Application Layer: E-mail, DNS

EECS 3214

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

22-Jan-18

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

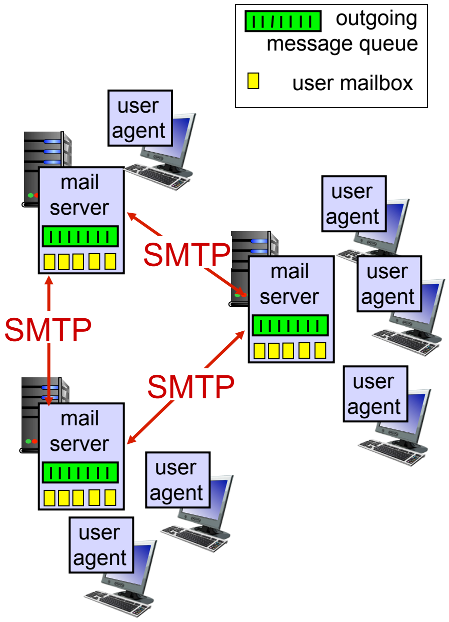
2.3 Electronic mail

Three major components:

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

User Agent

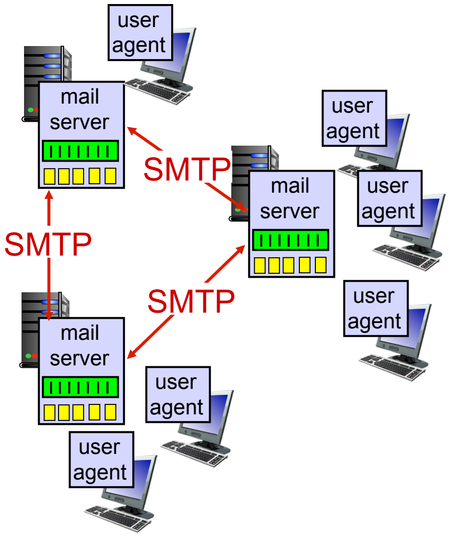
- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Outlook, Apple Mail, iPhone mail client
- outgoing, incoming messages stored on server



Electronic mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages
- *SMTP protocol* between mail servers to send email messages
  - client: sending mail server
  - “server”: receiving mail server



2.3.1 Electronic Mail: SMTP [RFC 5321]

- 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 ASCII

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message "to" `bob@someschool.edu`

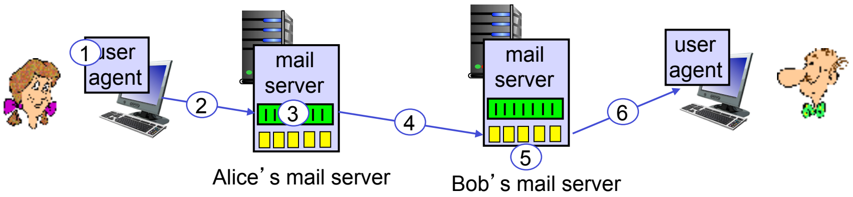
2) Alice' s UA sends message to her mail server; message placed in message queue

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



Sample SMTP interaction

blue text:
handshaking

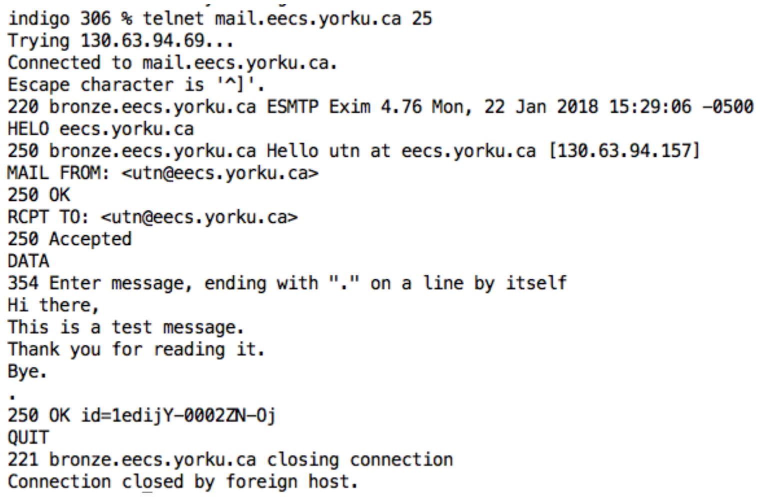
```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?\r\n
C: .\r\n
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Try SMTP interaction for yourself

- `telnet servername 25`
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

SMTP Interaction: Example



