CMSC 332
Computer Networking
Email and DNS

Professor Szajda

Review

- Last lecture we talked about design principles, and the application protocols HTTP and FTP
  - Text commands sent over a port (recall telnet example)
  - Difference in *statefullness*
  - HTTP and FTP are primarily *pull* protocols

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
- 2.5 DNS
- 2.6 P2P Applications

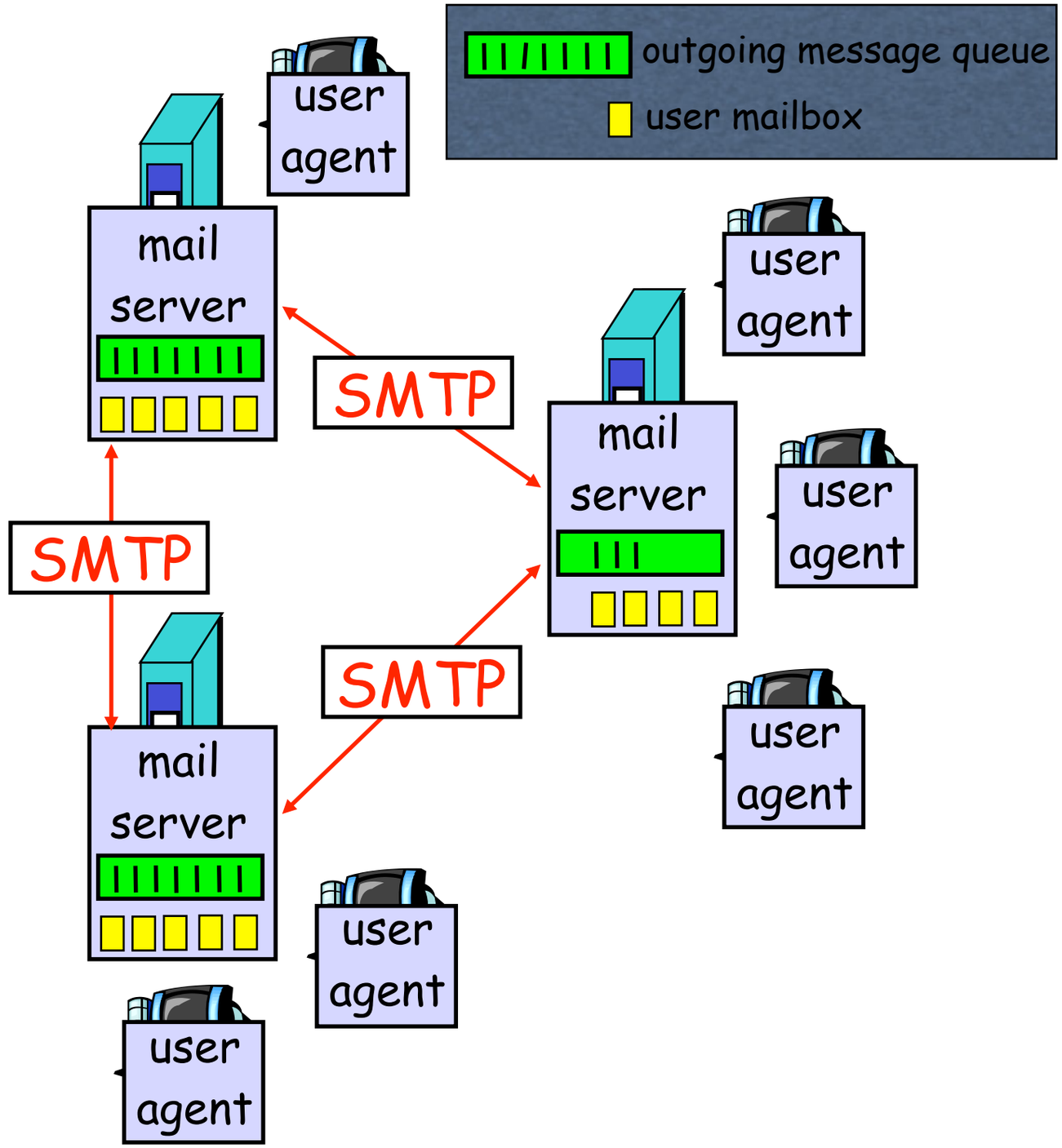
Electronic Mail

Three major components:

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

User Agent

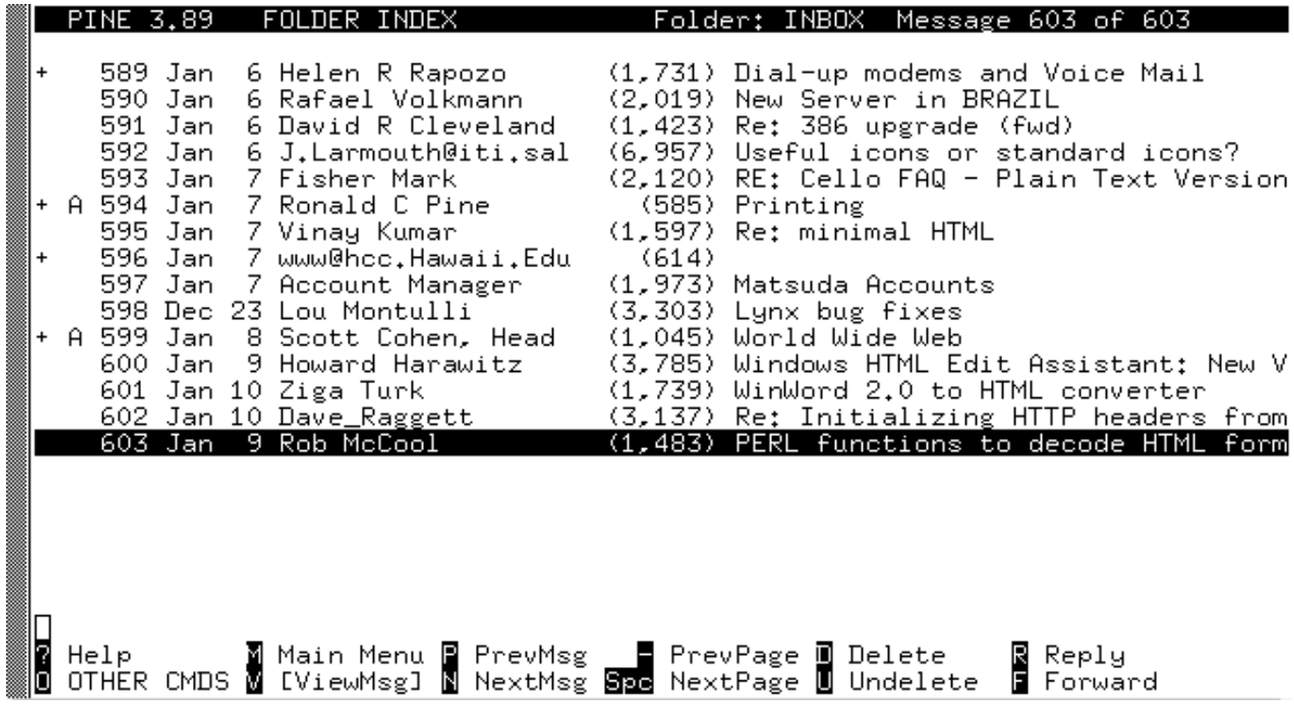
- a.k.a. "mail reader"
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, pine, Apple Mail, GMail
- outgoing, incoming messages stored on server



Electronic Mail

Three major components:

