# Computer Networks

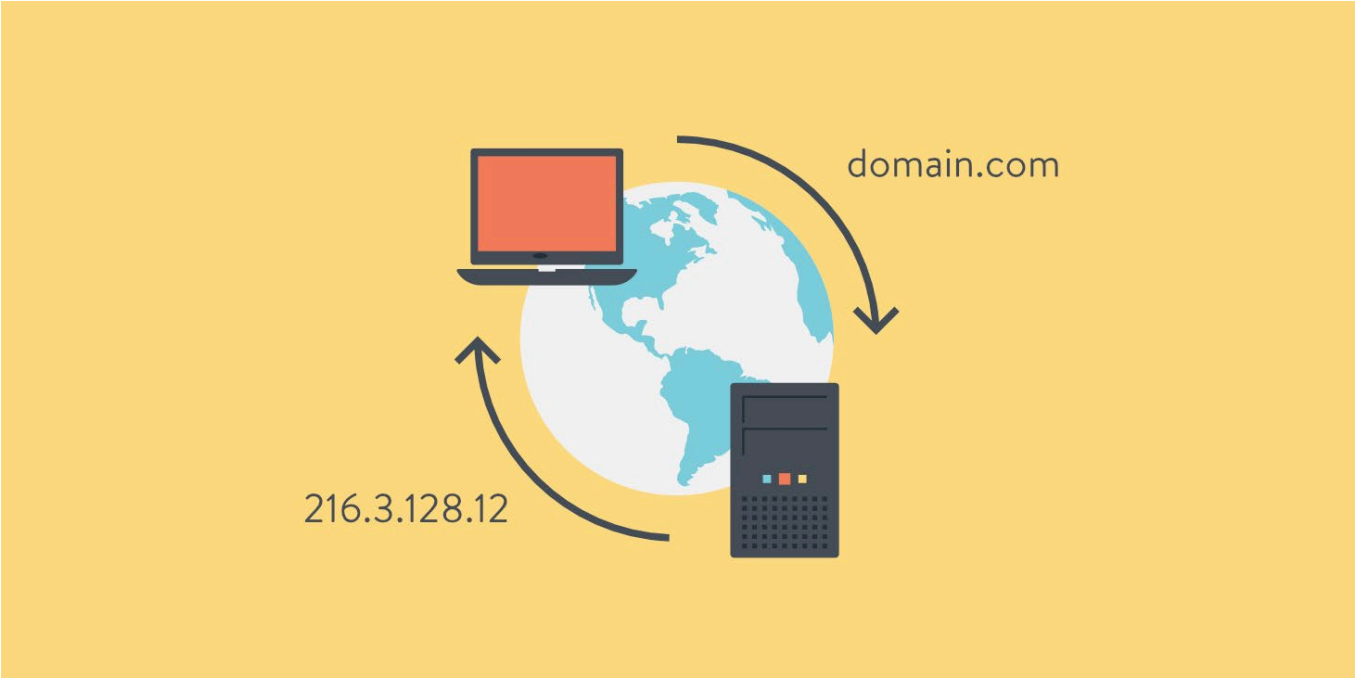
Lecturer: ZHANG Ying

Fall semester 2022

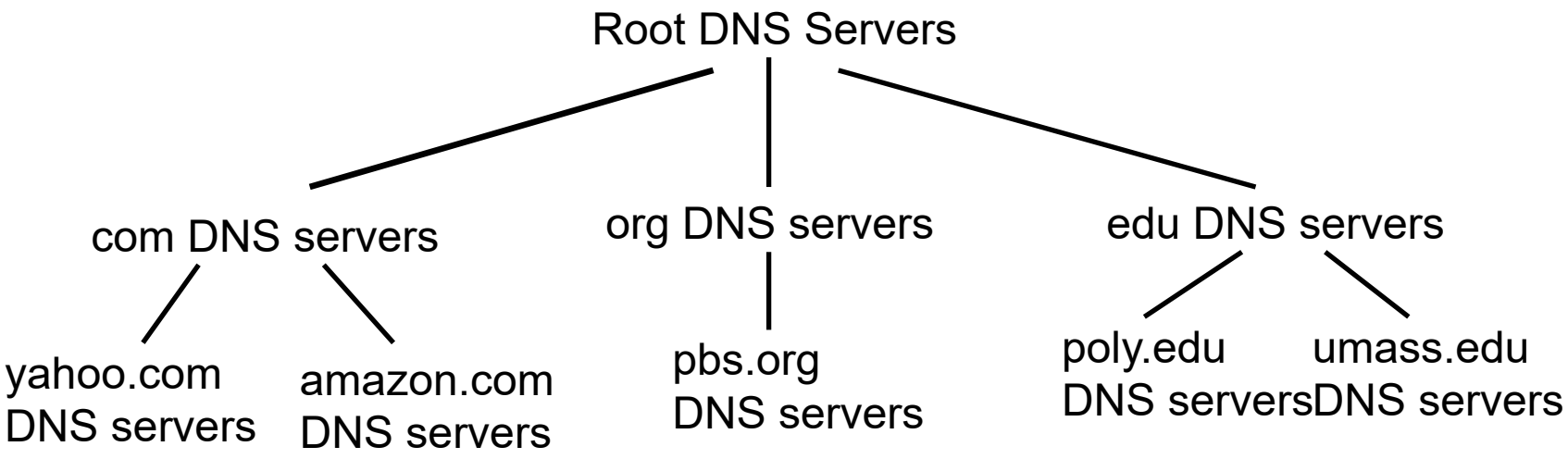
# Chapter 2
# Application Layer

# DNS: domain name system

DNS: map between IP address and name



# DNS: a distributed, hierarchical database




*client wants IP for www.amazon.com;*

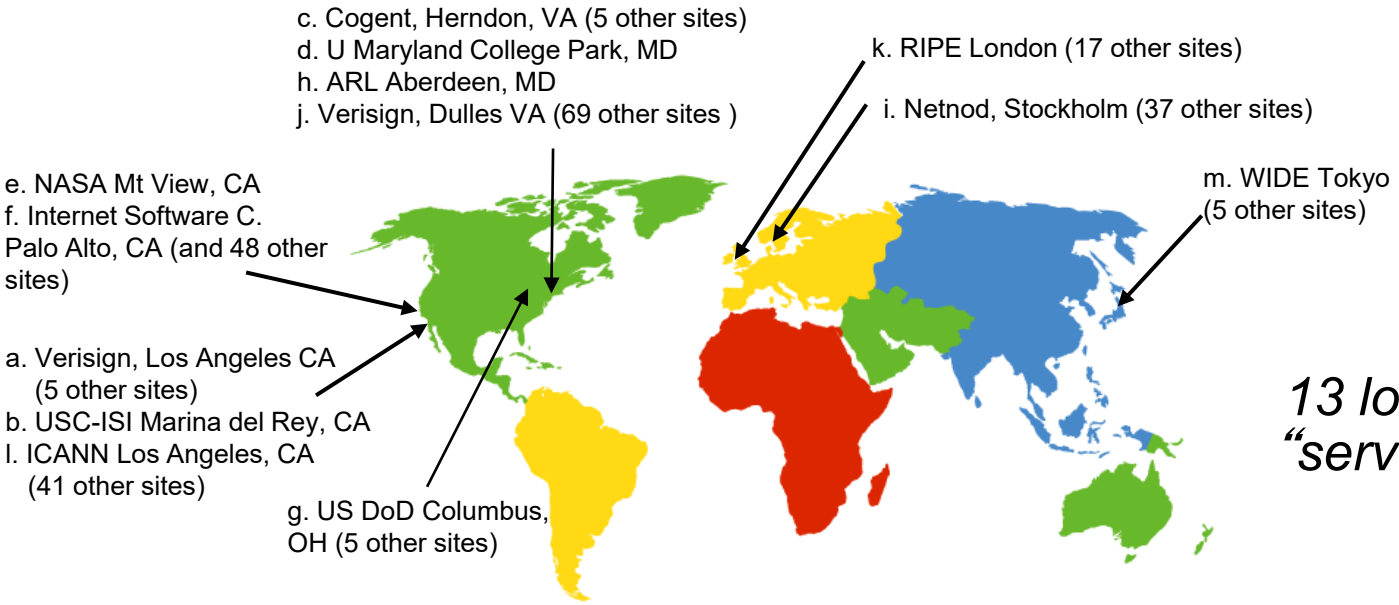
- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

