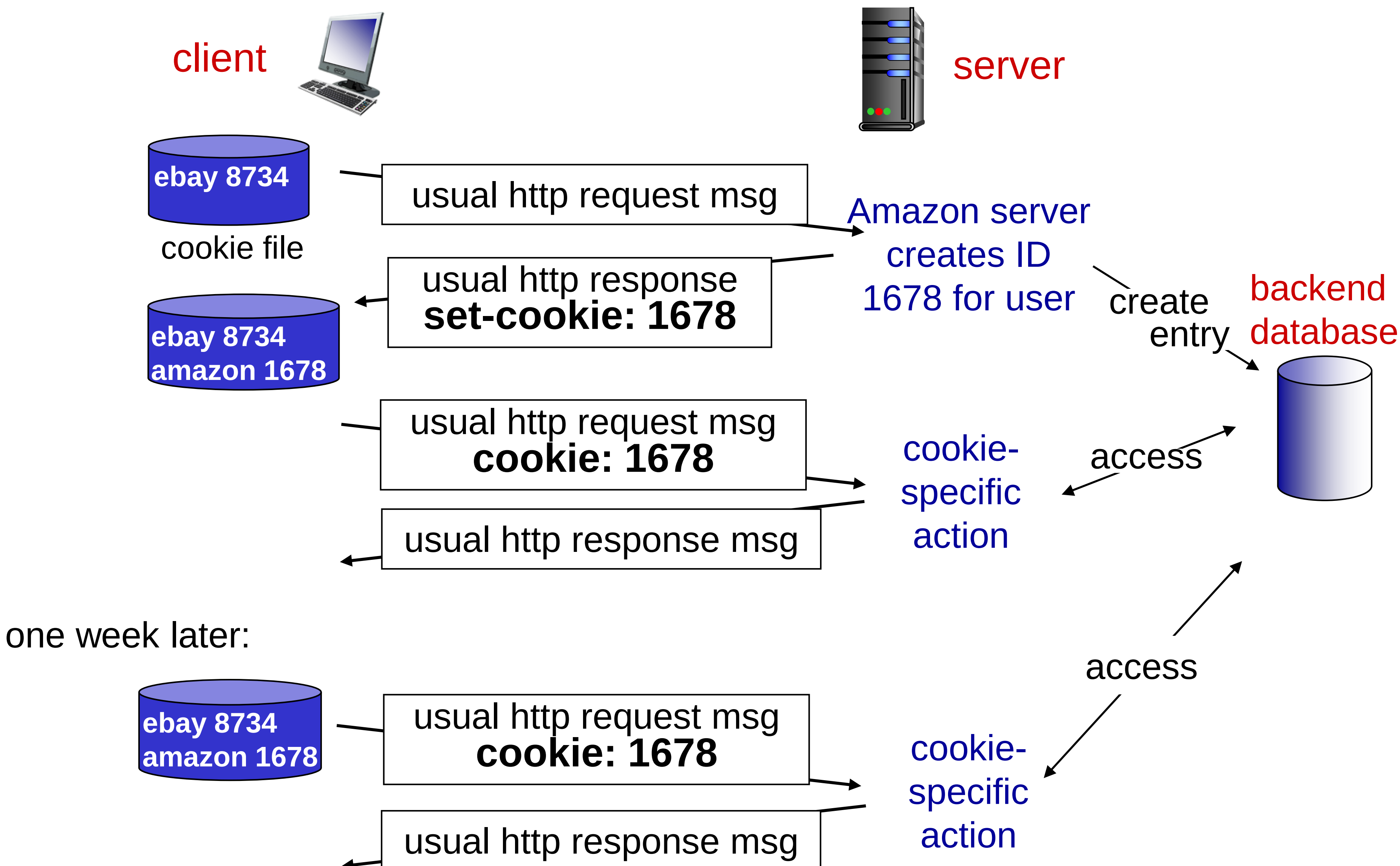
Overview of last class

Cookies: keeping "state" (cont.)