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



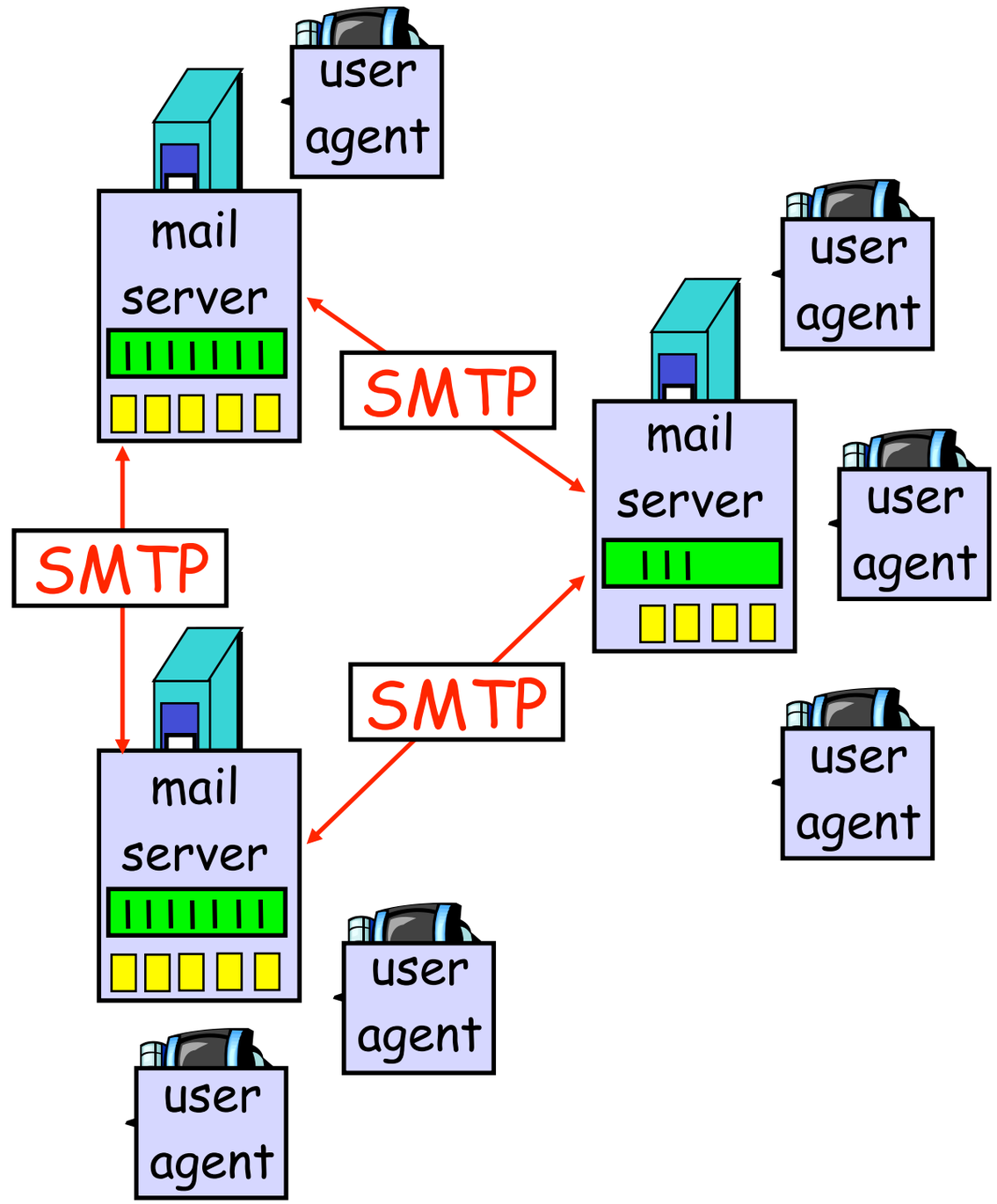
User Agent

- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, pine, Apple Mail, GMail
- outgoing, incoming messages stored on server

Electronic Mail: mail servers

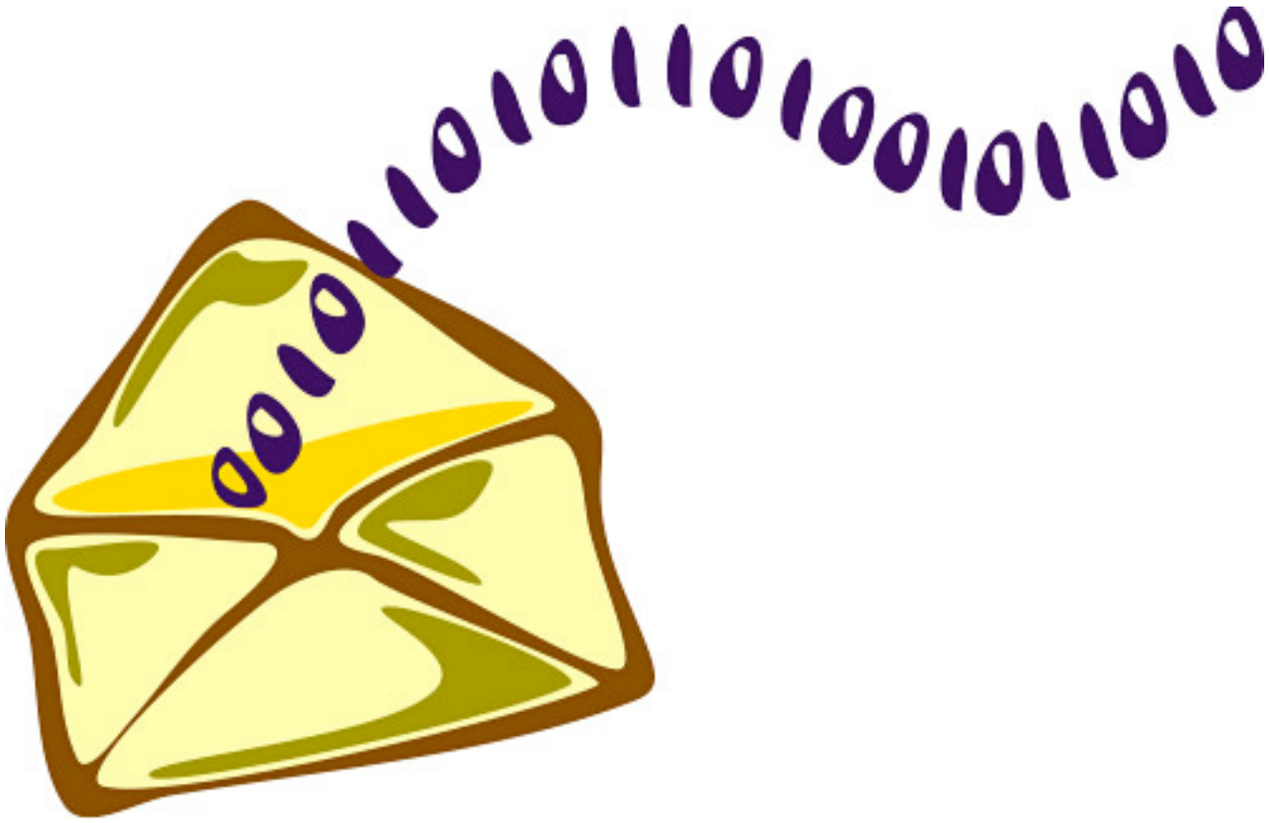
Mail Servers

- 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



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
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII



7-bit ASCII

- The original ASCII character code, provided 128 different characters, numbered 0 to 127. ASCII and 7-bit ASCII are synonymous. Since the 8-bit byte is the common storage element, ASCII leaves room for 128 additional characters, which are used to represent a host of foreign language and other symbols (see the code page). If none of the additional character combinations is used (128-255), the first bit of the byte is 0.

- Squeeze into Seven Bits: Early Internet mail systems supported only 7-bit ASCII codes. In order to transmit executable programs and multimedia files over such systems, these files, which use all eight bits of the byte, must be turned into a 7-bit format using such encoding methods as MIME. This is why some e-mail attachments are larger than their original file size. Their 8-bit data have been converted to 7-bit characters, which adds extra bytes to encode them. See 8-bit ASCII and ASCII.

