

Application layer: overview

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



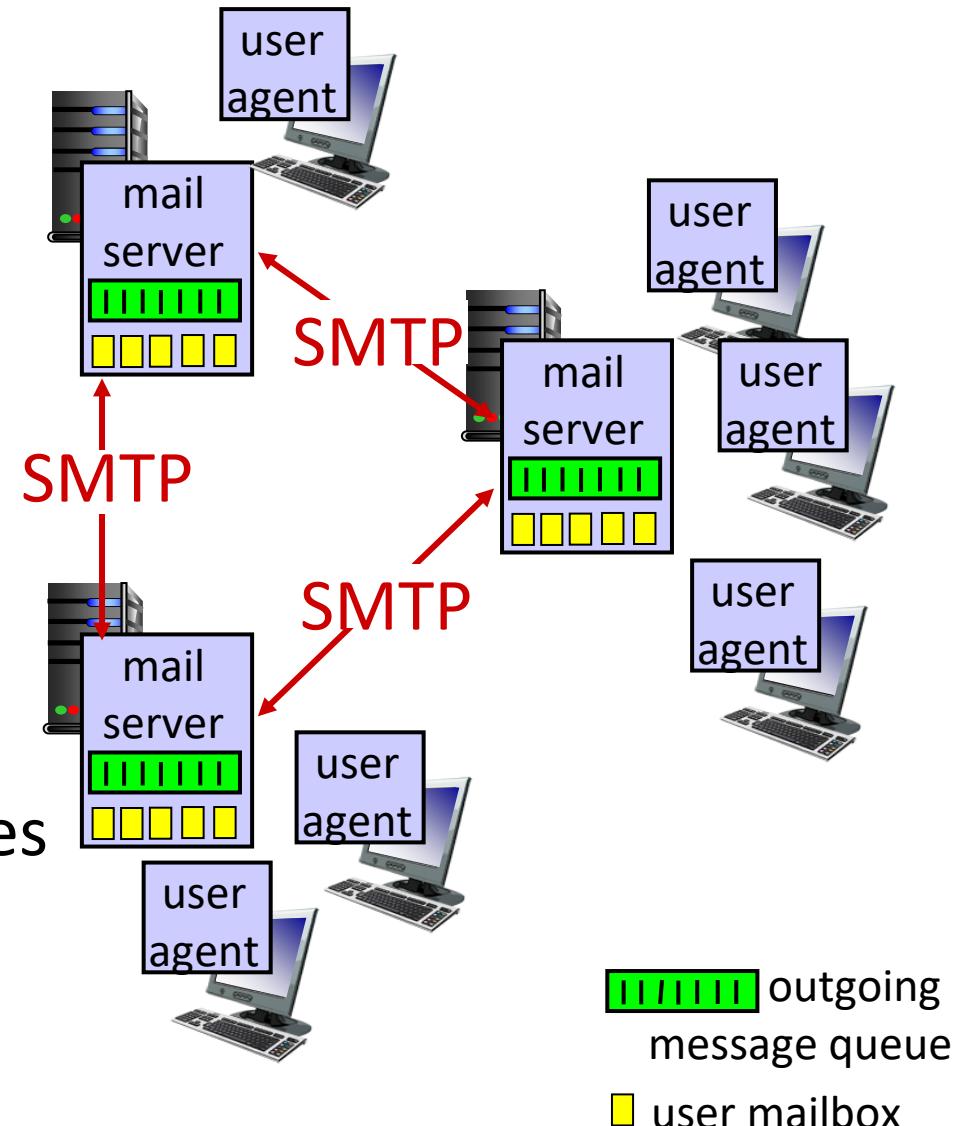
E-mail

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, iPhone mail client
- outgoing, incoming messages stored on server



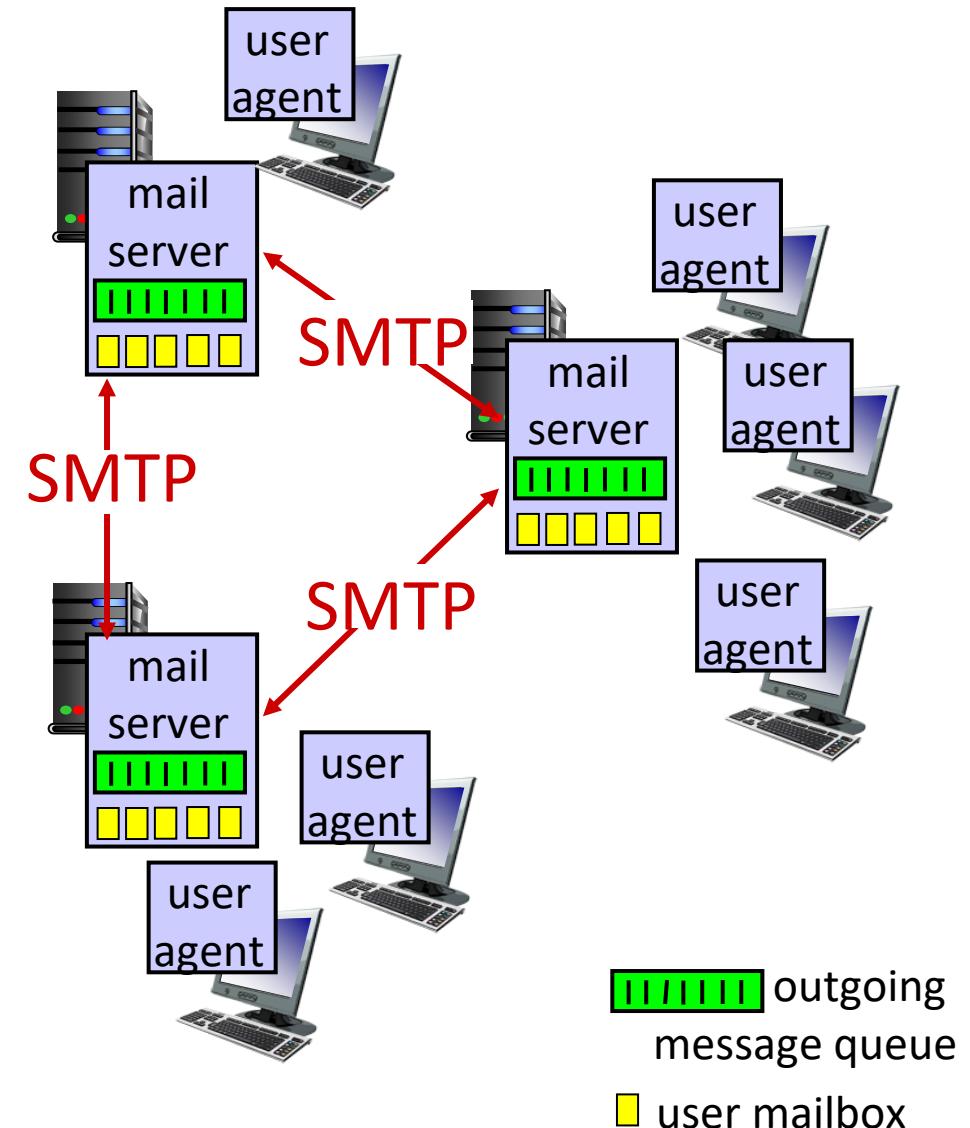
E-mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages

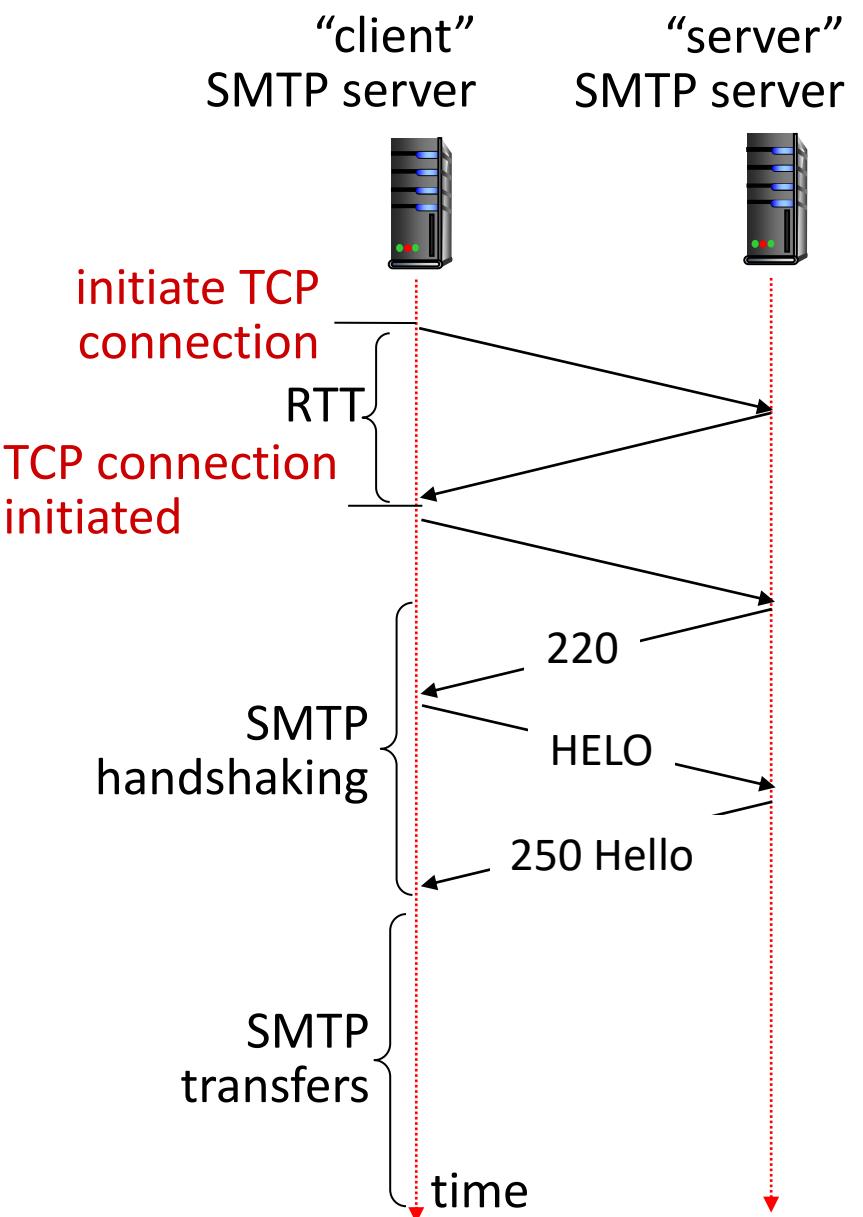
SMTP protocol between mail servers to send email messages

- client: sending mail server
- “server”: receiving mail server



SMTP RFC (5321)

- uses TCP to reliably transfer email message from client (mail server initiating connection) to server, port 25
 - direct transfer: sending server (acting like client) to receiving server
- three phases of transfer
 - SMTP handshaking (greeting)
 - SMTP transfer of messages