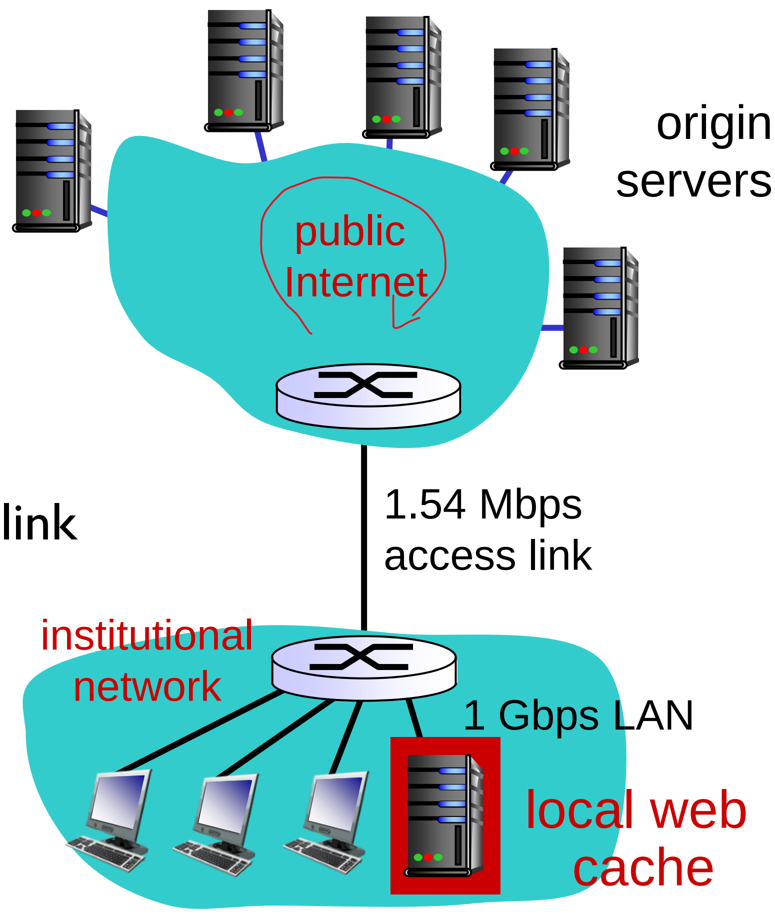


Caching example: install local cache

Calculating access link utilization, delay with cache:

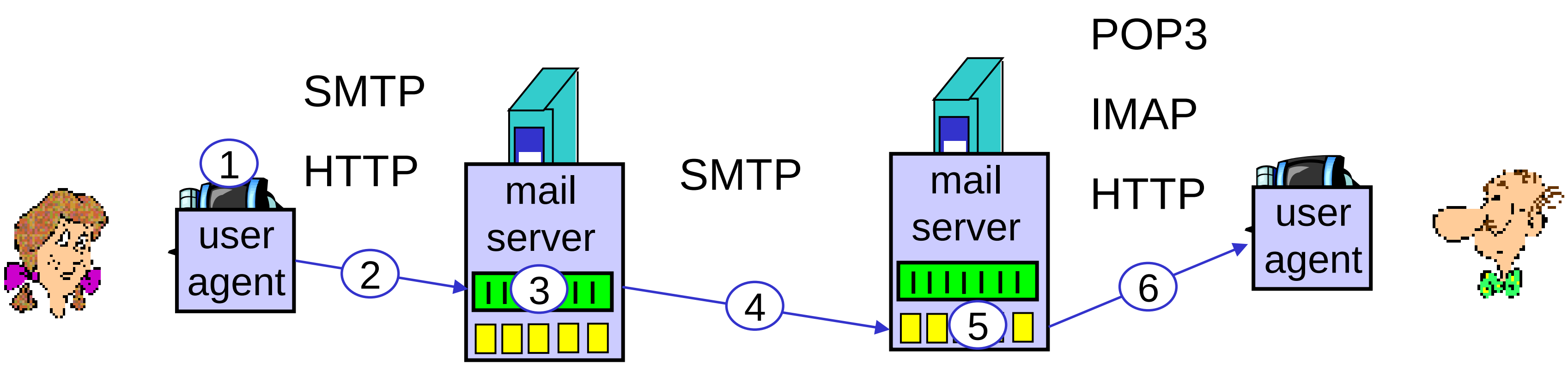
- suppose cache hit rate is 0.4
  - 40% requests satisfied at cache, 60% requests satisfied at origin
- access link utilization:
  - 60% of requests use access link
- data rate to browsers over access link = 0.6*1.50 Mbps = .9 Mbps
  - utilization = 0.9/1.54 = .58
- total delay
  - = 0.6 * (delay from origin servers) +0.4 * (delay when satisfied at cache)
  - = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs
  - less than with 154 Mbps link (and cheaper too!)

Prob.



Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "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



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
- The MIME (multipurpose Internet mail extension) extensions for non-ascii data

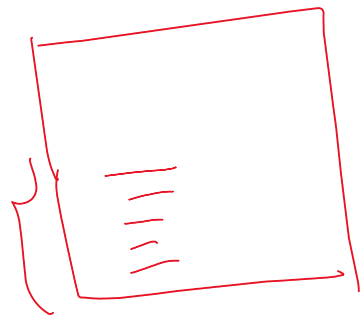
Q: why persistent mode is selected?

comparison with HTTP:

- HTTP: pull
- SMTP: 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

Alice → Bob

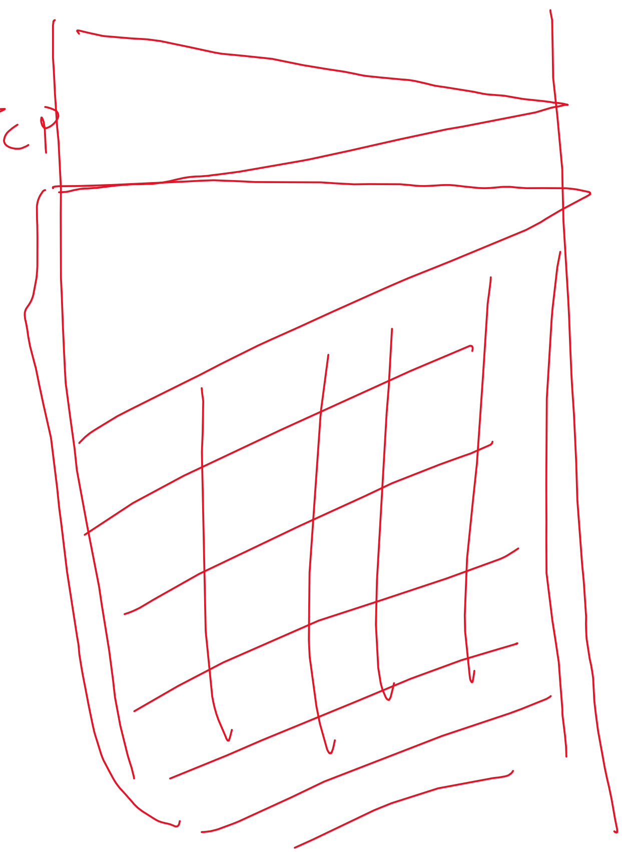
objects



each objects normally small size

- persistent : push all the objects one by one

TCP



- non-persistent
  - o non-parallel → bad choice
  - o parallel → collection management

congestion/flow control Incur cost

Class Today

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 #

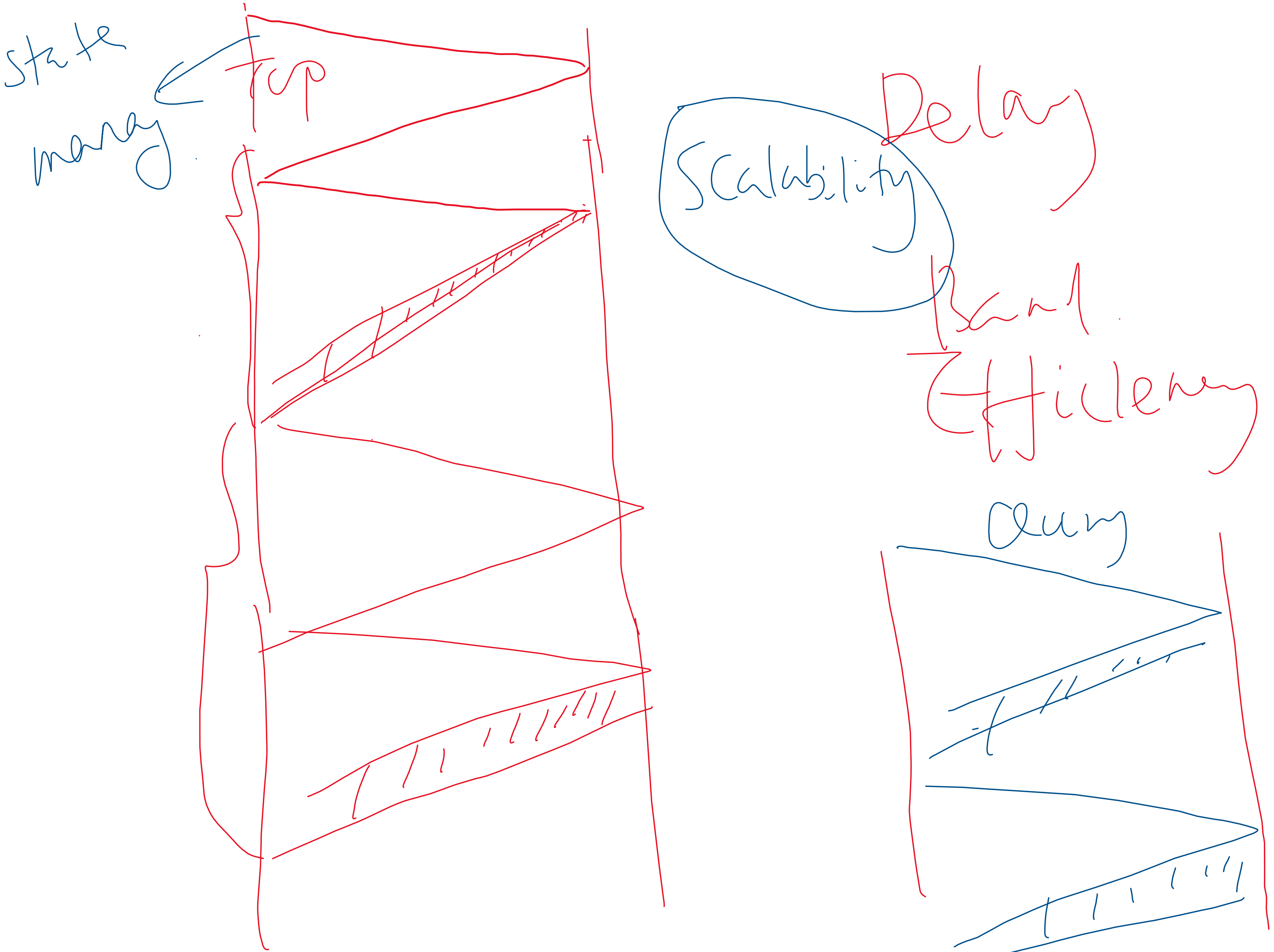
Internet hosts, routers:

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

Q: how to map between IP address and name?

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"
- *Runs over UDP: why not TCP?*


state
mang
Tcp
Scalability
Delay
Band
Efficieny

DNS: services, structure

DNS services

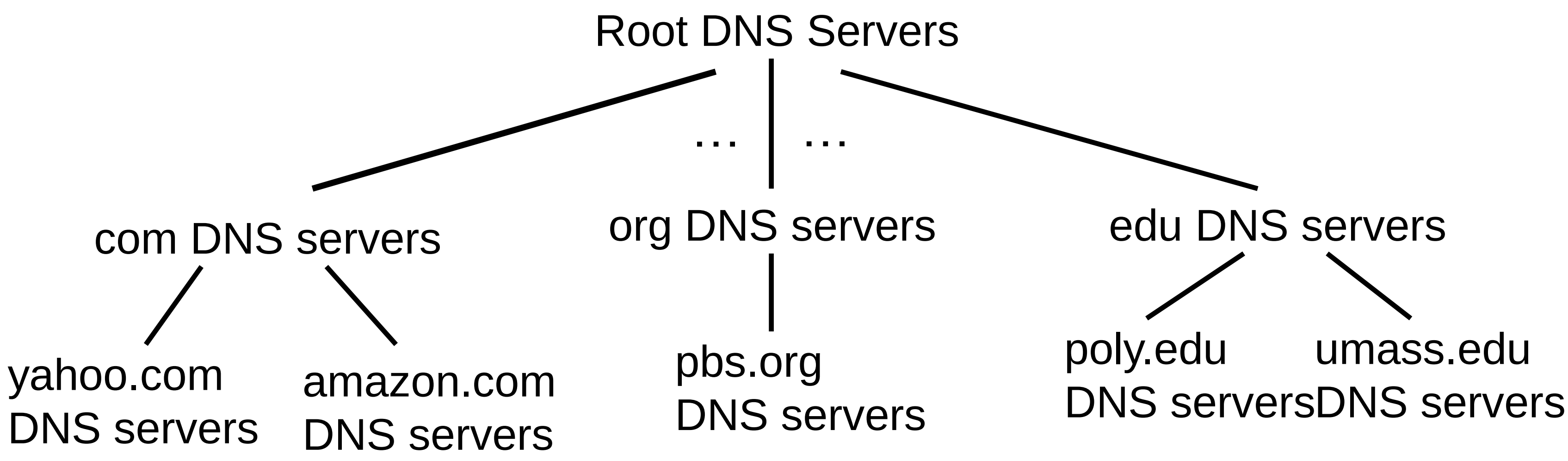
- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

why not centralize DNS?

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

A: doesn't scale!

