

Chapter 2

Application Layer

Reti di Elaboratori
Corso di Laurea in Informatica
Università degli Studi di Roma "La Sapienza"
Canale A-L

Prof.ssa Chiara Petrioli

Parte di queste slide sono state prese dal materiale associato al libro
Computer Networking: A Top Down Approach, 5th edition.

All material copyright 1996-2009

J.F Kurose and K.W. Ross, All Rights Reserved

Thanks also to Antonio Capone, Politecnico di Milano, Giuseppe Bianchi and
Francesco LoPresti, Un. di Roma Tor Vergata

DNS: domain name system

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit in IPv4) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - 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
 - 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

why not centralize DNS?

DNS: services, structure

DNS services

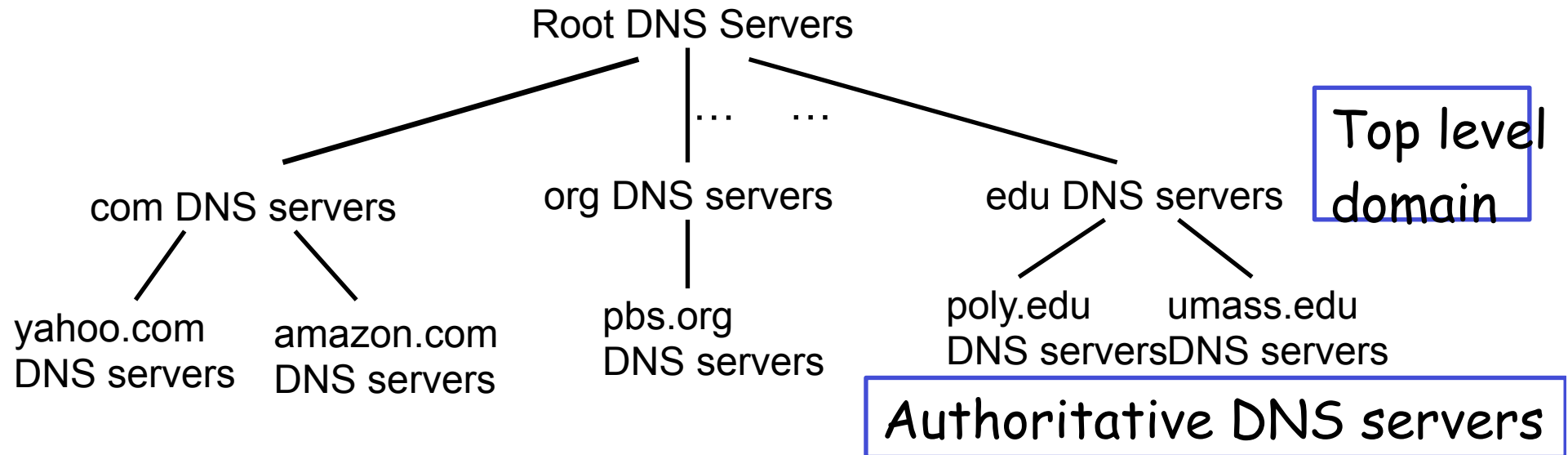
- ❑ hostname to IP address translation
- ❑ host aliasing
 - canonical, alias names
- ❑ mail server aliasing
- ❑ load distribution
 - replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

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

A: doesn't scale!

DNS: a distributed, hierarchical database

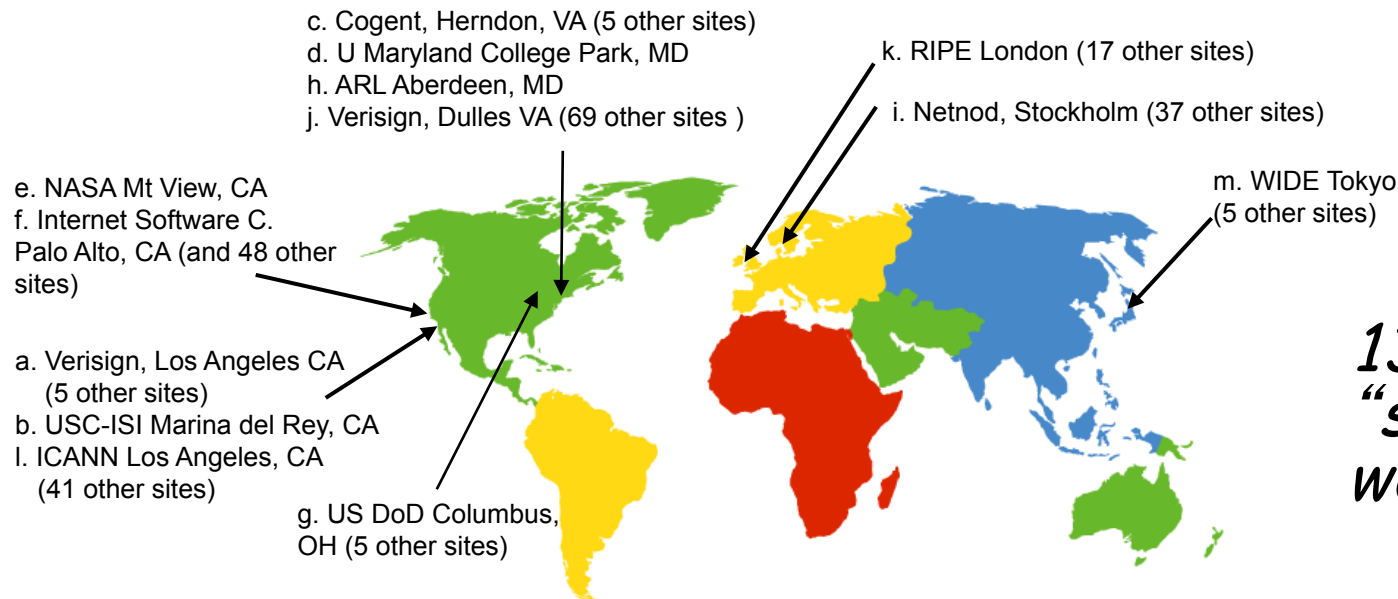


client wants IP for www.amazon.com; 1st approx:

- ❑ 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

- ❑ contacted by local name server that can not resolve name
- ❑ root name server:
 - could contacts authoritative name server if name mapping not known (in recursive queries)
 - gets mapping
 - returns mapping to local name server



*13 root name
“servers”
worldwide*

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp, eu
- 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 name server