8-bit ASCII

- Virtually all modern versions of SMTP can handle 8-bit ASCII via the 8BITMIME extension
- Client that wants to send 8-bit ASCII starts session with command EHLO, instead of HELO
- An Extended SMTP (ESMTP) server responds to EHLO with 8BITMIME keyword, if it is “8-bit clean”
- Once received, client can send 8-bit ASCII without need for conversion to 7-bit

Scenario: Alice sends message to Bob

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

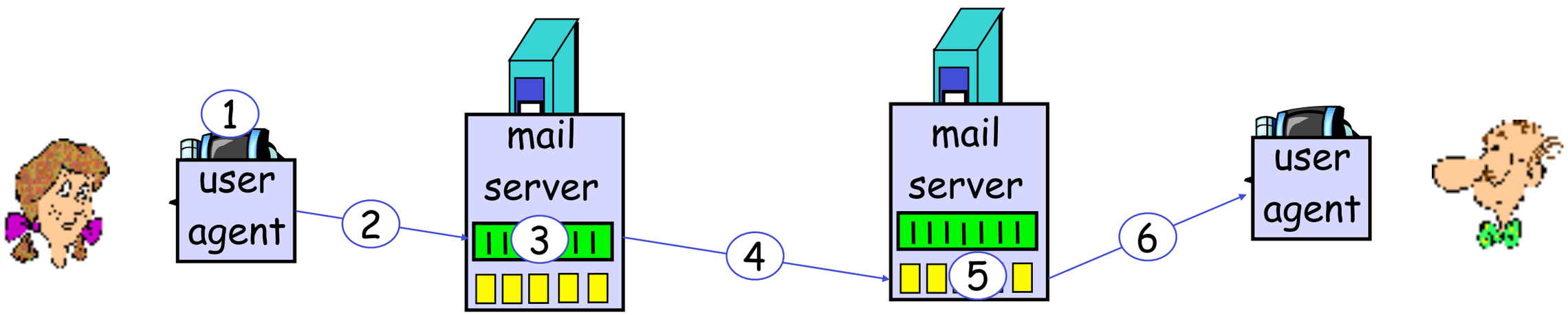
2) Alice's UA sends message to her mail server; message placed in message queue

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



Scenario: Alice sends message to Bob

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

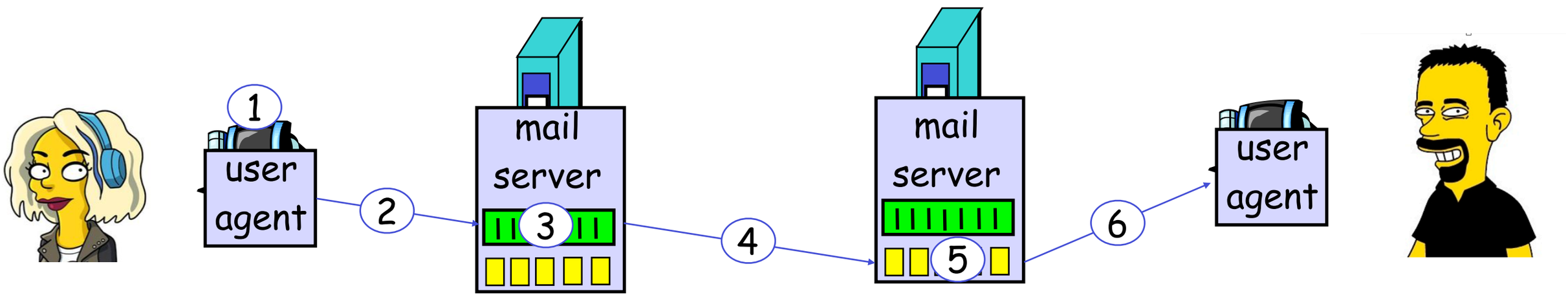
2) Alice's UA sends message to her mail server; message placed in message queue

