

# **CSC4200/5200 – COMPUTER NETWORKING**

**Instructor: Susmit Shannigrahi**

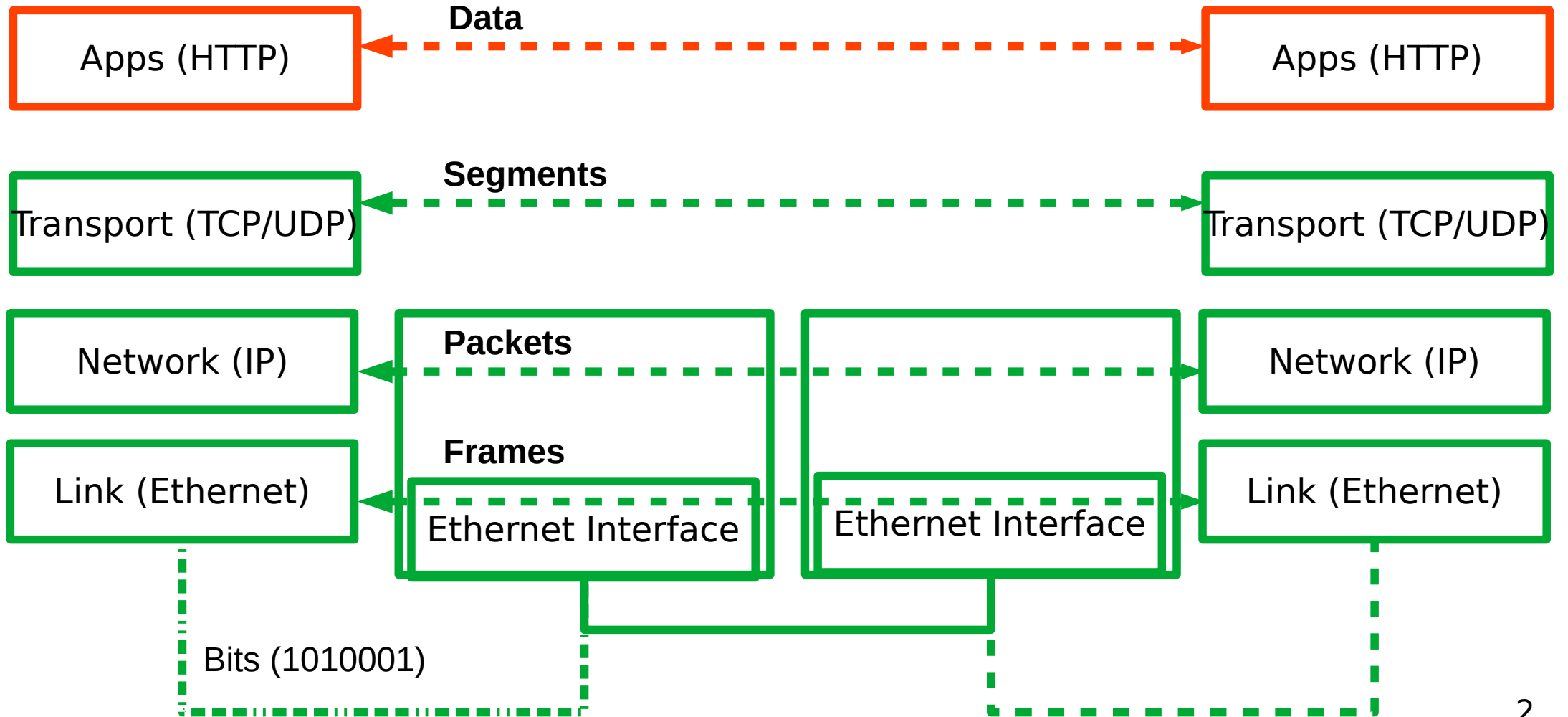
**NETWORKED APPLICATIONS – EMAIL AND DNS**