# 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  - has local cache of recent name-to-address translation pairs (but may be out of date!)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

# root DNS 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

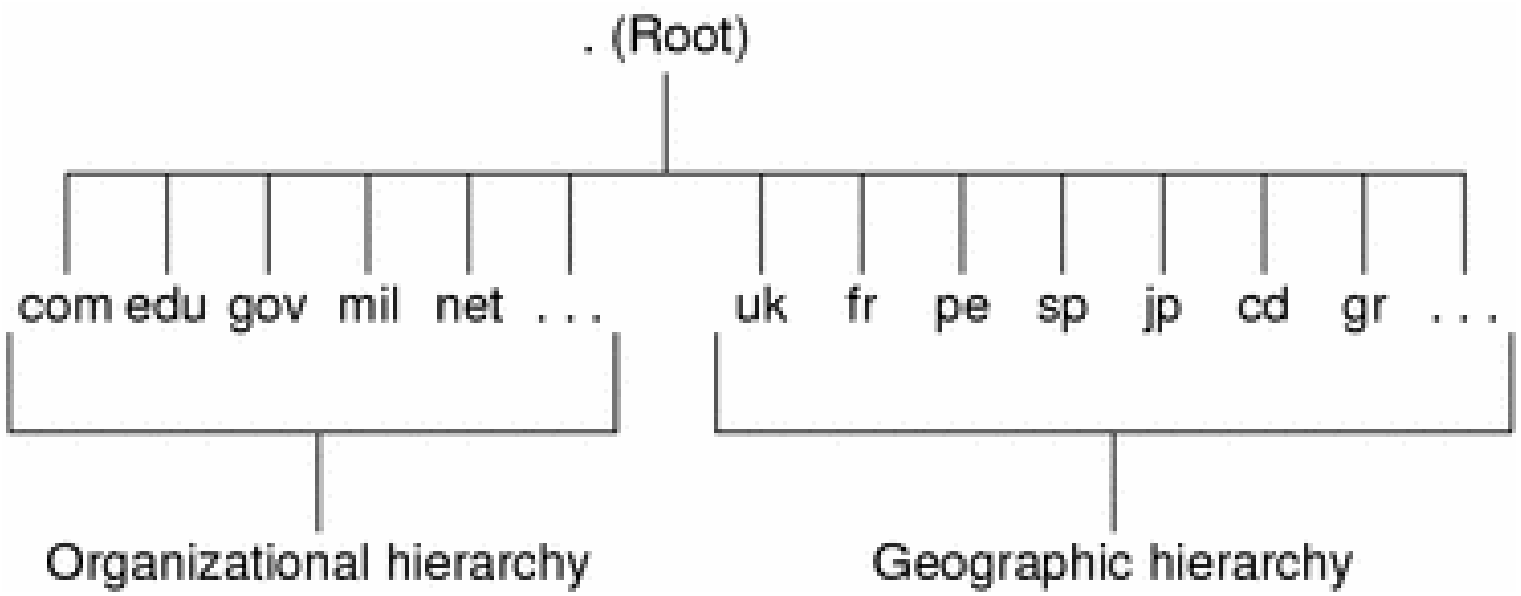



*13 logical root name "servers" worldwide*

# top-level domain (TLD) servers:

Divided into two distinct sub-categories:

- Organizational Hierarchy
- Geographical Hierarchy



# Organizational Hierarchy

| Domain | Purpose |
|---|---|
| .com | commercial organizations |
| .edu | educational organizations |
| .gov | government institutions |
| .mil | military groups |
| .net | major network support centers |
| .org | Nonprofit organizations and others |
| .int | International organizations |

# authoritative DNS servers

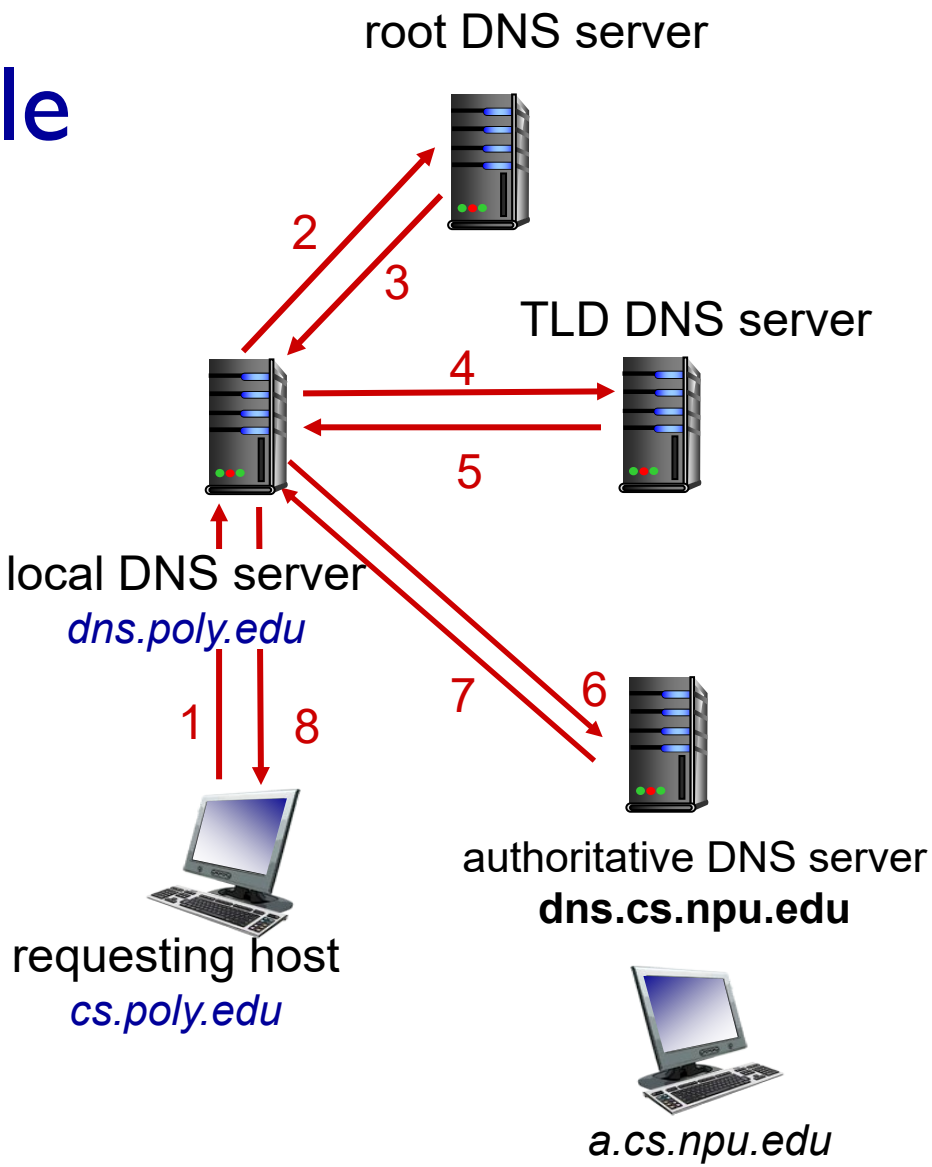
- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