 - SMTP closure
- command/response interaction (like HTTP)
 - commands: ASCII text
 - response: status code and phrase

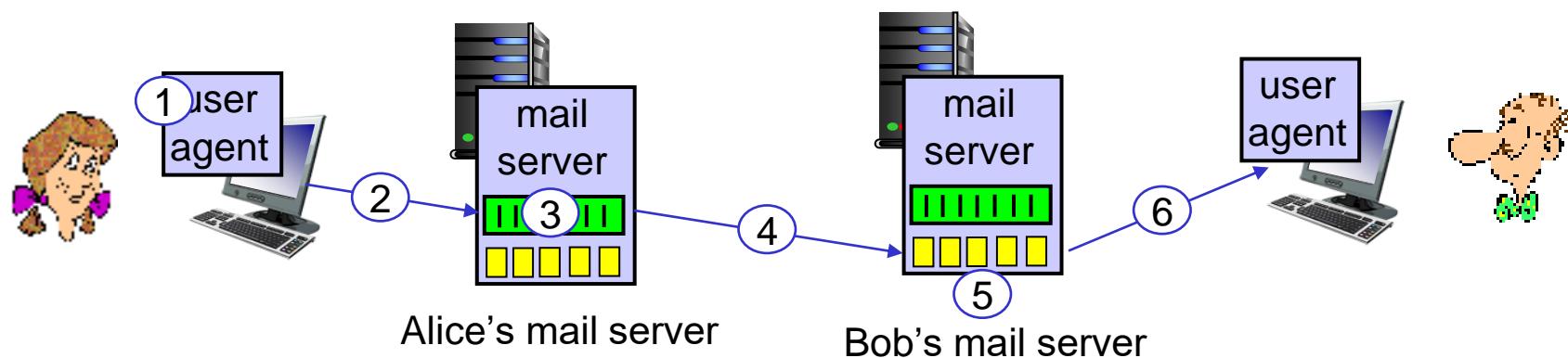


Scenario: Alice sends e-mail to Bob

- 1) Alice uses UA to compose e-mail message “to” bob@someschool.edu
- 2) Alice’s UA sends message to her mail server using SMTP; message placed in message queue
- 3) client side of SMTP at mail server 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

```
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: observations

comparison with HTTP:

- HTTP: client pull
- SMTP: client 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
- 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

Mail message format

SMTP: protocol for exchanging e-mail messages, defined in RFC 5321