**sshannigrahi@tntech.edu**

**GTA: dereddick42@students.tntech.edu**

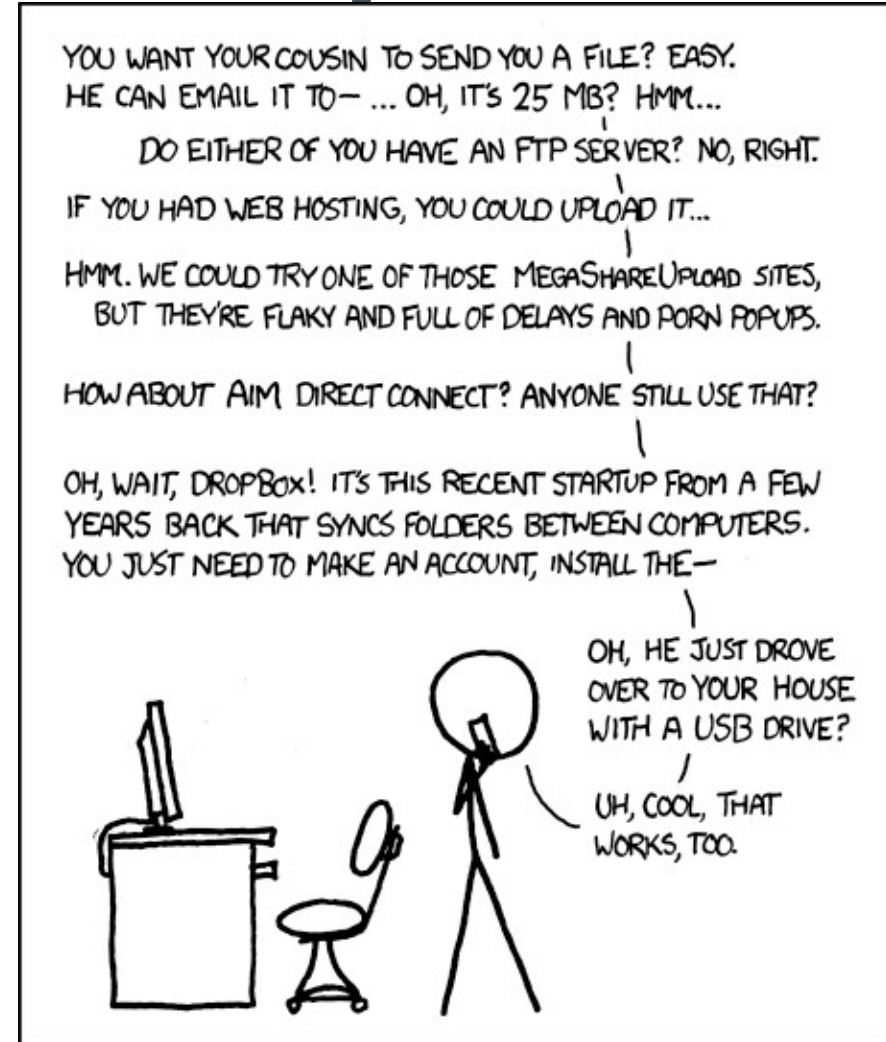
---





# How do you send the cat picture?

- Looked at Web
- How about email?



I LIKE HOW WE'VE HAD THE INTERNET FOR DECADES, YET "SENDING FILES" IS SOMETHING EARLY ADOPTERS ARE STILL FIGURING OUT HOW TO DO.

