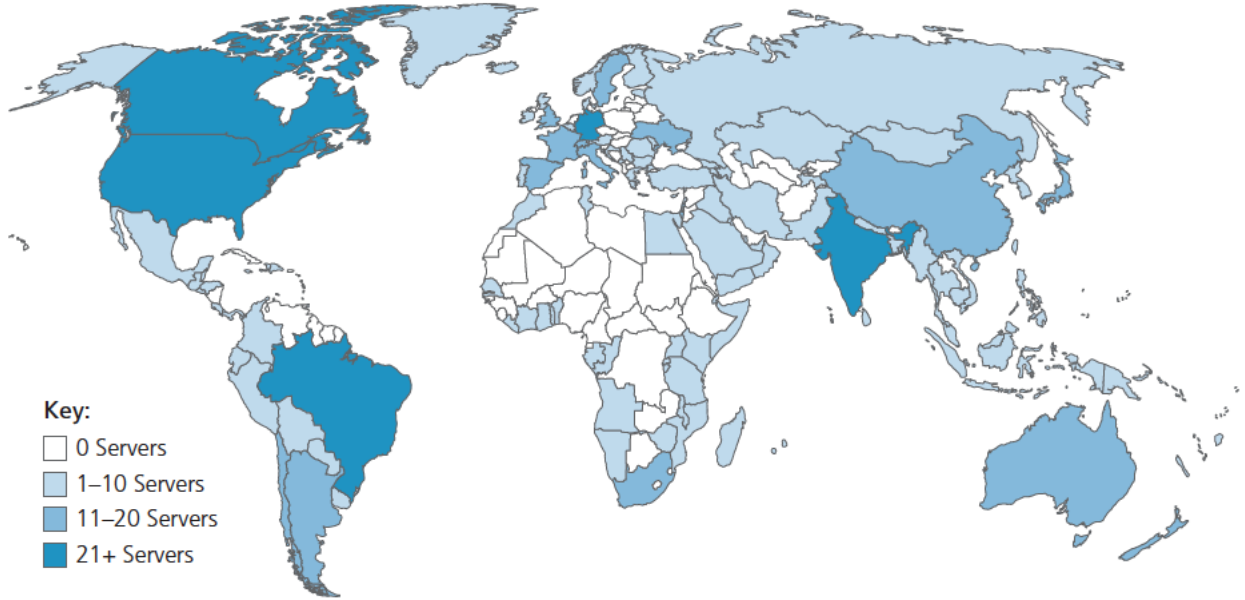
Client wants IP address for www.amazon.com; 1st approximation:

- 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

- official, contact-of-last-resort by name servers that can not resolve name
- *incredibly important* Internet function
  - Internet couldn't function without it!
  - DNSSEC – provides security (authentication and message integrity)
- ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain

13 logical root name "servers" worldwide each "server" replicated many times (~200 servers in US)



TLD: authoritative servers

Top-Level Domain (TLD) servers:

- **Generic top-level domains (gTLD):** .com, .org, .net, .edu, .aero, .jobs, .net, .edu
- **Country-code top-level domains (ccTLD):** all country domains, e.g.: .in .cn, .uk, .fr, .ca, .jp

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

Local DNS name servers

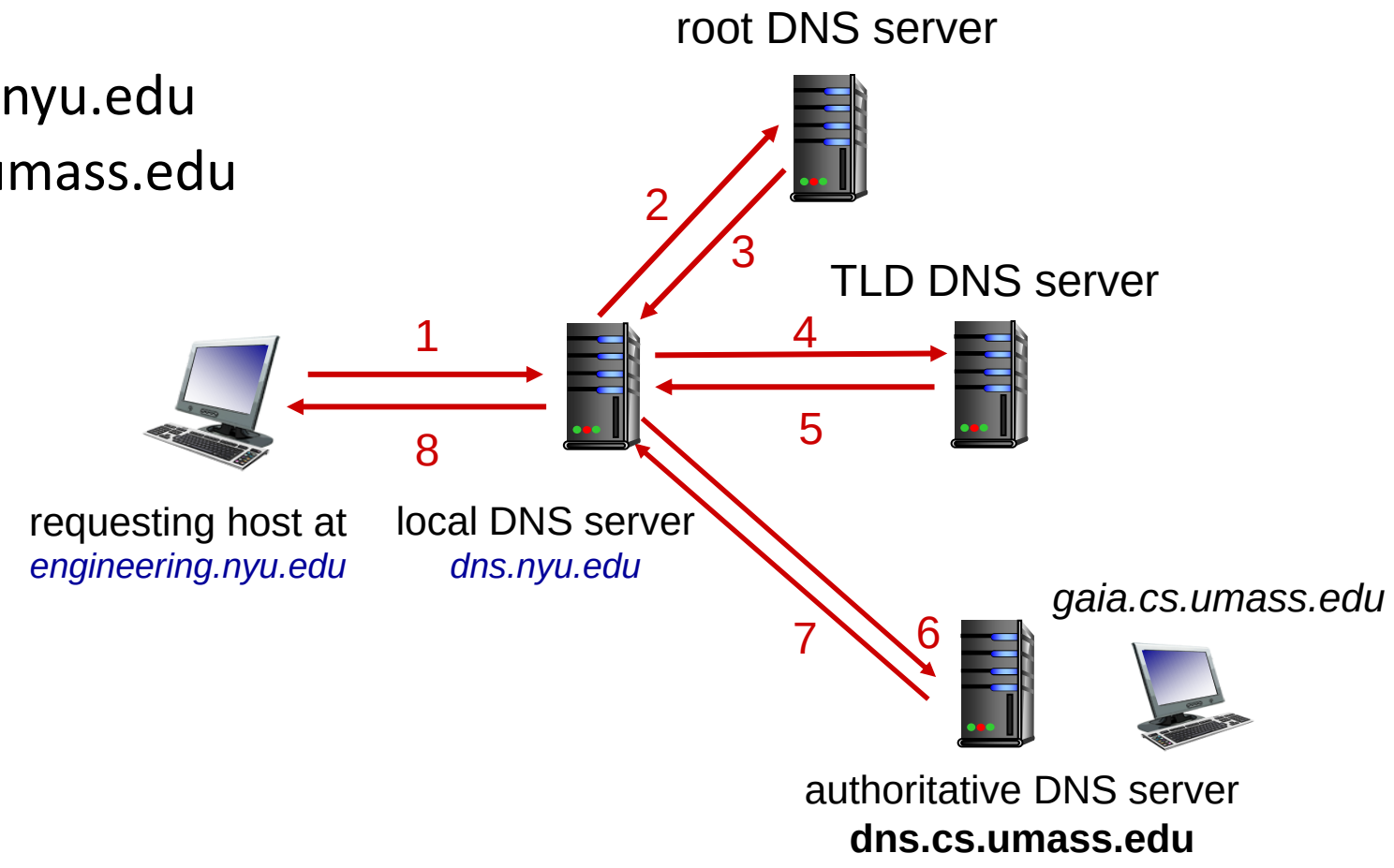
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy
  - The local DNS servers are statically configured with the identity of the root servers.

DNS name resolution: iterated query

Example: host at engineering.nyu.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"



DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

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