# DNS name resolution example

- host at cs.poly.edu wants IP address for a.cs. npu.edu

*iterated query:*

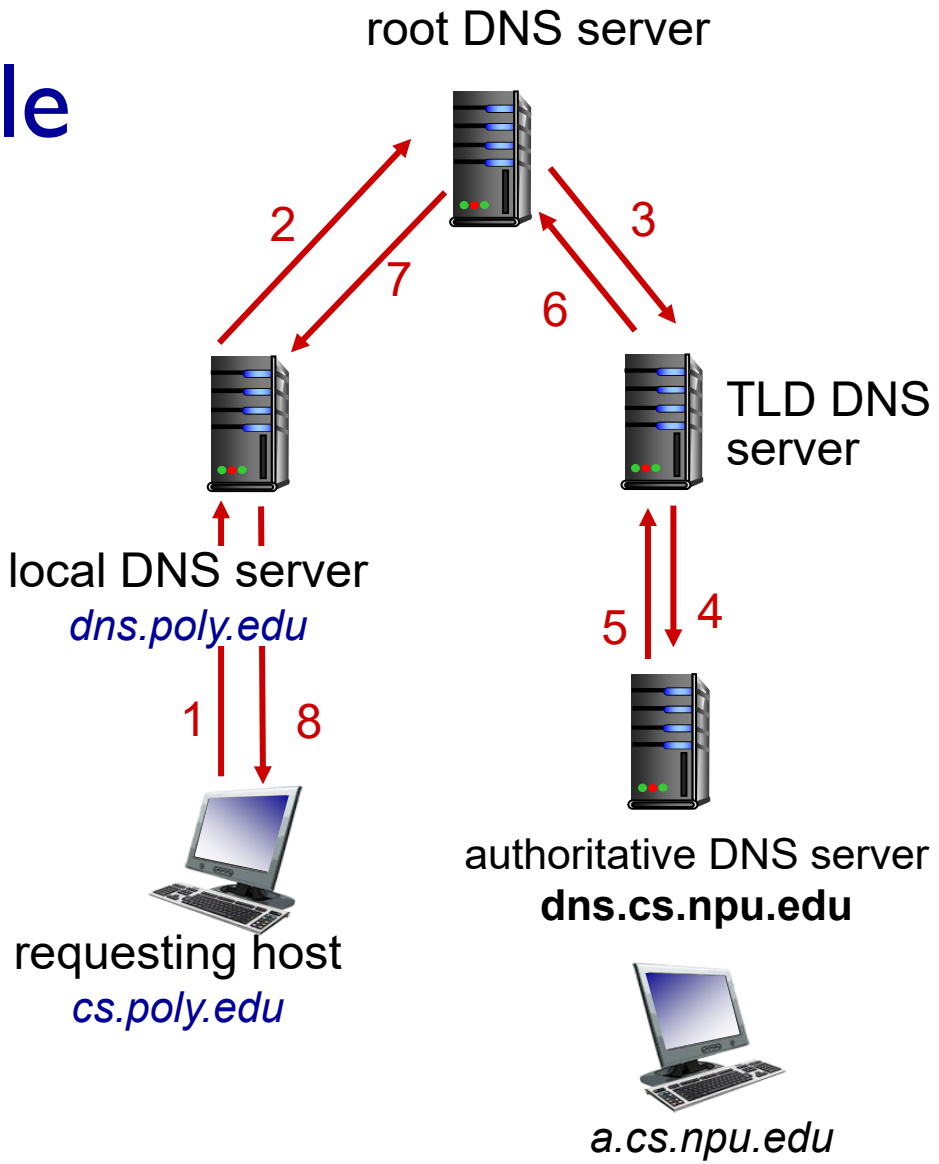
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



# DNS name resolution example

*recursive query:*

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



# 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

# 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

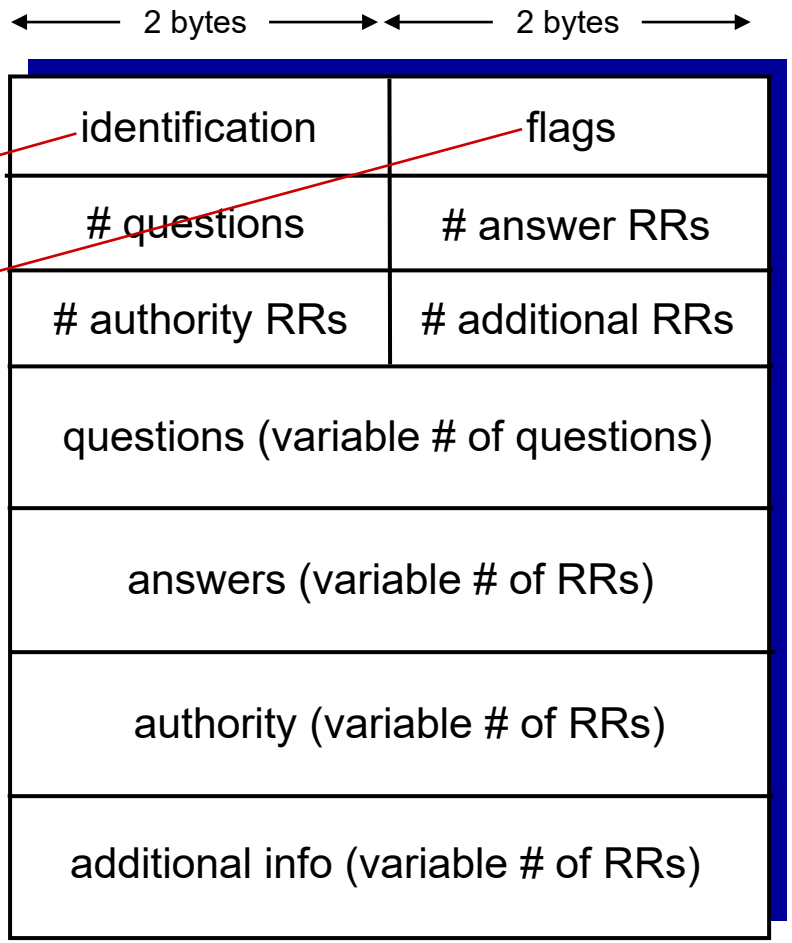
- **`value`** is name of mailserver associated with **`name`**