(like RFC 7231 defines HTTP)

RFC 2822 defines *syntax* for e-mail message itself (like HTML defines syntax for web documents)

- header lines, e.g.,

- To:

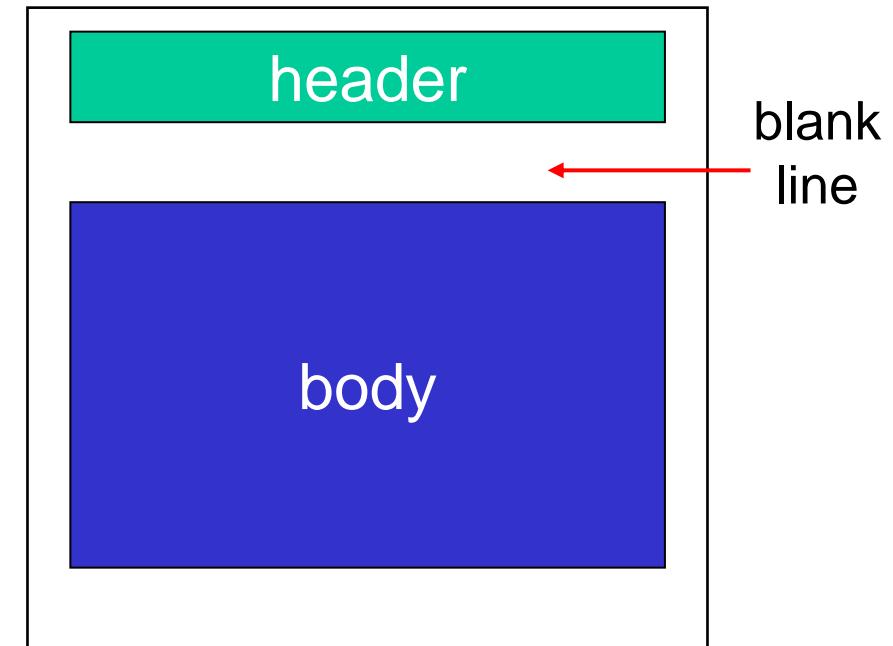
- From:

- Subject:

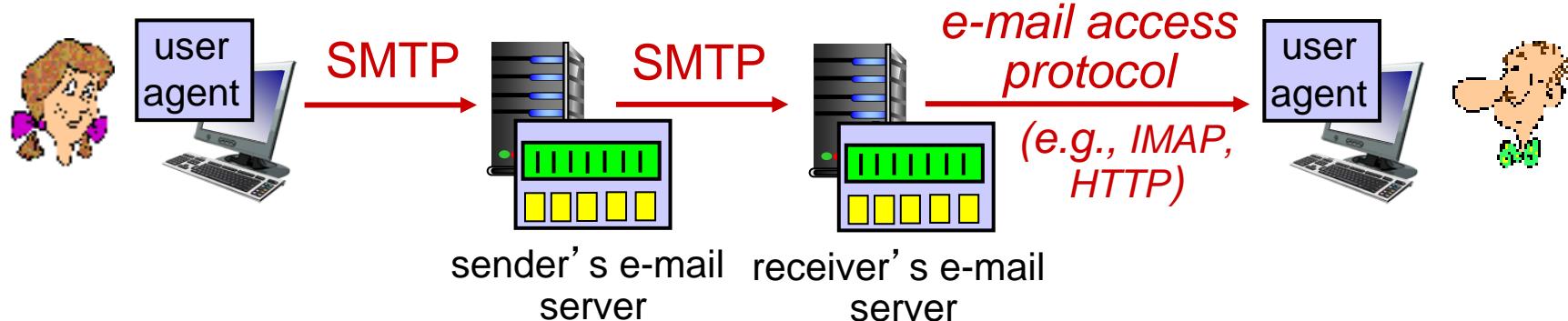
these lines, within the body of the email

message area different from SMTP MAIL FROM;
RCPT TO: commands!

- Body: the “message”, ASCII characters only



Retrieving email: mail access protocols



- **SMTP:** delivery/storage of e-mail messages to receiver's server
- mail access protocol: retrieval from server
 - **IMAP:** Internet Mail Access Protocol [RFC 3501]: messages stored on server, IMAP provides retrieval, deletion, folders of stored messages on server
- **HTTP:** gmail, Hotmail, Yahoo!Mail, etc. provides web-based interface on top of STMP (to send), IMAP (or POP) to retrieve e-mail messages

Application Layer: Overview

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



DNS: Domain Name System

people: many identifiers:

- SSN, name, passport #

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 (DNS):

- *distributed database implemented in hierarchy of many name servers*
- *application-layer protocol: hosts, DNS 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

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/day
- Akamai DNS servers alone: 2.2T DNS queries/day

Thinking about the DNS

humongous distributed database:

- ~ billion records, each simple

handles many *trillions* of queries/day:

- *many more reads than writes*
- *performance matters*: almost every Internet transaction interacts with DNS - msecs count!