```
indigo 306 % telnet mail.eecs.yorku.ca 25
Trying 130.63.94.69...
Connected to mail.eecs.yorku.ca.
Escape character is '^]'.
220 bronze.eecs.yorku.ca ESMTP Exim 4.76 Mon, 22 Jan 2018 15:29:06 -0500
HELO eecs.yorku.ca
250 bronze.eecs.yorku.ca Hello utn at eecs.yorku.ca [130.63.94.157]
MAIL FROM: <utn@eecs.yorku.ca>
250 OK
RCPT TO: <utn@eecs.yorku.ca>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
Hi there,
This is a test message.
Thank you for reading it.
Bye.
.
250 OK id=1edijY-0002ZN-0j
QUIT
221 bronze.eecs.yorku.ca closing connection
Connection closed by foreign host.
```

More on SMTP

- SMTP uses persistent connections
- SMTP requires message (header and body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message

Comparison with HTTP (2.3.2)

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- SMTP: message in 7-bit ASCII
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message

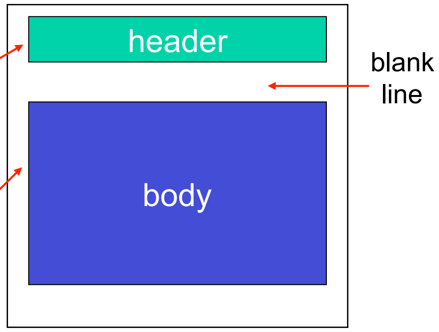
2.3.3 Mail message format

SMTP: protocol for exchanging email messages

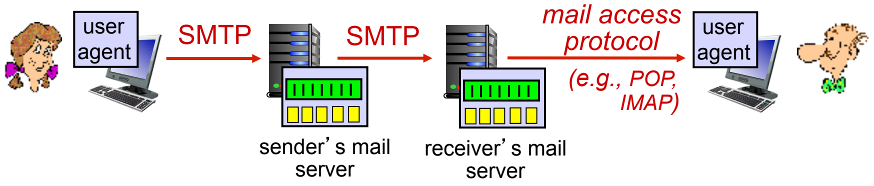
RFC 5322: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

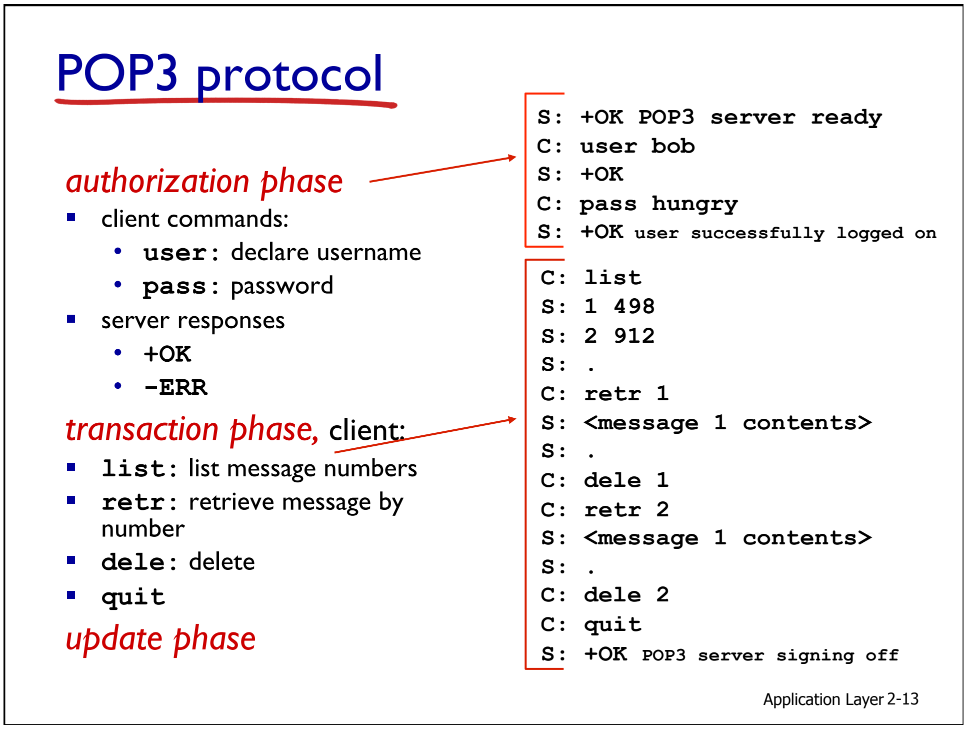
  different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the "message"
  - ASCII characters only



2.3.4 Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization and 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

authorization phase

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

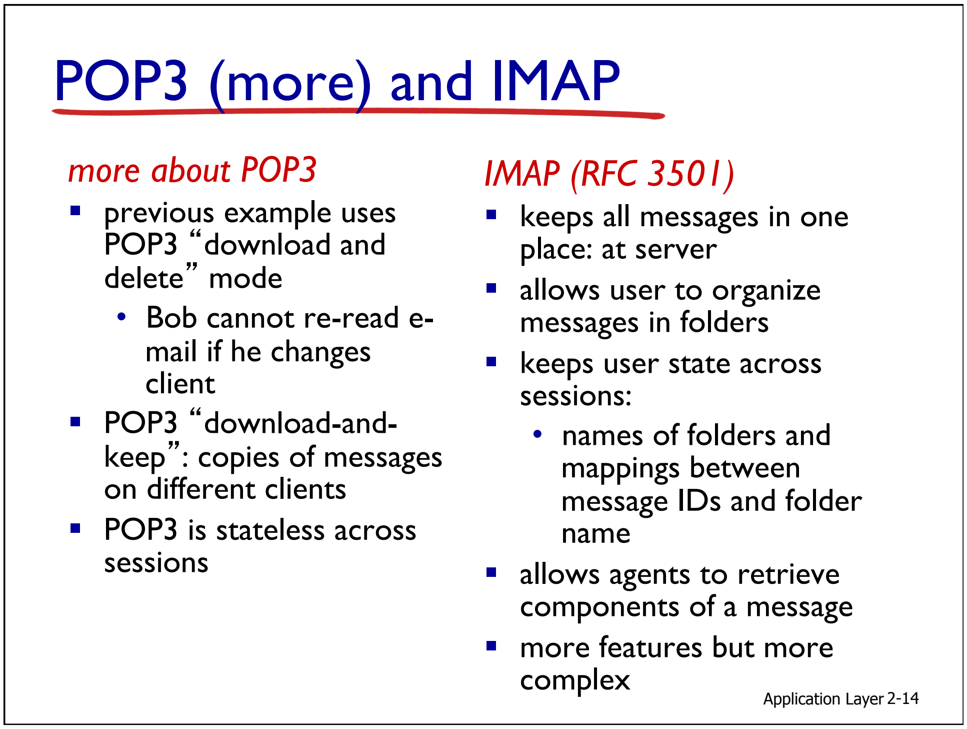
transaction phase, client:

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

update phase

```
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

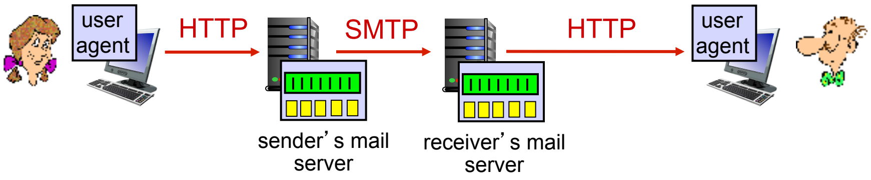
more about POP3

- 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 (RFC 3501)

- 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
- allows agents to retrieve components of a message
- more features but more complex

Web-based E-mail



- sender’s agent to sender’s server: HTTP
- sender’s server to receiver's server: SMTP
- receiver's server to receiver's agent: HTTP

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:

- IP address (32 bit) - used for addressing datagrams
- "name" or URL, 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 *DNS 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

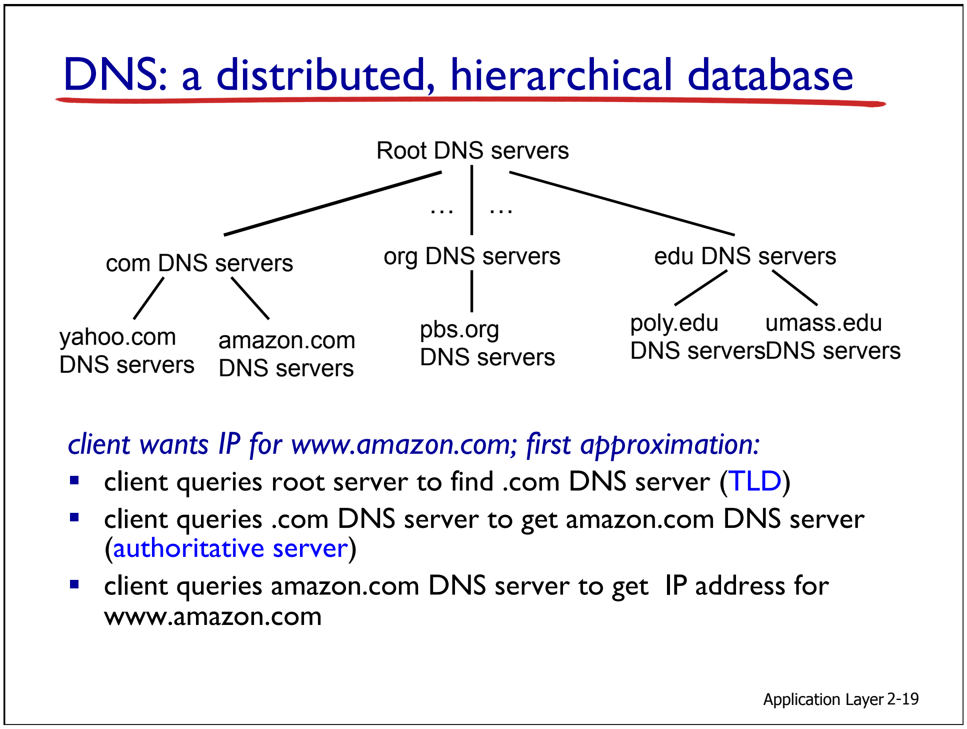
- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name
  - DNS server sends entire list, rotating the ordering

why not centralize DNS?

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

A: doesn't scale!

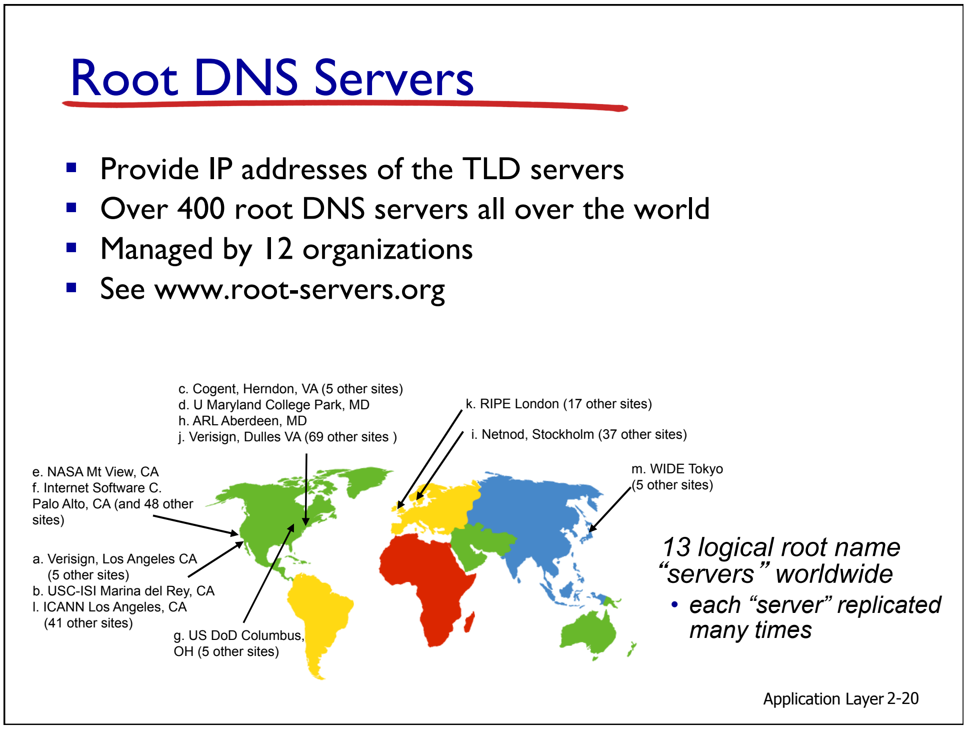
- HTTP client uses the first address on the list



DNS: a distributed, hierarchical database

client wants IP for www.amazon.com; first approximation:

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



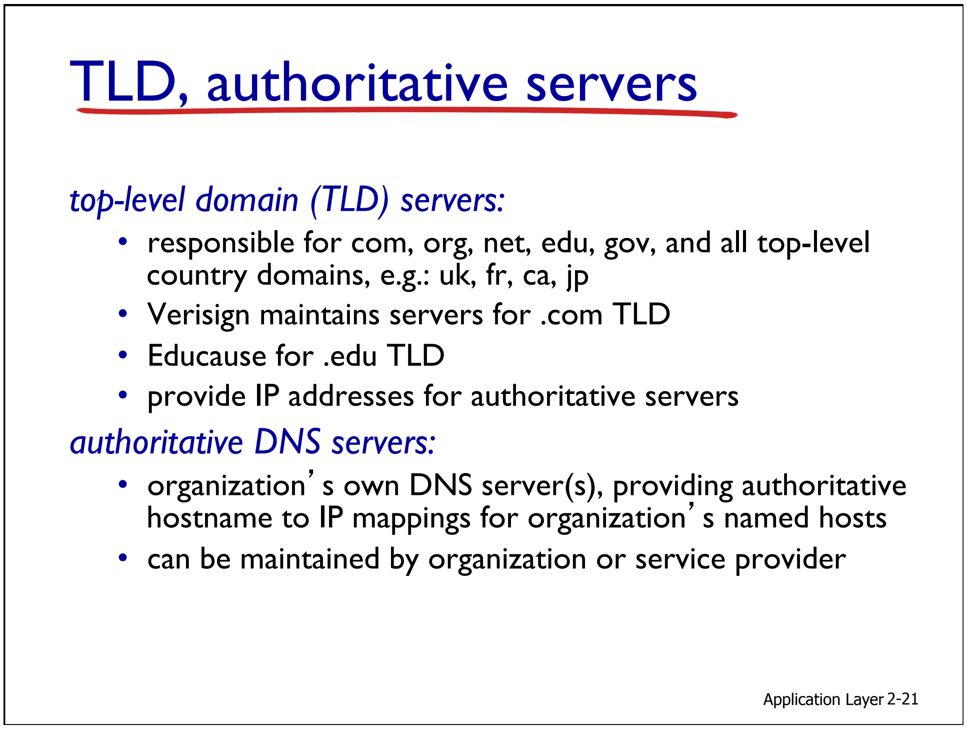
Root DNS Servers

- Provide IP addresses of the TLD servers
- Over 400 root DNS servers all over the world
- Managed by 12 organizations
- See www.root-servers.org

c. Cogent, Herndon, VA (5 other sites)
d. U Maryland College Park, MD
h. ARL Aberdeen, MD
j. Verisign, Dulles VA (69 other sites)
k. RIPE London (17 other sites)
i. Netnod, Stockholm (37 other sites)
m. WIDE Tokyo (5 other sites)
e. NASA Mt View, CA
f. Internet Software C. Palo Alto, CA (and 48 other sites)
a. Verisign, Los Angeles CA (5 other sites)
b. USC-ISI Marina del Rey, CA
l. ICANN Los Angeles, CA (41 other sites)
g. US DoD Columbus, OH (5 other sites)

13 logical root name "servers" worldwide

- *each "server" replicated many times*



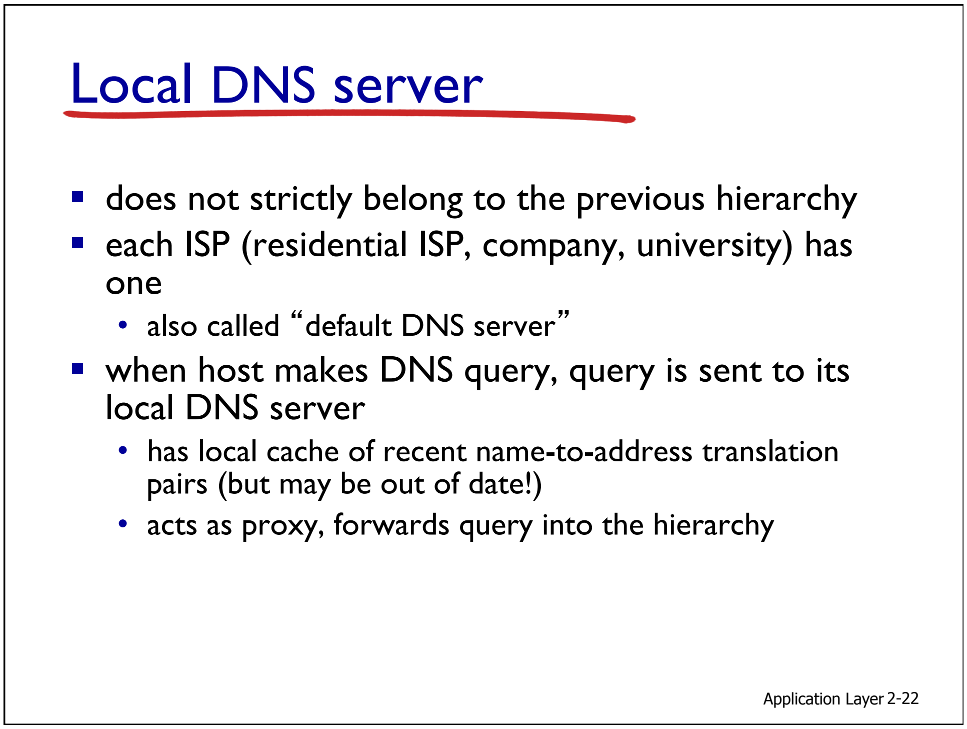
TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, gov, and all top-level country domains, e.g.: uk, fr, ca, jp
- Verisign maintains servers for .com TLD
- Educause for .edu TLD
- provide IP addresses for authoritative servers

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

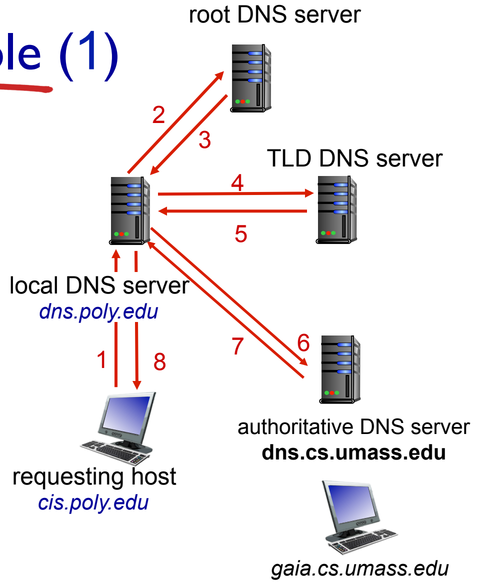
- does not strictly belong to the previous hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default DNS server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)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

DNS name resolution example (1)

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

iterated query:

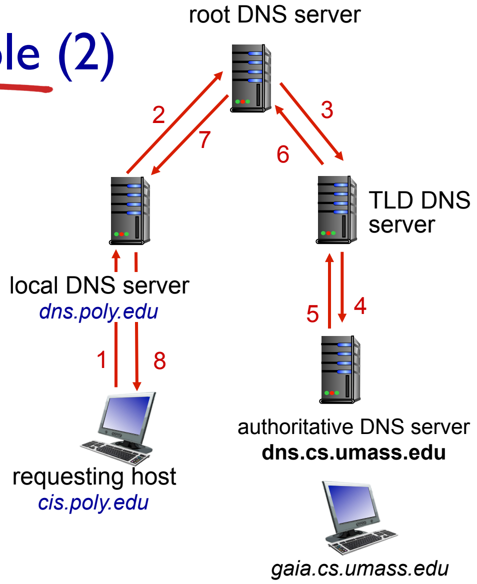
- contacted server replies with name and IP address of next server to contact
- “I don’ t know this name, but ask this server”