DNS: a distributed, hierarchical database

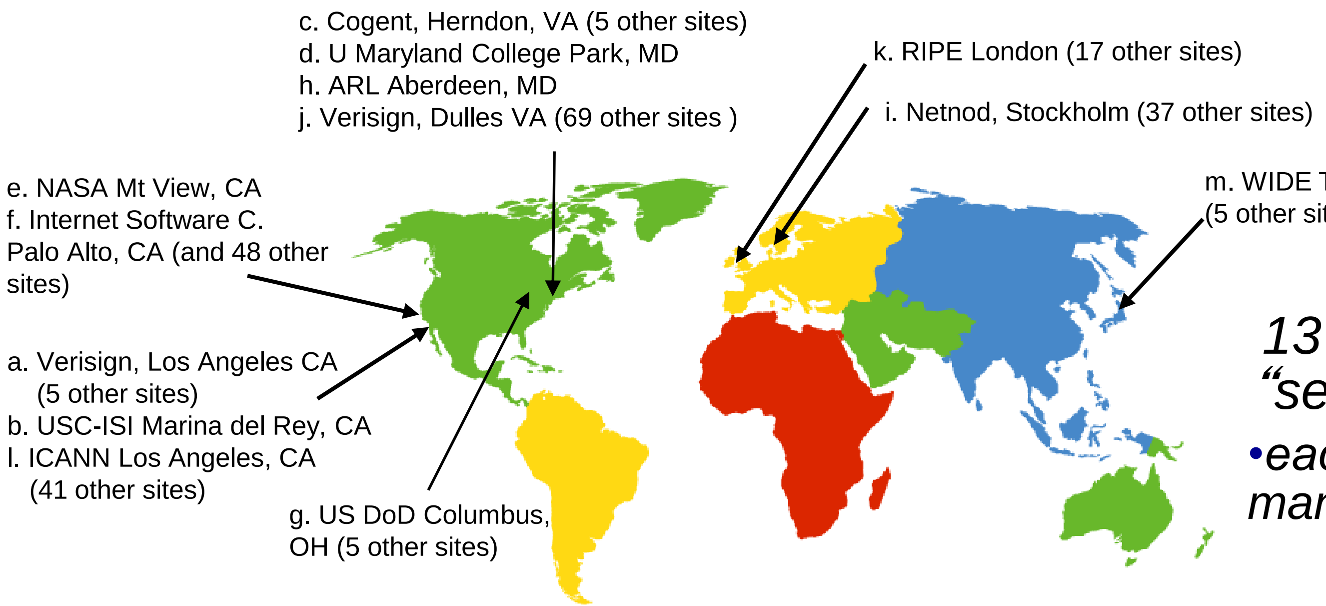


client wants IP for www.amazon.com:

- 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:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



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, aero, jobs, museums, and all top-level country domains, e.g.: 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 name server

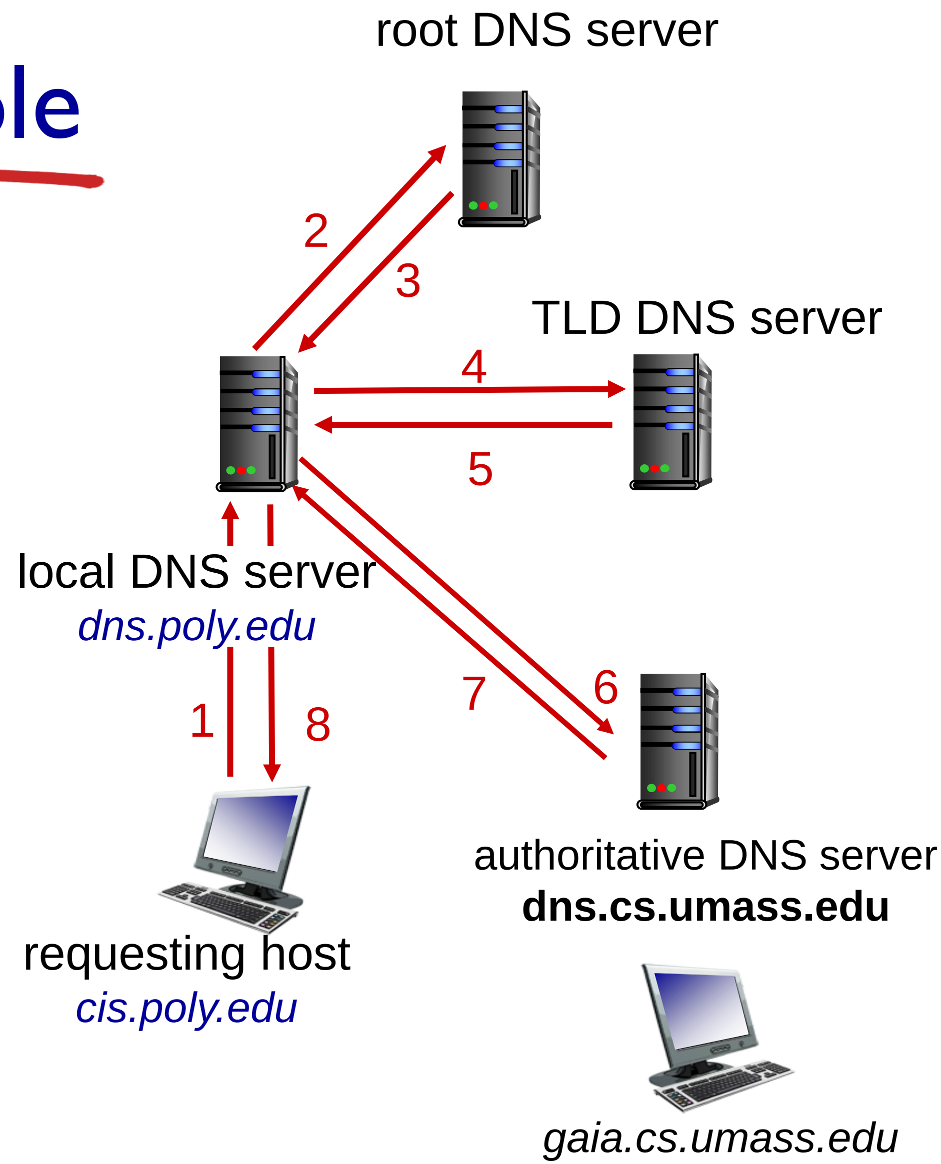
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts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