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



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
```

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)

Well, sometimes…

SMTP: final words

- SMTP uses persistent connections
  - Just like...?
- SMTP used to require 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

But Not Final ASCII Word

- ASCII was indeed originally conceived as a 7-bit code. This was done well before 8-bit bytes became ubiquitous, and even into the 1990s you could find software that assumed it could use the 8th bit of each byte of text for its own purposes ("not 8-bit clean"). Nowadays people think of it as an 8-bit coding in which bytes 0x80 through 0xFF have no defined meaning, but that's a retcon (retroactive conversion)
- There are dozens of text encodings that make use of the 8th bit; they can be classified as ASCII-compatible or not, and fixed- or variable-width. ASCII-compatible means that regardless of context, single bytes with values from 0x00 through 0x7F encode the same characters that they would in ASCII. You don't want to have anything to do with a non-ASCII-compatible text encoding if you can possibly avoid it; naive programs expecting ASCII tend to misinterpret them in catastrophic, often security-breaking fashion. They are so deprecated nowadays that (for instance) HTML5 forbids their use on the public Web, with the unfortunate exception of UTF-16. I'm not going to talk about them any more.

But Not Final ASCII Word

- A fixed-width encoding means what it sounds like: all characters are encoded using the same number of bytes. To be ASCII-compatible, a fixed-width encoding must encode all its characters using only one byte, so it can have no more than 256 characters. The most common such encoding nowadays is Windows-1252, an extension of ISO 8859-1

- There's only one variable-width ASCII-compatible encoding worth knowing about nowadays, but it's very important: UTF-8, which packs all of Unicode into an ASCII-compatible encoding. You really want to be using this if you can manage it.

But Not Final ASCII Word

- As a final note, "ASCII" nowadays takes its practical definition from Unicode, not its original standard (ANSI X3.4-1968), because historically there were several dozen variations on the ASCII 127-character repertoire -- for instance, some of the punctuation might be replaced with accented letters to facilitate the transmission of French text. Nowadays all of those variations are obsolescent, and when people say "ASCII" they mean that the bytes with value 0x00 through 0x7F encode Unicode codepoints U+0000 through U+007F. This will probably only matter to you if you ever find yourself writing a technical standard.

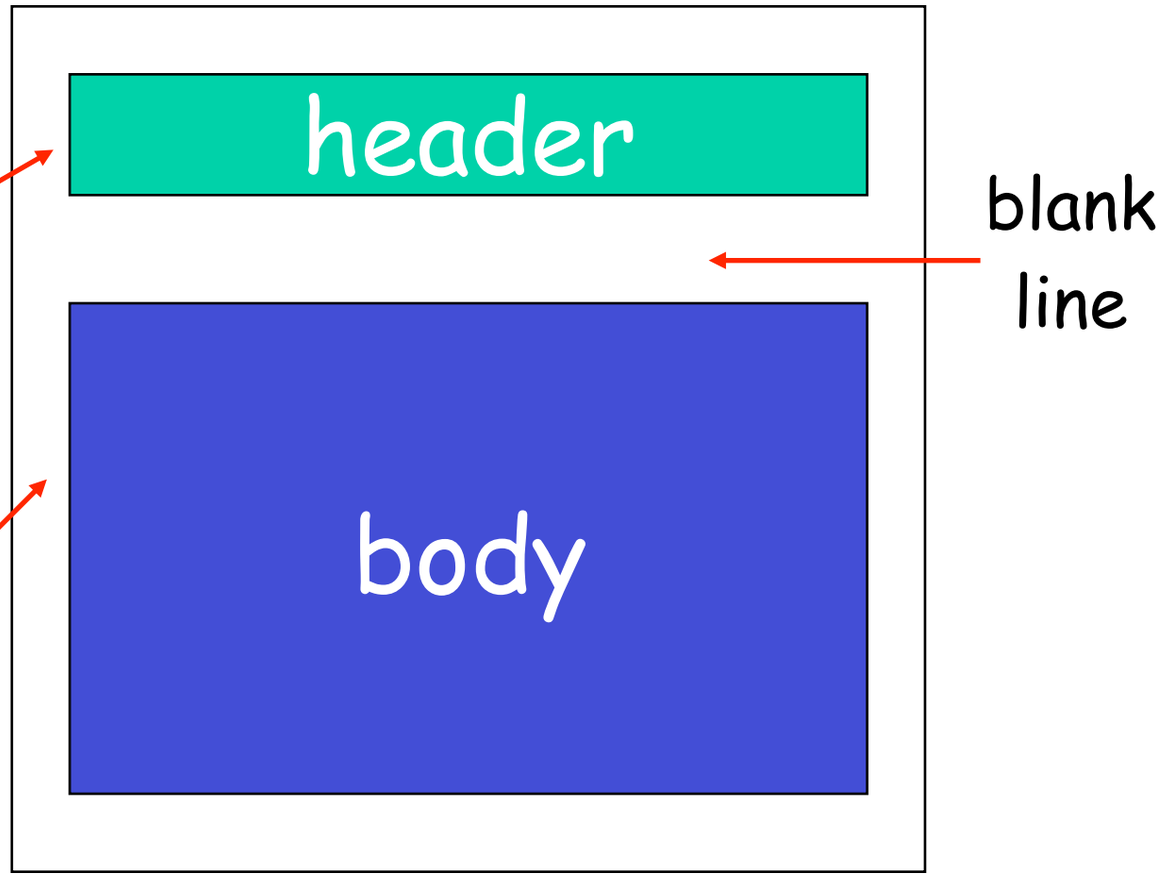
Mail message format

SMTP: protocol for exchanging email msgs

RFC 2822: standard for text message format:

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

  different from SMTP commands!
- body
  - the "message", ASCII characters only



Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- "Multipurpose Internet Mail Extensions"
- additional lines in msg header declare MIME content type

MIME version

method used to encode data

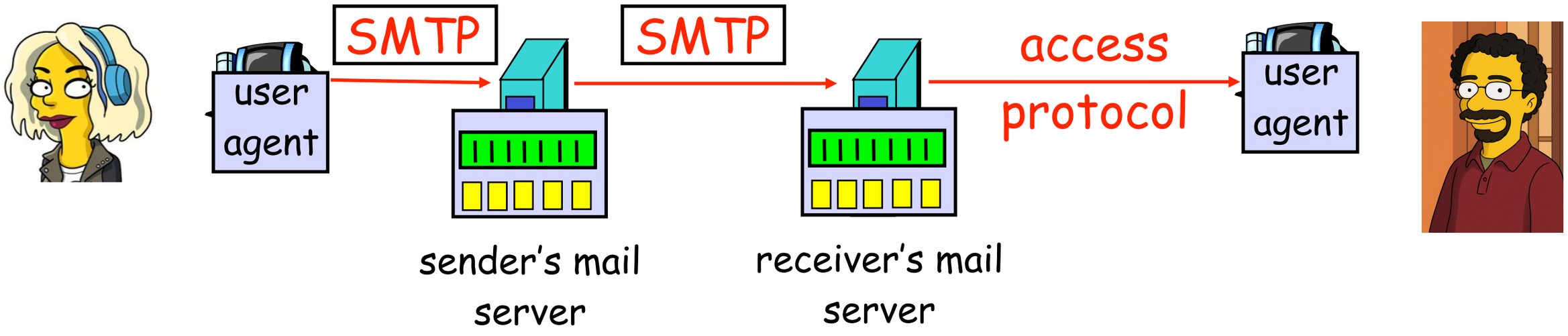