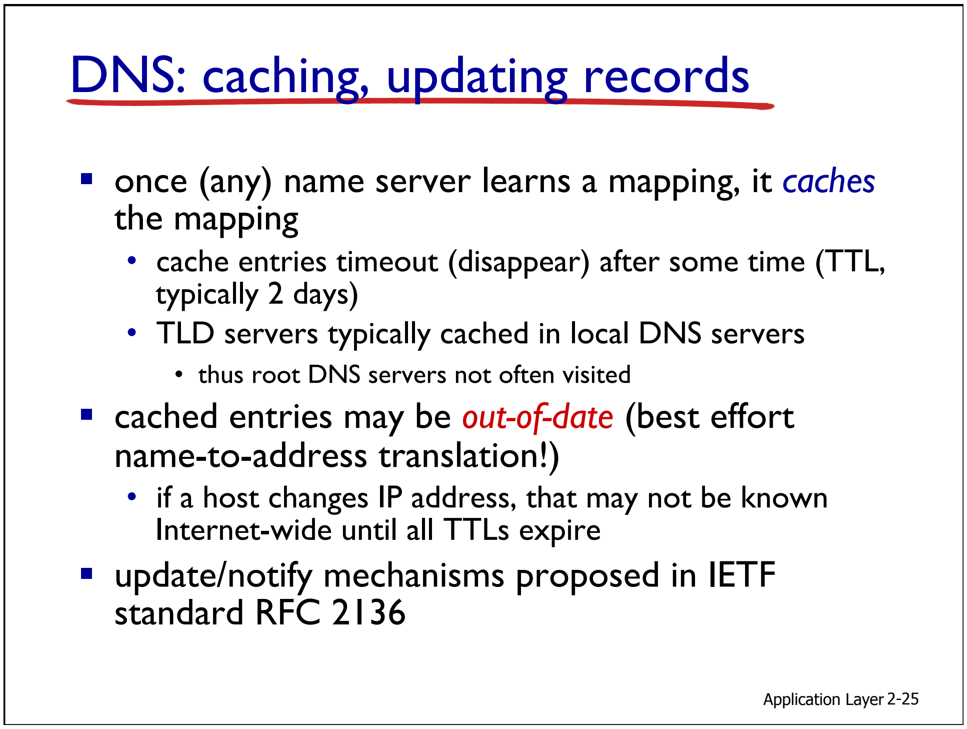


DNS name resolution example (2)

recursive query:

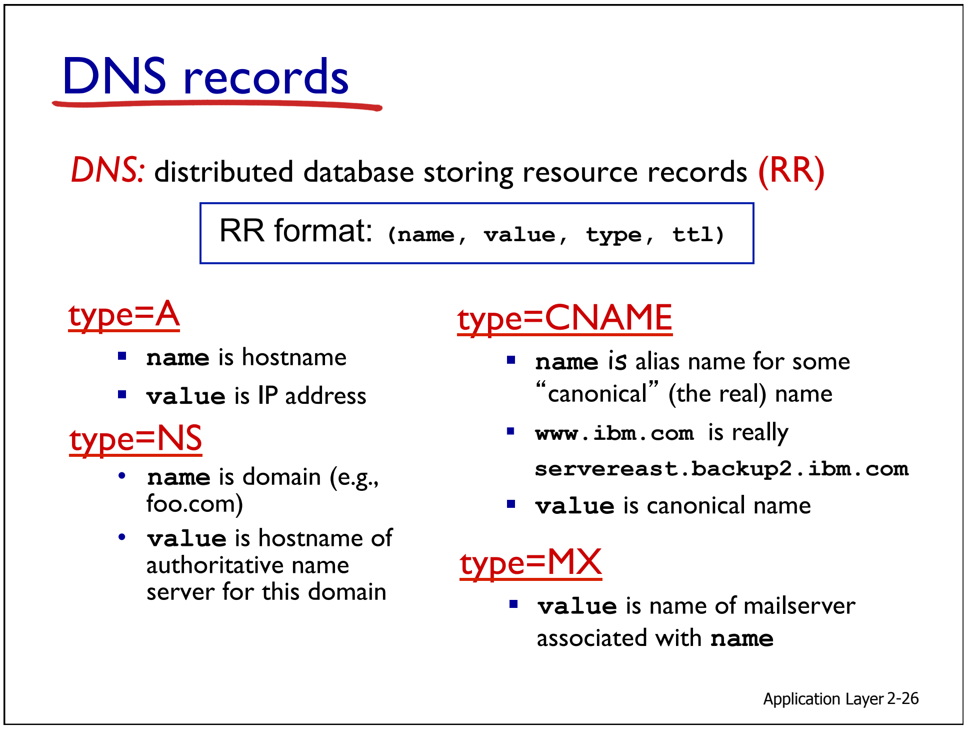
- puts burden of name resolution on contacted DNS server
- heavy load at upper levels of hierarchy?





DNS: caching, updating records

- once (any) name server learns a mapping, it *caches* the mapping
  - cache entries timeout (disappear) after some time (TTL, typically 2 days)
  - TLD servers typically cached in local DNS servers
    - thus root DNS servers not often visited
- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if a host changes IP address, that may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed in 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 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

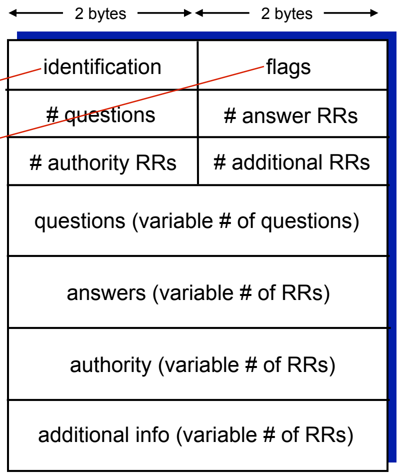
- **`value`** is name of mailserver associated with **`name`**

DNS 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
  - reply is authoritative
  - recursion desired
  - recursion available



2 bytes | 2 bytes

identification | flags

# questions | # answer RRs

# authority RRs | # additional RRs

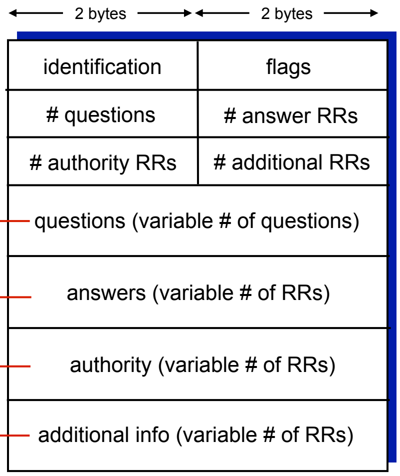
questions (variable # of questions)

answers (variable # of RRs)

authority (variable # of RRs)

additional info (variable # of RRs)

DNS messages (cont.)



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

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used

Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions or some others)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD servers:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- create authoritative server type A record for web server www.networkuptopia.com; type MX record for mail server mail.networkutopia.com

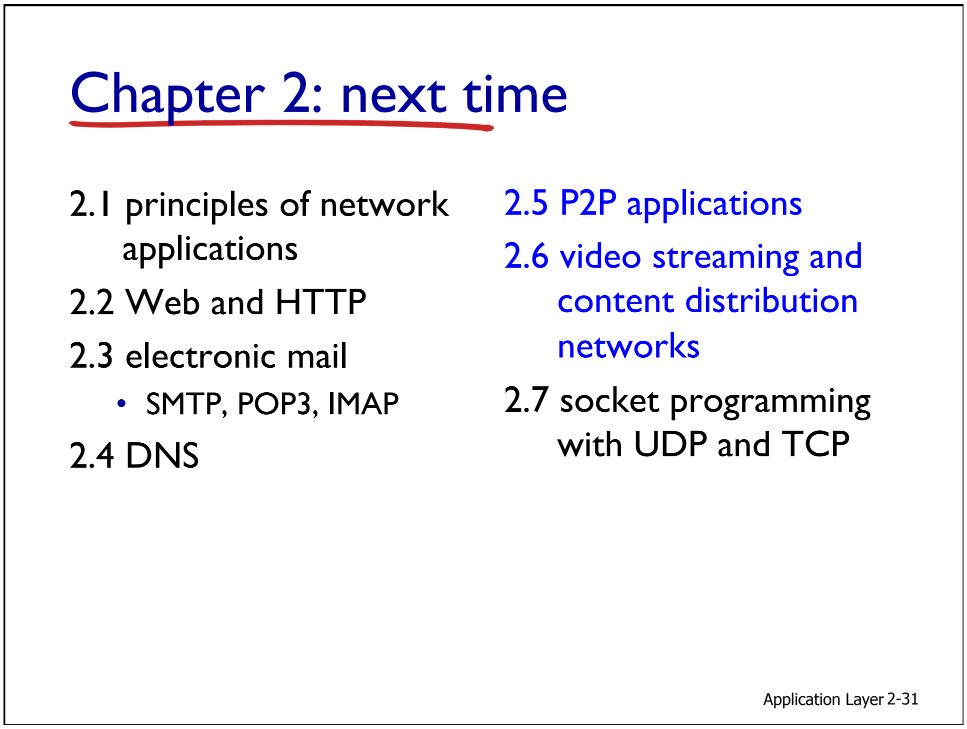
Attacking DNS

DDoS attacks

- bombard root servers with traffic
  - not successful to date
  - traffic filtering (of ICMP messages)
  - local DNS servers cache IPs of TLD servers, allowing root server to be bypassed
- bombard TLD servers
  - potentially more dangerous
  - mitigation: caching in local DNSs

Other attacks

- man-in-middle
  - intercept queries
  - send bogus replies
- DNS poisoning
  - send bogus relies to DNS server, which caches them
- difficult to implement in practice



Chapter 2: next time

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