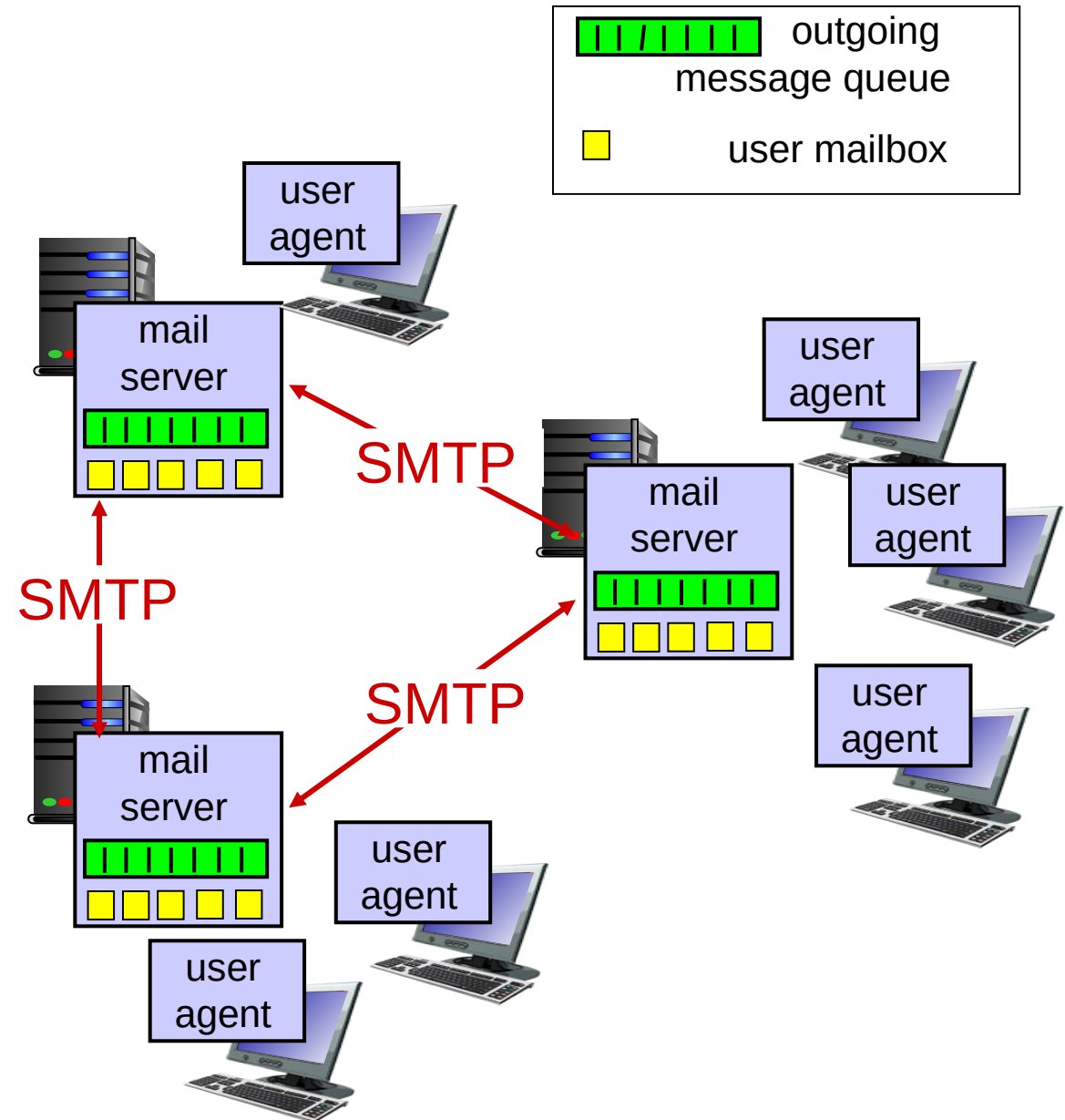
# Electronic mail

## *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



# Electronic Mail: SMTP [RFC 2821]

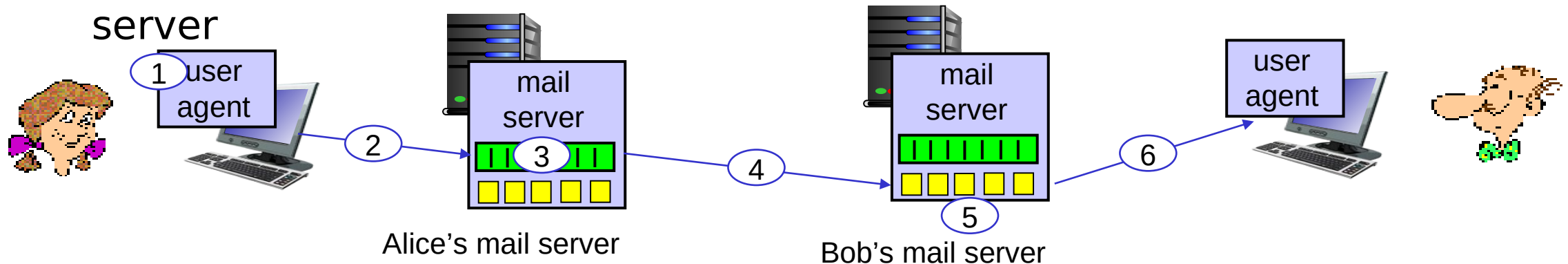
---

- 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

# 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

---

```
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
```

# Mail message format

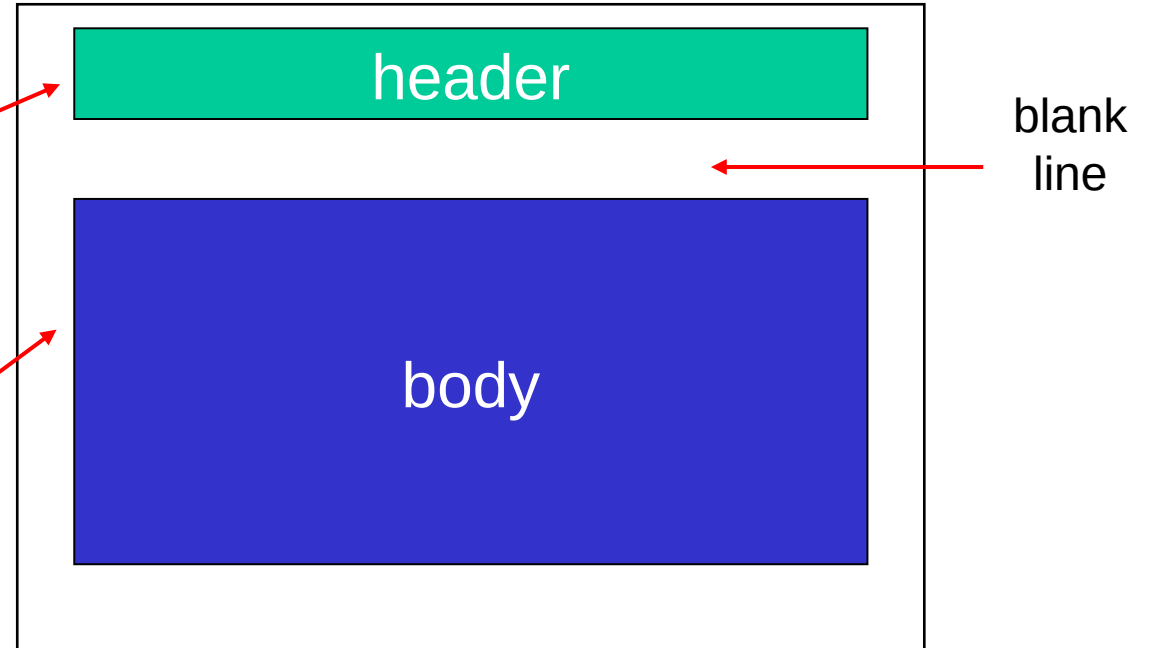
SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

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

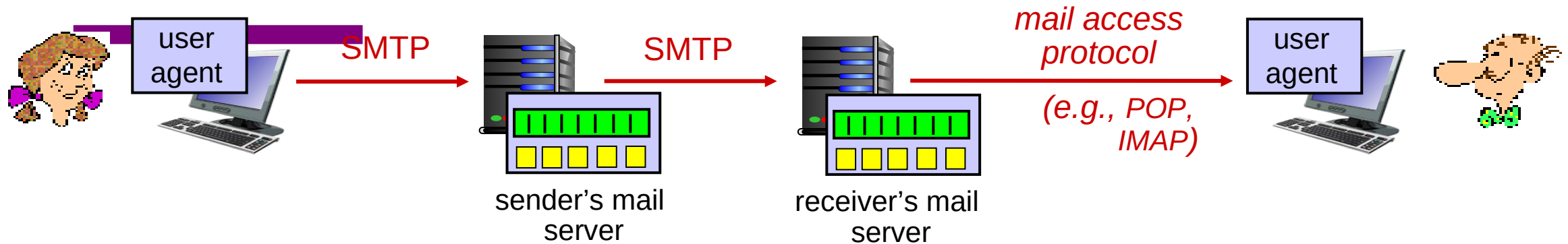
*different from* SMTP MAIL  
FROM, RCPT TO: commands!

- Body: the “message”
  - ASCII characters only





# Mail access protocols

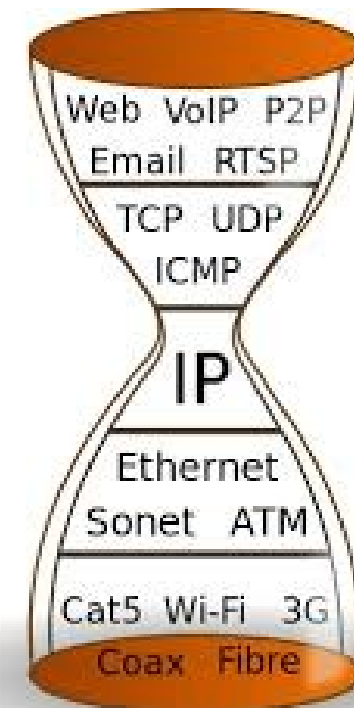
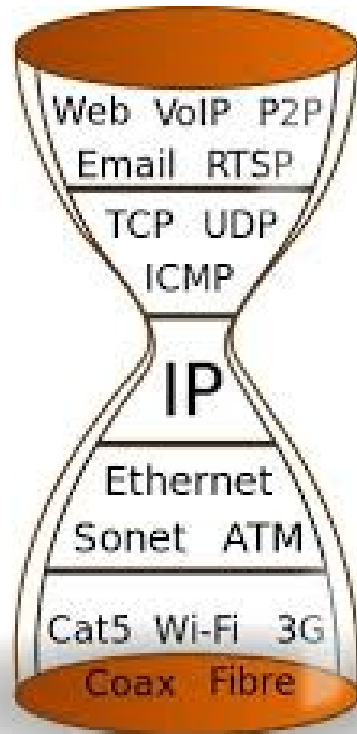


- **SMTP**: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - **POP**: Post Office Protocol [RFC 1939]: authorization, download
  - **IMAP**: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server
  - **HTTP**: gmail, Hotmail, Yahoo! Mail, etc.

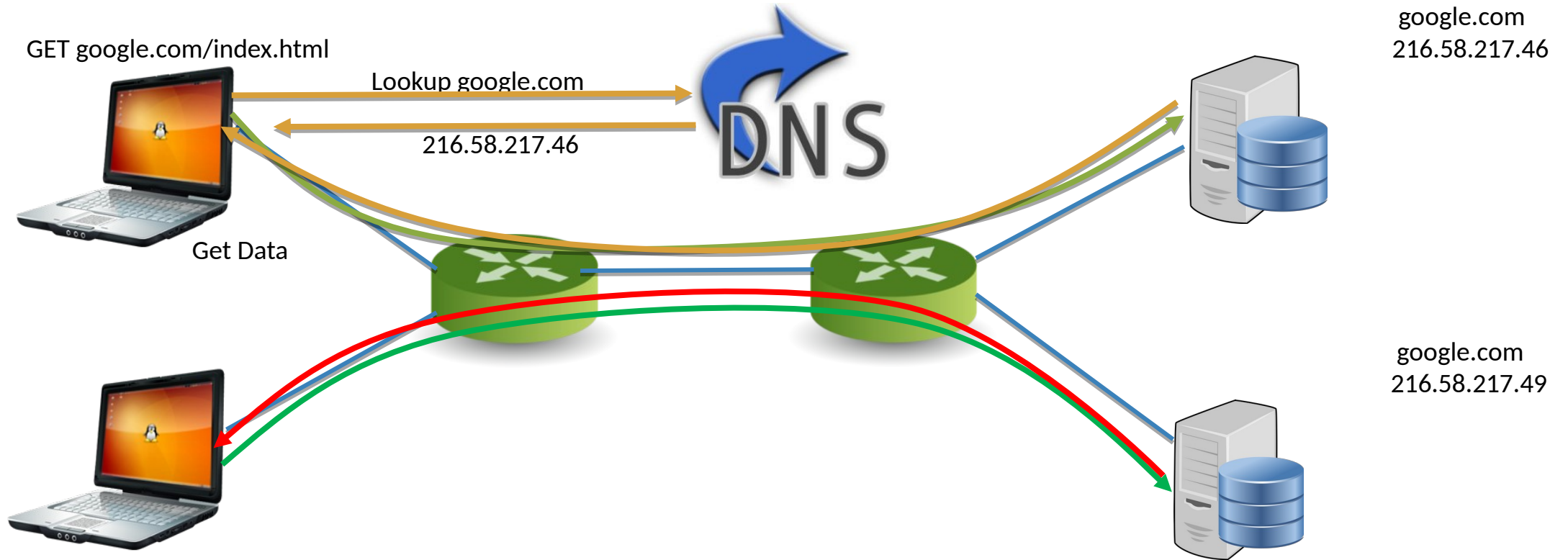
# IP Based Communication

---

[youtube.com/catvideo1](https://www.youtube.com/watch?v=catvideo1)



# IP Based Communication



# DNS – IP to Name

---

People: Good with names

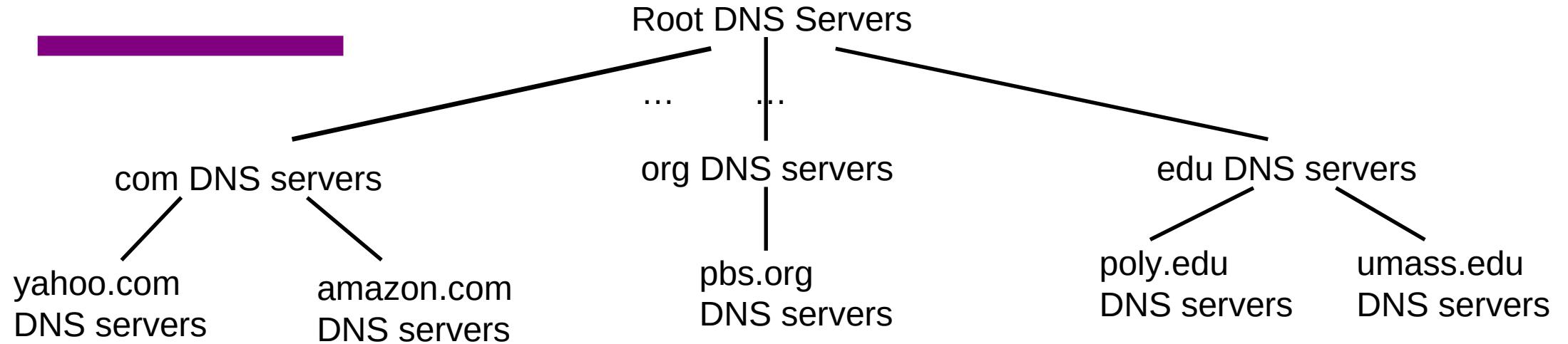
Machines: Good with numbers

Ask a person to remember 100s of Ips

- May not work well

DNS maps IP addresses to human readable names.

# DNS: a distributed, hierarchical database

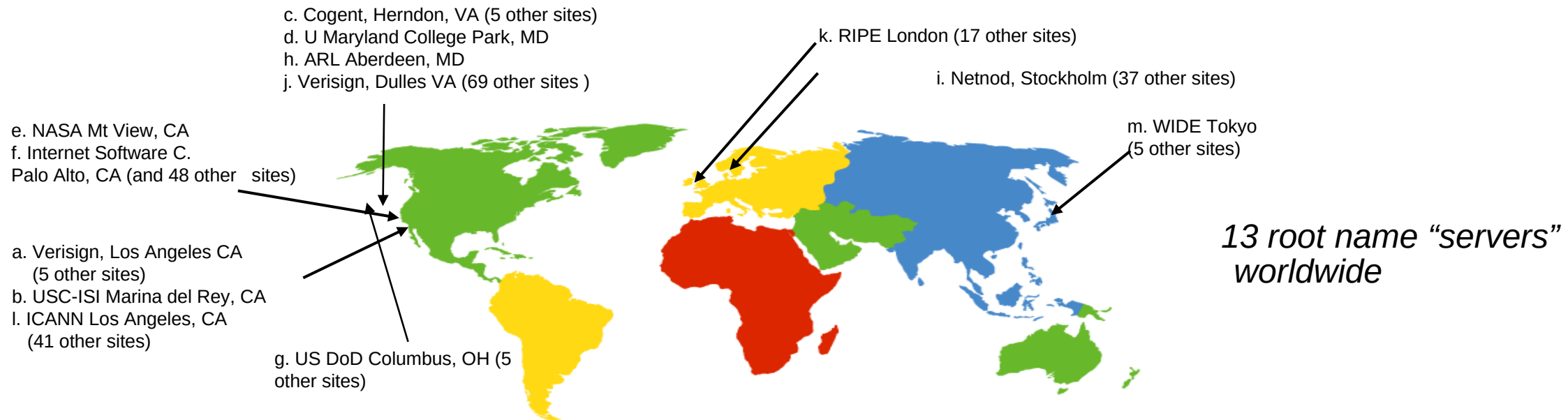


*client wants IP for [www.amazon.com](http://www.amazon.com);*

- 1) client queries root server to find com DNS server
- 2) client queries .com DNS server to get amazon.com DNS server
- 3) client queries amazon.com DNS server to get IP address for [www.amazon.com](http://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



# 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
  - Served from cache
  - Looked up
  - **Attack?**

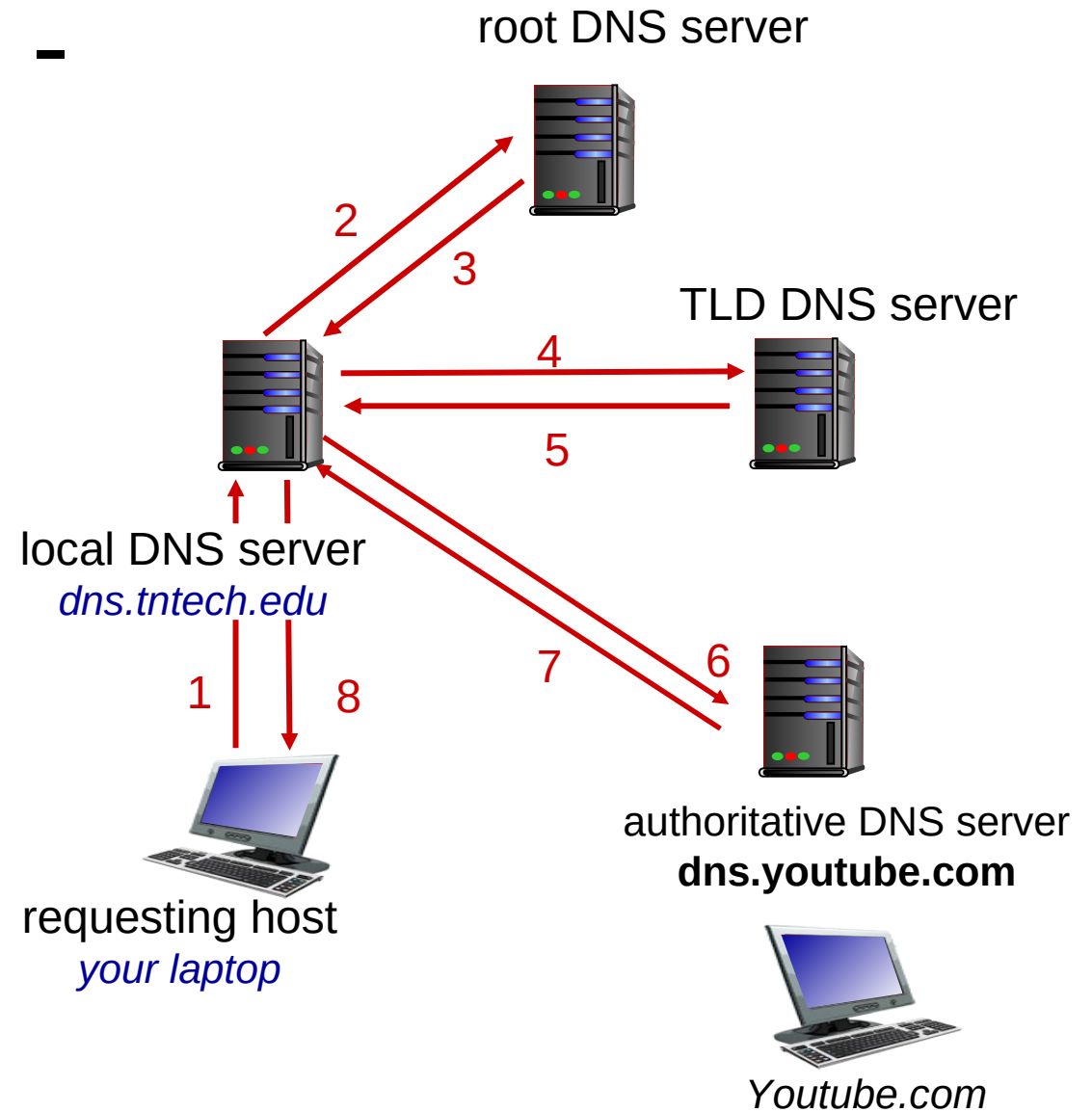


# DNS name resolution example - Iterative

- host at tntech.edu wants IP address for youtube.com

## *iterated query:*

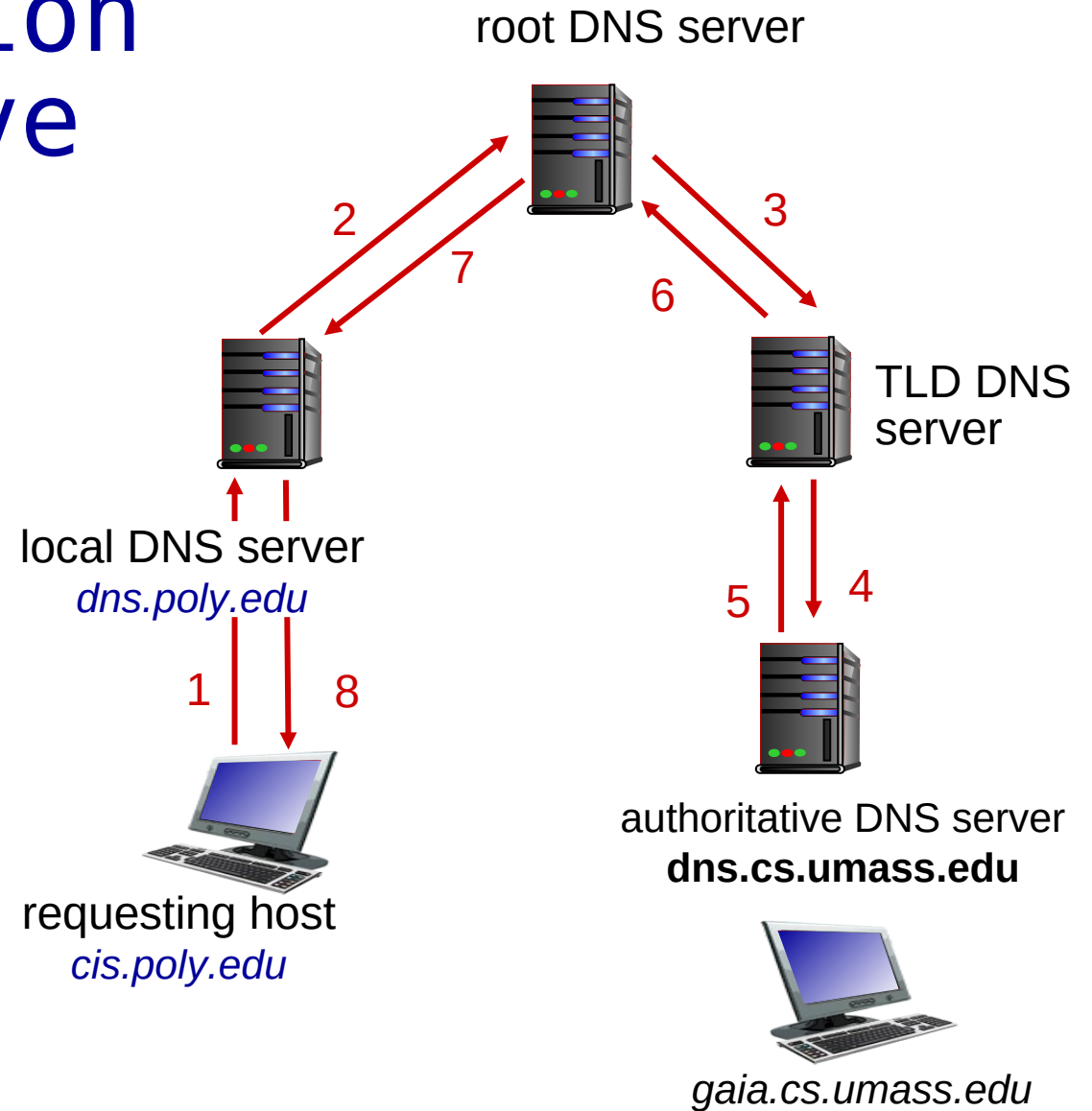
- ❖ contacted server replies with name of server to contact
- ❖ “I don’t know this name, but ask this server”



# DNS name resolution example- Recursive

## *recursive query:*

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

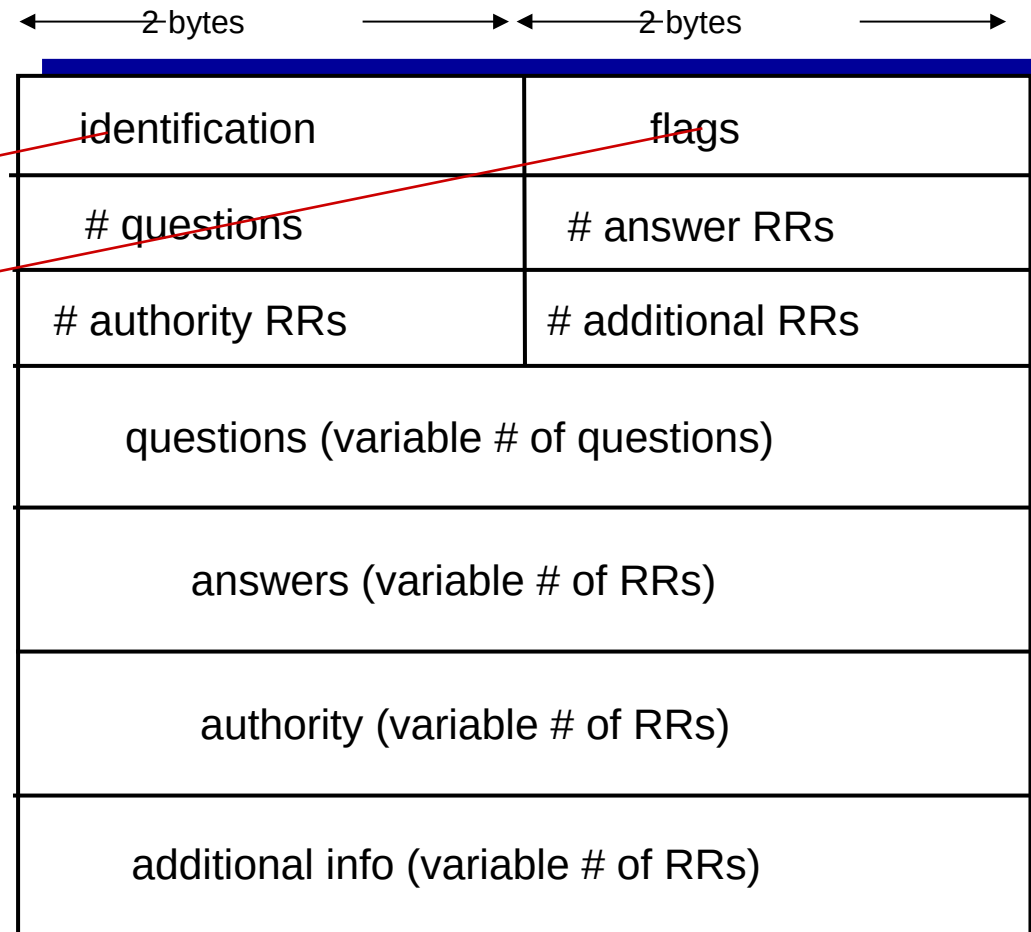


# DNS protocol, messages

- *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



# Inserting records into DNS

---

example: new startup “tornadoguard”

- register name tornadoguard.com at *DNS registrar* (godaddy, gandi.net)
  - Tell them the IP of your local DNS server and name
  - registrar inserts two RRs into .com TLD 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 replies to DNS server, which caches

## Exploit DNS for DDoS

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