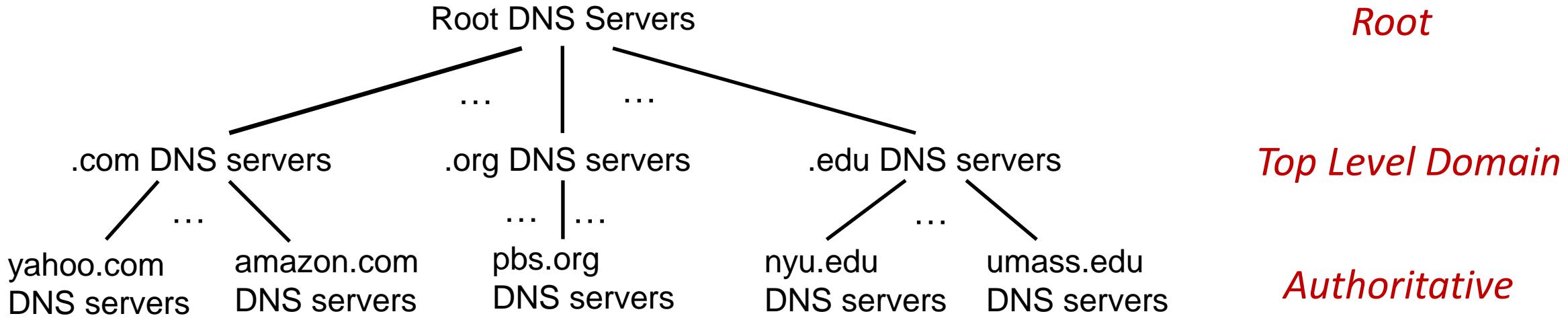
organizationally, physically decentralized:

- millions of different organizations responsible for their records

“bulletproof”: reliability, security



DNS: a distributed, hierarchical database

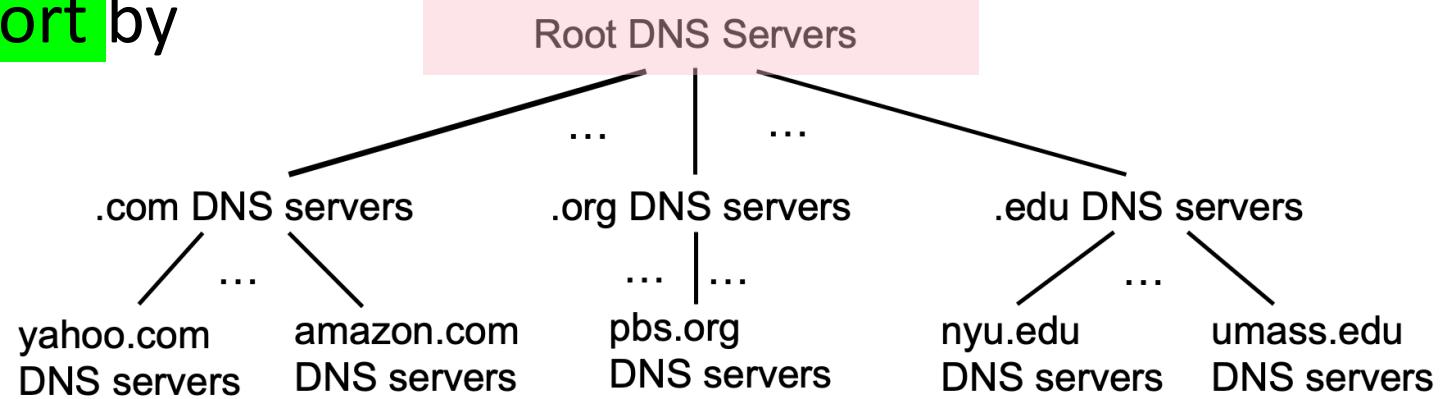


Client wants IP address for www.amazon.com; 1st approximation:

- 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

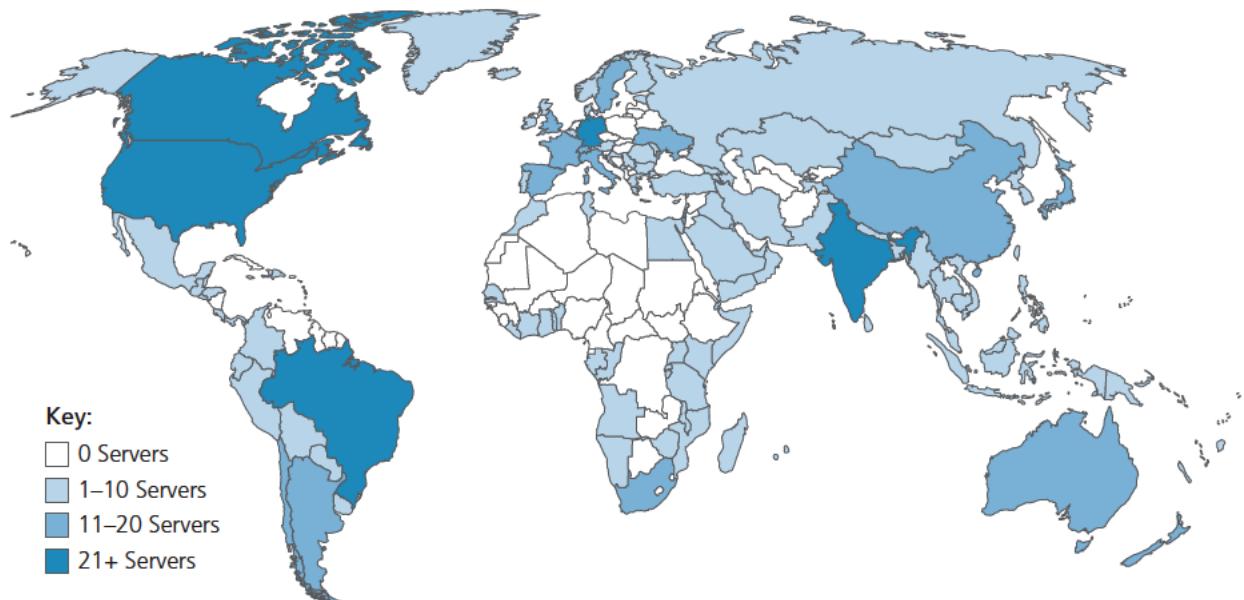
- official, **contact-of-last-resort** by name servers that can not resolve name