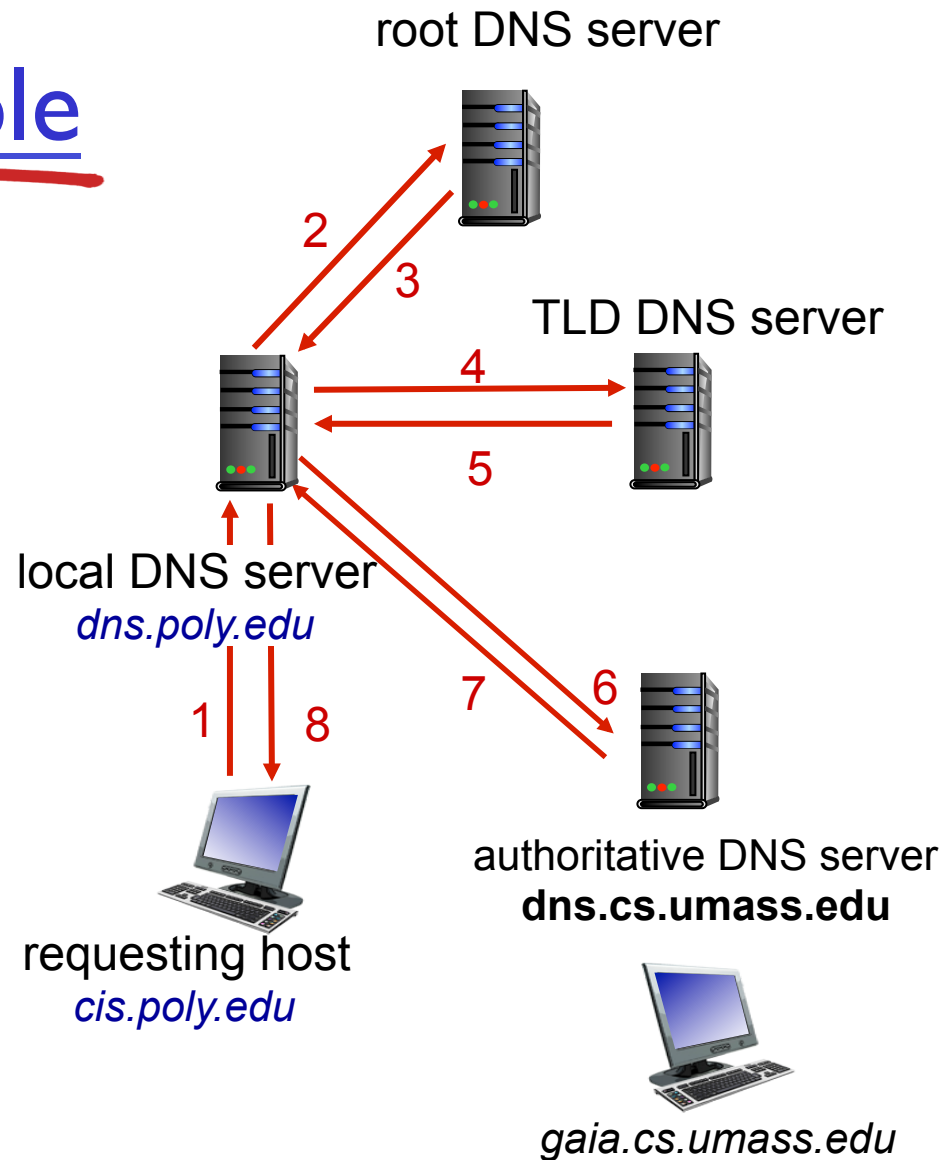
- ❑ does not strictly belong to hierarchy
- ❑ each ISP (residential ISP, company, university) has one
 - also called “default name server”
- ❑ when host makes DNS query, query is sent to its local DNS server
 - has local cache of recent name-to-address translation pairs (but may be out of date!)
 - acts as proxy, forwards query into hierarchy

DNS name resolution example

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

iterated query:

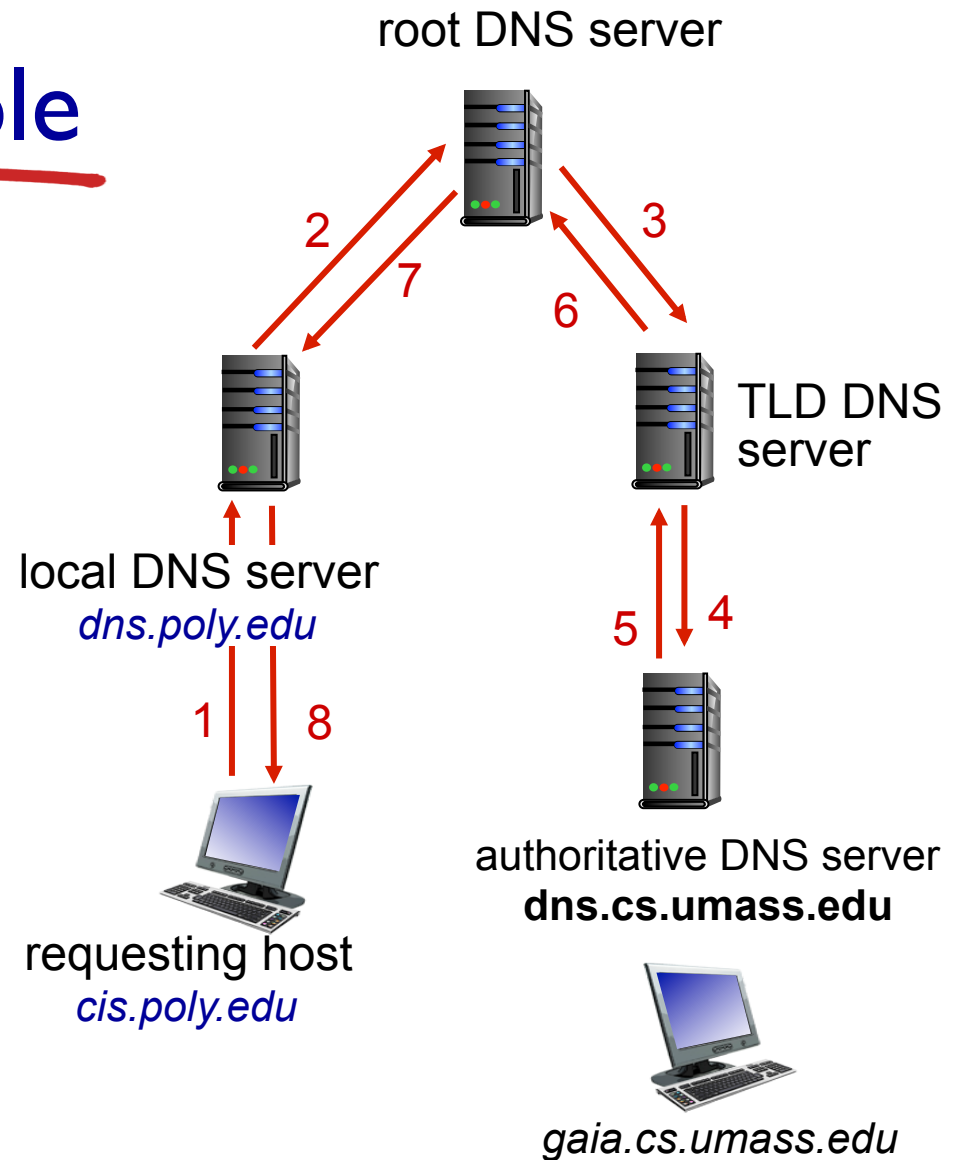
- ❖ contacted server replies with name of server to contact
- ❖ “I don’t know this name, but ask this server”



