CS 305: Computer Networks

Fall 2022

Lecture 4：Application Layer

Ming Tang

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

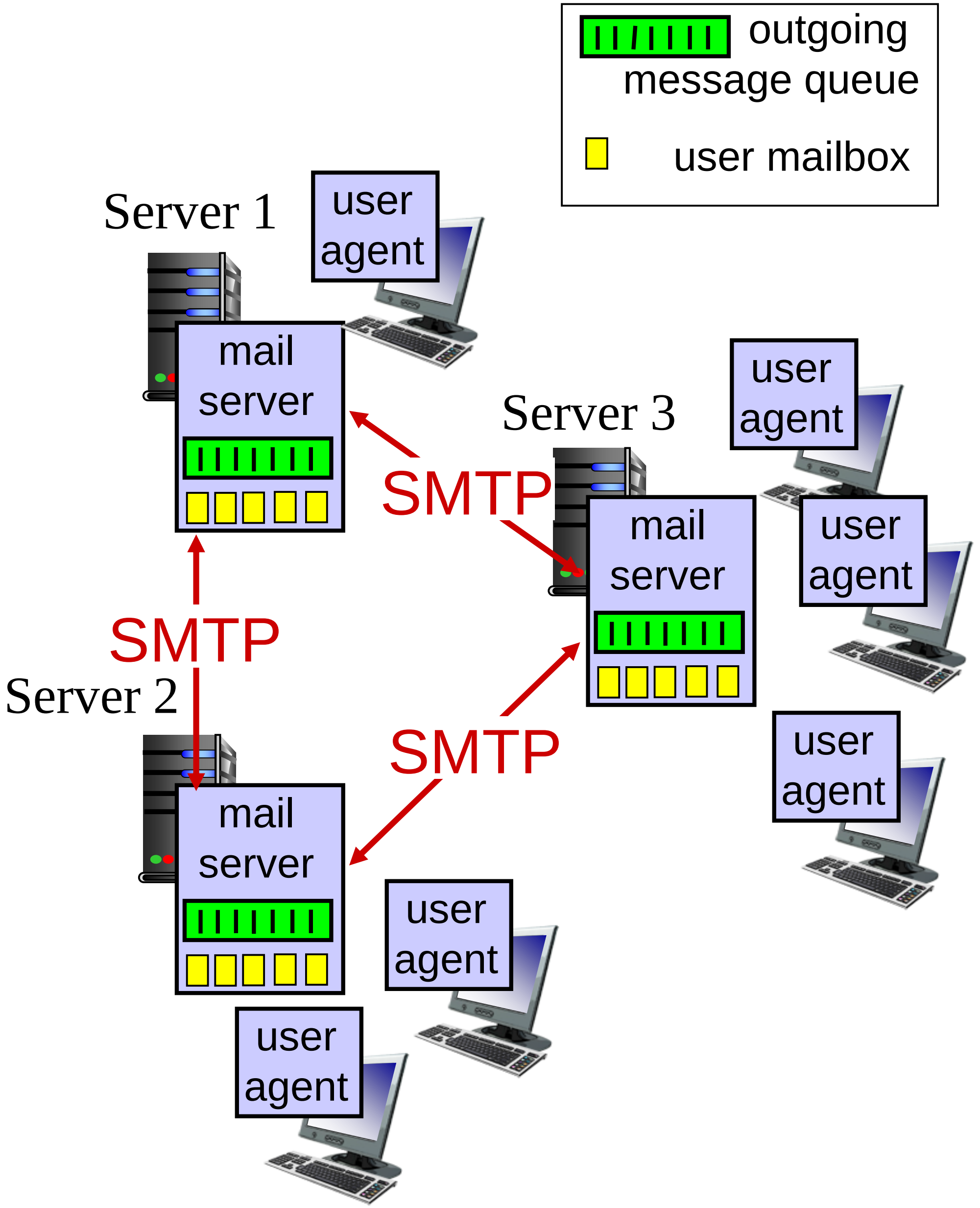
2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Electronic mail



Three major components:

- user agents
- mail servers
- simple mail transfer protocol (SMTP): use TCP

User Agent

- a.k.a. "mail reader"
- Allow users to read, reply to, forward, save and compose messages
- e.g., Outlook, iPhone mail client

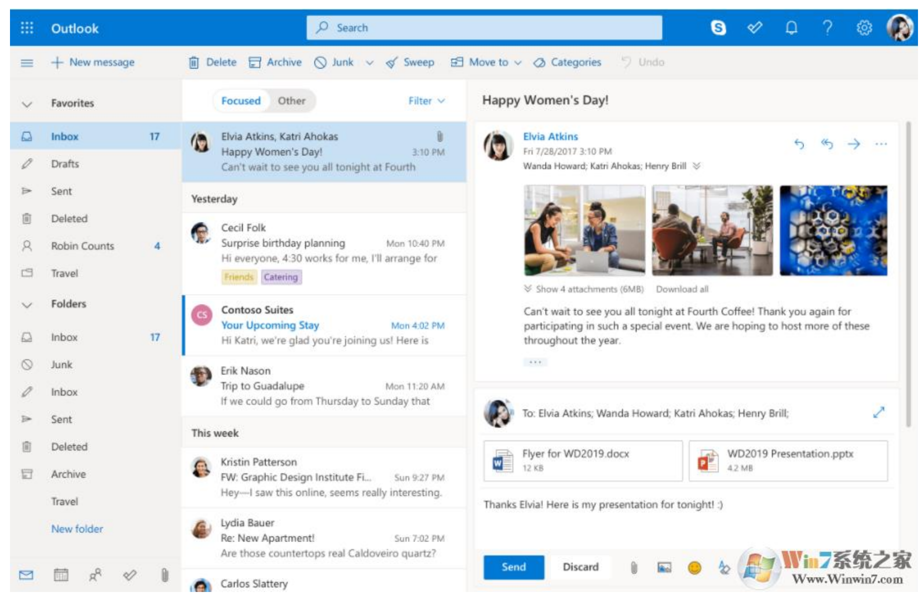
Electronic mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol (SMTP): use TCP

User Agent

- a.k.a. “mail reader”
- Allow users to read, reply to, forward, save and compose messages
- e.g., Outlook, iPhone mail client