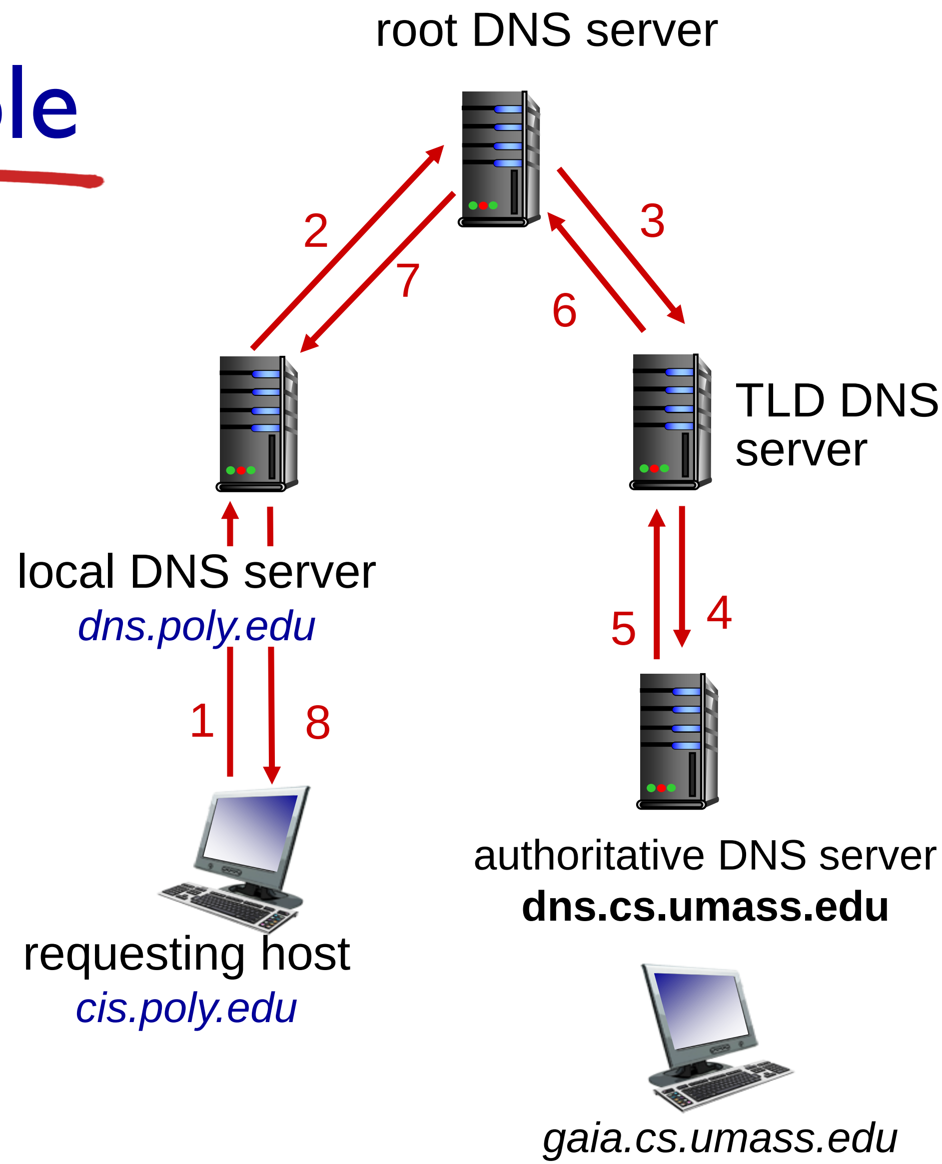
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



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 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