multimedia data type, subtype, parameter declaration

encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-->server) and download
  - IMAP: Internet Mail Access Protocol [RFC 3501]
    - more features (more complex)
    - 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 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more) and IMAP

More about POP3

- Previous example uses "download and delete" mode.
- Bob cannot re-read e-mail if he changes client
- "Download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- Keep all messages in one place: the server
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
- 2.5 DNS
- 2.6 P2P Applications

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: map between IP addresses and name ?

Domain Name System:

- distributed database implemented in hierarchy of many name servers
- application-layer protocol host, routers, name servers to communicate to resolve names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS

DNS services

- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names
- Mail server aliasing
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

Put another way, what are the disadvantages of centralized control of a large system, with a large geographical footprint?

Discuss…

DNS

DNS services

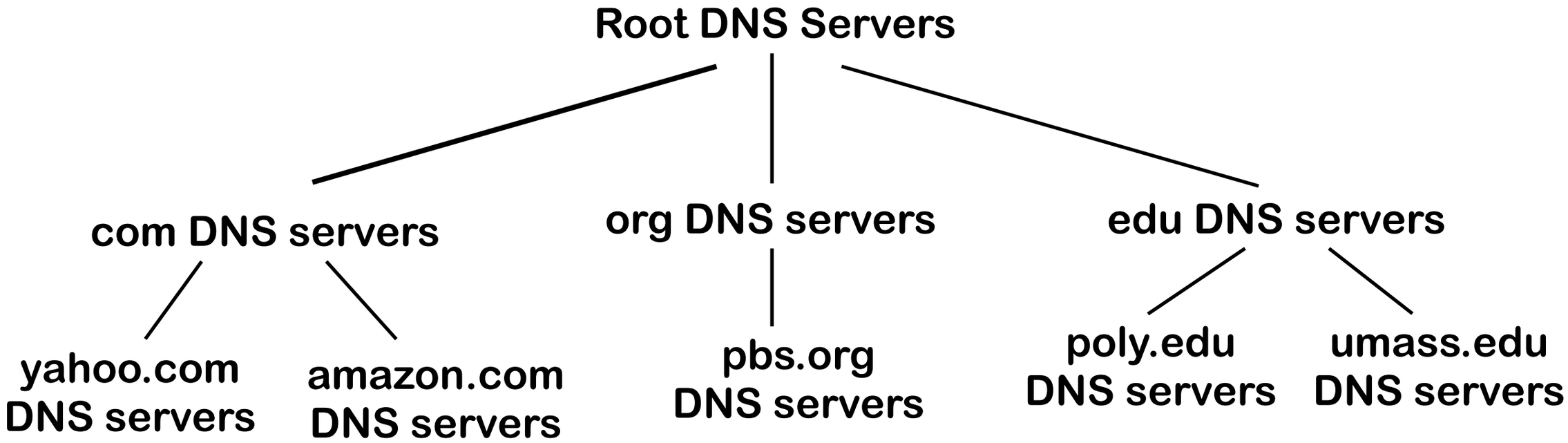
- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names
- Mail server aliasing
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

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

In summary, *it doesn't scale*!

Distributed, Hierarchical Database