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, message integrity)
- ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain

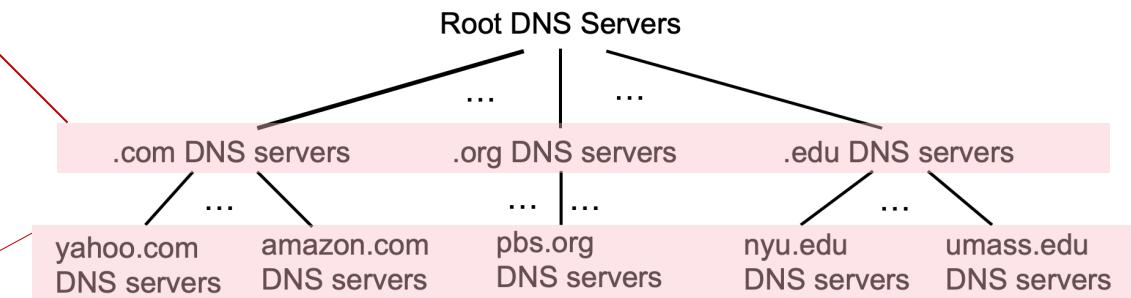
13 logical root name “servers” worldwide each “server” replicated many times (~200 servers in US)



Top-Level Domain, and authoritative servers

Top-Level Domain (TLD) servers:

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

Local DNS name servers

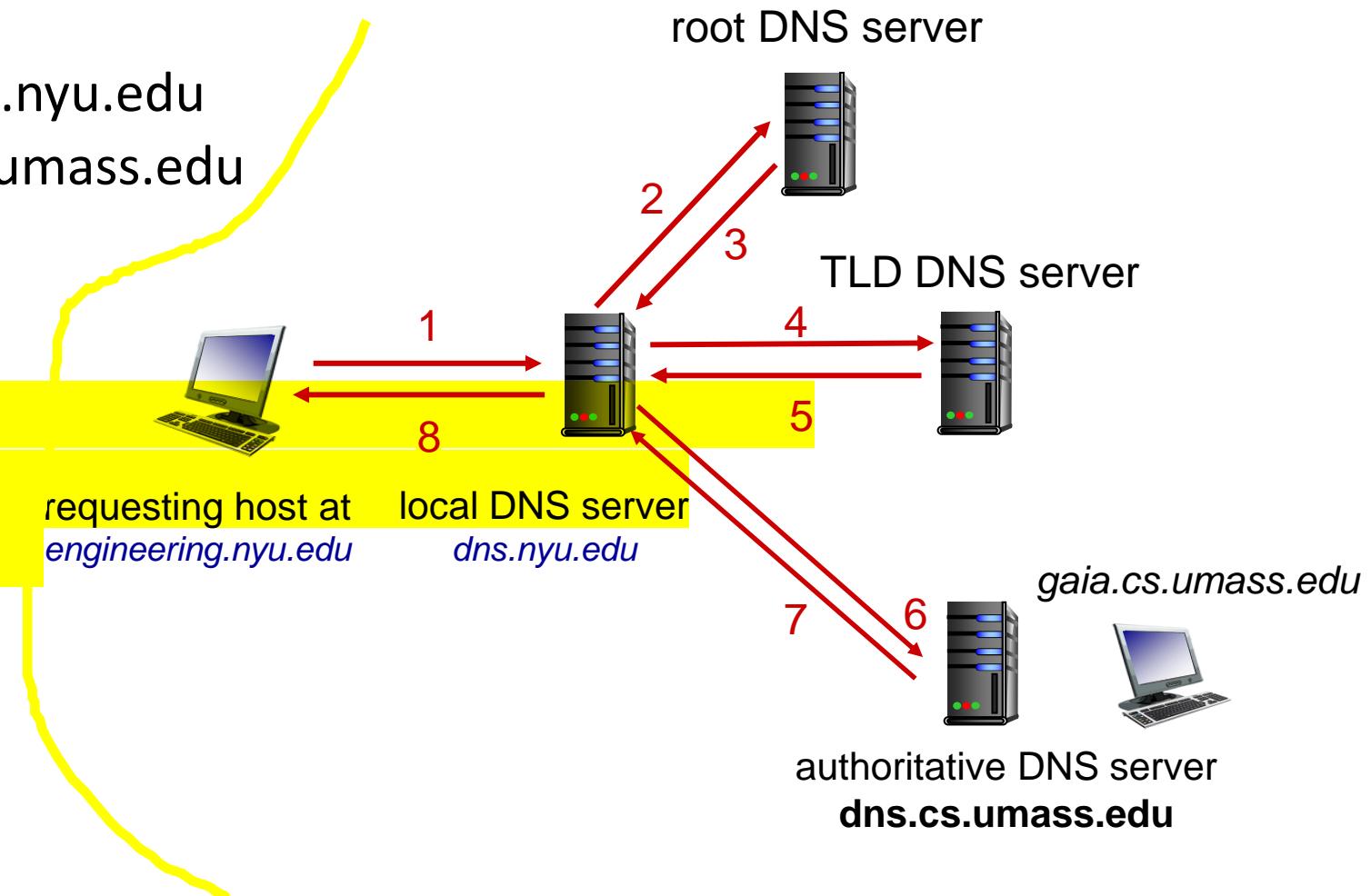
- when host makes DNS query, it is sent to its *local* DNS server