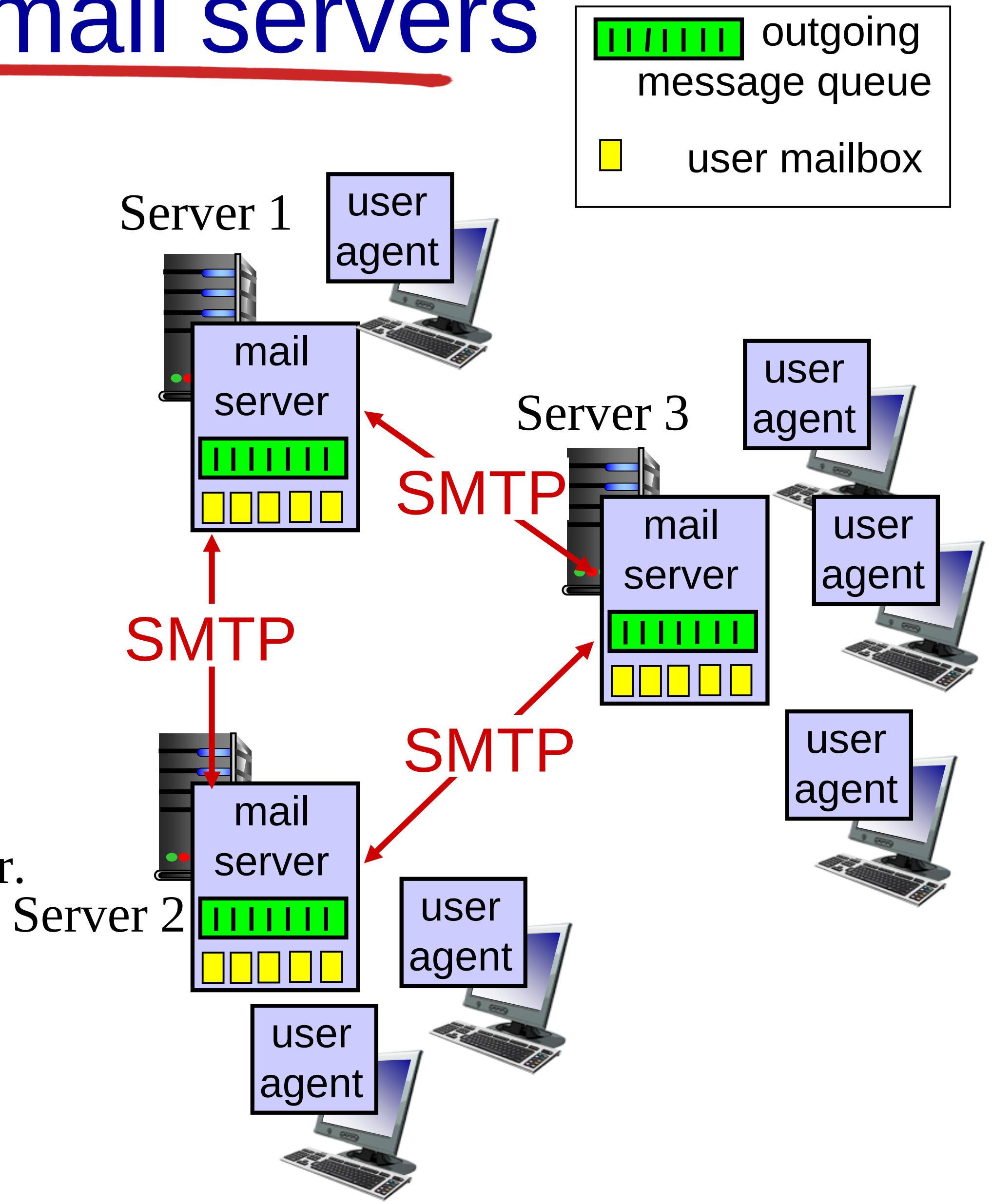


Electronic mail: mail servers

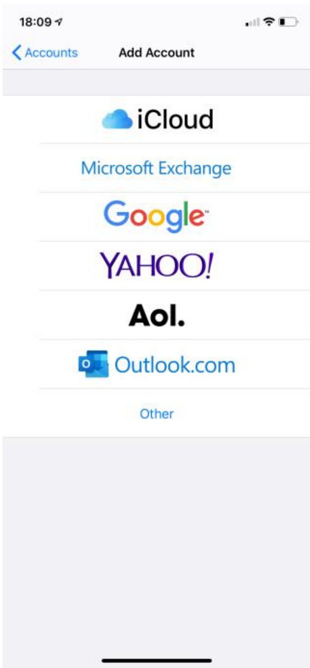
Mail servers:

- *mailbox* contains outgoing, incoming messages
- *message queue* of outgoing (to be sent) mail messages
- *SMTP* between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server

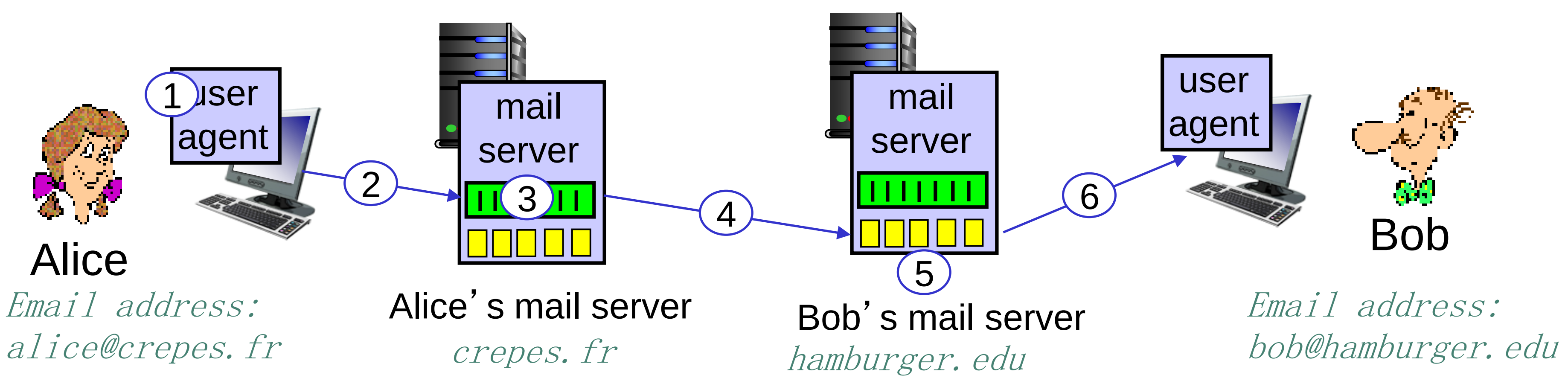
Both client and server sides of SMTP run on every mail server.



Scenario: Alice sends message to Bob

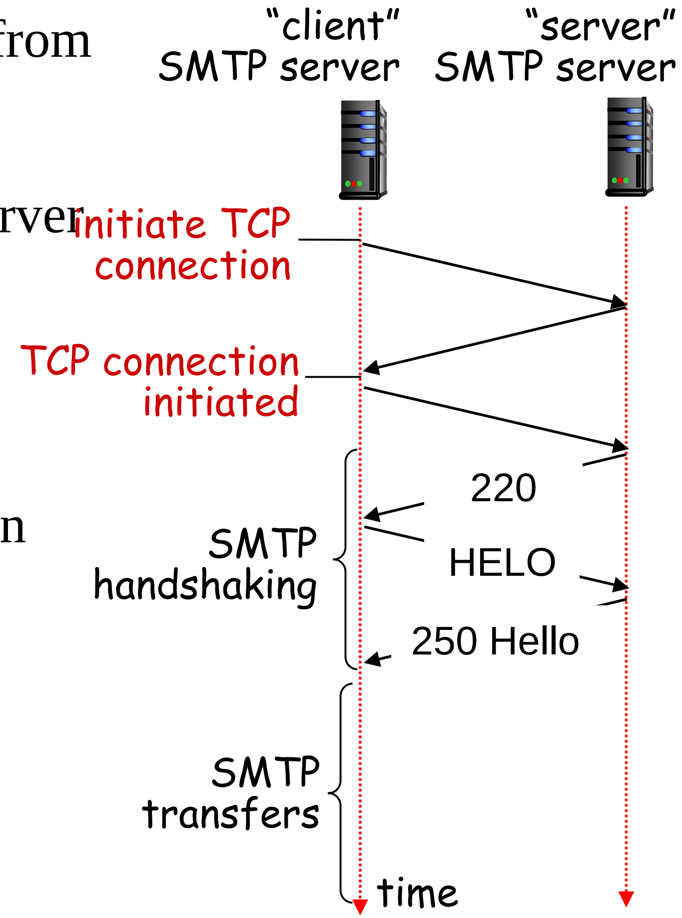


1) Alice uses user agent to compose message "to" `bob@hamburger.edu`
2) Alice's user agent sends message to her mail server; message placed in message queue
3) client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message



Electronic Mail: SMTP [RFC 2821]

- Uses TCP to reliably transfer email message from client to server, port 25
  - If fail, new attempt after a while (e.g., 30 minutes)
- Direct transfer: sending server to receiving server
  - Direct connection, no intermediate mail server
- Three phases of transfer
  - handshaking (greeting): indicate email address
  - transfer of messages: persistent connection
  - closure
- Command/response interaction (like HTTP)
  - commands: ASCII text
  - response: status code and phrase
- Messages (header + body) must be in ASCII
  - Binary multimedia data → ASCII
  - For HTTP, headers are encoded with ASCII



Sample SMTP interaction

The following are exactly the lines the client (C: crepes.fr) and server (S: hamburger.edu) send after they establishing TCP connections.

commands
response (status code + phrase)

SMTP handshaking

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
```

SMTP transfers

```
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
```

Repeat to send multiple messages

Closure