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

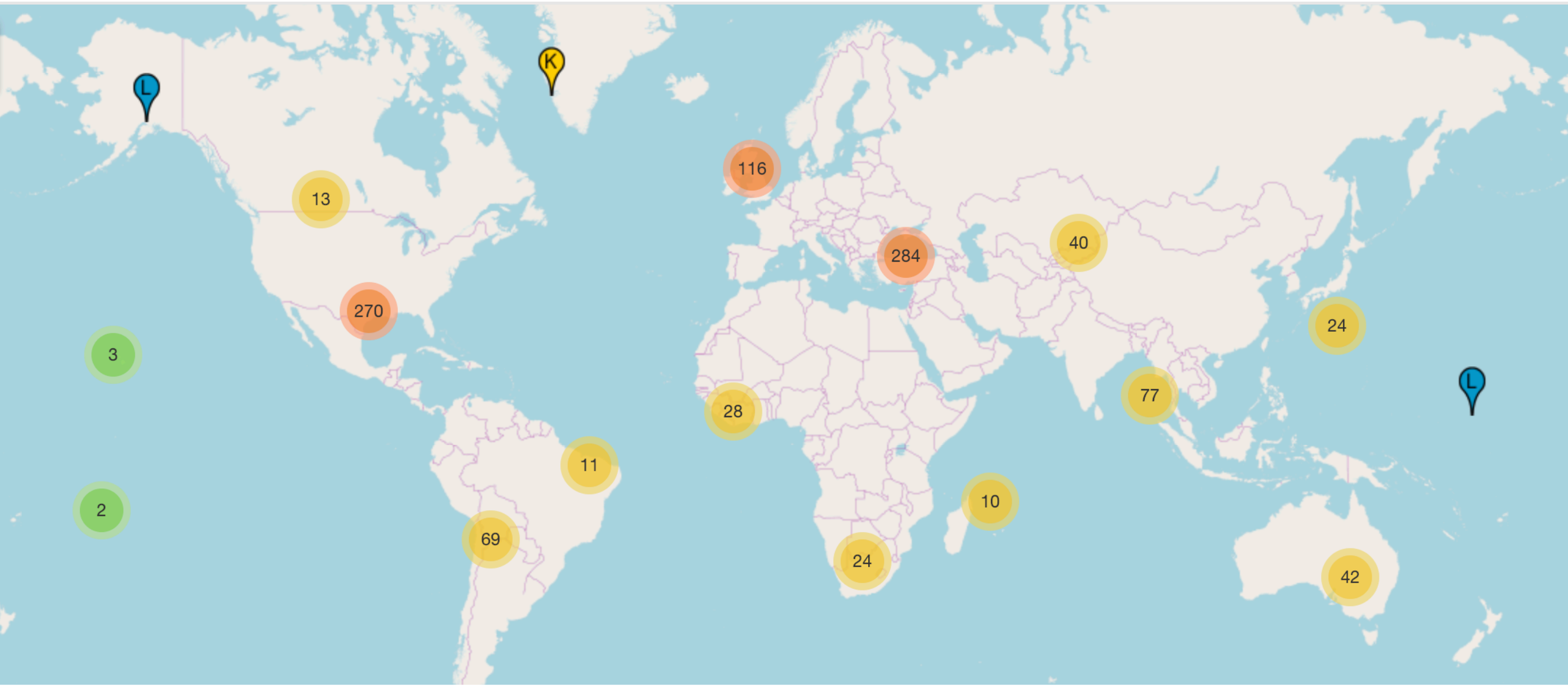
- Client queries a root server to find com DNS server
- Client queries com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com

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 root name servers worldwide

Root Name Servers



1016 servers controlled by 12 operators (using anycast)

Root Name Servers

1988 servers controlled by 12 operators (using anycast)

TLD and Authoritative Servers

- Top-level domain (TLD) servers: responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Verisign maintains servers for com TLD
  - Educause for edu TLD

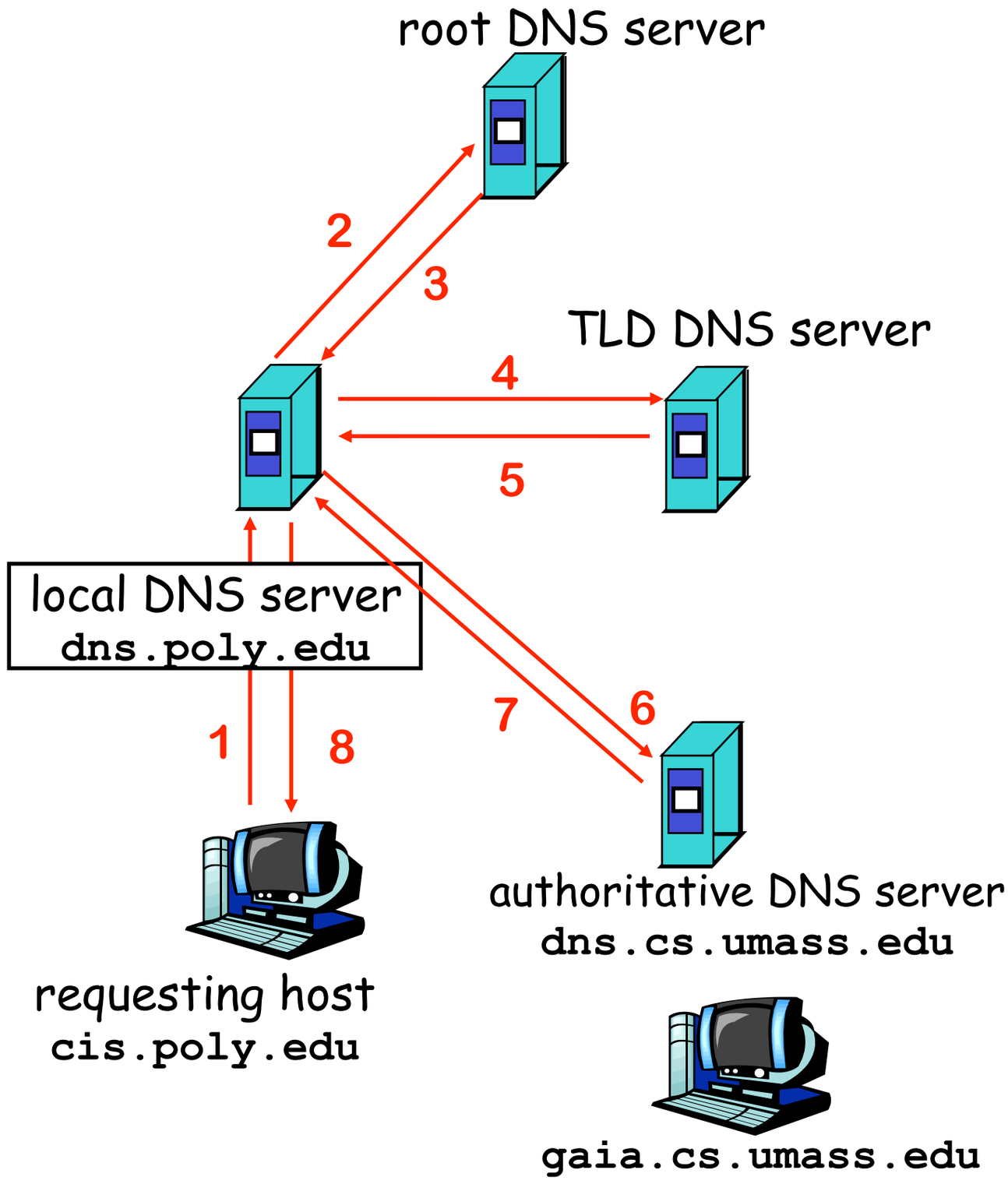
- Authoritative DNS servers: organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail).
  - Can be maintained by organization or service provider

Local Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
  - Also called "default name server"
- When a host makes a DNS query, query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy.

Example

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



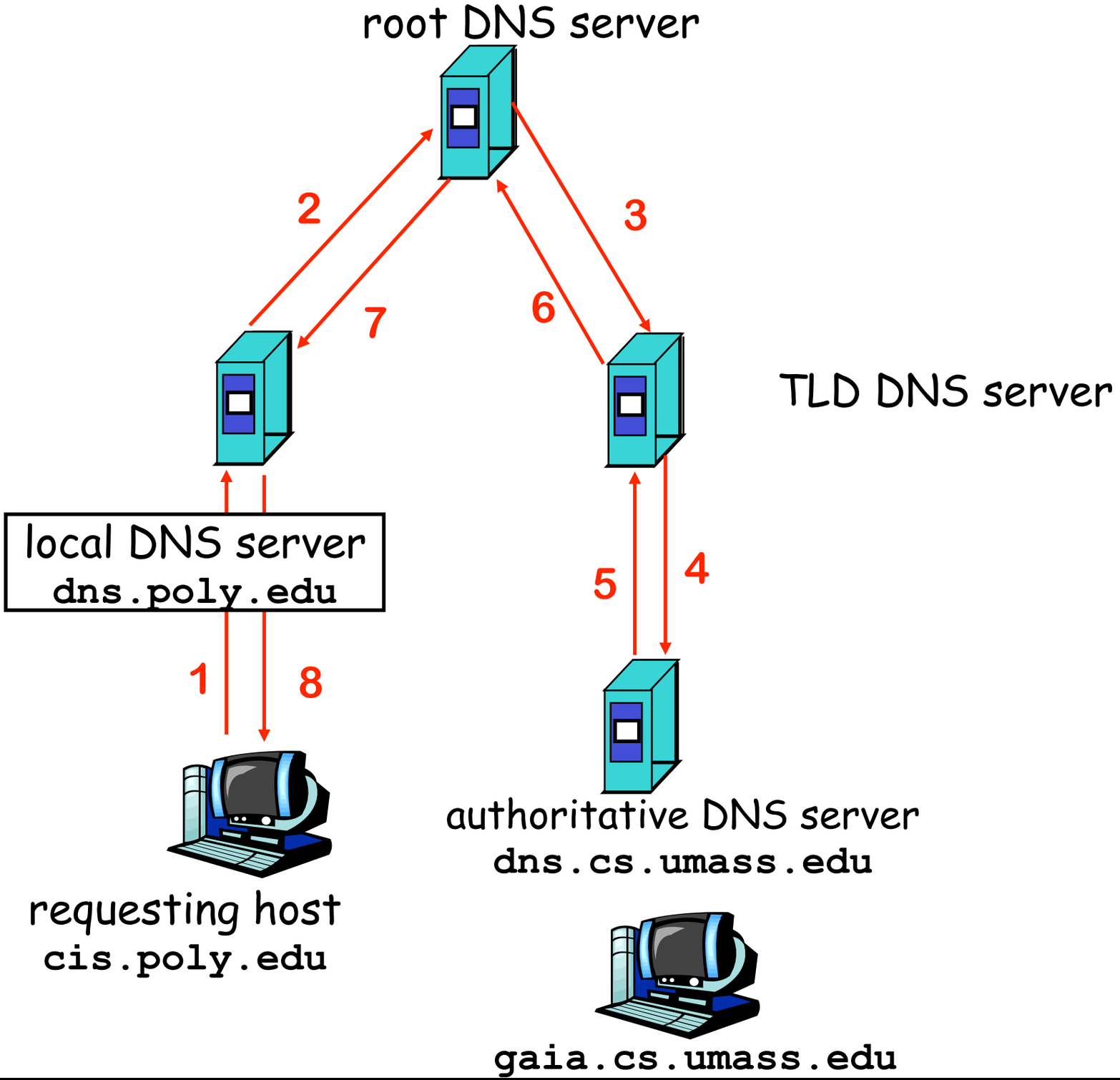
Recursive queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

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



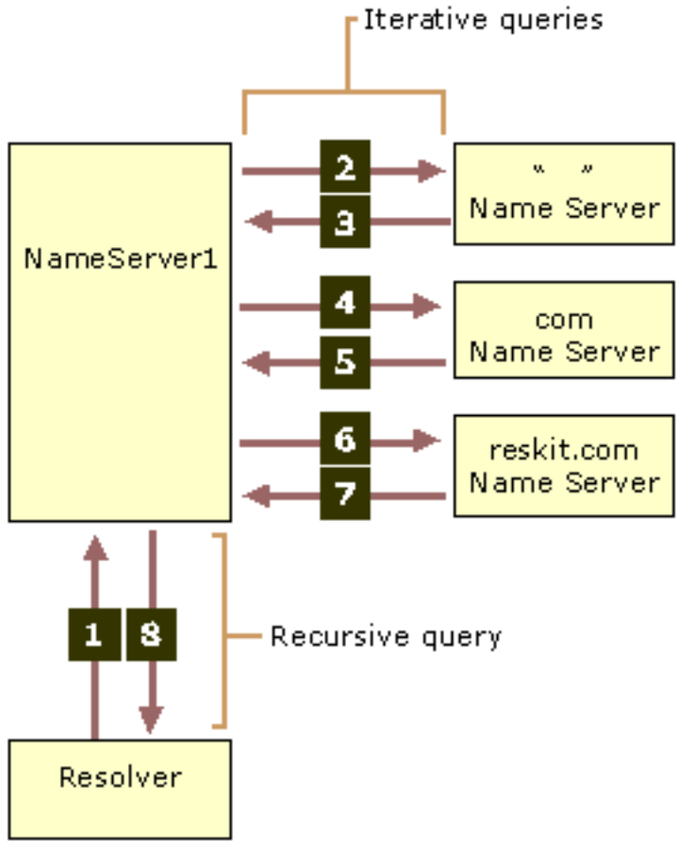
Recursive queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

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



DNS: caching and updating records

- once (any) name server learns mapping, it *caches* the mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited
- update/notify mechanisms