# DNS protocol, messages

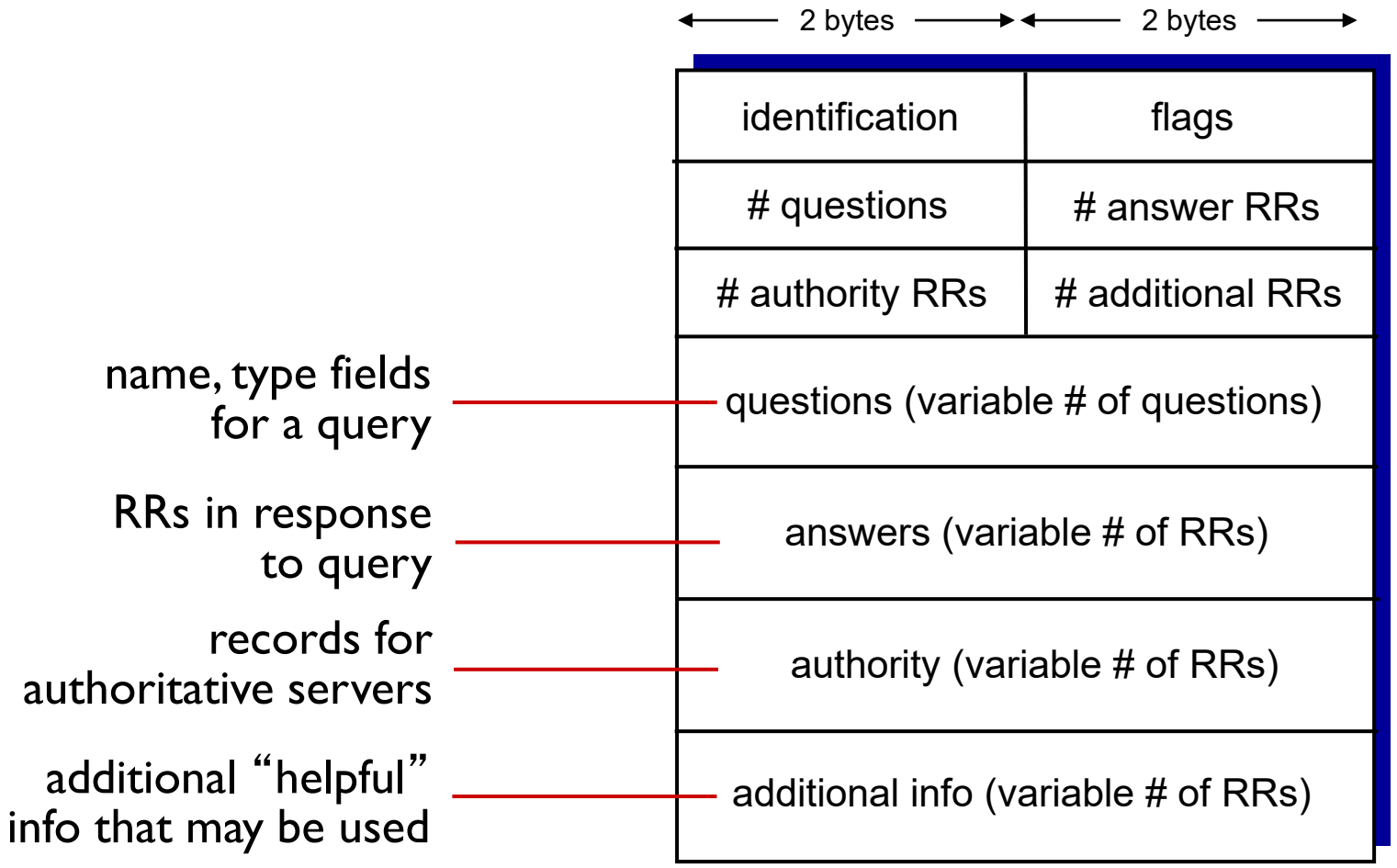
- *query* and *reply* messages, both with same *message format*

message header

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



# DNS protocol, messages



# Chapter Outline

1 principles of network applications

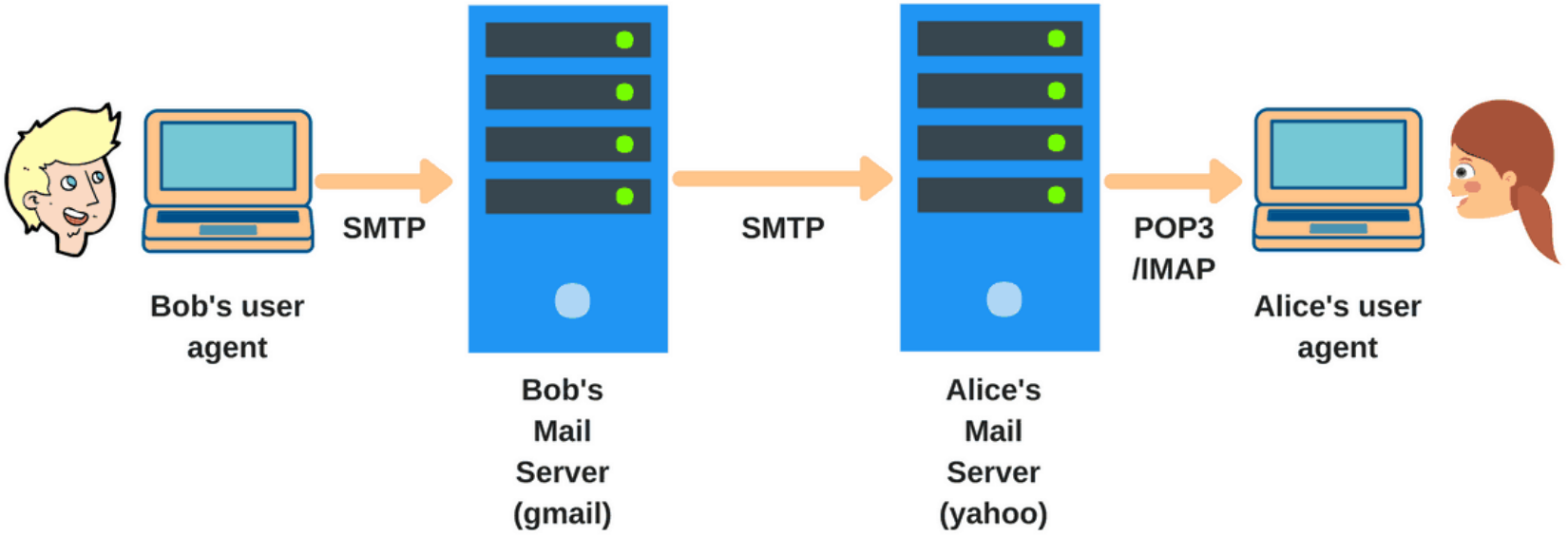
2 Web and HTTP

3 electronic mail

# Electronic mail

- **Electronic mail** (**email** or **e-mail**) is a method of exchanging messages ("mail") between people using electronic devices.
- A-synchronous
- Inexpensive
- Rich contents