DNS name resolution 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
- ❑ update/notify mechanisms proposed IETF standard
 - RFC 2136

DNS records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

type=A

- **name** is hostname
- **value** is IP address
(relay.bar.foo.com,
145.37.93.126,A)

type=NS

- **name** is domain (e.g.,
foo.com)
- **value** is hostname of
authoritative name
server for this domain
(foo.com,dns.foo.com,NS)

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

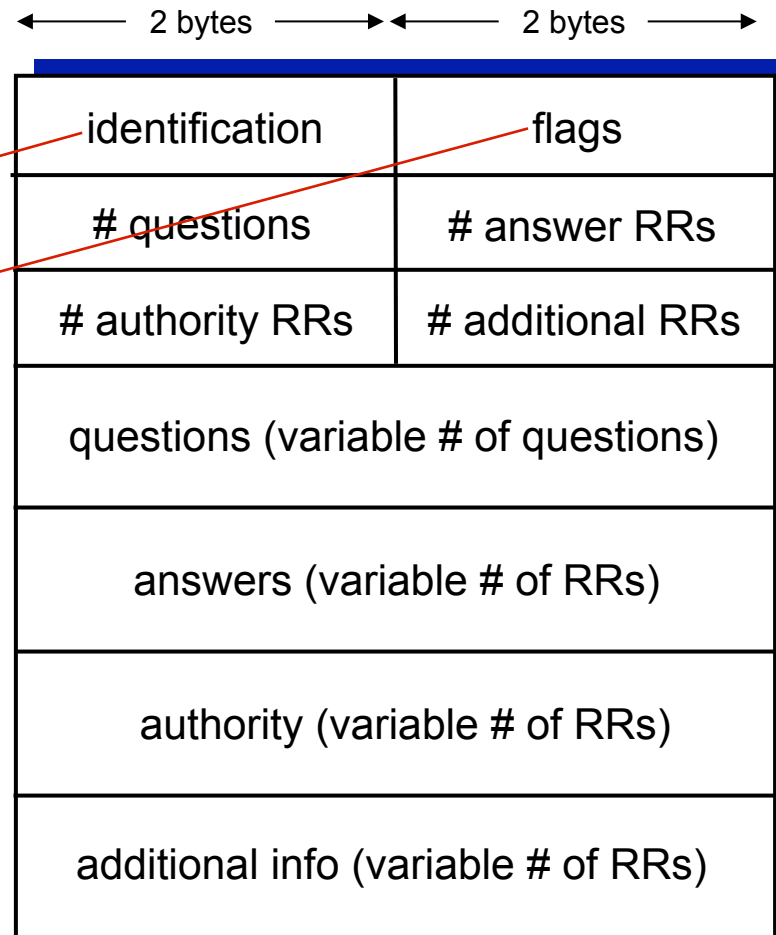
- **value** is name of
mailserver associated with
name

DNS protocol, messages

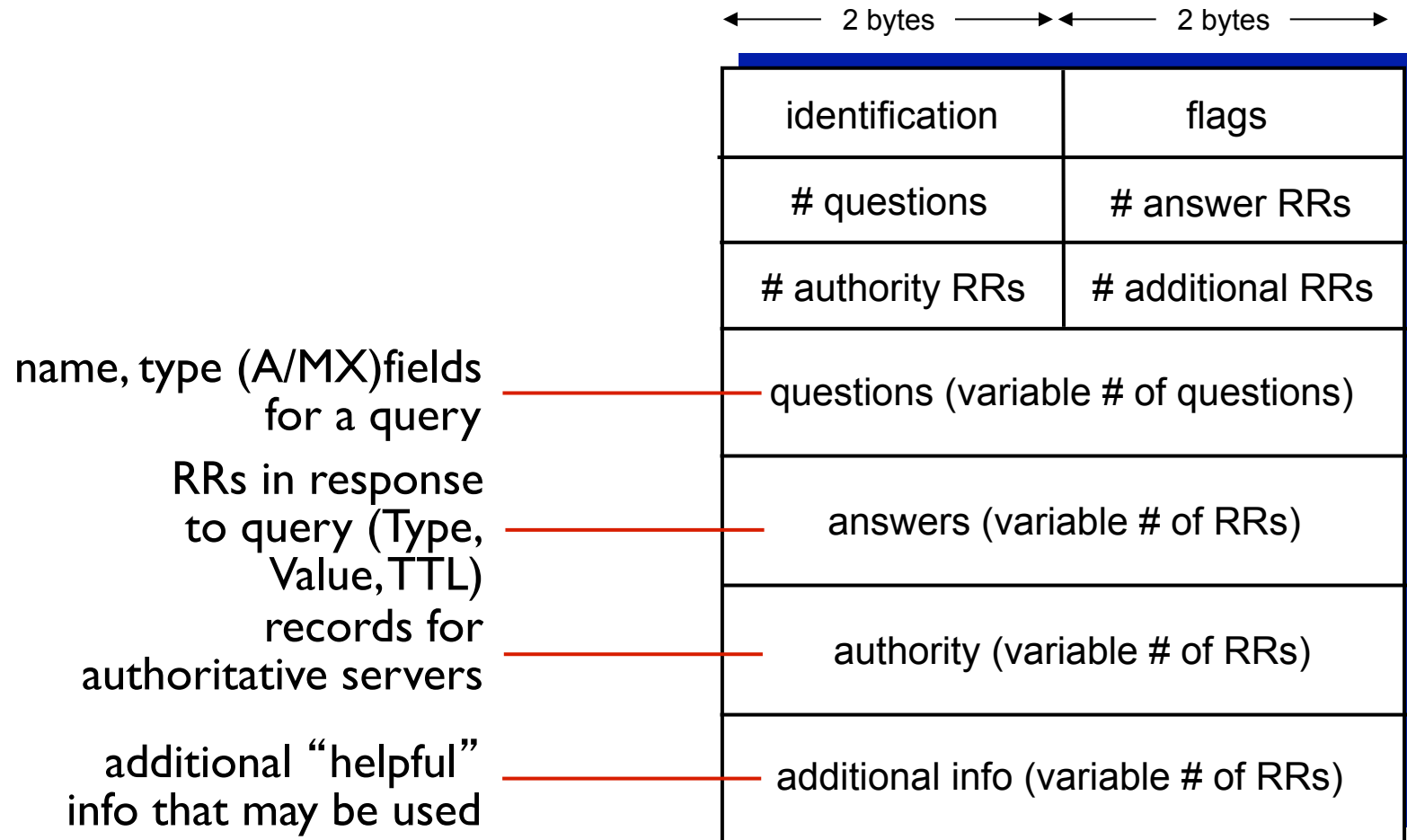
□ *query* and *reply* messages, both with same *message format*