```
C: QUIT
S: 221 hamburger.edu closing connection
```

SMTP: Closing Observations

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Comparison with HTTP:

- HTTP: pull
- SMTP: push

- HTTP: ASCII in header
- SMTP: ASCII in header and body

- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message

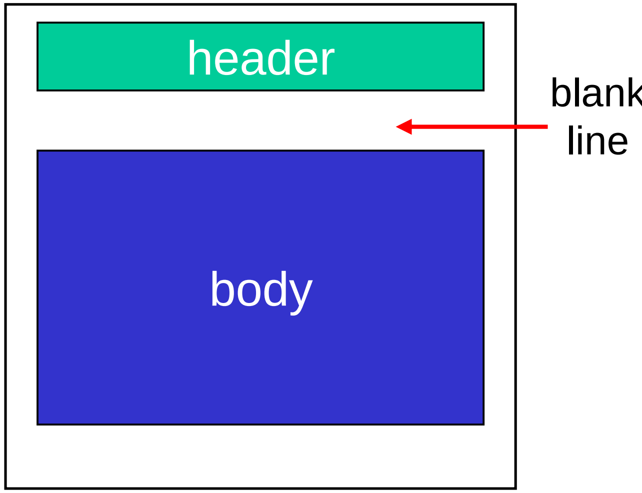
Mail message format

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

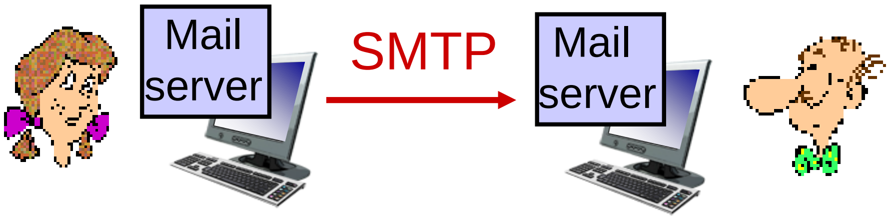
Mail message format (RFC 2822) defines *syntax* for e-mail message itself (like HTML)

- 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

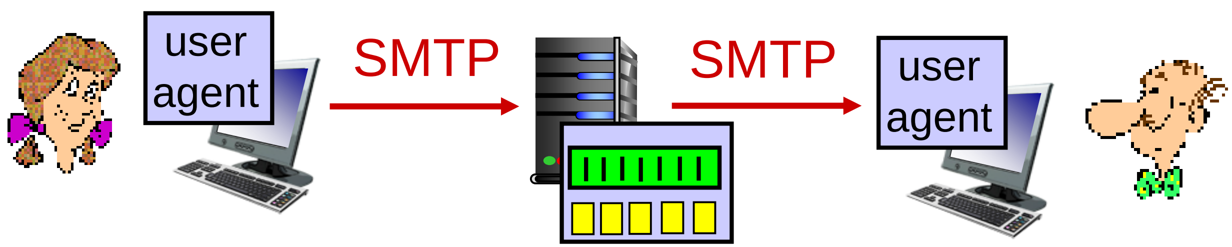


Alternative Choices?



Why not having mail servers directly on user’s local PC?

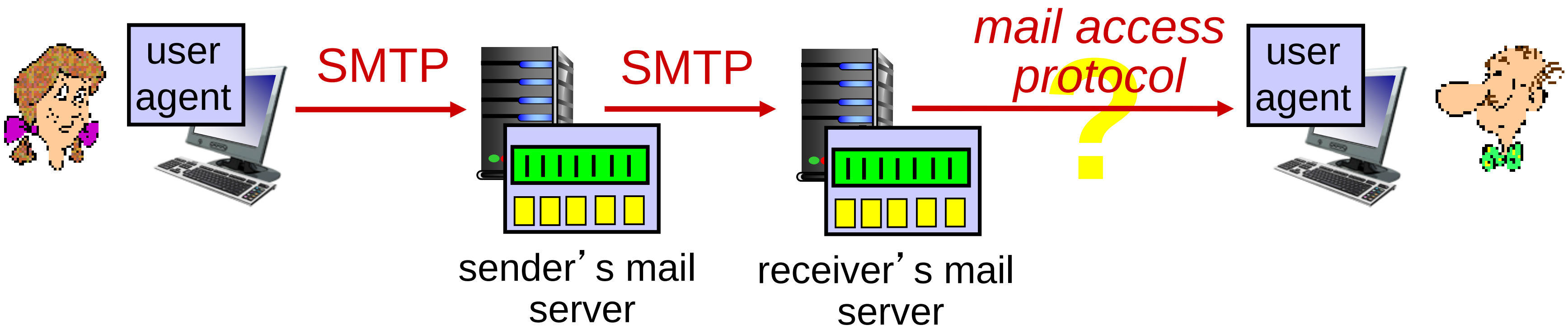
- Recall that a mail server manages mailboxes and runs the client and server sides of SMTP.
- If Bob’s mail server were to reside on his local PC, then Bob’s PC would have to remain always on in order to receive new mail.



Why not letting Alice send to Bob’s mail server directly?

- Bob’s mail sever may fail; need to repeatedly send the message until success.

Mail access protocols



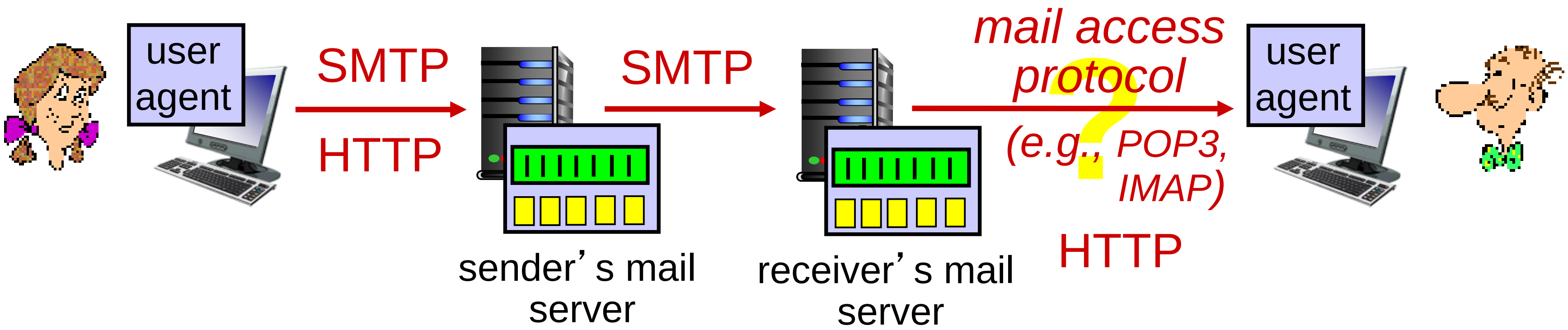
SMTP: delivery to receiver's server

Mail access protocols: How does Bob obtain his message?

SMTP?

No! Because obtaining message is a pull operation.

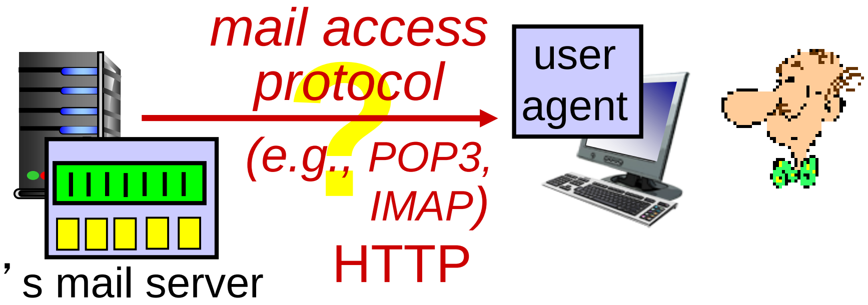
Mail access protocols



Mail access protocol: retrieval from server

- POP3: Post Office Protocol 3: authorization, download
  - TCP, port 110
- IMAP: Internet Mail Access Protocol: more features, including manipulation of stored messages on server
- HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 protocol



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**

Update phase

- After **Quit**, the mail server deletes the messages marked as deletion