# How email works – example




Bob@gmail.com

Alice@yahoo.com

# Electronic mail

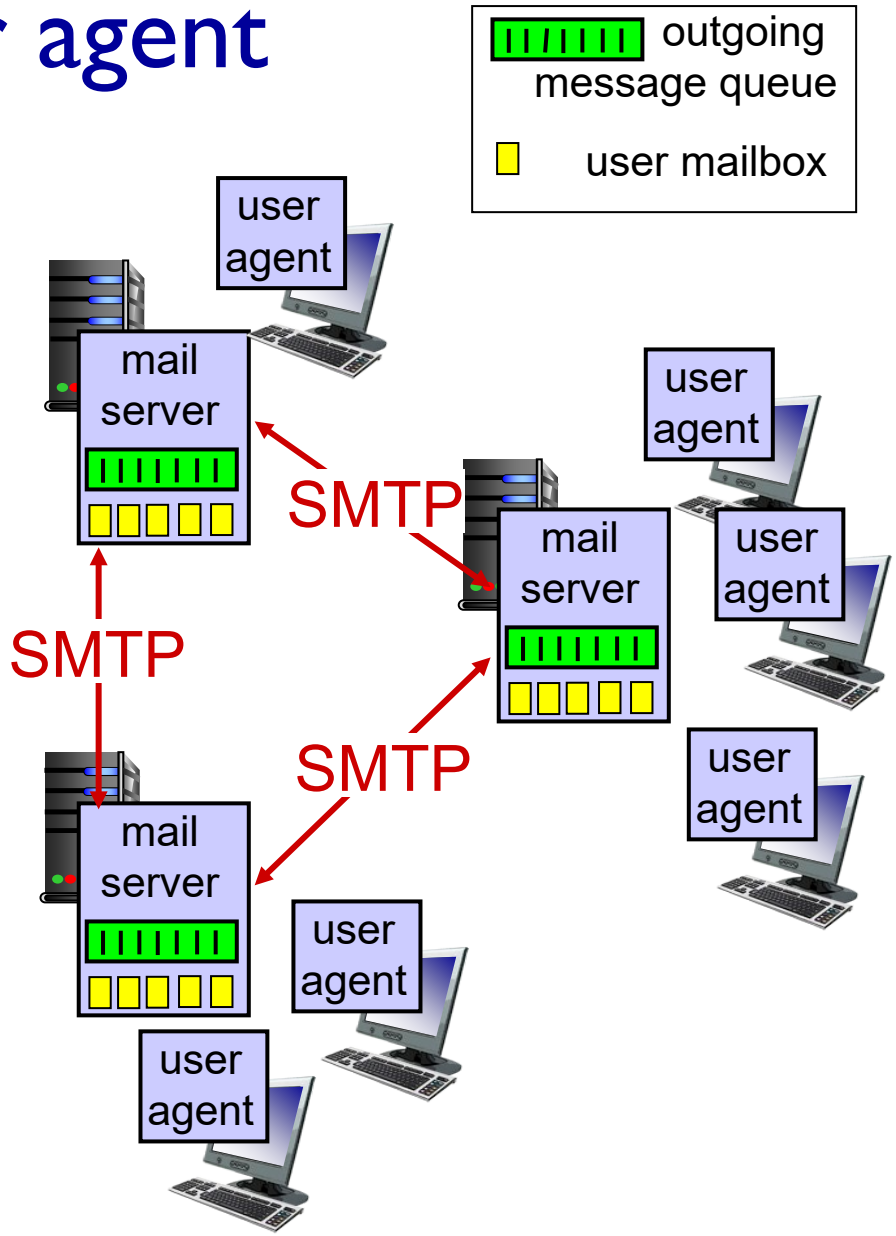
*Major components:*

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

# Electronic mail: user agent

## *User Agent*

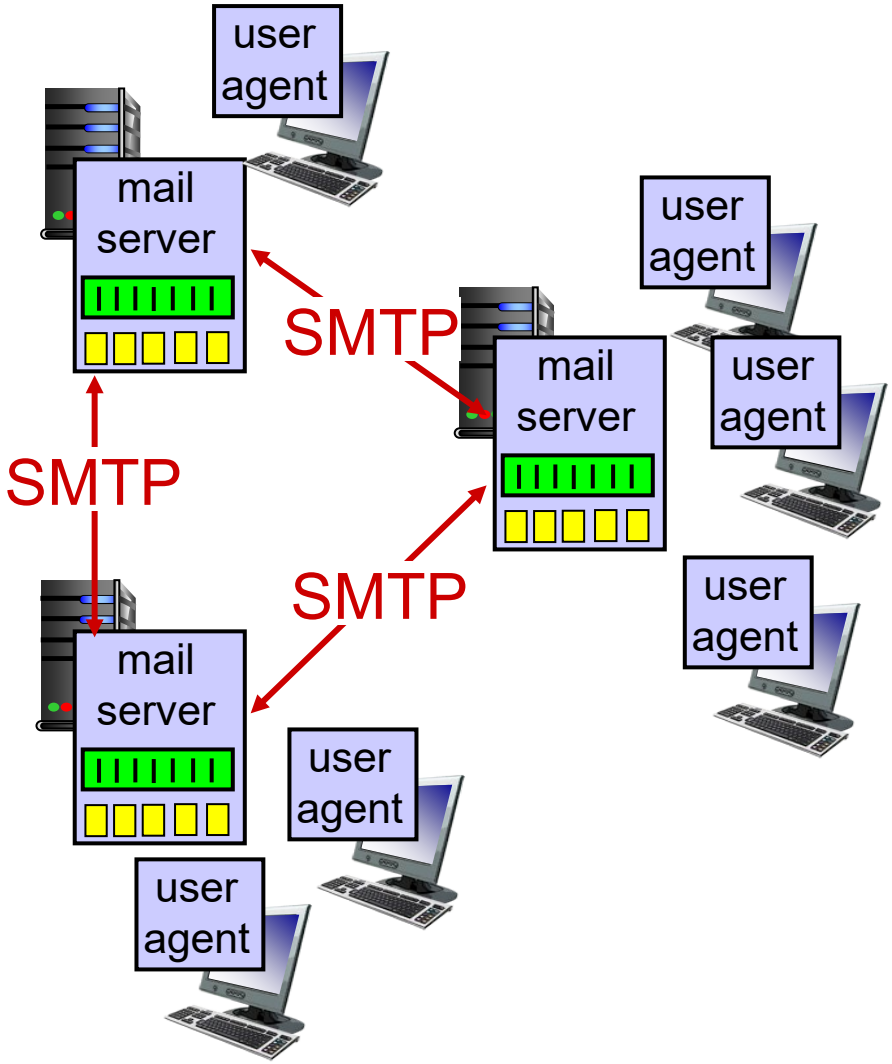
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Outlook, iPhone mail client
- outgoing, incoming messages stored on server



# Electronic mail: mail servers

mail servers:

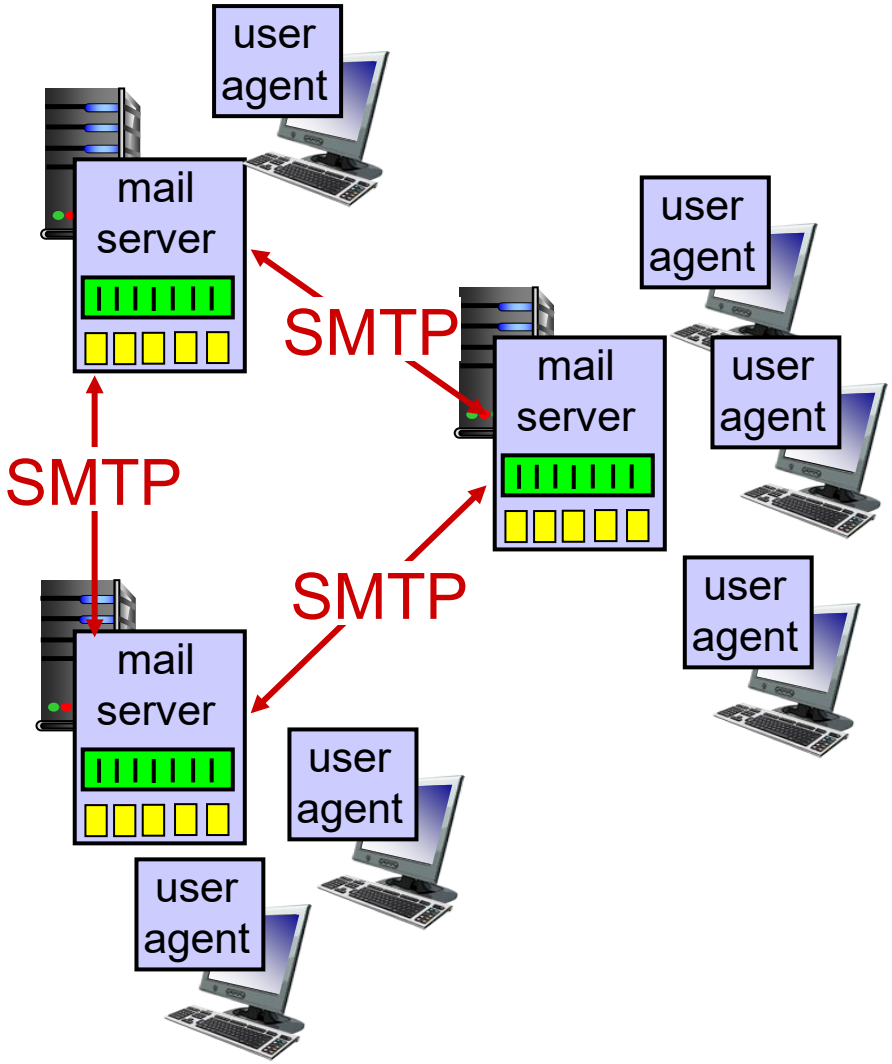
- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages



# Electronic mail: SMTP protocol

*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

# 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 fries?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

# SMTP vs HTTP

SMTP

- SMTP: push
- SMTP: multiple objects sent in multipart message

HTTP

- HTTP: pull
- HTTP: each object encapsulated in its own response message

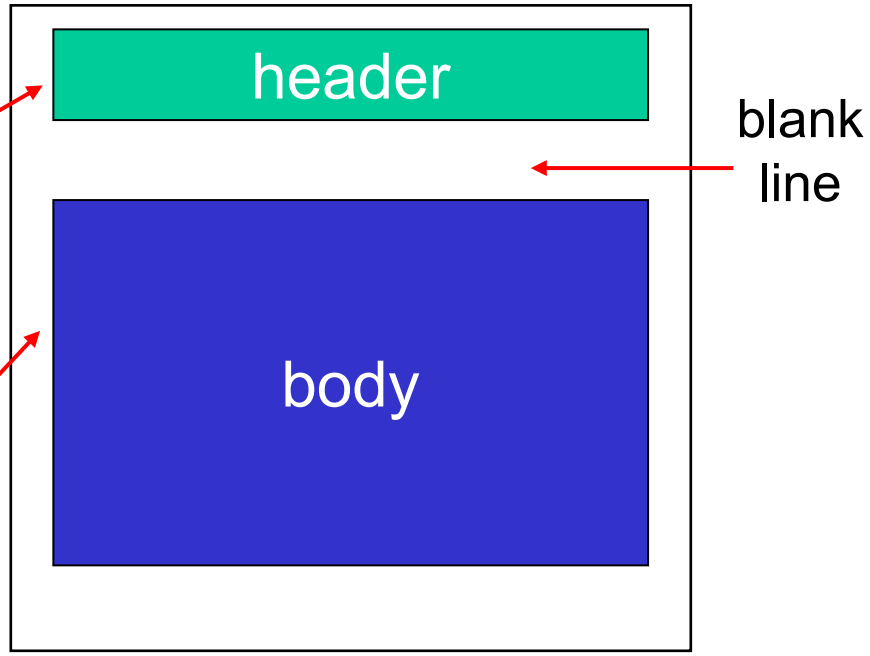
Common: both have ASCII command/response interaction, status codes (but may have different format requirement)

# Mail message format

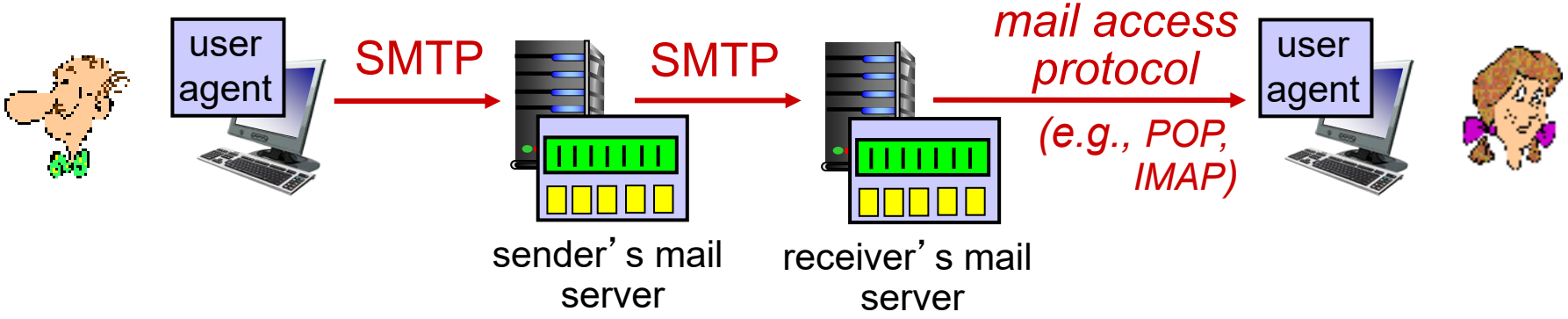
SMTP: protocol for exchanging email messages

RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:
- Body: the "message"
  - ASCII characters only



# 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

Three main phases: authorization; transaction; update

*authorization phase*

- client commands:
  - **`user`**: declare username
  - **`pass`**: password
- server responses
  - **`+OK`**
  - **`-ERR`**

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

# POP3 protocol

*transaction phase,* client:

- **`list`:** list message numbers
- **`retr`:** retrieve message by number
- **`dele`:** delete
- **`Quit`**

**Two modes:**

- Download and delete
  - Bob cannot re-read e-mail if he changes client
- Download and keep
  - copies of messages on different clients

```
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
```

# IMAP

- Another mail access protocol
- 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

*IMAP has many more features than POP3, but it is also significantly more complex*

# Chapter Outline

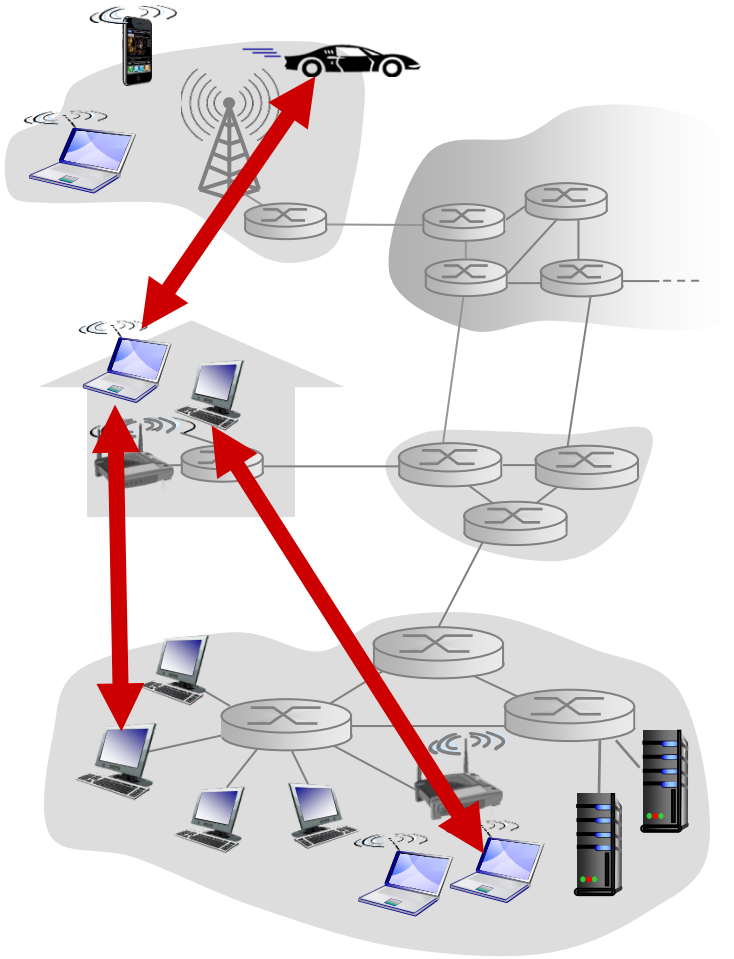
1. principles of network applications
2. Web and HTTP
3. DNS
4. electronic mail
5. P2P applications

# Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

*examples:*

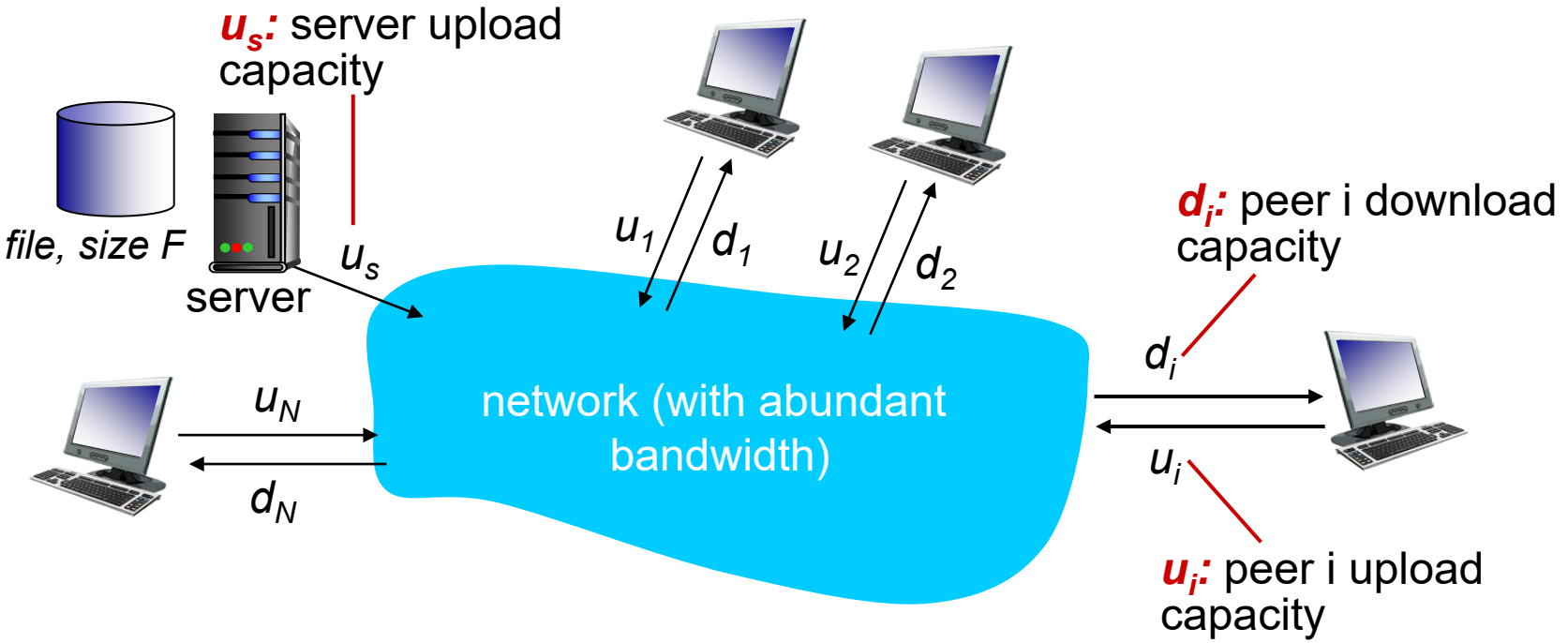
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



# File distribution: client-server vs P2P

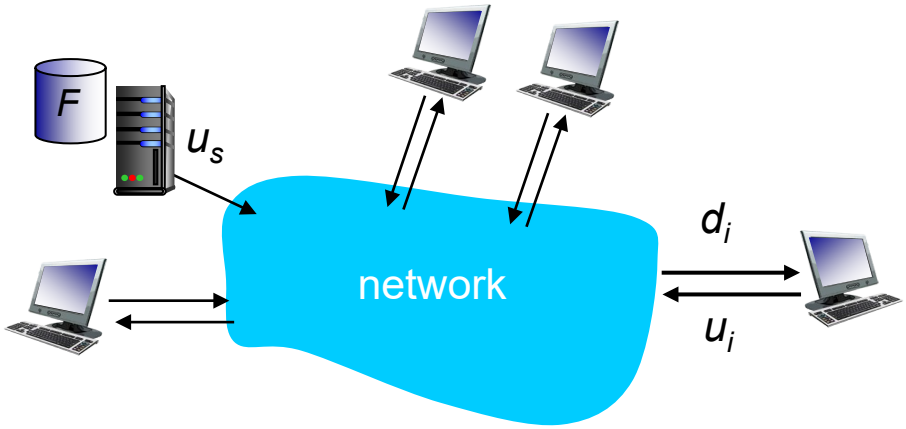
*Question:* how much time to distribute file (size *F*) from one server to *N* peers?