 - Local DNS server returns reply, answering:
 - from its **local cache** of recent name-to-address translation pairs (possibly out of date!)
 - **forwarding request into DNS hierarchy for resolution**
 - each ISP has local DNS name server; to find yours:
 - MacOS: % scutil --dns
 - Windows: >ipconfig /all
- local DNS server doesn't strictly belong to hierarchy

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”

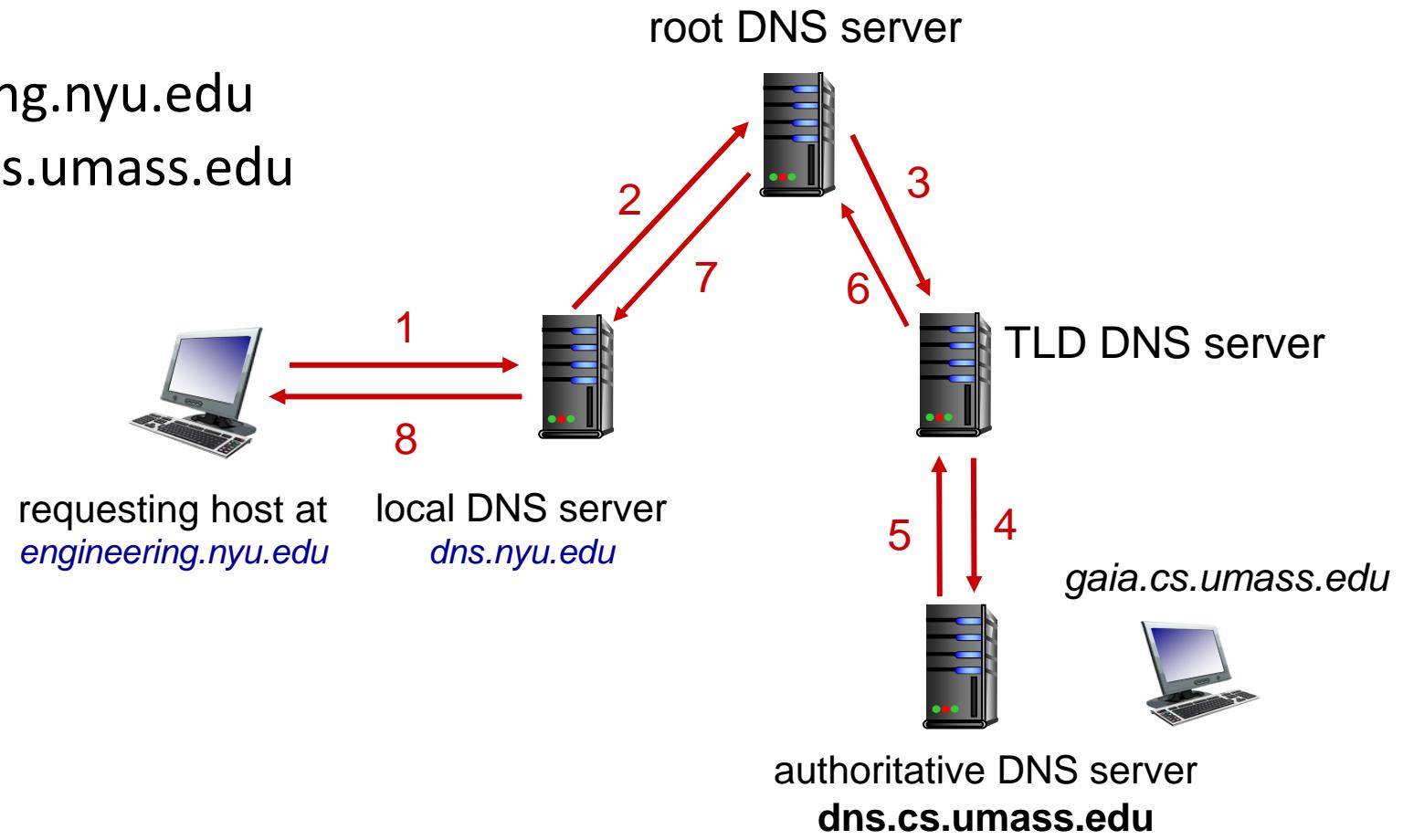


DNS name resolution: recursive query

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?