```
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
```

Download-and-delete mode

Download-and-keep mode？

POP3 (more) and IMAP

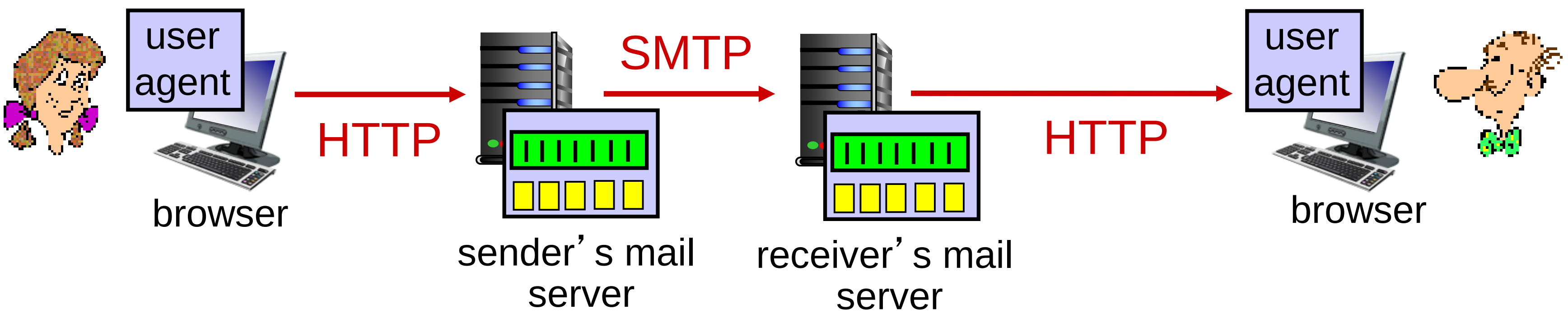
More about POP3

- 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

IMAP

- Maintain a folder hierarchy 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
- Obtain components of messages

Web-based Email



Web-based emails are provided by gmail, Hotmail, Yahoo! Mail, etc.

- The user agent is an ordinary web browser

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

DNS: domain name system

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- hostname, e.g., www.yahoo.com - used by humans
- IP address (32 bit) - used for addressing datagrams

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 and name servers communicate to *resolve* names (address/name translation)

DNS Services

- **hostname to IP address translation**
- **host aliasing**
  - canonical, alias hostnames
  - **www.ibm.com** (alias) is really **servereast.backup2.ibm.com** (canonical)
  - From supplied alias hostname to canonical hostname
- **mail server aliasing**
- **load distribution**
  - replicated Web servers: many IP addresses correspond to one name
  - rotation distributes the traffic (rotate the ordering of IP addresses)

DNS Services

1. An application invoke the client side of DNS, specifying the hostname that needs to be translated.
2. DNS in the user's host takes over, sending a query message into the network. All DNS query and reply messages are sent within UDP datagrams to port 53.