- peer upload/download capacity is limited resource



# 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: $F/d_{min}$



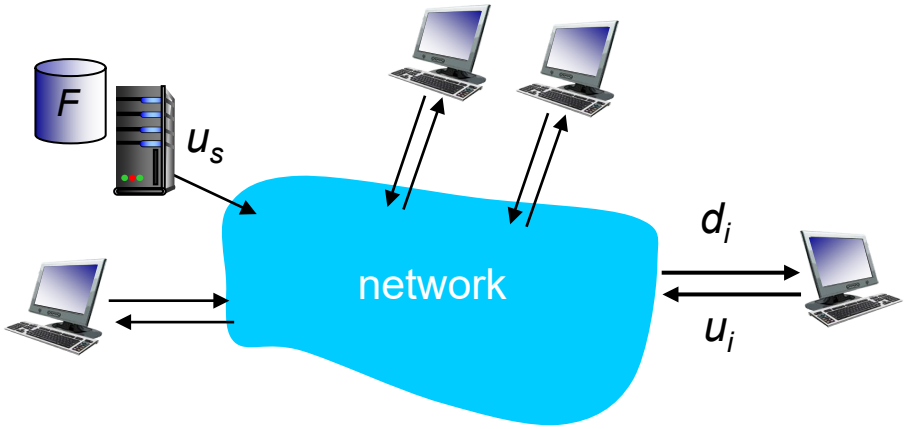

*time to distribute F to N clients using client-server approach*

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

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*

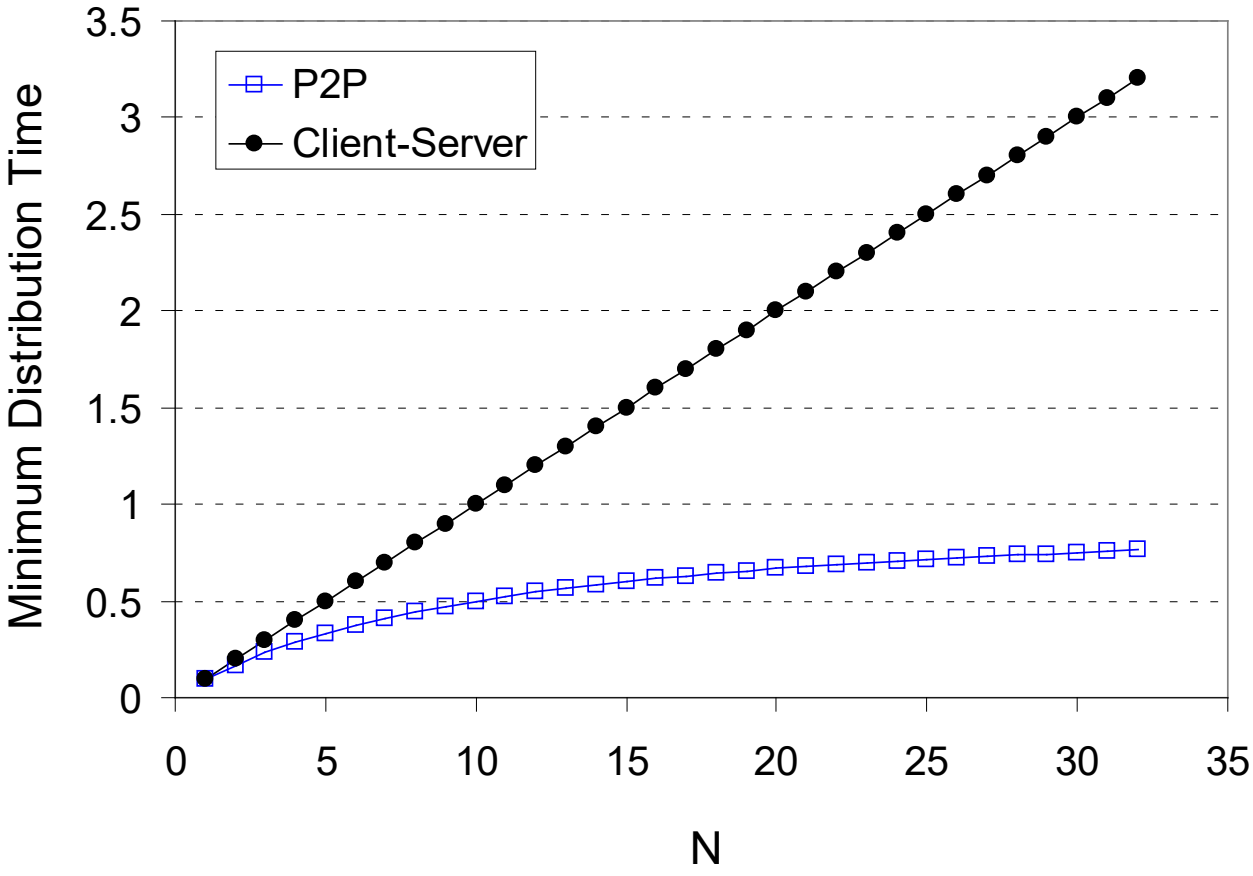
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

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, $u_s = 10u$, $d_{min} \geq u_s$

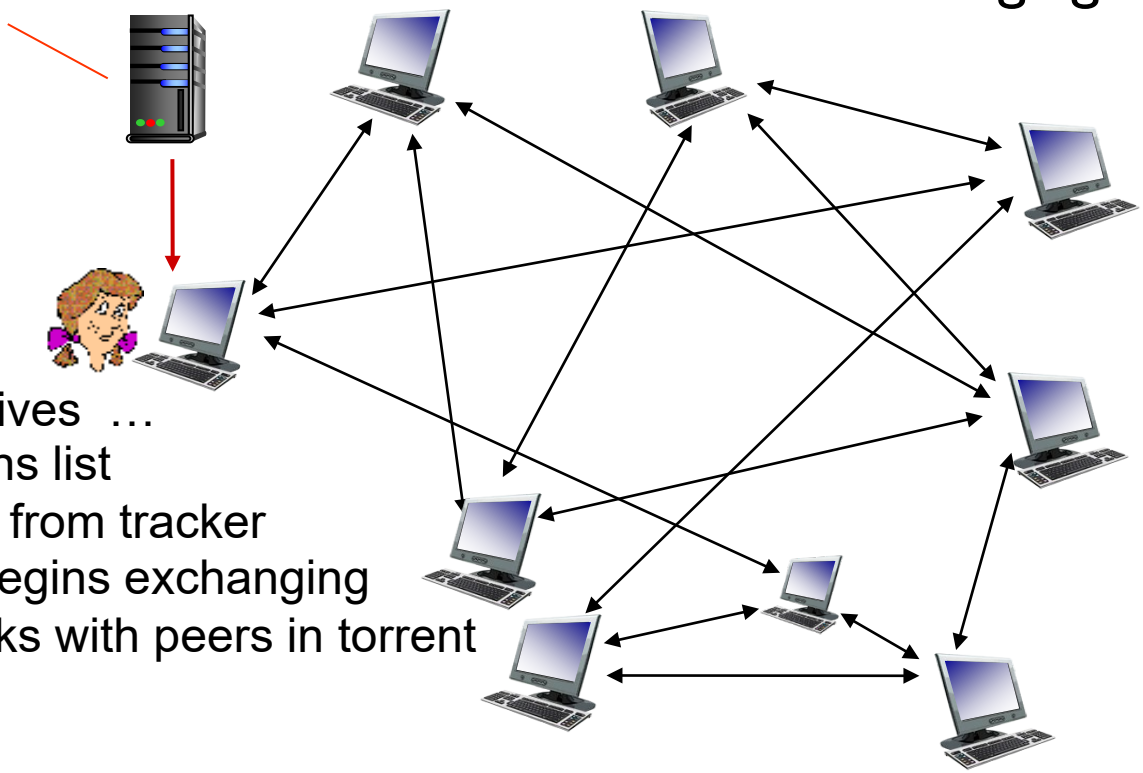


# P2P file distribution: BitTorrent

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

*tracker:* tracks peers participating in torrent

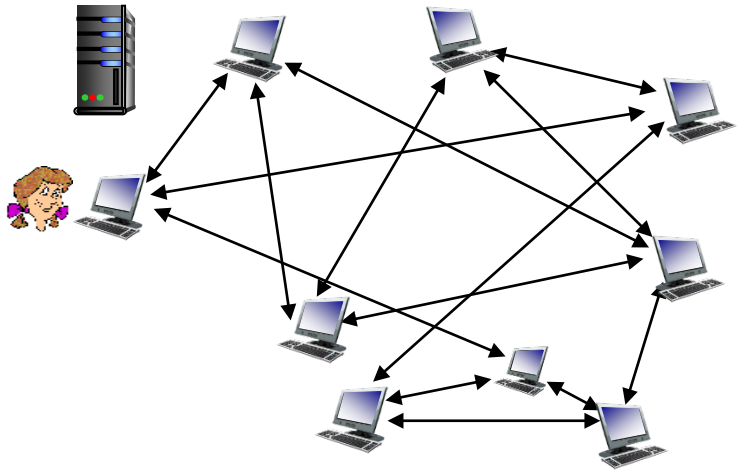
*torrent:* group of peers exchanging chunks of a file



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