Caching DNS Information

- once (any) name server learns mapping, it *caches mapping, and immediately returns a cached mapping* in response to a query
 - caching improves response time
 - cache entries timeout (disappear) after some time (TTL)
 - TLD servers typically cached in local name servers
- **cached entries may be *out-of-date***
 - if named host changes IP address, may not be known Internet-wide until all TTLs expire!
 - *best-effort name-to-address translation!*

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

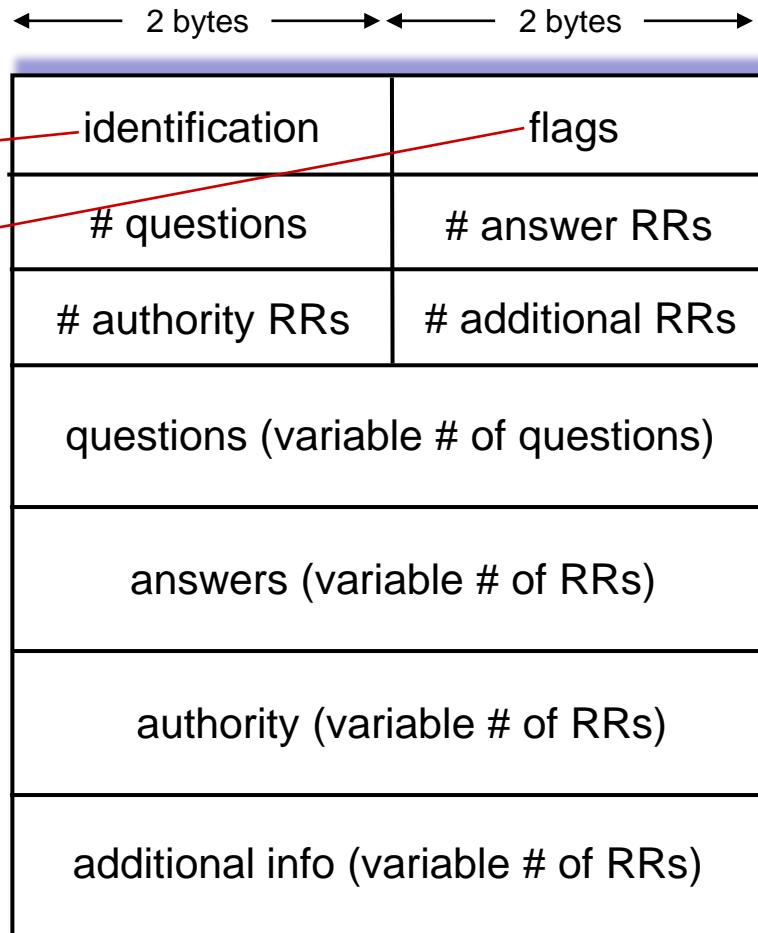
- value is name of SMTP mail server associated with name

DNS protocol messages

DNS *query* and *reply* messages, both have same *format*:

message header:

- identification: 16 bit # for query,
reply to query uses same #
- flags:
 - query or reply
 - recursion desired
 - recursion available
 - reply is authoritative



DNS protocol messages

DNS *query* and *reply* messages, both have same *format*:

↔ 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)	

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)

Getting your info into the 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 NS, A RRs into .com TLD server:
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
- create authoritative server locally with IP address 212.212.212.1
 - type A record for www.networkuptopia.com
 - type MX record for networkutopia.com

DNS security

DDoS attacks

- bombard root servers with traffic
 - not successful to date
 - traffic filtering
 - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers
 - potentially more dangerous

Spoofing attacks

- intercept DNS queries, returning bogus replies
 - DNS cache poisoning
 - RFC 4033: DNSSEC authentication services