  - RFC 2136, 3007
  - http://www.ietf.org/html.charters/dnsind-charter.html

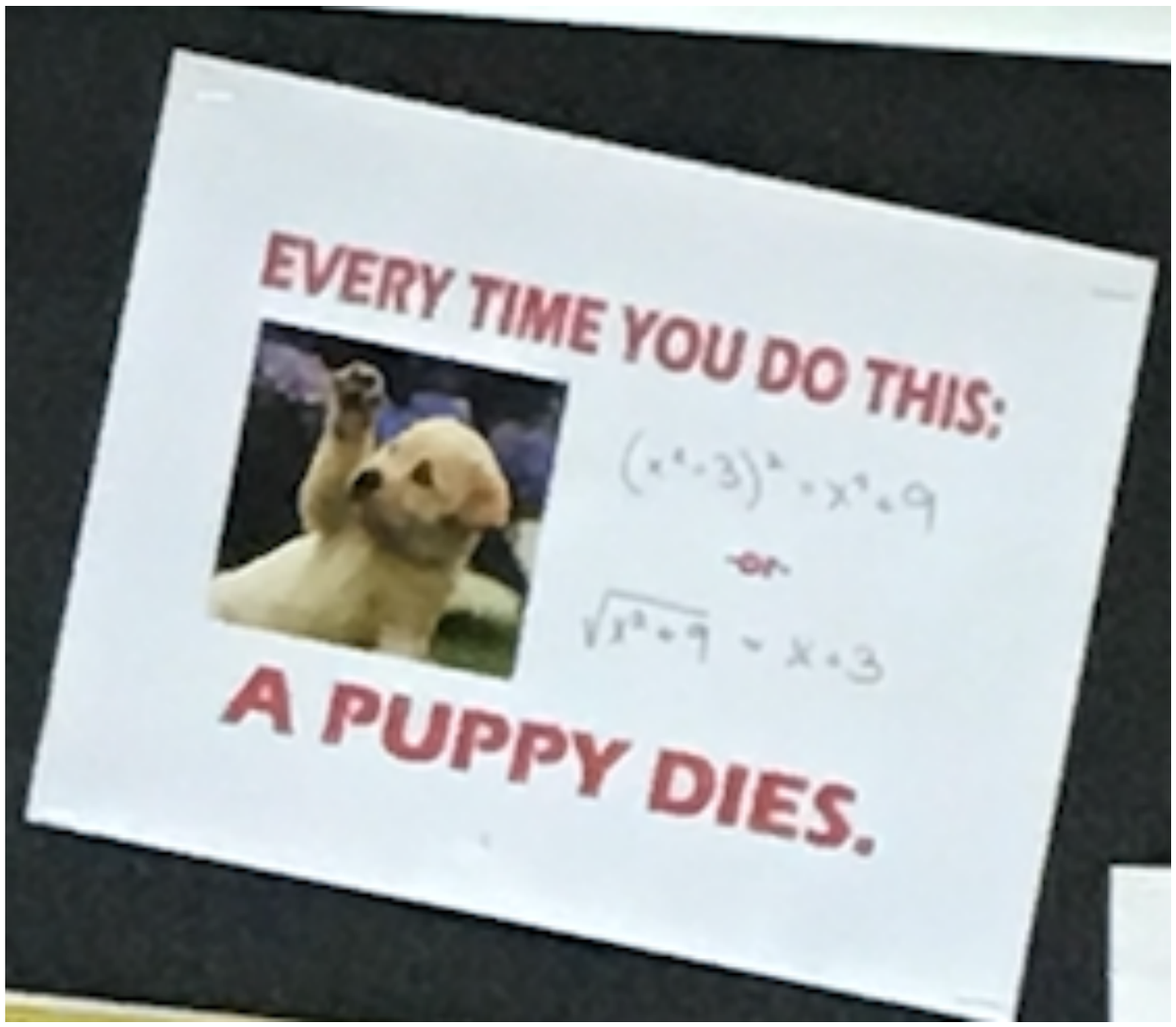
DNS records

DNS: distributed db 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


Asymptotic high fives
The fun never ends!

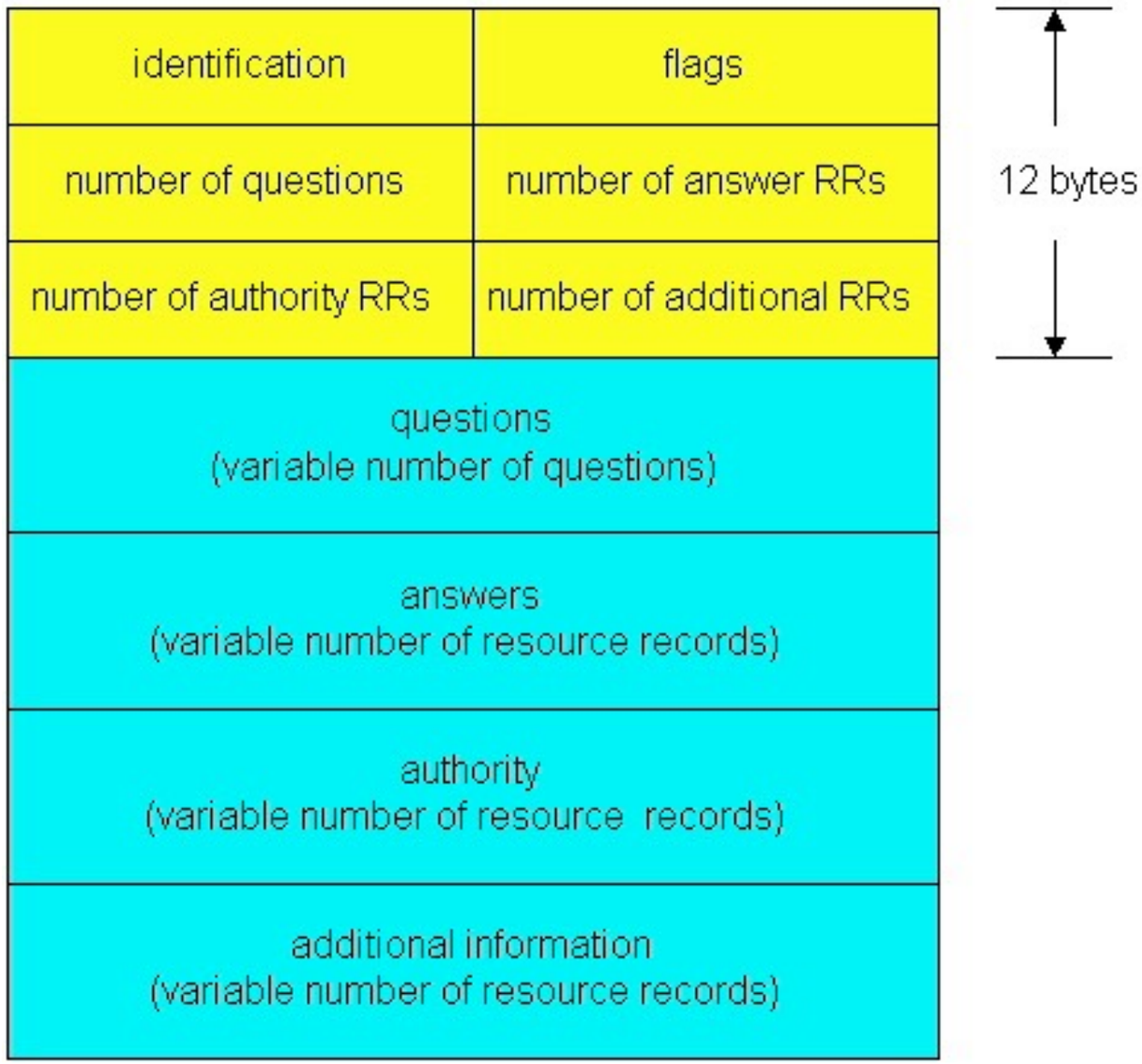

EVERY TIME YOU DO THIS:
A PUPPY DIES.

DNS protocol, messages

DNS protocol : 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



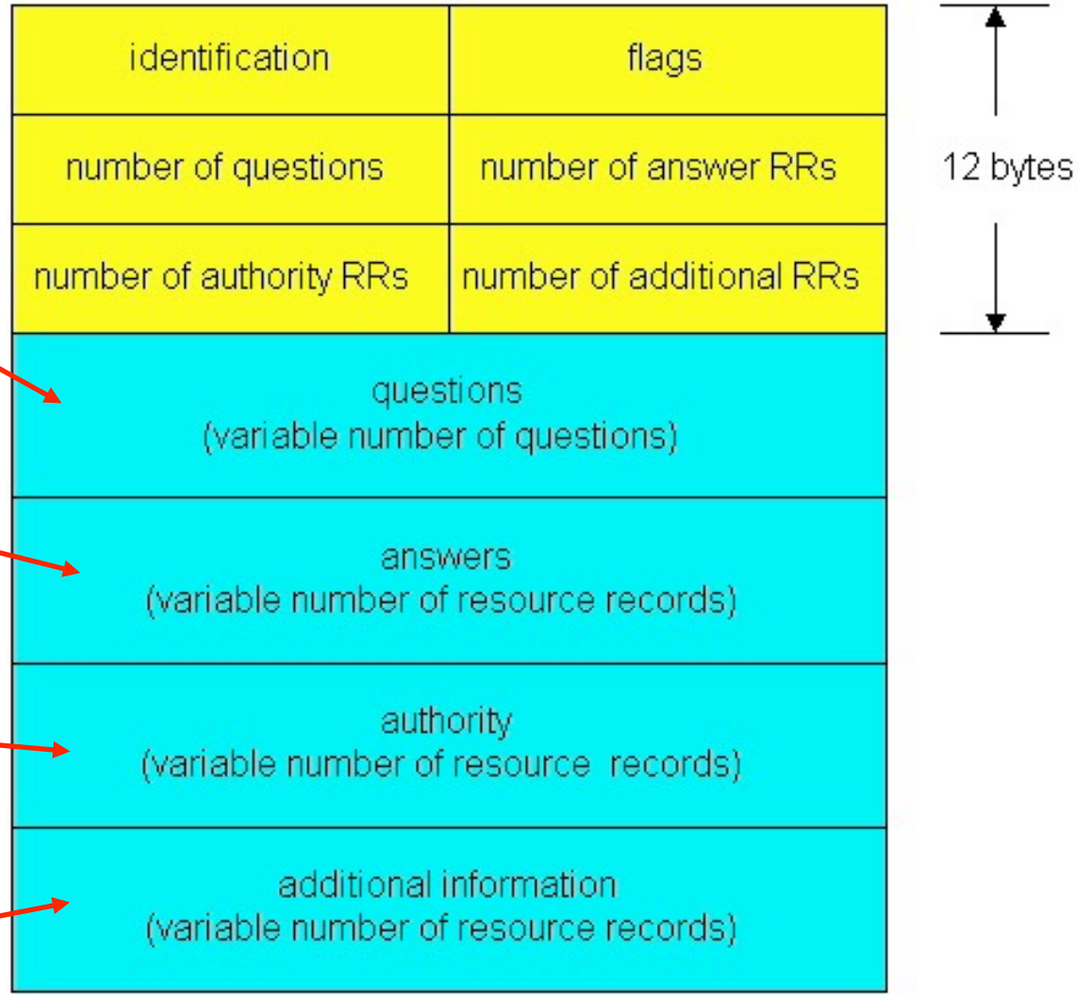
DNS protocol, messages

Name, type fields for a query

RRs in response to query

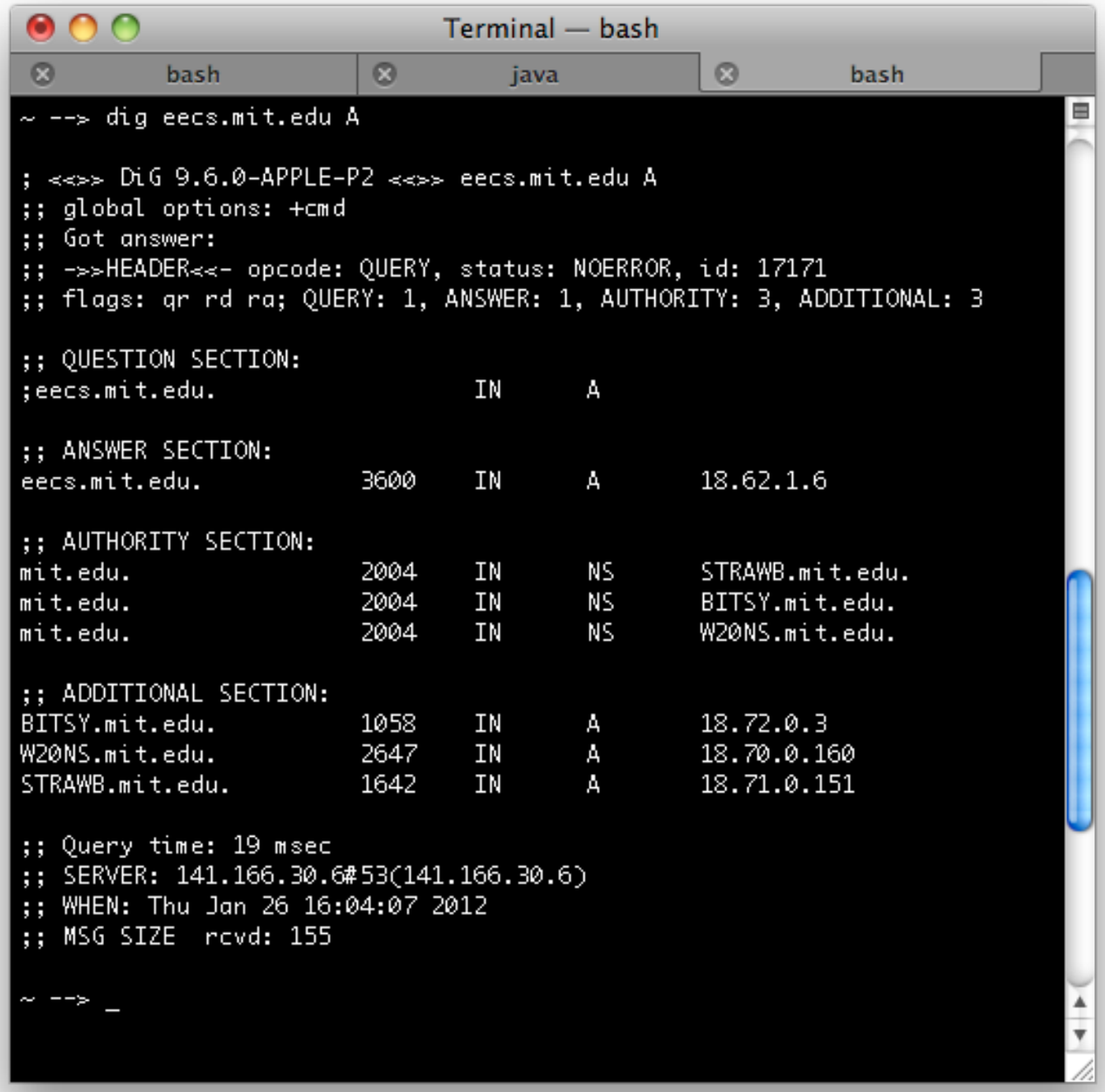
records for authoritative servers

additional "helpful" info that may be used



Viewing DNS Queries

- Text recommends nslookup
- I use dig



Inserting records into DNS

- Example: just created startup "Network Utopia"
- Register name networkuptopia.com at a registrar (e.g., Network Solutions)
  - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:

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

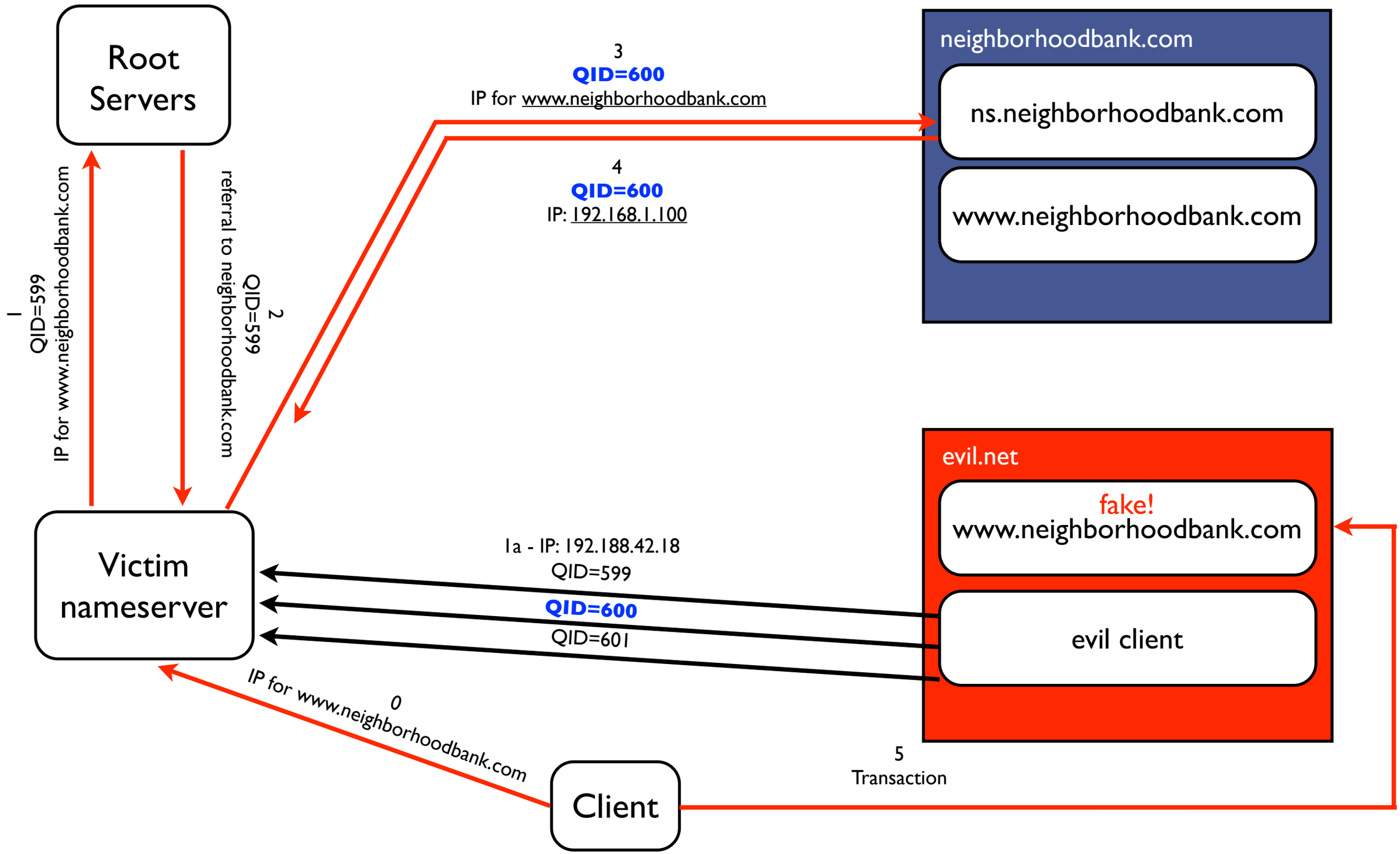
- Put in authoritative server Type A record for www.networkuptopia.com and Type NS record for networkutopia.com
- How do people get the IP address of your Web site?

DNS Security Issues

- Given that so many different servers can respond to your request, *how do you know that what you get back is correct?*
  - Are you sure that you spoke to the resolver you think you spoke to?

- What happens if you manage to give a resolver false look-up information?

DNS Cache Poisoning



DNS Attacks - Real?

- Golden Shield Project
- Kaminsky Attack
- Others?
  - Why is it difficult to know?