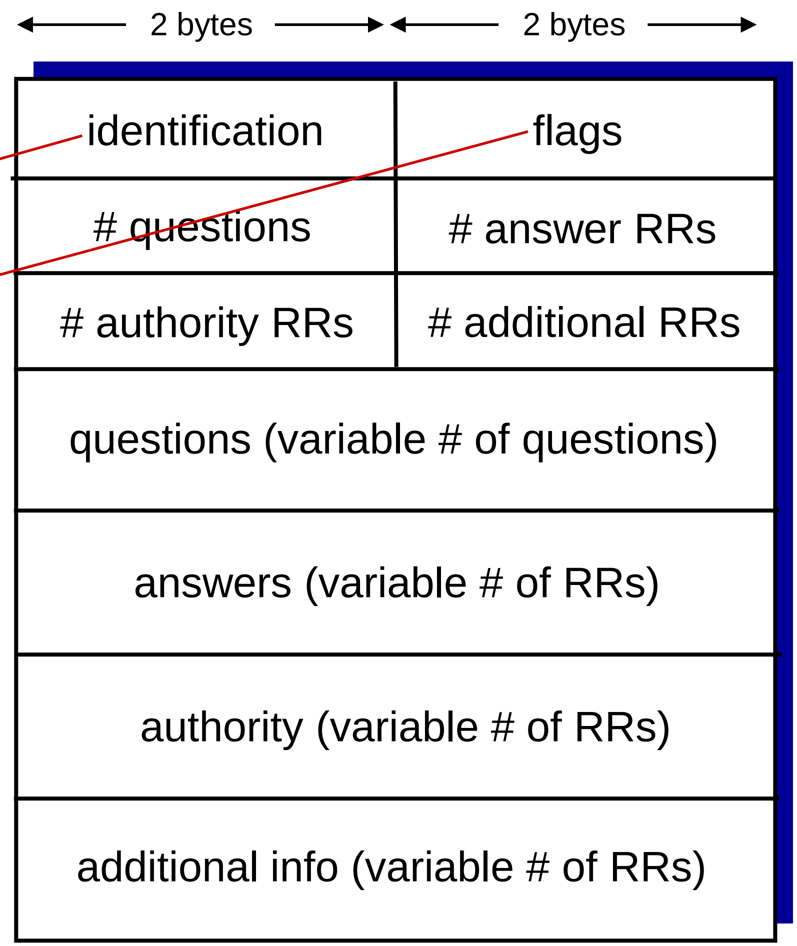
- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

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

DNS protocol, messages

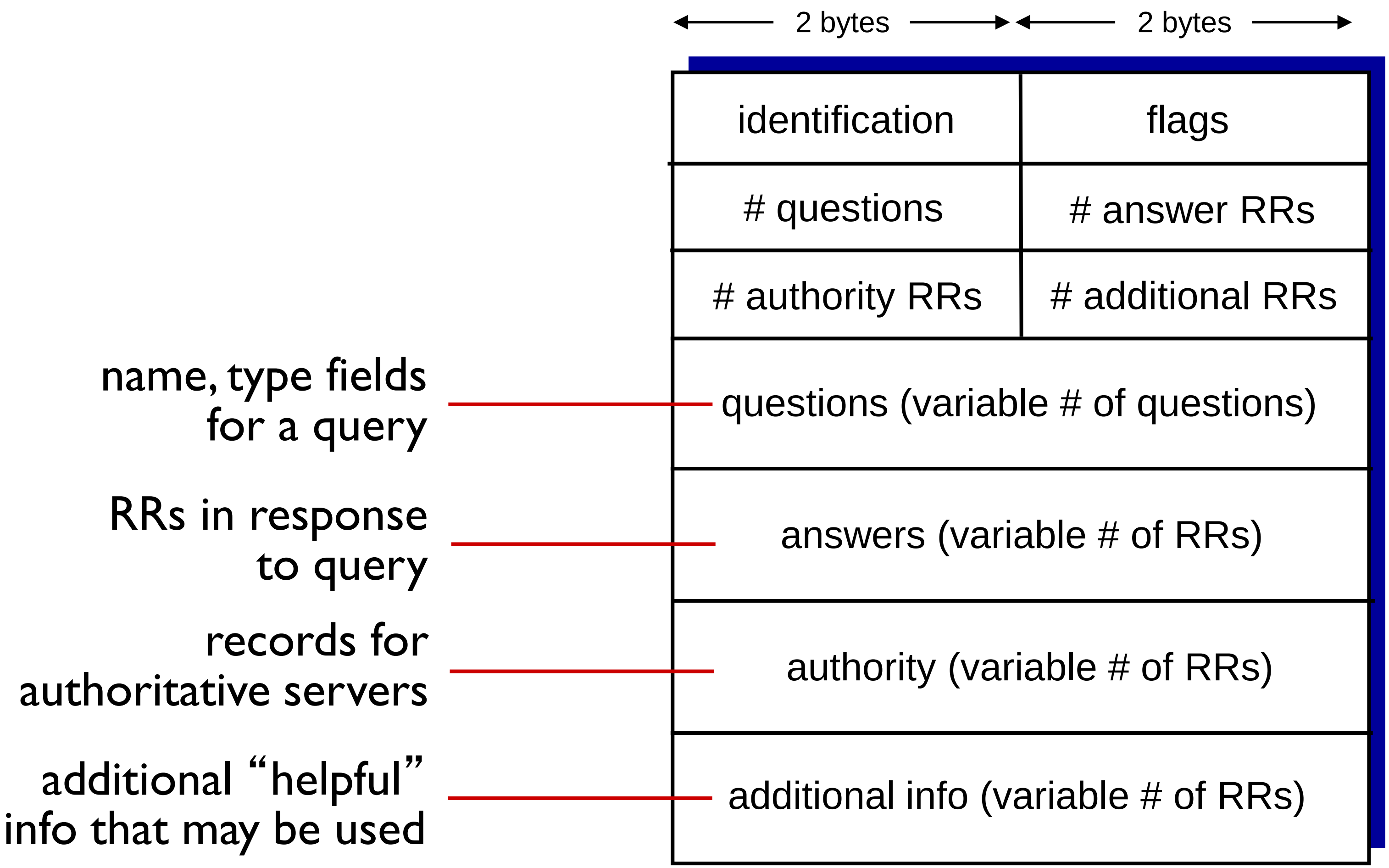
- *query* and *reply* messages, both with same *message format*