msg 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



Inserting records into DNS

- ❑ example: new startup “Network Utopia”
- ❑ register name networkutopia.com at *DNS registrar* (e.g., Network Solutions)
 - provide names, IP addresses of authoritative name 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)
- ❑ create authoritative server type A record for www.networkutopia.com; type MX record for networkutopia.com

Attacking DNS

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
- ❖ DNS poisoning
 - Send bogus replies to DNS server, which caches

Exploit DNS for DDoS

- ❖ Send queries with spoofed source address: target IP
- ❖ Requires amplification

Perche' UDP?

- ❑ Less overhead
 - ❑ Messaggi corti
 - ❑ Tempo per set-up connessione di TCP lungo
 - ❑ Un unico messaggio deve essere scambiato tra una coppia di server (nella risoluzione contattati diversi server—se si usasse TCP ogni volta dovremmo mettere su la connessione!!)
- ❑ Se un messaggio non ha risposta entro un timeout?
 - ❑ Semplicemente viene riinviato dal resolver (problema risolto dallo strato applicativo)

Porta usata per il DNS: 53!!

Chapter 2: outline

2.1 principles of network applications

- app architectures
- app requirements

2.2 Web and HTTP

2.3 FTP

2.4 electronic mail

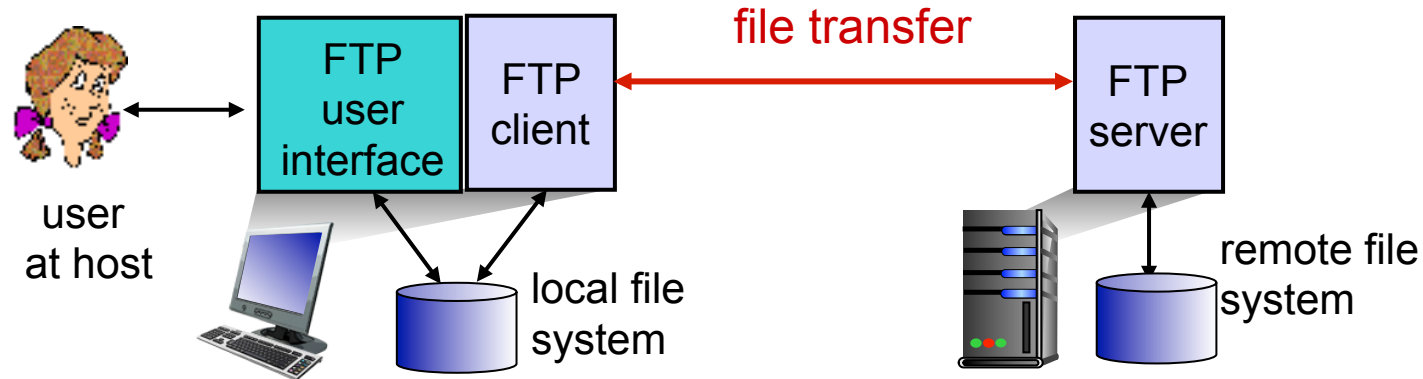
- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 socket programming with UDP and TCP

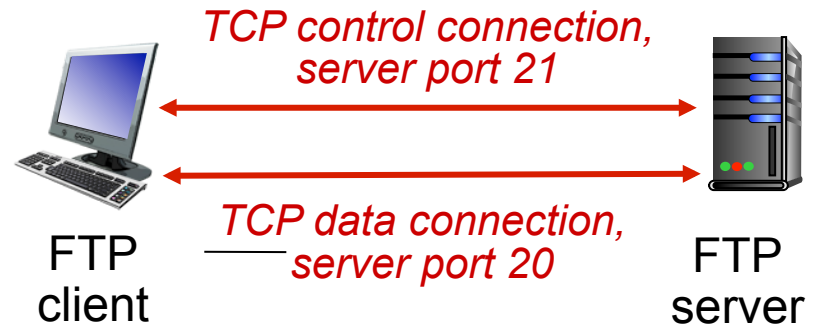
FTP: the file transfer protocol



- ❖ transfer file to/from remote host
- ❖ client/server model
 - *client*: side that initiates transfer (either to/from remote)
 - *server*: remote host
- ❖ ftp: RFC 959
- ❖ ftp server: port 21

FTP: separate control, data connections

- ❑ FTP client contacts FTP server at port 21, using TCP
- ❑ client authorized over control connection
- ❑ client browses remote directory, sends commands over control connection
- ❑ when server receives file transfer command, *server* opens 2nd TCP data connection (for file) to client
- ❑ after transferring one file, server closes data connection



- ❖ server opens another TCP data connection to transfer another file
- ❖ control connection: *“out of band”*
- ❖ FTP server maintains *“state”*: current directory, earlier authentication

FTP commands, responses

sample commands:

- ❑ sent as ASCII text over control channel
- ❑ **USER *username***
- ❑ **PASS *password***
- ❑ **LIST** return list of file in current directory
- ❑ **RETR *filename*** retrieves (gets) file
- ❑ **STOR *filename*** stores (puts) file onto remote host

sample return codes

- ❑ status code and phrase (as in HTTP)
- ❑ **331 Username OK, password required**
- ❑ **125 data connection already open; transfer starting**
- ❑ **425 Can't open data connection**
- ❑ **452 Error writing file**

Chapter 2: outline

2.1 principles of network applications

- app architectures
- app requirements

2.2 Web and HTTP

2.3 FTP

2.4 electronic mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 socket programming with UDP and TCP

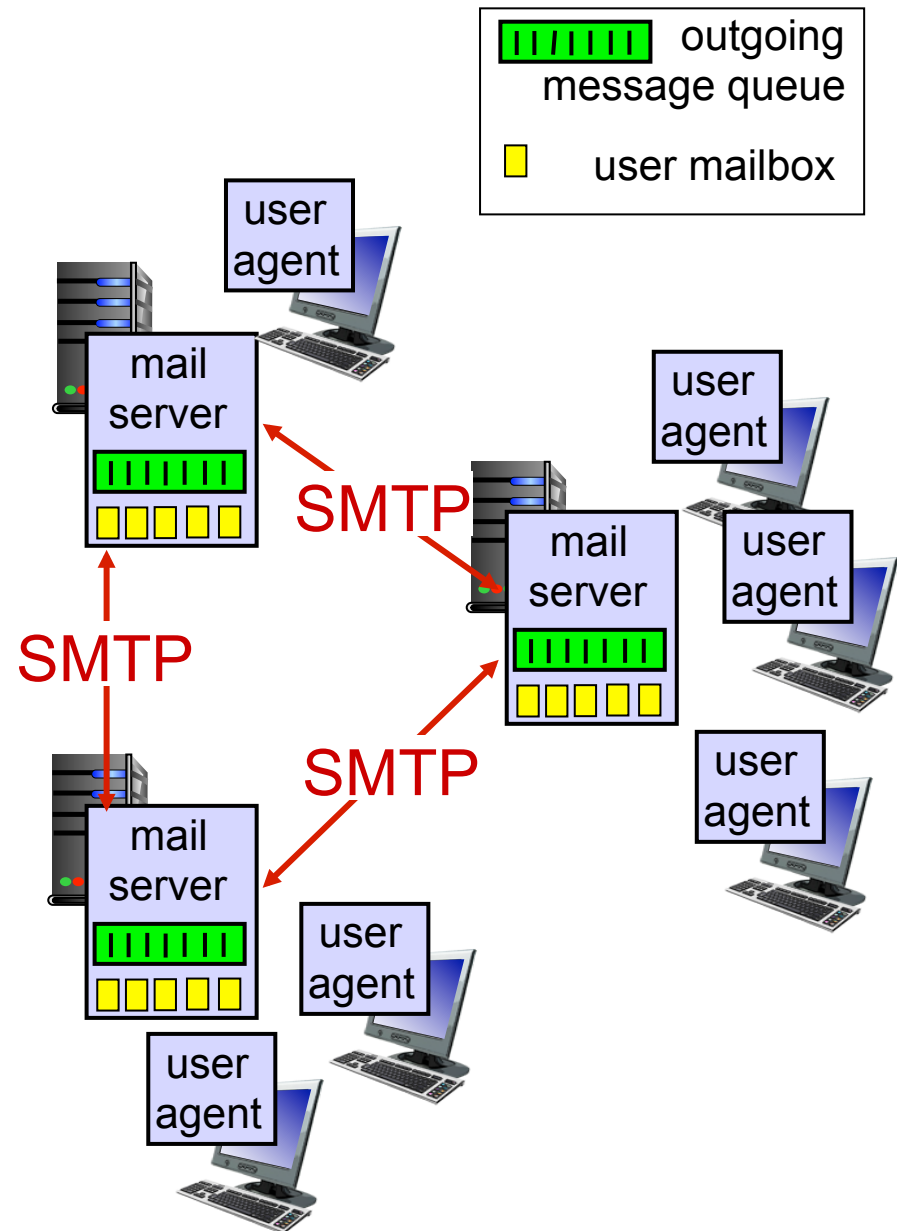
Electronic mail

Three major components:

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

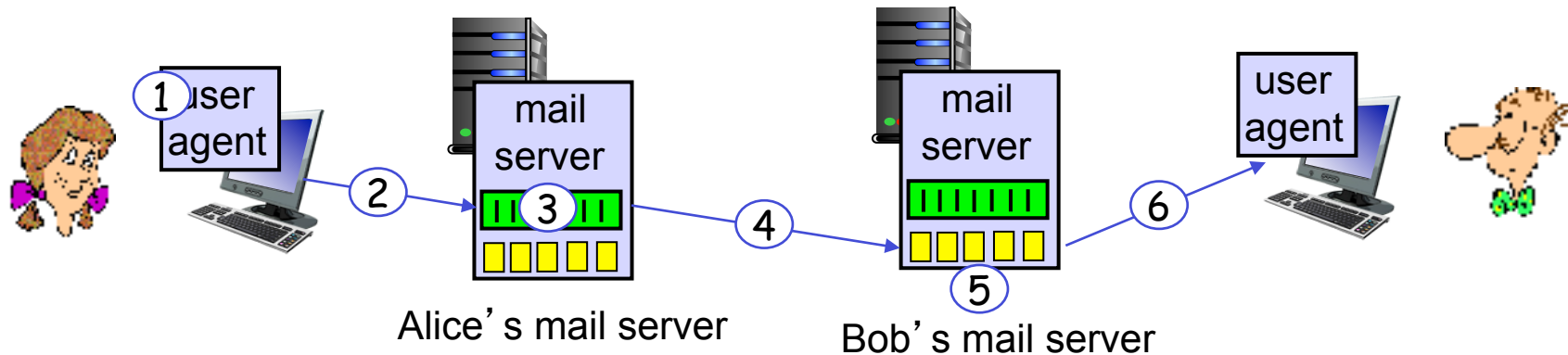
User Agent

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



Scenario: Alice sends message to Bob

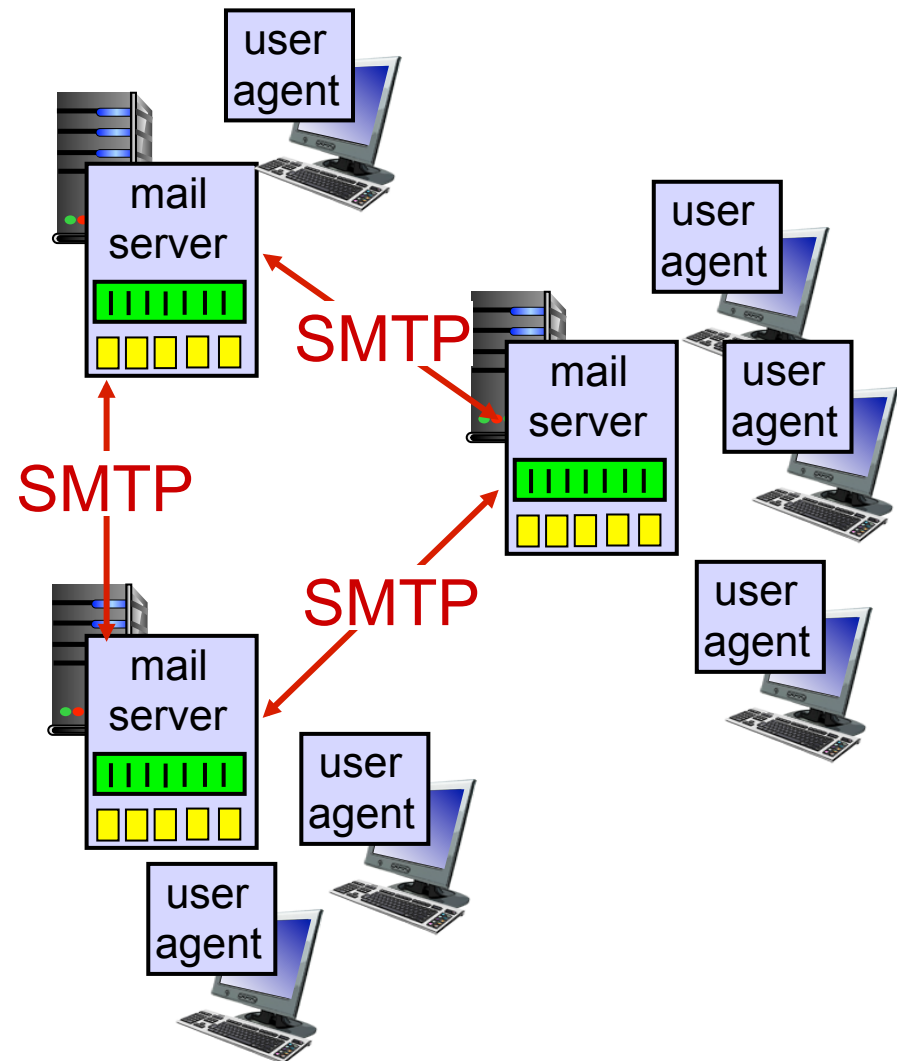
- 1) Alice uses UA to compose message "to" bob@someschool.edu
- 2) Alice's UA sends message to her mail server; message placed in message queue
- 3) client side of SMTP opens TCP connection with Bob's mail server
- 4) SMTP client sends Alice's message over the TCP connection
- 5) Bob's mail server places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



Electronic 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



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, FTP)
 - **commands:** ASCII text
 - **response:** status code and phrase
- ❑ messages must be in 7-bit ASCII

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: final words

- ❑ 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

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

Mail message format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

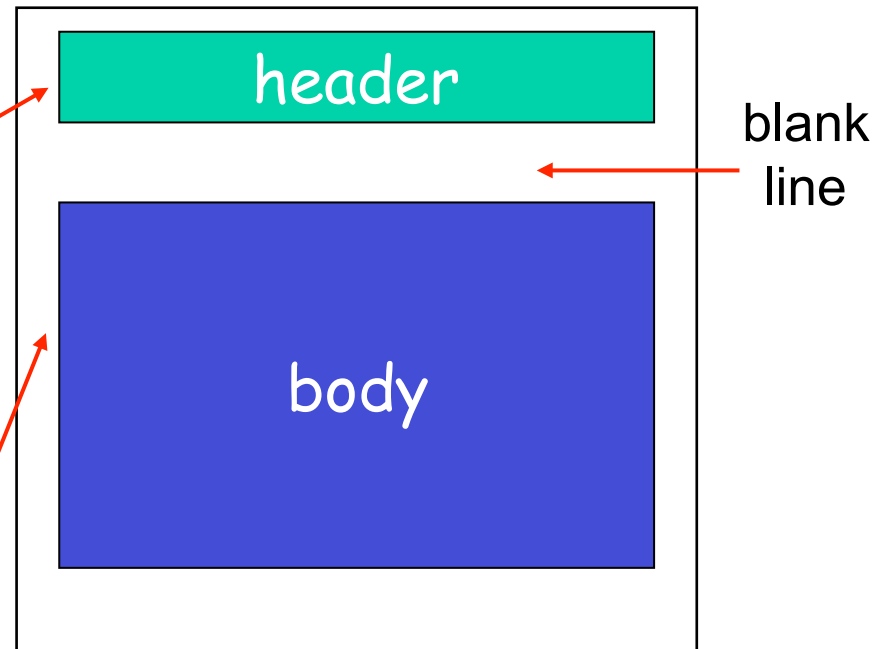
□ header lines, e.g.,

- To:
- From:
- Subject:

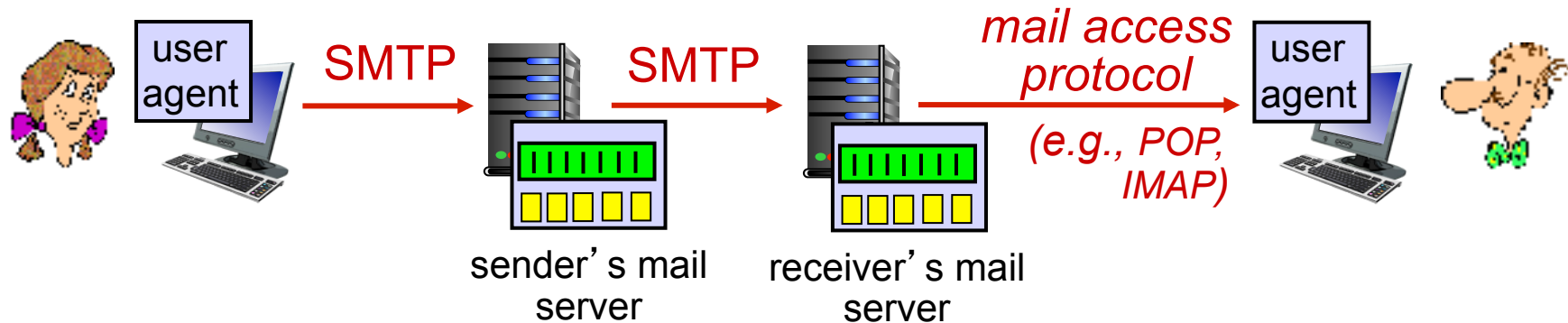
different from SMTP MAIL FROM, RCPT TO: commands!

□ 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 msgs 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**

```
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
```

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

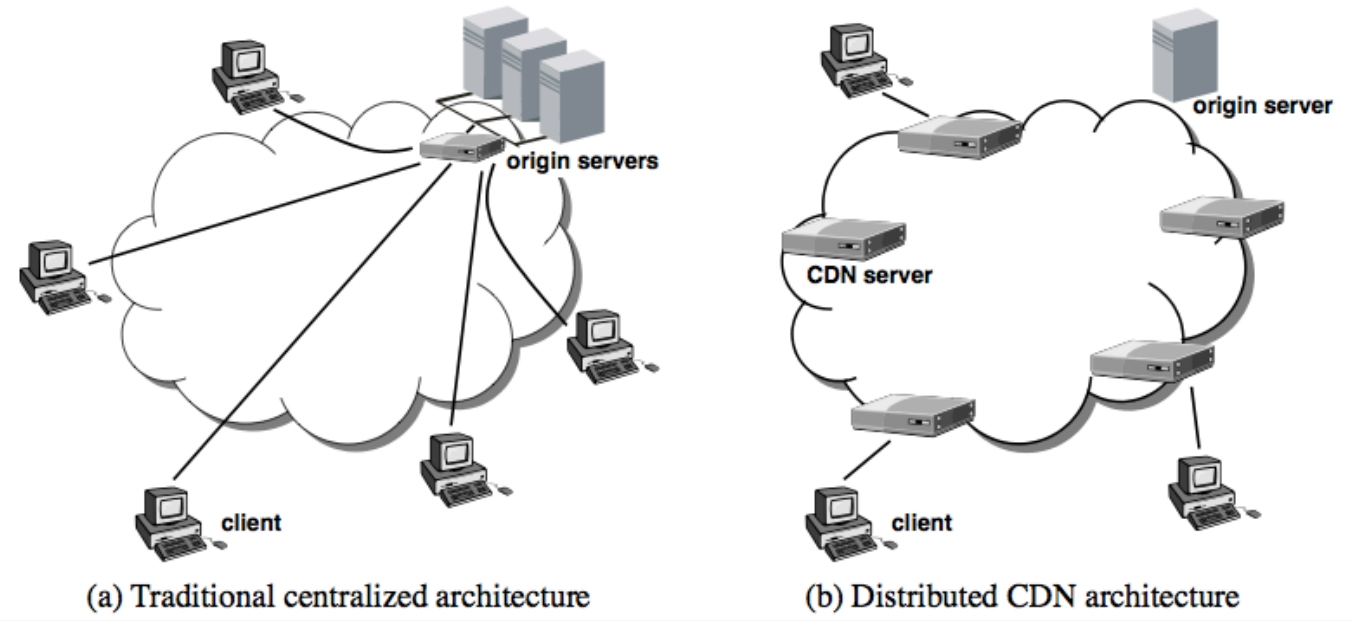
IMAP

- ❑ 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

Content Delivery Networks

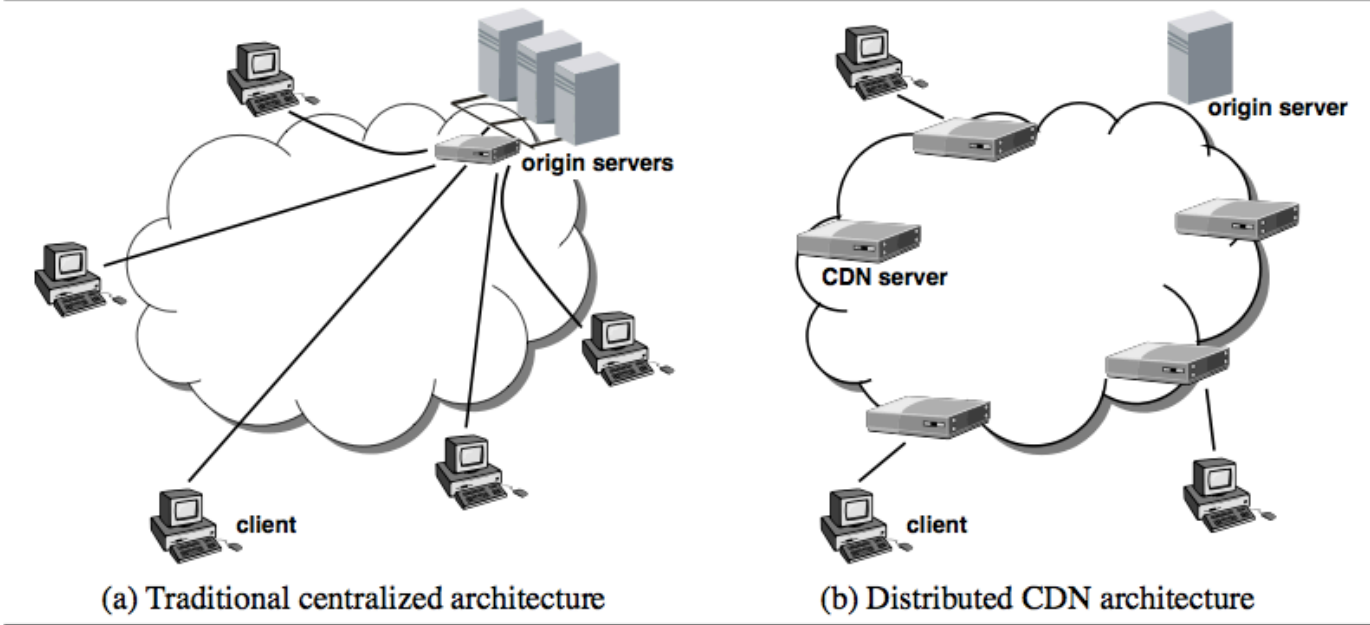
- ❑ We have seen the extensive use of caching for reducing latencies in resolving names and accessing web content
- ❑ Is this enough?
 - Origin servers may still have to be accessed to maintain consistency
- ❑ Caching
 - What to cache
 - How to maintain consistency
 - How to invalidate or update in case an inconsistency is detected
- ❑ More here:<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.586&rep=rep1&type=pdf>

Content Delivery Networks



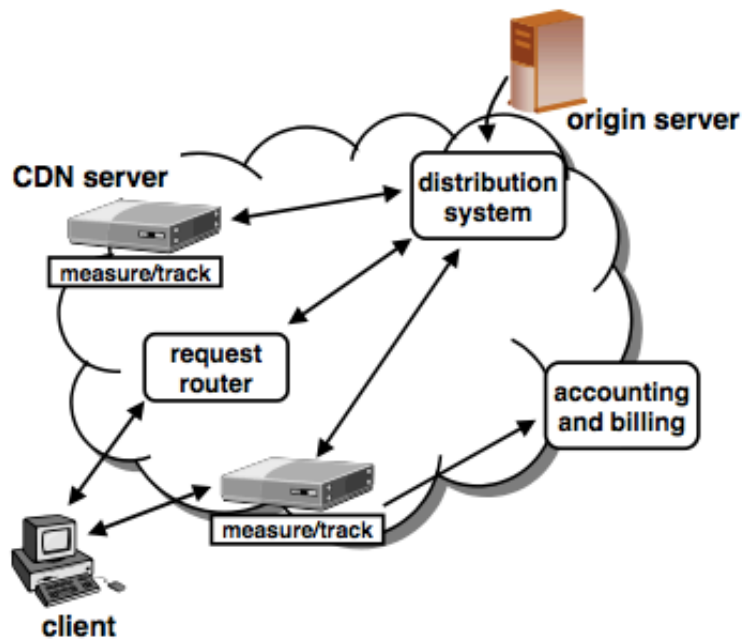
- improving client-perceived response time by bringing content closer to the network edge, and thus closer to end-users
- off-loading work from origin servers by serving larger objects, such as images and multimedia, from multiple CDN servers
- reducing content provider costs by reducing the need to invest in more powerful servers or more bandwidth as user population increases

Content Delivery Networks

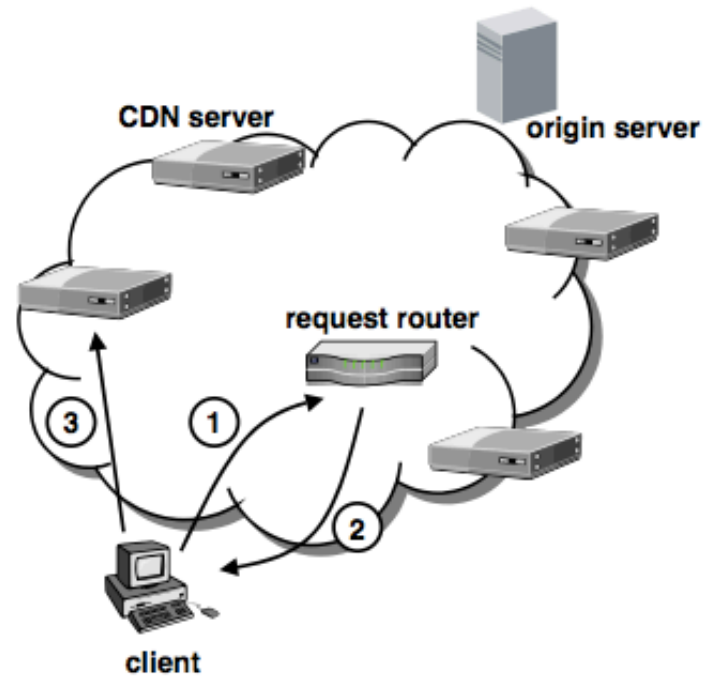


- improving site availability by replicating content in many distributed locations

Content Delivery Networks

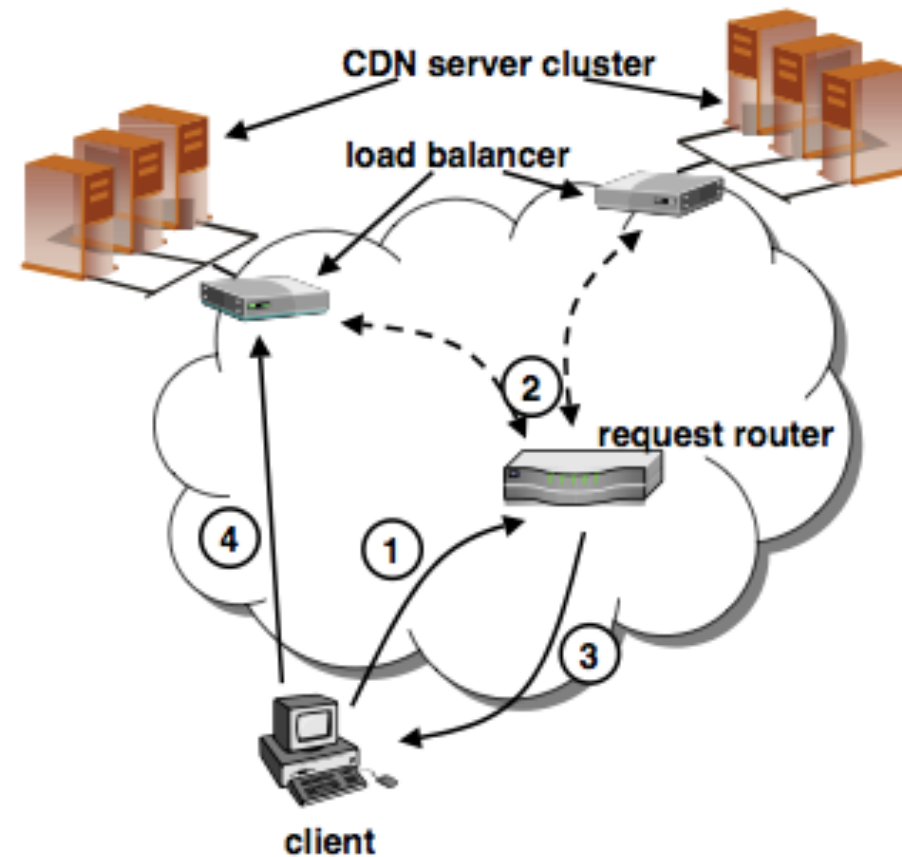


(a) CDN architectural elements



(b) CDN request-routing

Content Delivery Networks



- ❑ HTTP Redirect
- ❑ DNS Redirect