3. After a delay, ranging from milliseconds to seconds, DNS in the user's host receives a DNS reply message that provides the desired mapping.
4. The mapping is then passed to the invoking application.

From the perspective of the invoking application in the user's host, DNS is a black box providing a simple, straightforward translation service.

Why UDP?

- smaller data packets
- doesn't need consistent data to work
- fast speed

DNS Structure

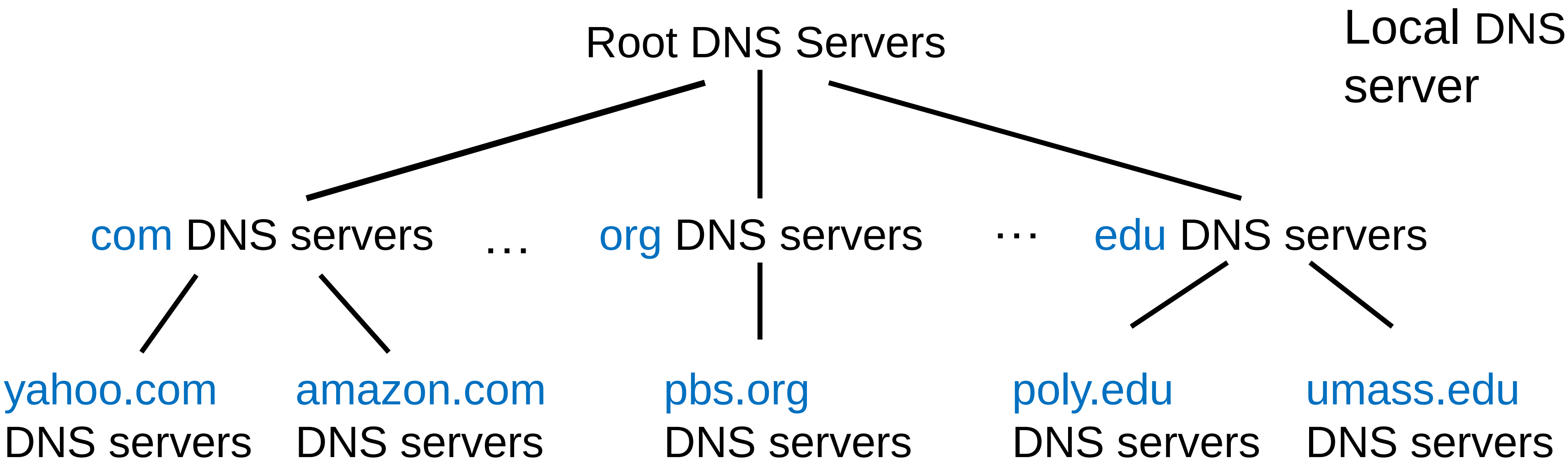
Centralized DNS:
Clients simply direct all queries to the single DNS server, and the DNS server responds directly to the querying clients.

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Maintenance: huge database, update frequently

A: doesn‘t scale!

DNS: a distributed, hierarchical database

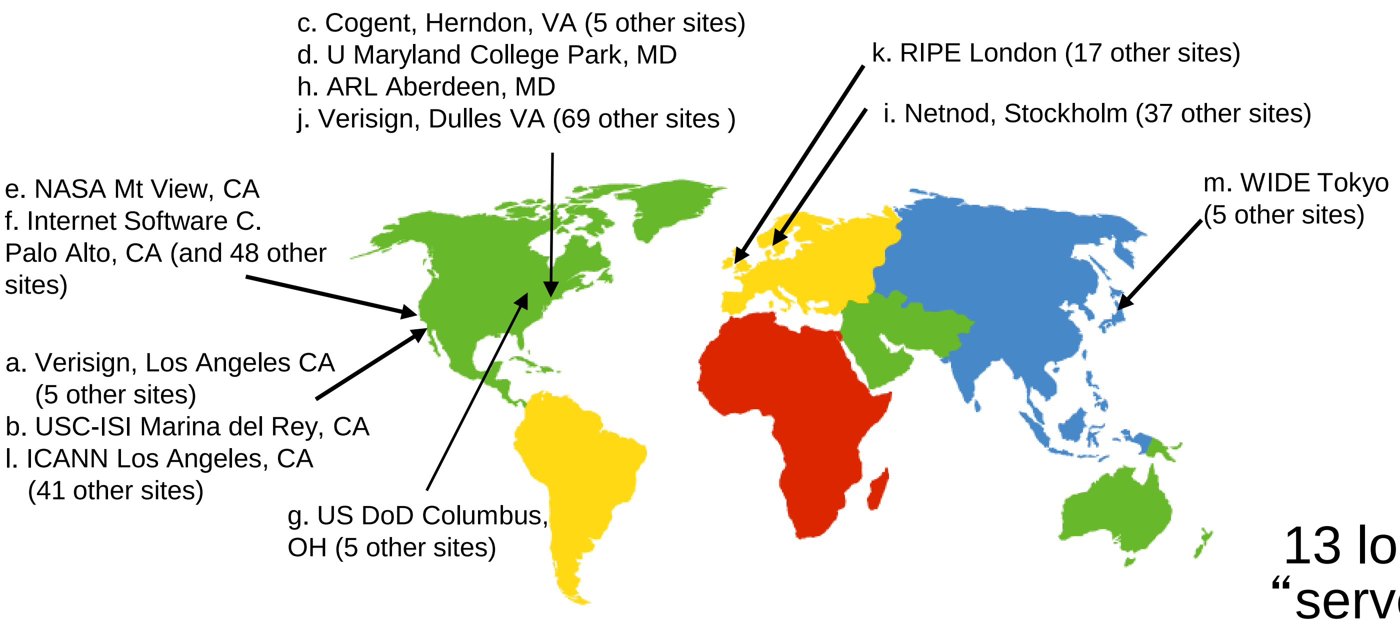


Client wants IP for www.amazon.com:

- Root DNS Servers: client queries root server to find .com DNS server
- Top-Level Domain (TLD) DNS: client queries .com DNS server to get amazon.com DNS server
- Authoritative DNS servers: client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: root servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Provide the IP addresses of the TLD servers



13 logical root name "servers" worldwide

TLD, authoritative servers

Top-level domain (TLD) servers:

- Top-level domains: com, org, net, edu, aero, jobs, museums; top-level country domains: uk, fr, ca, jp
- *Network Solutions* maintains servers for .com TLD
- *Educause* for .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 server

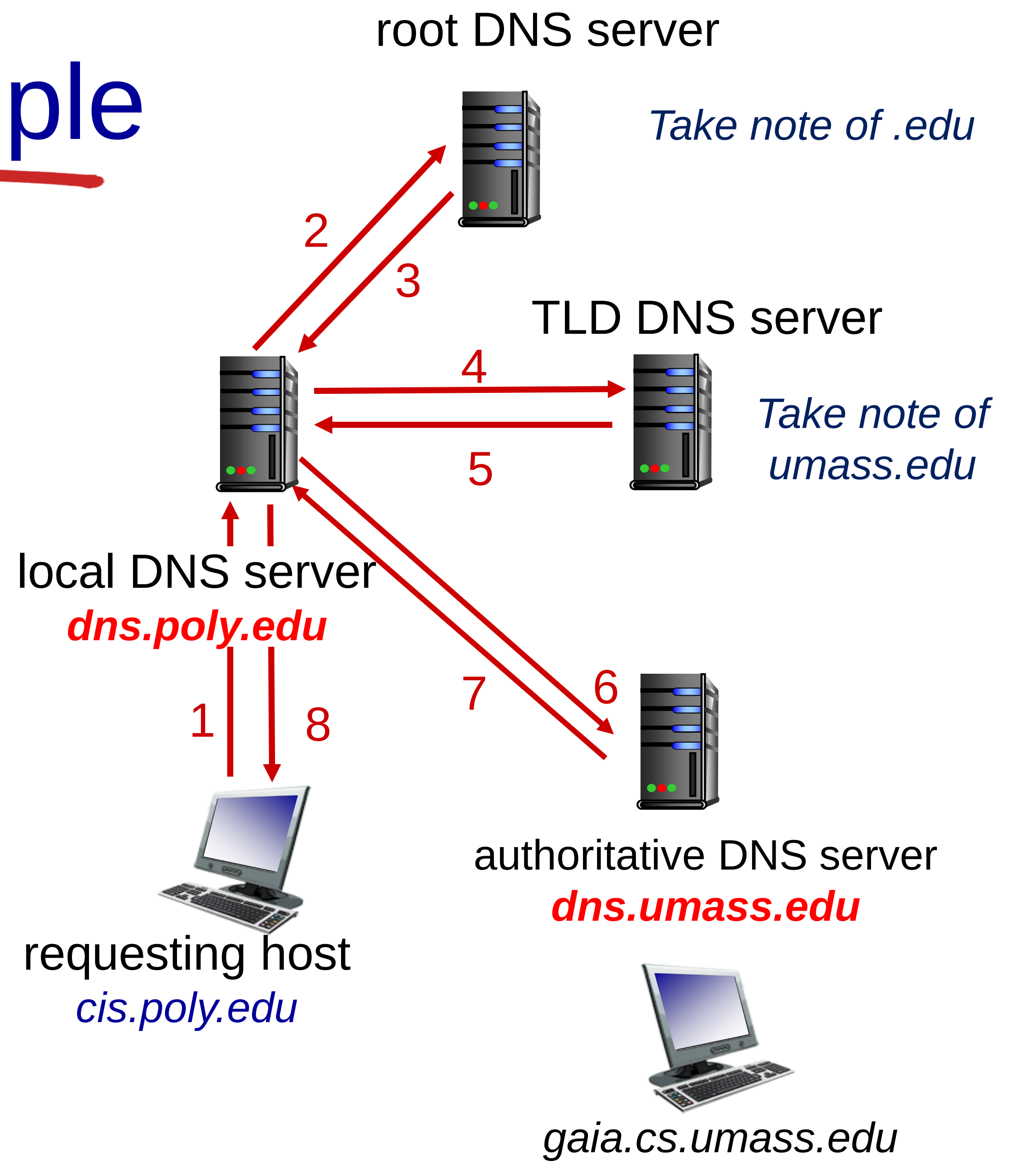
- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one
  - also called "default name server"
- When a host connects to an ISP, the ISP provides the IP addresses of one or more of local DNS servers
  - A host's local DNS server may be typically "close to" the host
- 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

DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Iterated query:

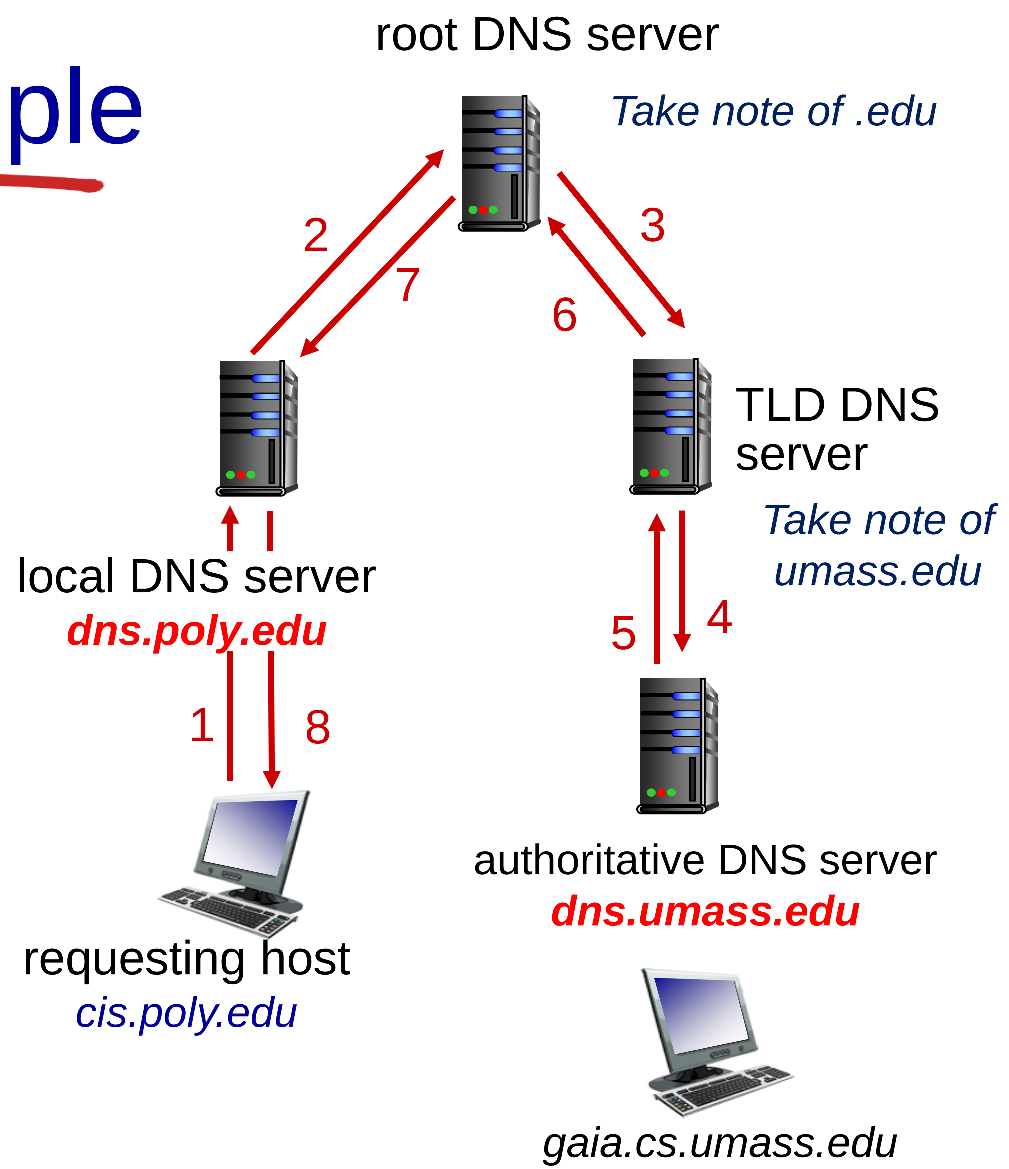
- contacted server replies with the name of another server to contact
- "I don't know this name, but ask this server"



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
  - TLD servers typically cached in local name servers
  - thus root DNS servers not often visited
- Cached entries may be *out-of-date*
  - cache entries timeout (disappear) after some time
  - if name host changes IP address, may not be known Internet-wide until all time-to-lives (TTLs) expire
- Update/notify mechanisms proposed IETF standard
  - RFC 2136

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 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

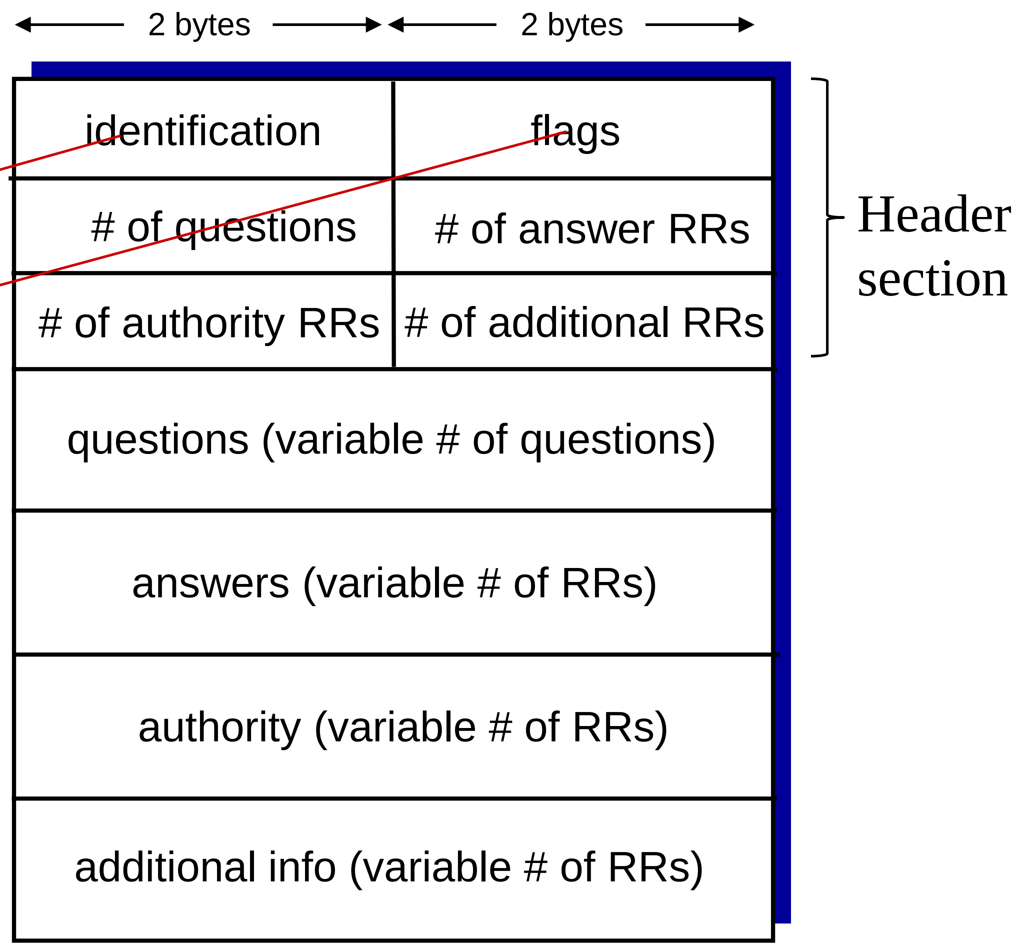
- **value** is canonical name of the mailserver with **name** (alias name)

DNS protocol, messages

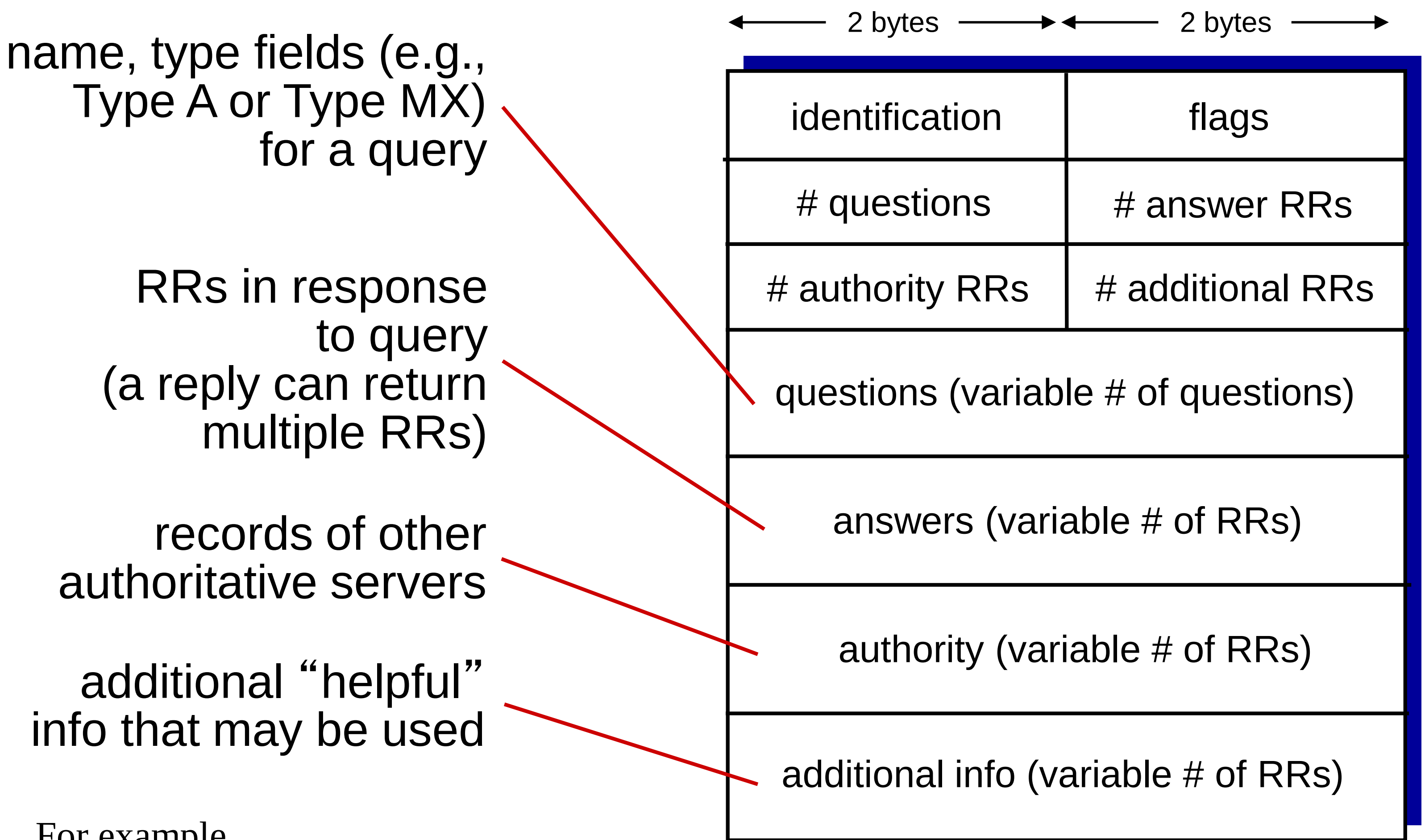
Query and reply messages, both with same message format

message header

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



DNS protocol, messages



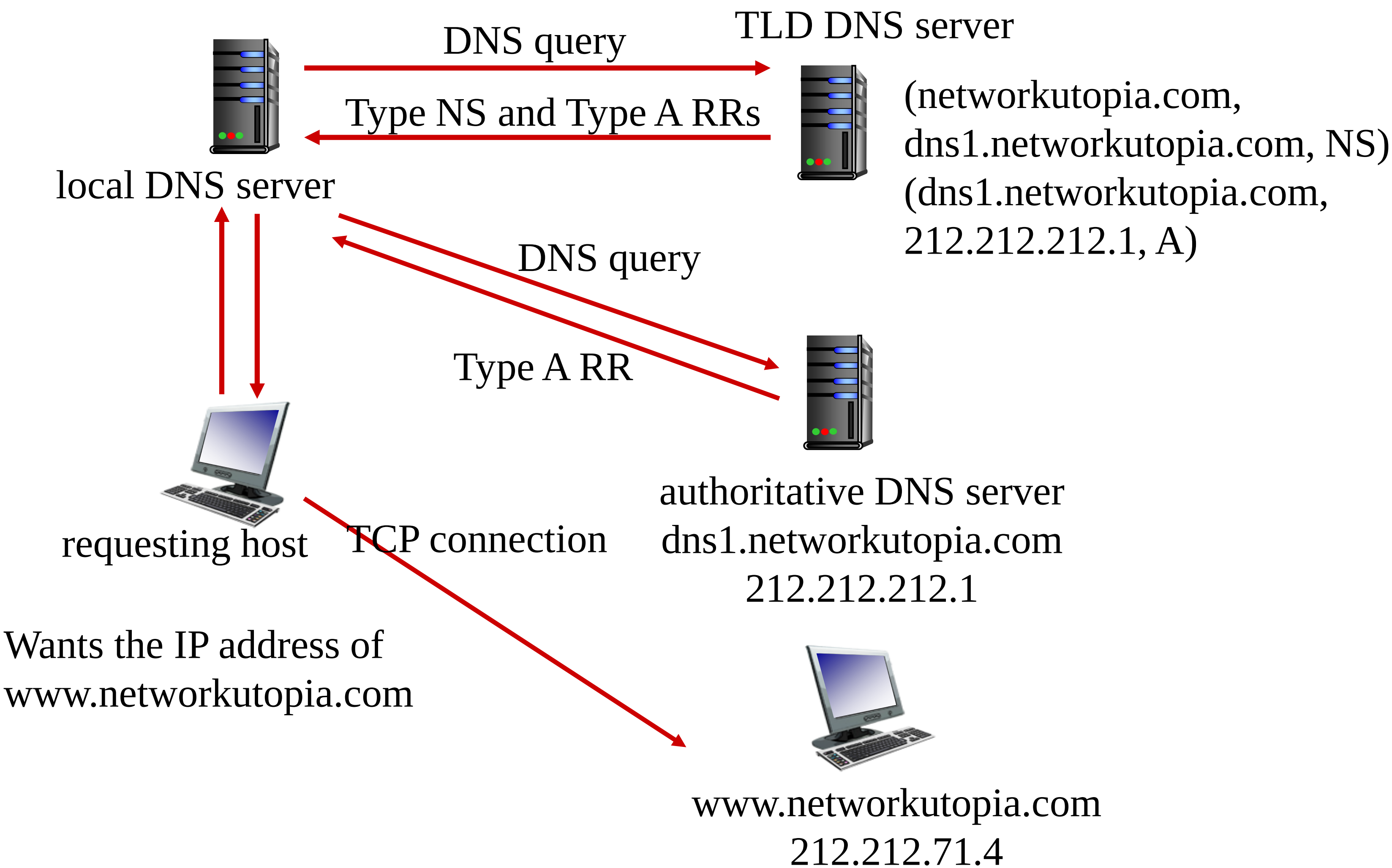
For example,

- Answer field in a reply to an MX query: an RR providing the canonical hostname of a mail server.
- Additional section: a Type A record providing the IP address for the canonical hostname of the mail server.

Inserting records into DNS

- Example: new startup "Network Utopia"
- Register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative DNS server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)

Inserting records into DNS



Attacking DNS

Distributed denial-of-service (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

Redirect attacks

- man-in-middle
  - Intercept queries；bogus reply
- DNS poisoning
  - Send bogus replies to DNS server

Exploit DNS for DDoS

- target IP
- Redirect an unsuspecting Web user to attack Web site

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

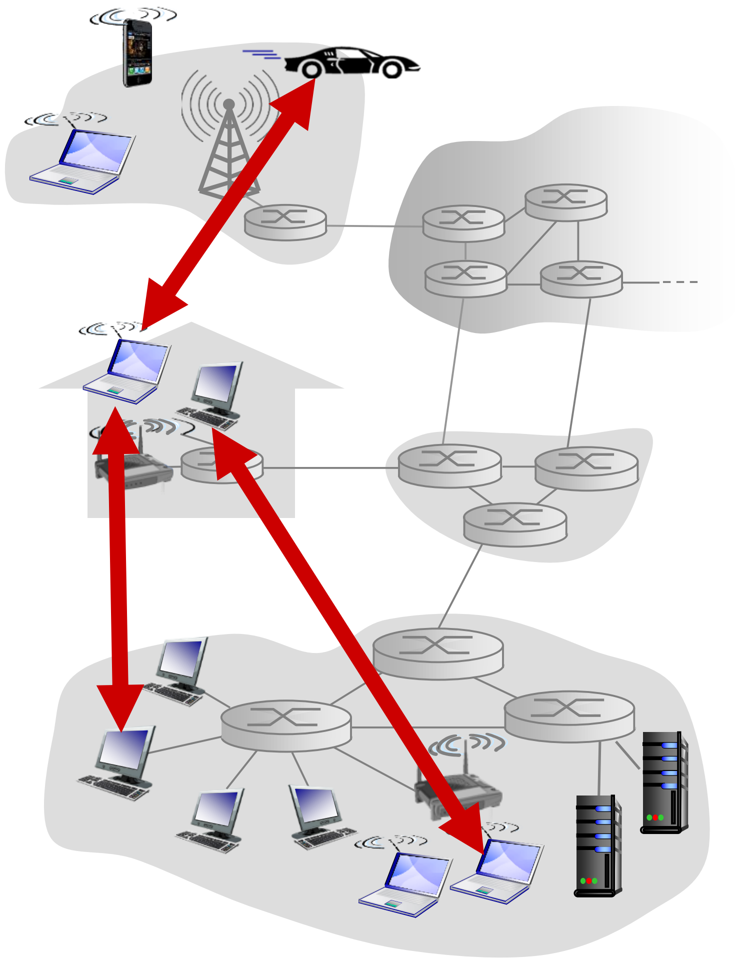
2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP

Pure P2P architecture

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



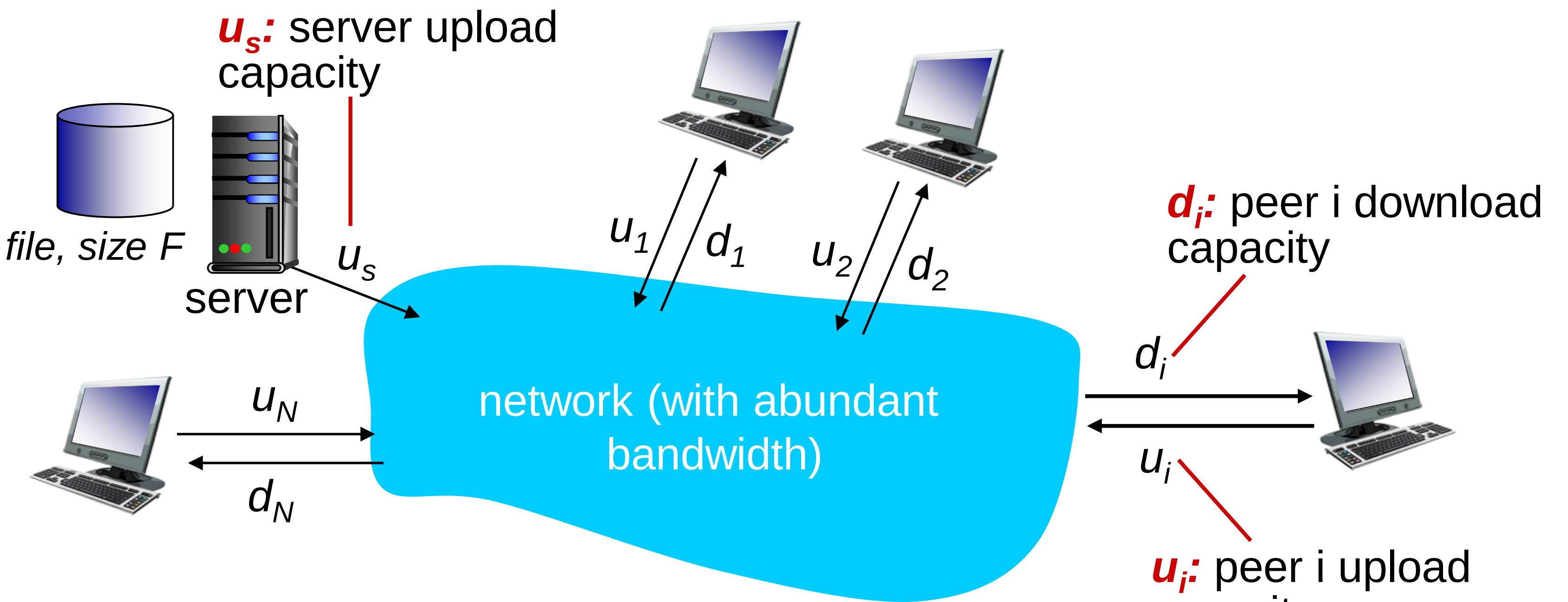
Examples:

- 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*?

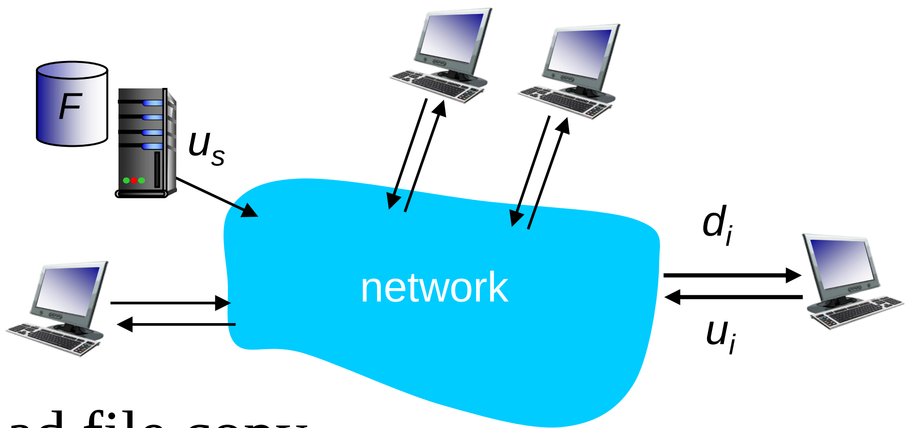
- peer upload/download capacity is limited resource



The **distribution time** is the time it takes to get a copy of the file to all N peers.

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}

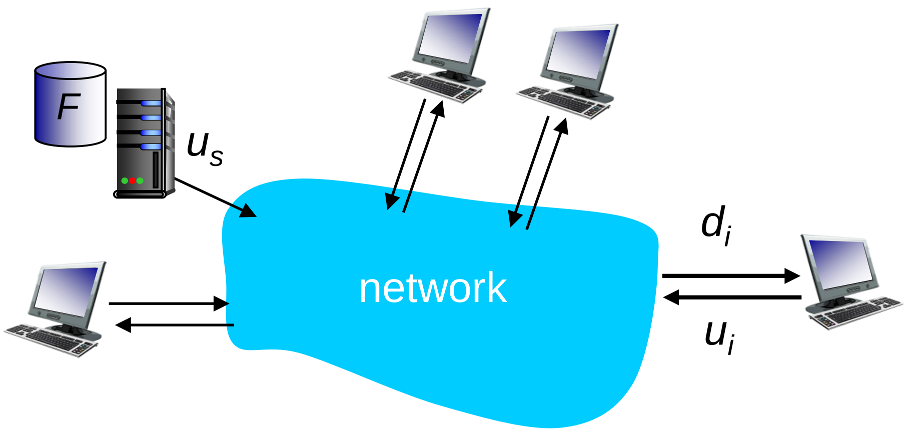
time to distribute F to N clients using client-server approach

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

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 downloading: each client must download file copy
  - min client download time: F/d_{min}
- Clients and server: delivering a total of 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

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

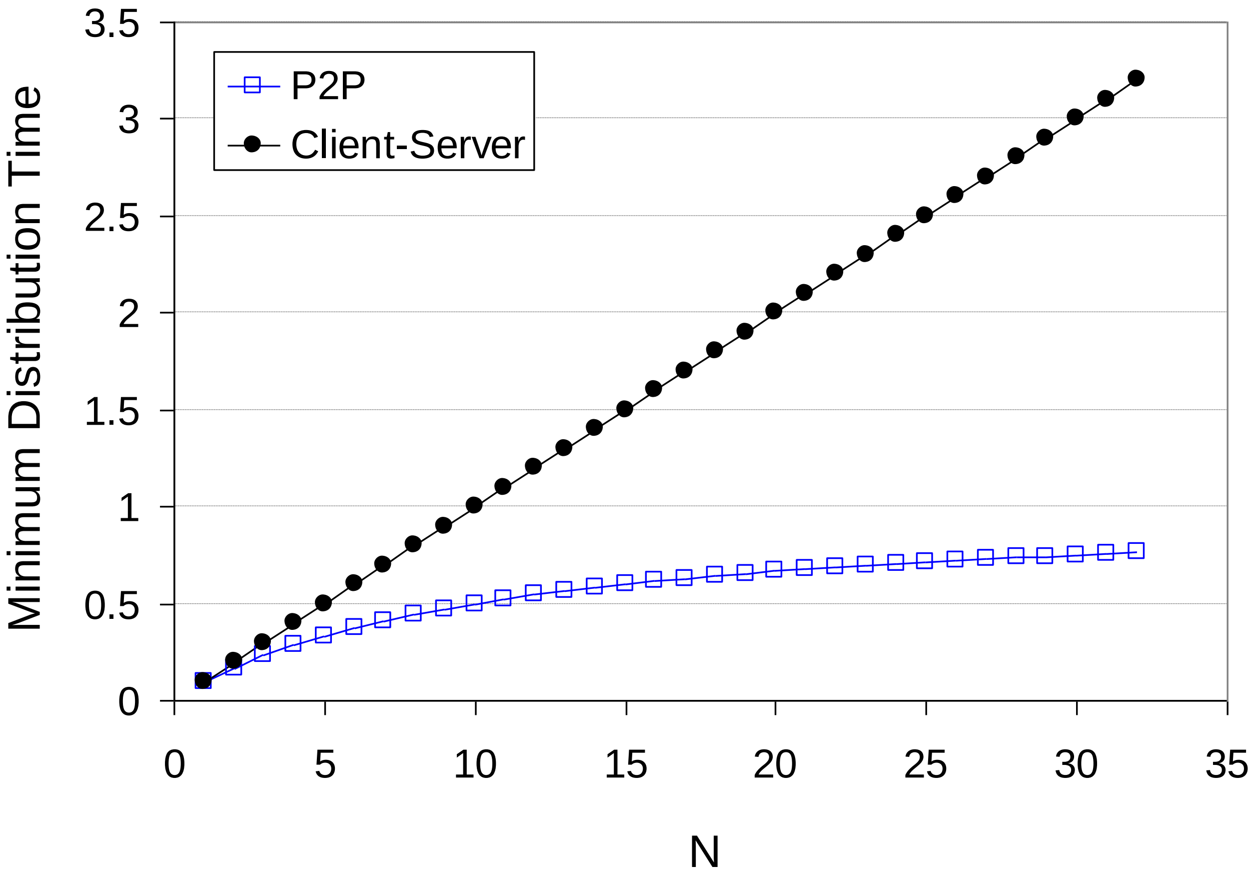
If each peer can redistribute a bit as soon as it receives the bit, then there is a scheme that actually achieves this lower bound

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

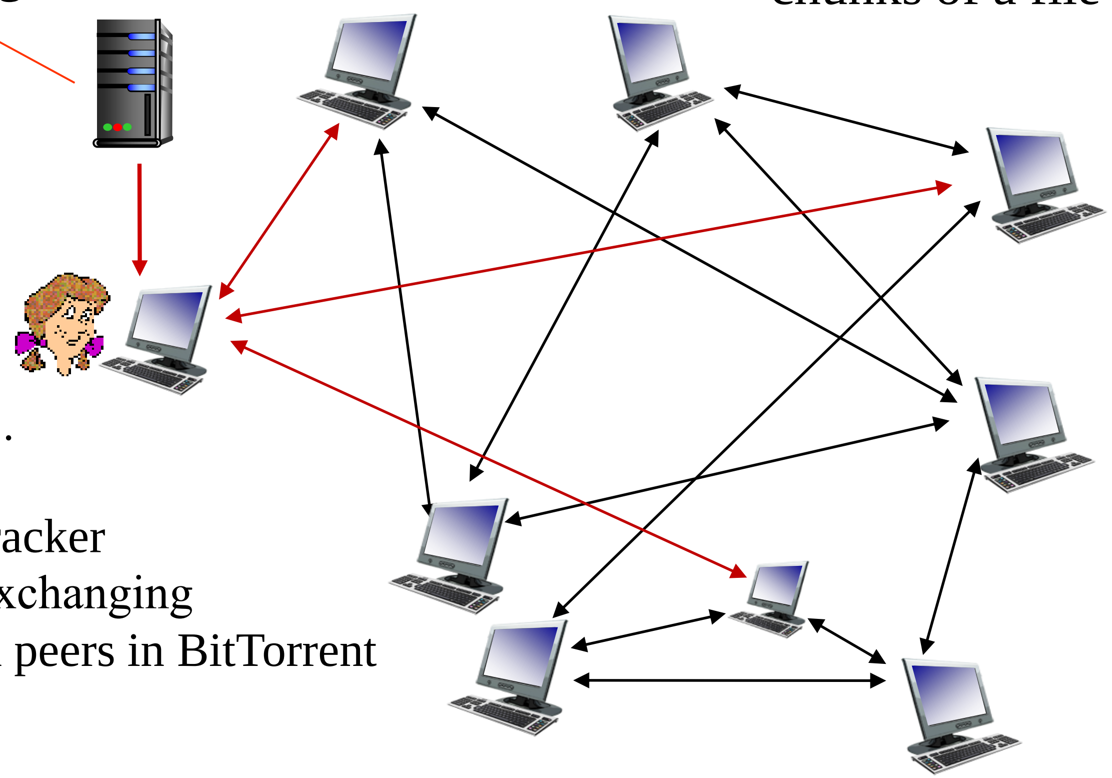


P2P file distribution: BitTorrent

- File divided into 256Kb chunks
- Peers in BitTorrent send/receive file chunks

tracker: tracks peers participating in BitTorrent

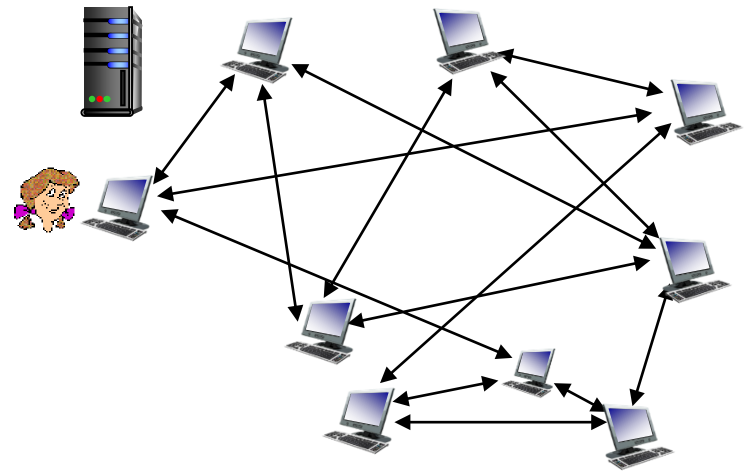
torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in BitTorrent

P2P file distribution: BitTorrent

- Peer joining BitTorrent:
  - 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
  - Peers may come and go
  - The neighbors may change
- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in BitTorrent

BitTorrent: requesting, sending file chunks

Q1: which chunks should she request first from her neighbors?
Q2: to which of her neighbors should she send requested 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

sending chunks: tit-for-tat

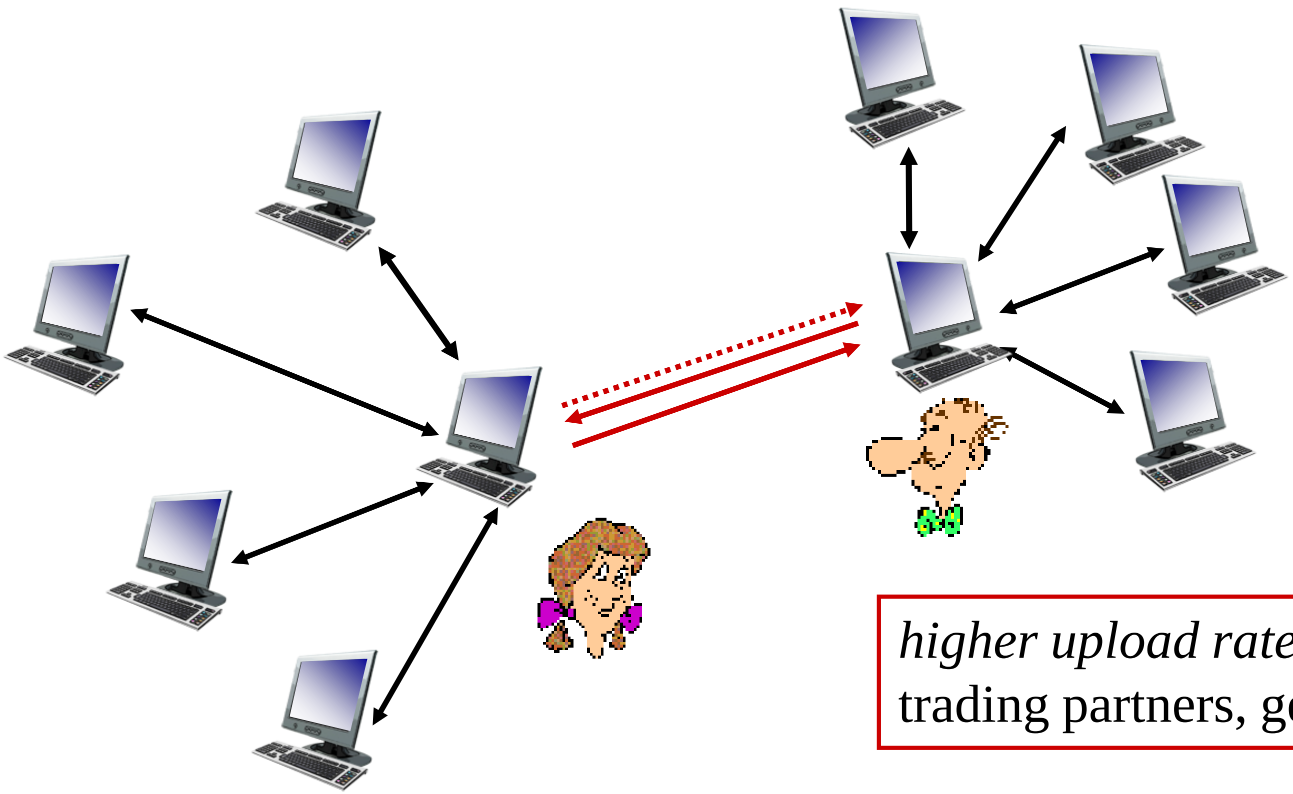
- Alice sends chunks to those four peers currently sending her chunks at highest rate
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster!

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks

2.7 socket programming with UDP and TCP