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



Header flags format

Field	Description	Length (bits)
QR	Indicates if the message is a query (0) or a reply (1)	1
OPCODE	The type can be QUERY (standard query, 0), IQUERY (inverse query, 1), or STATUS (server status request, 2)	4
AA	Authoritative Answer, in a response, indicates if the DNS server is authoritative for the queried hostname	1
TC	TrunCation, indicates that this message was truncated due to excessive length	1
RD	Recursion Desired, indicates if the client means a recursive query	1
RA	Recursion Available, in a response, indicates if the replying DNS server supports recursion	1
Z	Zero, reserved for future use	3
RCODE	Response code, can be NOERROR (0), FORMERR (1, Format error), SERVFAIL (2), NXDOMAIN (3, Nonexistent domain), etc.[34]	4

https://en.wikipedia.org/wiki/Domain_Name_System

```
Domain Name System (query)
    Transaction ID: 0x178e
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  Queries
    image.google.com: type A, class IN
        Name: image.google.com
        [Name Length: 16]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 7]

0000  02 00 00 00 00 04 3c 22  fb c2 d2 51 08 00 45 00   ······<"  ···Q··E·
0010  00 3e 61 1d 00 00 40 11  ab c4 c0 a8 00 05 d0 43   ·>a···@·  ·······C
0020  dc dc de 06 00 35 00 2a  ae 19 17 8e 01 00 00 01   ·····5·*  ········
0030  00 00 00 00 00 00 05 69  6d 61 67 65 06 67 6f 6f   ·······i  mage·goo
0040  67 6c 65 03 63 6f 6d 00  00 01 00 01               gle·com·  ····
```

https://cabulous.medium.com/dns-message-how-to-read-query-and-response-message-cfebcb4fe817

```
v Domain Name System (response)
    Transaction ID: 0x178e
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 0
  > Queries
  v Answers
    > image.google.com: type CNAME, class IN, cname images.google.com
    > images.google.com: type CNAME, class IN, cname images.l.google.com
    > images.l.google.com: type A, class IN, addr 172.217.1.14
    [Request In: 5]
    [Time: 0.038051000 seconds]
```

```
0000  3c 22 fb c2 d2 51 02 00  00 00 00 04 08 00 45 00   <"···Q·· ······E·
0010  00 7a 15 b0 40 00 3b 11  bb f5 d0 43 dc dc c0 a8   ·z··@·;· ···C····
0020  00 05 00 35 de 06 00 66  e1 b0 17 8e 81 80 00 01   ···5···f ········
0030  00 03 00 00 00 00 05 69  6d 61 67 65 06 67 6f 6f   ·······i mage·goo
0040  67 6c 65 03 63 6f 6d 00  00 01 00 01 c0 0c 00 05   gle·com· ········
0050  00 01 00 00 00 3c 00 09  06 69 6d 61 67 65 73 c0   ·····<·· ·images·
0060  12 c0 2e 00 05 00 01 00  09 3a 80 00 0b 06 69 6d   ··.····· ·:····im
0070  61 67 65 73 01 6c c0 12  c0 43 00 01 00 01 00 00   ages·l·· ·C······
0080  01 2c 00 04 ac d9 01 0e                            ·,······
```

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 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.networkuptopia.com; type MX record for networkutopia.com and type A record for the email server

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 relies to DNS server, which caches

exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification