# Couche application

- FTP
- SMTP / POP3 / IMAP
- DNS

*Computer Networks. Tanenbaum*

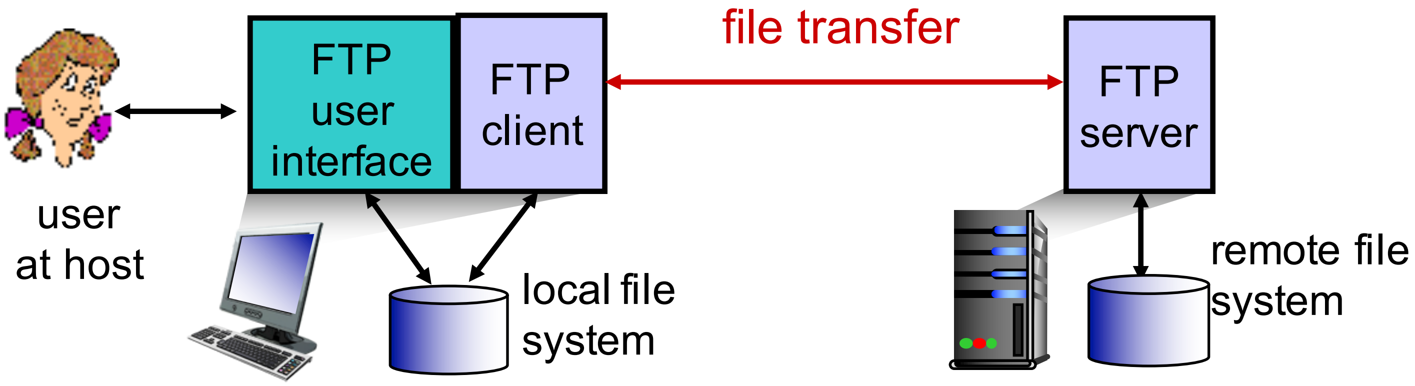
*Computer Networking. Kurose&Ross*

Carole Delporte

# Couche application

## FTP

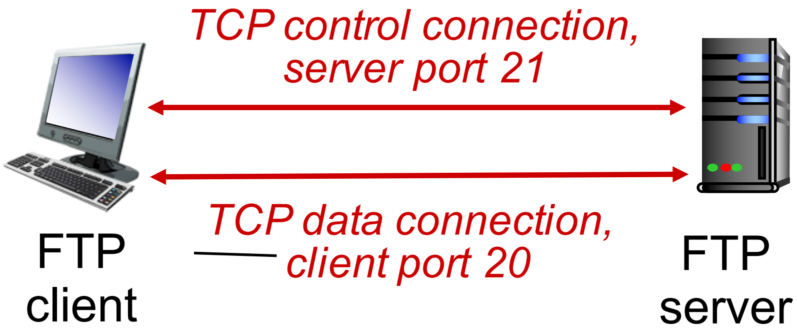
# FTP: the file transfer protocol



- 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 port 20
- after transferring one file, server closes data connection



TCP control connection, server port 21
TCP data connection, client port 20
FTP client
FTP server

- server opens another TCP data connection to transfer another file
- control connection: *"out of band"* (http is "in-band")
- FTP server maintains "state": current directory, earlier authentication

# FTP commands, responses

*sample commands (client→ server):*

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

*sample return codes ( serveur→ client)*

- 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`**

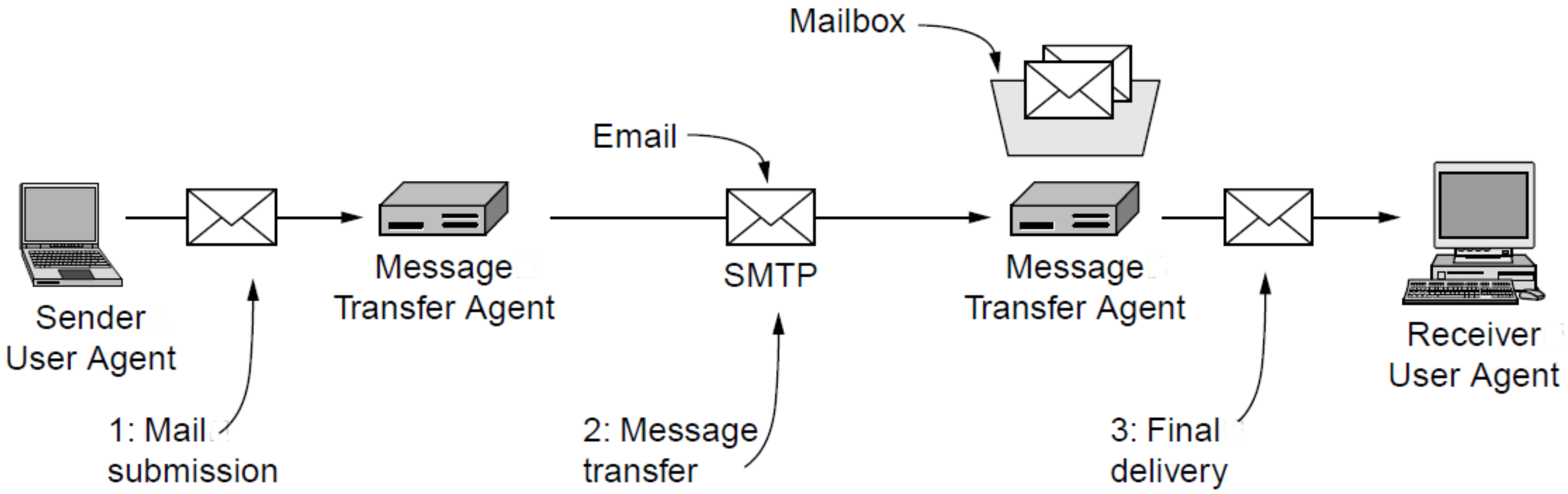
# Couche application

electronic mail

- SMTP, POP3, IMAP

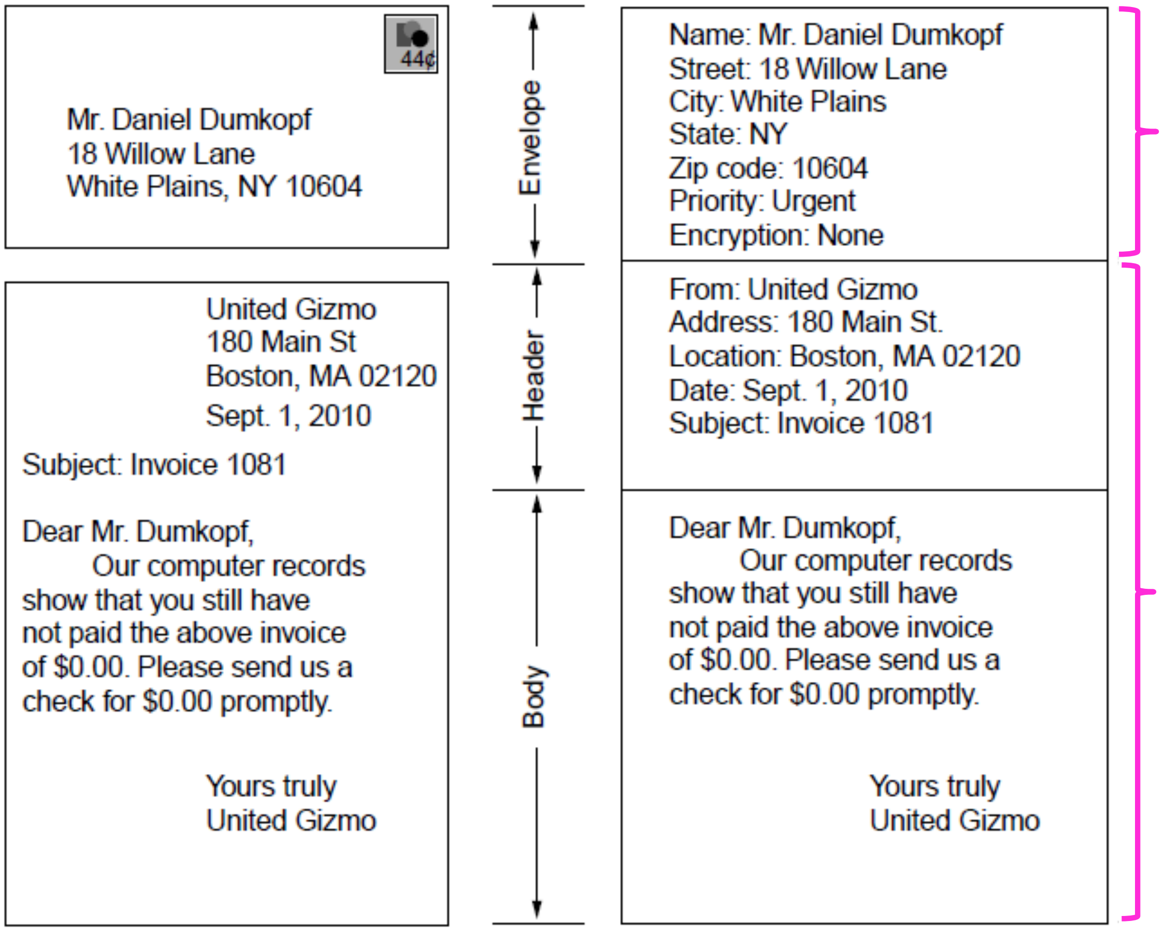
# Architecture et Services

The key components and steps (numbered) to send email



Architecture of the email system

# Architecture and Services (2)



Mr. Daniel Dumkopf
18 Willow Lane
White Plains, NY 10604

United Gizmo
180 Main St
Boston, MA 02120
Sept. 1, 2010

Subject: Invoice 1081

Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.

Yours truly
United Gizmo

Name: Mr. Daniel Dumkopf
Street: 18 Willow Lane
City: White Plains
State: NY
Zip code: 10604
Priority: Urgent
Encryption: None

From: United Gizmo
Address: 180 Main St.
Location: Boston, MA 02120
Date: Sept. 1, 2010
Subject: Invoice 1081

Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.

Yours truly
United Gizmo

Envelope

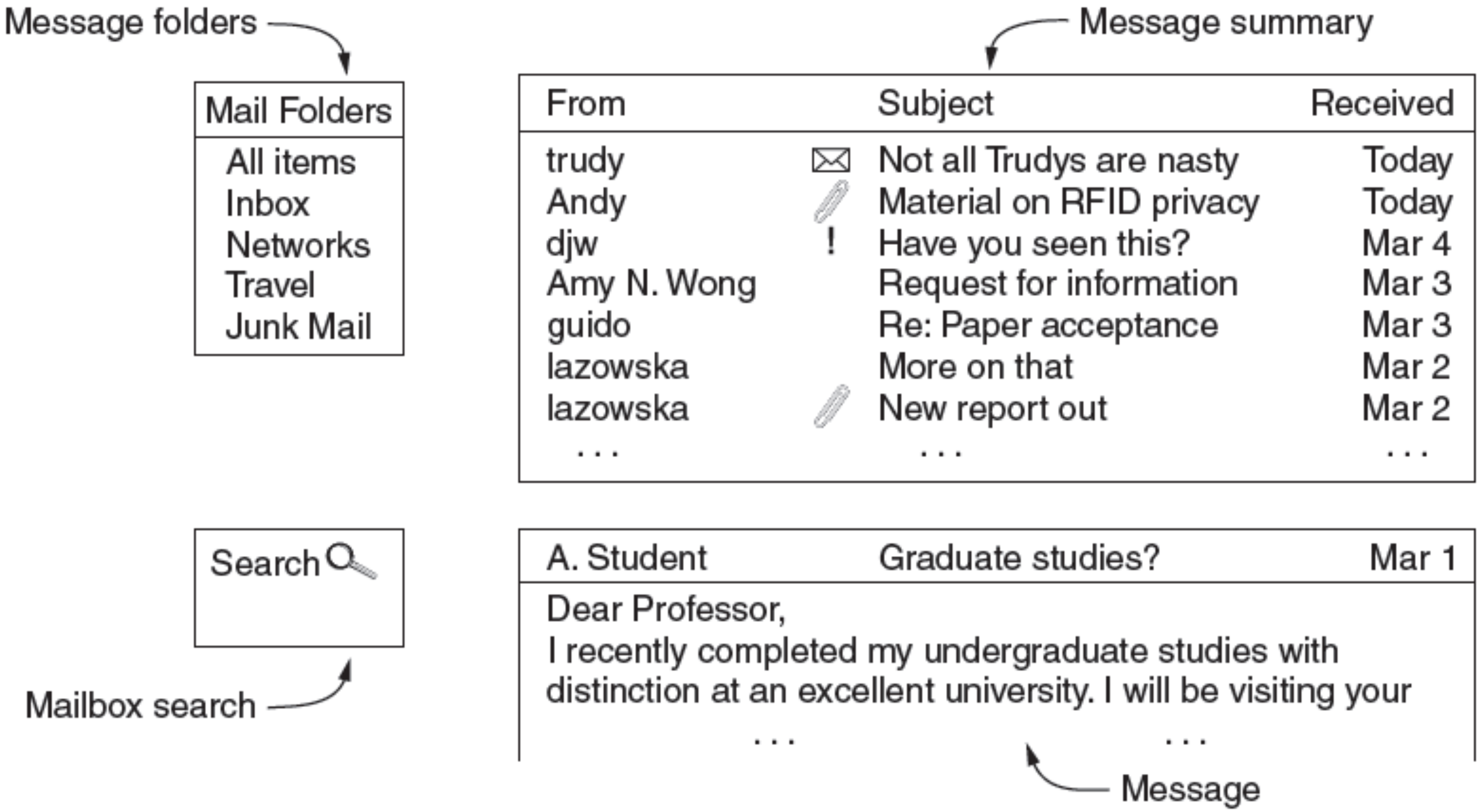
Message
(= header
and body)

Paper mail

Electronic mail

# The User Agent

What users see – interface elements of a typical user agent



Mail Folders

- All items
- Inbox
- Networks
- Travel
- Junk Mail

| From | | Subject | Received |
|---|---|---|---|
| trudy | ✉ | Not all Trudys are nasty | Today |
| Andy | 📎 | Material on RFID privacy | Today |
| djw | ! | Have you seen this? | Mar 4 |
| Amy N. Wong | | Request for information | Mar 3 |
| guido | | Re: Paper acceptance | Mar 3 |
| lazowska | | More on that | Mar 2 |
| lazowska | 📎 | New report out | Mar 2 |
| . . . | | . . . | . . . |

Search

| A. Student | Graduate studies? | Mar 1 |
|---|---|---|

Dear Professor,
I recently completed my undergraduate studies with distinction at an excellent university. I will be visiting your
. . . . . .

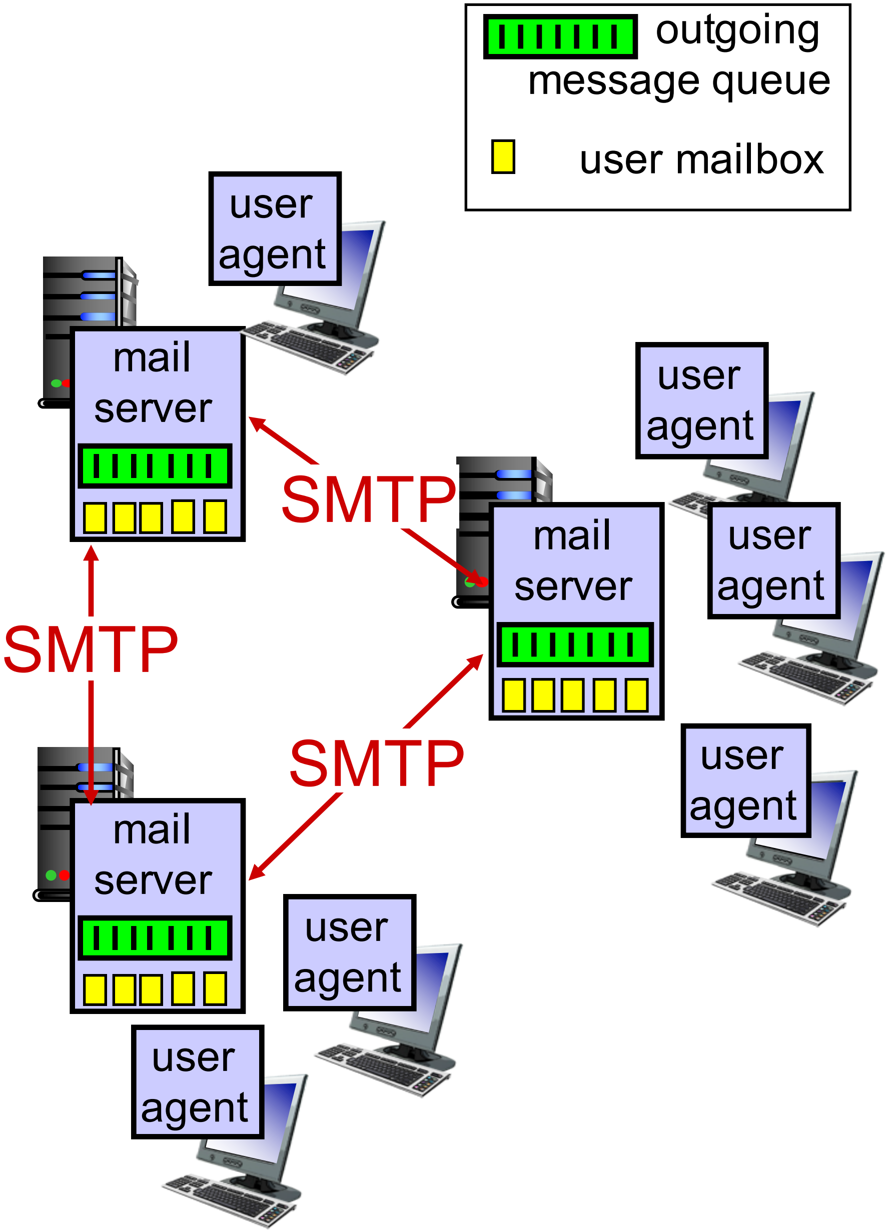
# Electronic mail

## *Three major components:*

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

## *User Agent*

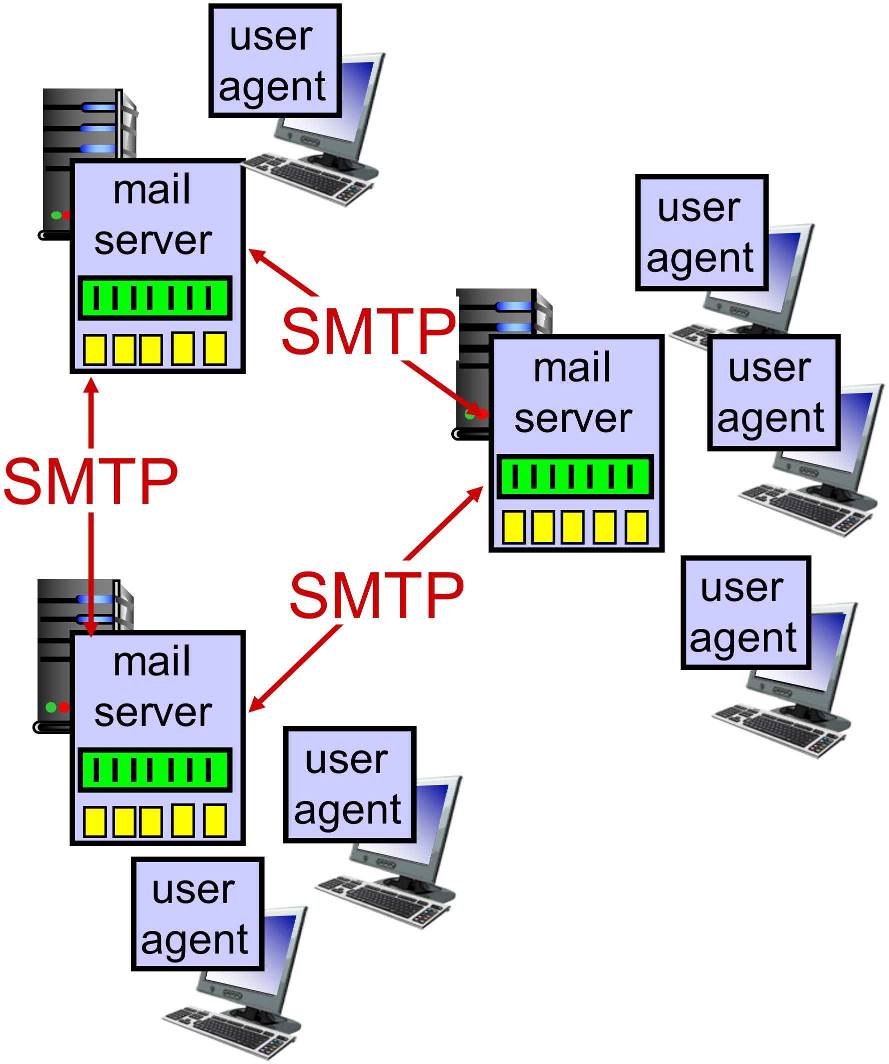
- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server



# Electronic mail: 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 ASCI

# Scenario: Alice sends message to Bob

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

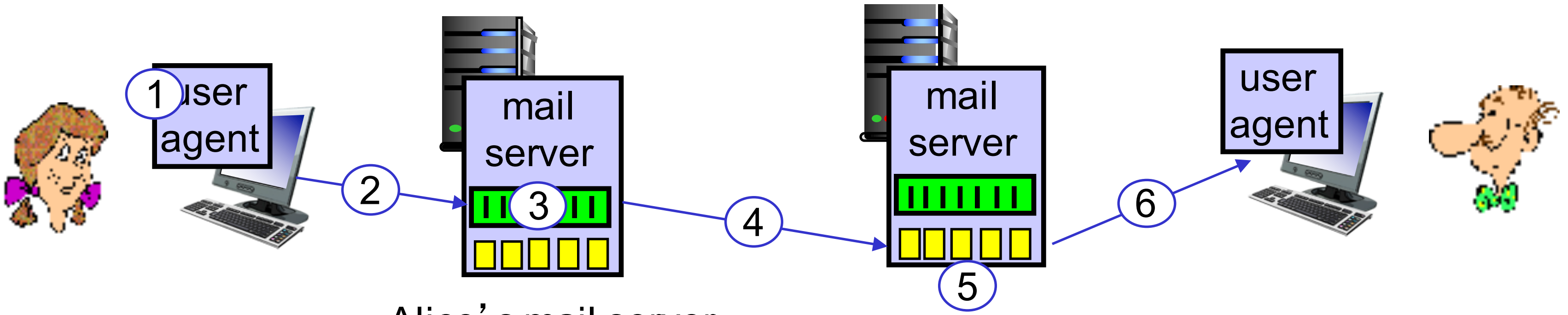
2) Alice's UA sends message to her mail server; message placed in message queue

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



Alice's mail server

Bob's mail server

Carole Delporte

# 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)

# Sample SMTP interaction

```
$ telnet smtp-auth.sfr.fr 587
Trying 93.17.128.23...
Connected to smtp-auth.sfr.fr.
Escape character is '^]'.
220 msfrf2308.sfr.fr ESMTP ABO **************************
HELO sfr.fr
250 msfrf2308.sfr.fr
MAIL FROM:<noel@sfr.fr>
250 2.1.0 Ok
RCPT TO: <cd@liafa.univ-paris-diderot.fr>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
BLABLA blabla
Et encore
.
250 2.0.0 Ok: queued as C25897000079
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
$
```

# 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

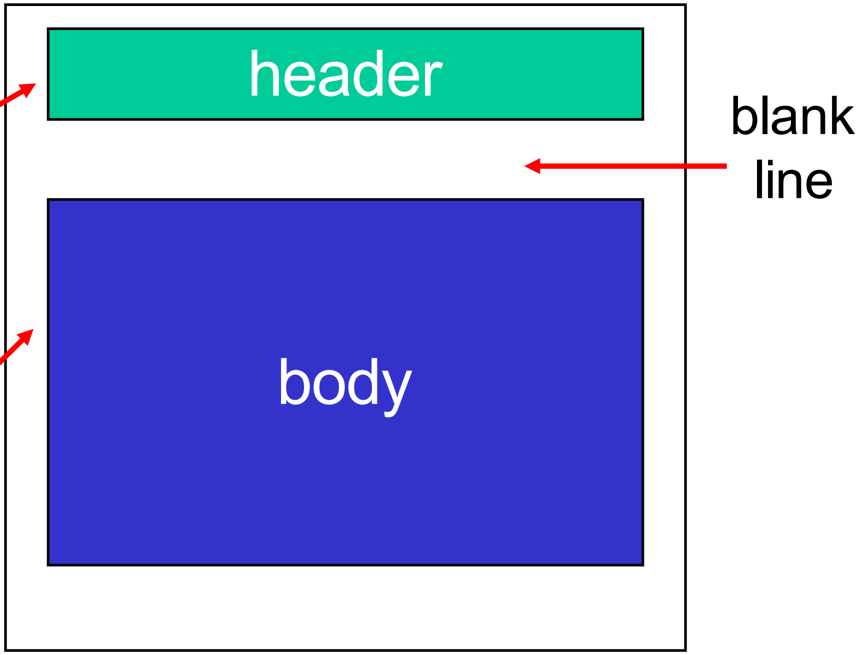
# Mail message format

SMTP: protocol for exchanging email msgs

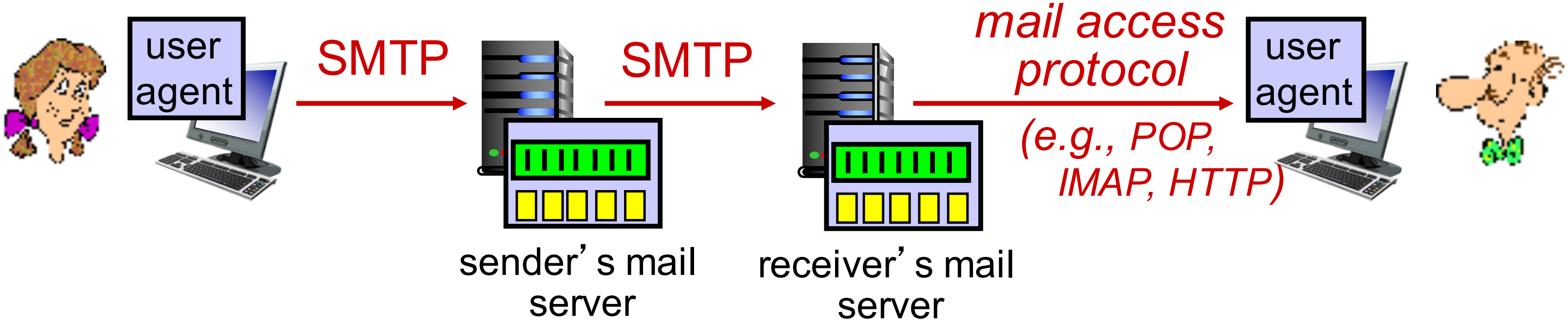
RFC 822: standard for text message format:

- 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

Port 110

*authorization phase*

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

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

*transaction phase,* client:

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

```
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
```

```
$ telnet nivose.informatique.univ-paris-diderot.fr 110
Trying 194.254.199.73...
Connected to nivose.informatique.univ-paris-diderot.fr.
Escape character is '^]'.
+OK Qpopper (version 4.1b18) at nivose starting.
user cd
+OK Password required for cd.
Pass ENCLAIR
+OK cd has 41 visible messages (0 hidden) in 397421 octets.
list
+OK 41 visible messages (397421 octets)
1 33935
…..
41 13037
.
quit
+OK Pop server at nivose signing off.
Connection closed by foreign host.
$
```

# 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

# envoiMailSimple.java

```
import java.util.*;
import javax.mail.Address;
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
// les 2 classes suivantes sont  utiles pour  le courrier electronique Internet
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class envoiMailSimple {
 public static void main(String[] args) {
  try {
  // emetteur du message (MAIL FROM:)
   Address emetteur = new InternetAddress("papi@dugrandnord.com",
   "Pere Noel");
// recepteur du message (RCPT TO:)
   Address receveur = new
InternetAddress("etudiant@informatique.univ-paris-diderot.fr");

  // positionnement de la propriete mail.host au serveur local
   Properties props = new Properties();
   props.put("mail.host", "ouindose.informatique.univ-paris-
diderot.fr");

  // demarrage d'une session de courrier
   Session mailConnection = Session.getInstance(props, null);

  // Construction du message rnvoyer par Internet
   Message msg = new MimeMessage(mailConnection);
   msg.setFrom(emetteur);
   msg.setRecipient(Message.RecipientType.TO, receveur);
   msg.setSubject("Bientot Noel");
   msg.setContent(" M'as tu envoye ta commande?\n j'attends",
    "text/plain");

   //Emission du message
   Transport.send(msg);
  }
  catch (Exception ex) {
   ex.printStackTrace();
  }   }}
```

# pop3Client.java

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;
import java.io.*;

public class pop3Client {
  public static void main(String[] args) {
    Properties props = new Properties();

    String host = "ouindose.informatique.univ-paris-diderot.fr";
    String username= "cd";
    String password ="enclair";
    String protocol = "pop3";

    try {
     Session session = Session.getDefaultInstance(props,null);
      Store store = session.getStore(protocol);
       store.connect(host, username, password);
      System.out.println("connection reussi");
```

# pop3Client.java

```
// Open the folder
      Folder inbox = store.getFolder("INBOX");
      if (inbox == null) {
        System.out.println("No INBOX");
        System.exit(1);
      }
      inbox.open(Folder.READ_ONLY);

 //lecture des messages
Message[] messages = inbox.getMessages();
      for (int i = 0; i < messages.length; i++) {
        System.out.println("----------– Message " + (i+1)
         + " ------------");
        messages[i].writeTo(System.out);
                 }
      inbox.close(false);

    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
    System.exit(0);    }}
```

Saisir le mot de passe:

```
Session session = Session.getDefaultInstance(props,new
MailAuthenticator("cd"));
```

Avec class MailAuthenticator qui étend la classe Authenticator

# MailAuthenticator.java

```
import javax.mail.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MailAuthenticator extends Authenticator {

  private JDialog passwordDialog = new JDialog(new JFrame(), true);
  private JLabel passwordLabel = new JLabel("Password: ");
  private String username;
  private JPasswordField passwordField = new JPasswordField(20);
  private JButton okButton = new JButton("OK");

  public MailAuthenticator(String u) {
    username = new String(u);
    Container pane = passwordDialog.getContentPane();
    pane.setLayout(new GridLayout(2, 1));
    JPanel p = new JPanel();
    JPanel p = new JPanel();
    p.add(passwordLabel);
    p.add(passwordField);
     p.add(okButton);
    pane.add(p);
    passwordDialog.pack();
    ActionListener al = new HideDialog();
    okButton.addActionListener(al);
    passwordField.addActionListener(al);
  }
  class HideDialog implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      passwordDialog.hide();
    }
  }
  public PasswordAuthentication getPasswordAuthentication() {
    passwordDialog.show();
    String password = new String(passwordField.getPassword());
    passwordField.setText("");
    return new PasswordAuthentication(username, password);
  }
}
```

# Récuperation des champs du message

```
Message[] messages = inbox.getMessages();
   for (int i = 0; i < messages.length; i++) {
     String from = InternetAddress.toString(messages[i].getFrom());
     if (from != null) System.out.println("From: " + from);
     String replyTo = InternetAddress.toString(
      messages[i].getReplyTo());
     if (replyTo != null) System.out.println("Reply-to: "
      + replyTo);
     String to = InternetAddress.toString(
      messages[i].getRecipients(Message.RecipientType.TO));
     if (to != null) System.out.println("To: " + to);
     String cc = InternetAddress.toString(
     messages[i].getRecipients(Message.RecipientType.CC));
     if (cc != null) System.out.println("Cc: " + cc);
```

```
 String subject = messages[i].getSubject();
  if (subject != null) System.out.println("Subject: " + subject);

  Date sent = messages[i].getSentDate();
  if (sent != null) System.out.println("Sent: " + sent);

  Date received = messages[i].getReceivedDate();
  if (received != null) System.out.println("Received: " + received);

  System.out.println();
}
```

# Couche application

## DNS

Carole Delporte

# 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

*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"

Carole Delporte

# 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?*

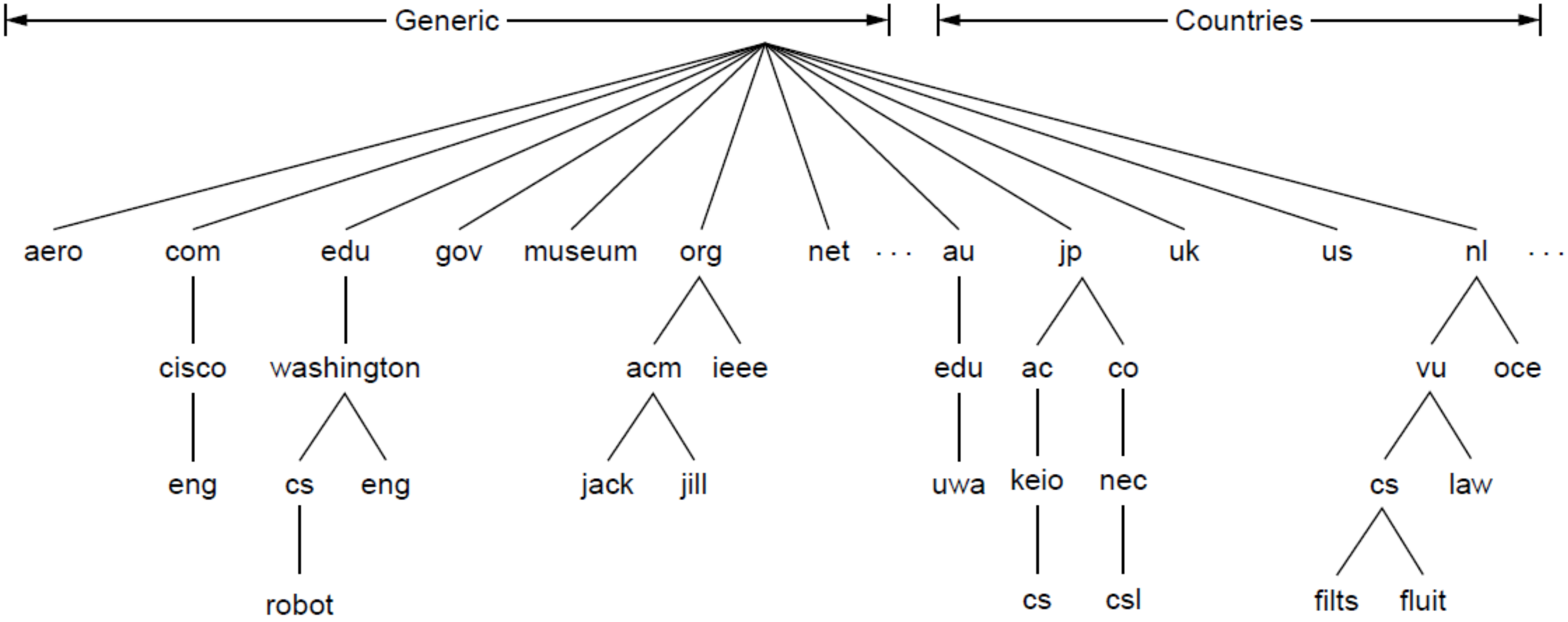
- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: *doesn't scale!*

# DNS Name space

DNS namespace is hierarchical from the root down

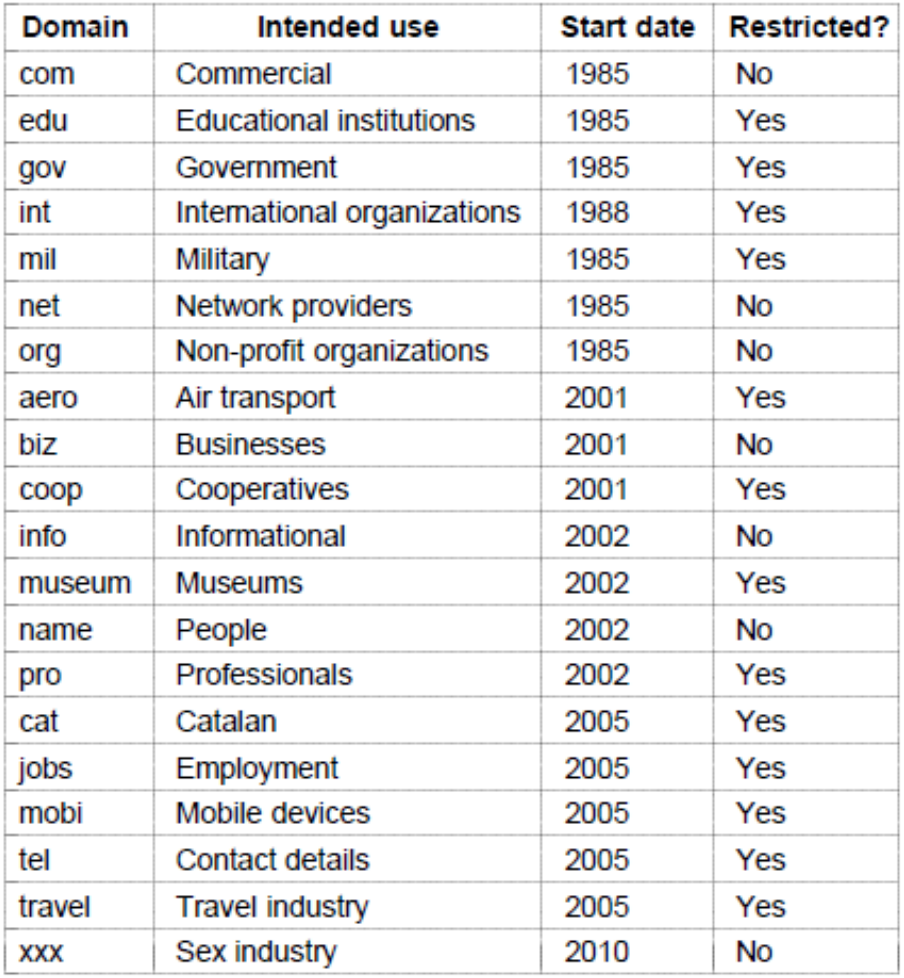
- Different parts delegated to different organizations



The computer *robot.cs.washington.edu*

# DNS Name Space

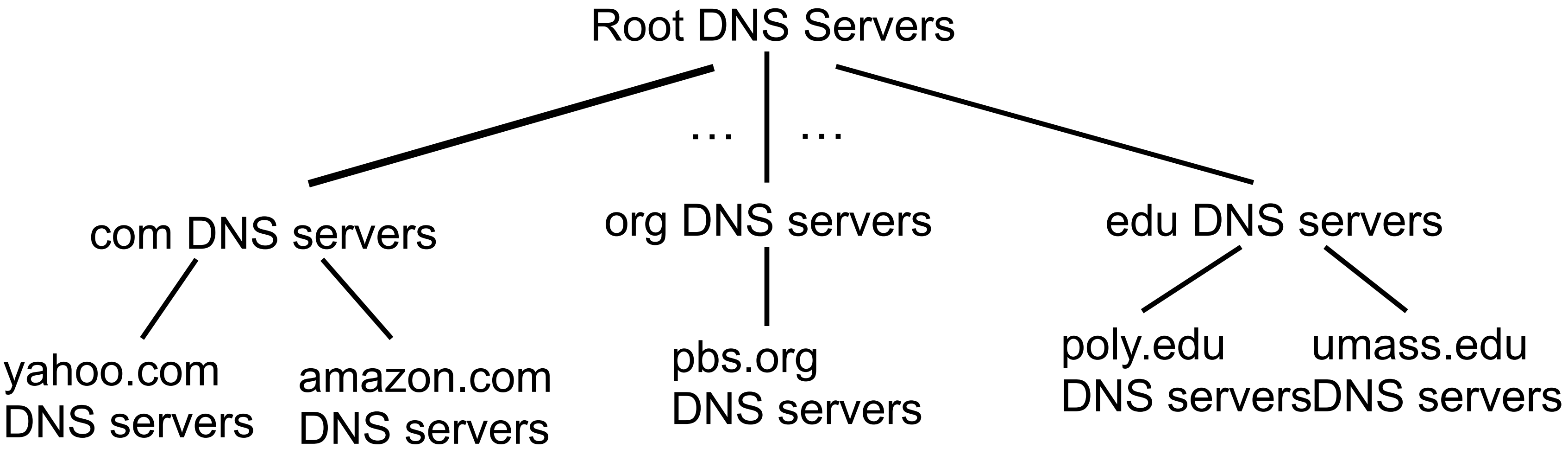
Generic top-level domains are controlled by ICANN who appoints registrars to run them



| Domain | Intended use | Start date | Restricted? |
|---|---|---|---|
| com | Commercial | 1985 | No |
| edu | Educational institutions | 1985 | Yes |
| gov | Government | 1985 | Yes |
| int | International organizations | 1988 | Yes |
| mil | Military | 1985 | Yes |
| net | Network providers | 1985 | No |
| org | Non-profit organizations | 1985 | No |
| aero | Air transport | 2001 | Yes |
| biz | Businesses | 2001 | No |
| coop | Cooperatives | 2001 | Yes |
| info | Informational | 2002 | No |
| museum | Museums | 2002 | Yes |
| name | People | 2002 | No |
| pro | Professionals | 2002 | Yes |
| cat | Catalan | 2005 | Yes |
| jobs | Employment | 2005 | Yes |
| mobi | Mobile devices | 2005 | Yes |
| tel | Contact details | 2005 | Yes |
| travel | Travel industry | 2005 | Yes |
| xxx | Sex industry | 2010 | No |

This one was controversial →

# DNS: a distributed, hierarchical database

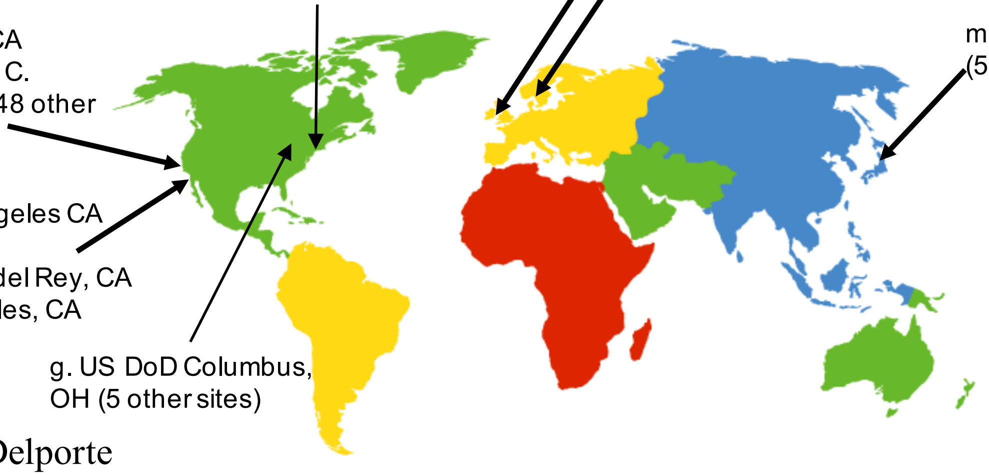


*client wants IP for www.amazon.com; 1st approx:*

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



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)

e. NASA Mt View, CA
f. Internet Software C.
Palo Alto, CA (and 48 other sites)

m. WIDE Tokyo
(5 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 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
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD
- .fr Association Française pour le Nommage Internet en Coopération (A.F.N.I.C.)

*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

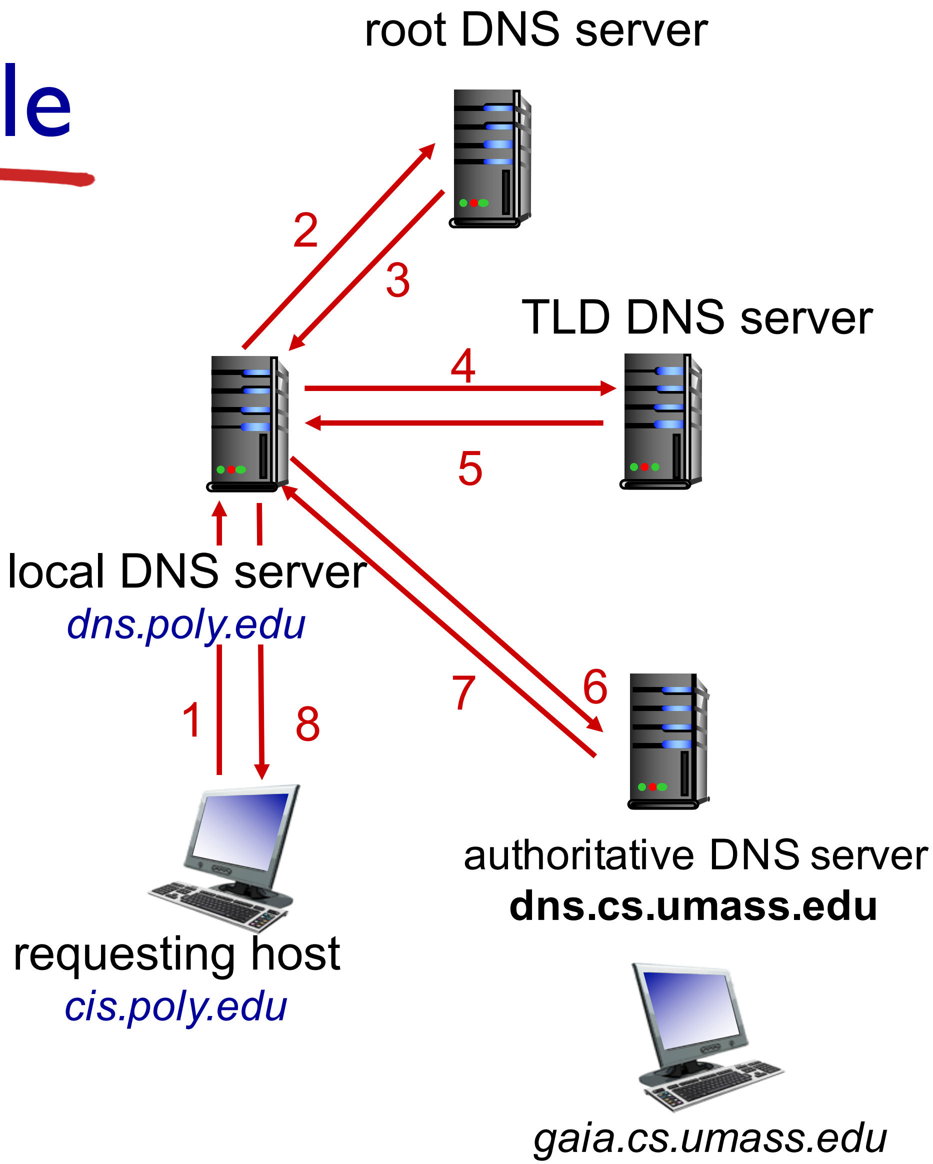
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts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

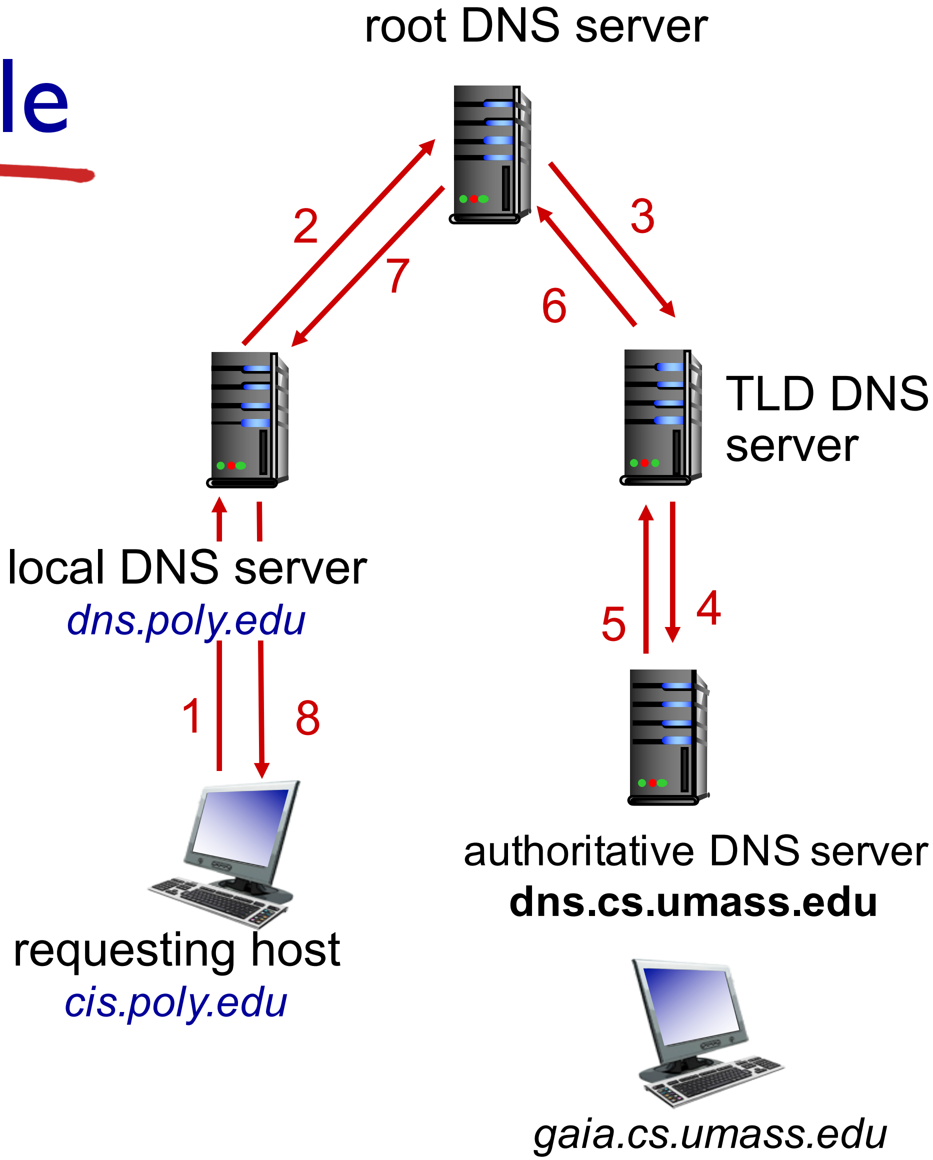
- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



# 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

## 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`**

# Enregistrements DNS

- nslookup
- dig

```
dig www.google.com

; <<>> DiG 9.8.3-P1 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30123
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         181             IN      A       173.194.65.103
www.google.com.         181             IN      A       173.194.65.147
www.google.com.         181             IN      A       173.194.65.104
www.google.com.         181             IN      A       173.194.65.105
www.google.com.         181             IN      A       173.194.65.106
www.google.com.         181             IN      A       173.194.65.99

;; Query time: 5 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Nov  6 22:24:54 2014
;; MSG SIZE  rcvd: 128
```

```
$ dig MX gmail.com

; <<>> DiG 9.8.3-P1 <<>> MX gmail.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4224
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;gmail.com.                         IN      MX

;; ANSWER SECTION:
gmail.com.                2614      IN      MX      20 alt2.gmail-smtp-in.l.google.com.
gmail.com.                2614      IN      MX      40 alt4.gmail-smtp-in.l.google.com.
gmail.com.                2614      IN      MX      30 alt3.gmail-smtp-in.l.google.com.
gmail.com.                2614      IN      MX      10 alt1.gmail-smtp-in.l.google.com.
gmail.com.                2614      IN      MX      5 gmail-smtp-in.l.google.com.

;; Query time: 17 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Nov  6 22:26:22 2014
;; MSG SIZE  rcvd: 150
```

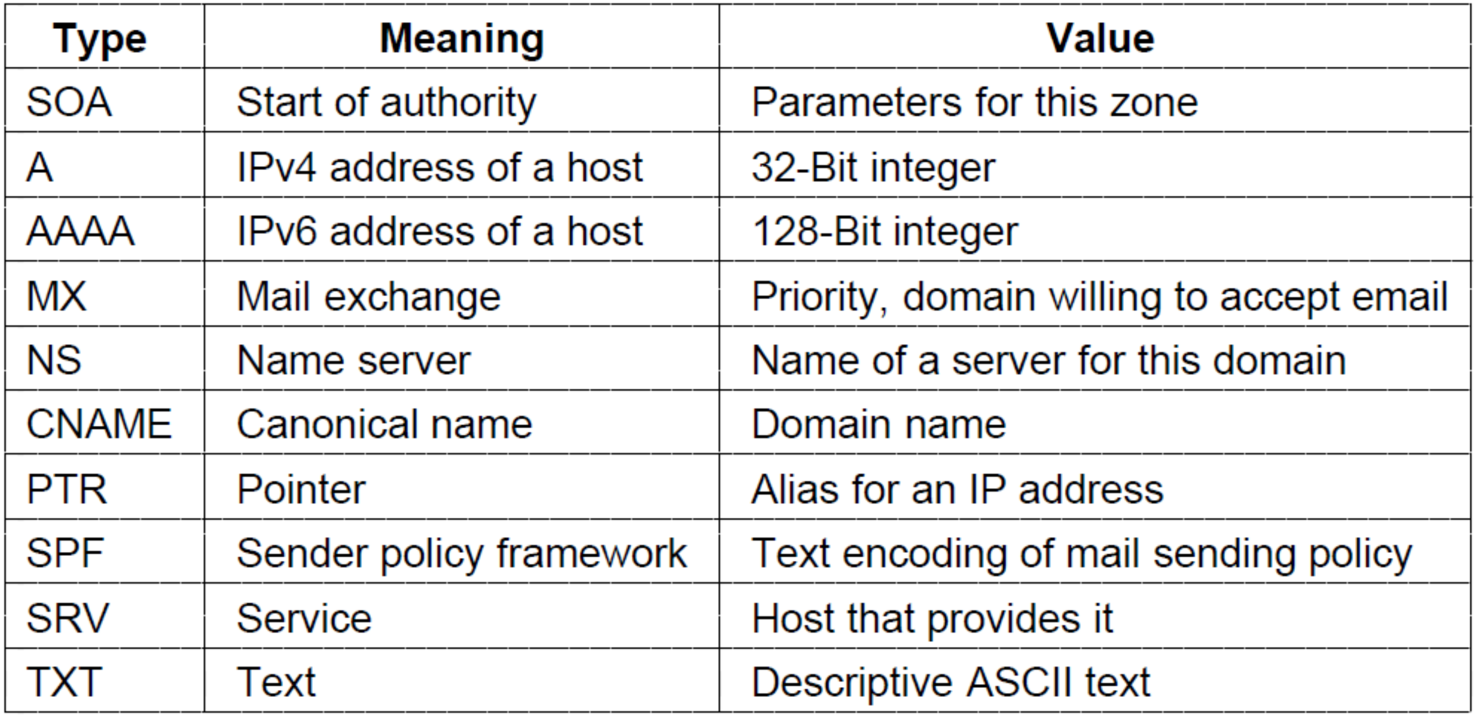
```
$dig au.edu
; <<>> DiG 9.7.2-P2 <<>> au.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36394
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;au.edu.                        IN     A

;; ANSWER SECTION:
au.edu.             10800   IN     A       168.120.16.231

;; AUTHORITY SECTION:
au.edu.             10800   IN     NS      abac.au.ac.th.
au.edu.             10800   IN     NS      ksc.au.ac.th.

…..
```

```
 $dig www.ibm.com

; <<>> DiG 9.7.2-P2 <<>> www.ibm.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21972
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 8, ADDITIONAL: 8

;; QUESTION SECTION:
;www.ibm.com.                  IN     A

;; ANSWER SECTION:
www.ibm.com.            3600    IN      CNAME   www.ibm.com.cs186.net.
www.ibm.com.cs186.net.  60      IN      CNAME   www.ibm.com.edgekey.net.
www.ibm.com.edgekey.net. 300    IN      CNAME   e3062.x.akamaiedge.net.
e3062.x.akamaiedge.net. 19      IN      A       23.223.231.66
```

```
;; AUTHORITY SECTION:
x.akamaiedge.net.       2696    IN      NS      n3x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      n5x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      a0x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      n1x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      n4x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      n0x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      a1x.akamaiedge.net.
x.akamaiedge.net.       2696    IN      NS      n2x.akamaiedge.net.

;; ADDITIONAL SECTION:
a0x.akamaiedge.net.     648     IN      AAAA    2a02:26f0:32:f000:f508:905:cbfb:3
a1x.akamaiedge.net.     192     IN      AAAA    2a02:26f0:32:f000:f508:4b39:89c7
n0x.akamaiedge.net.     1221    IN      A       217.212.239.56
…..
```

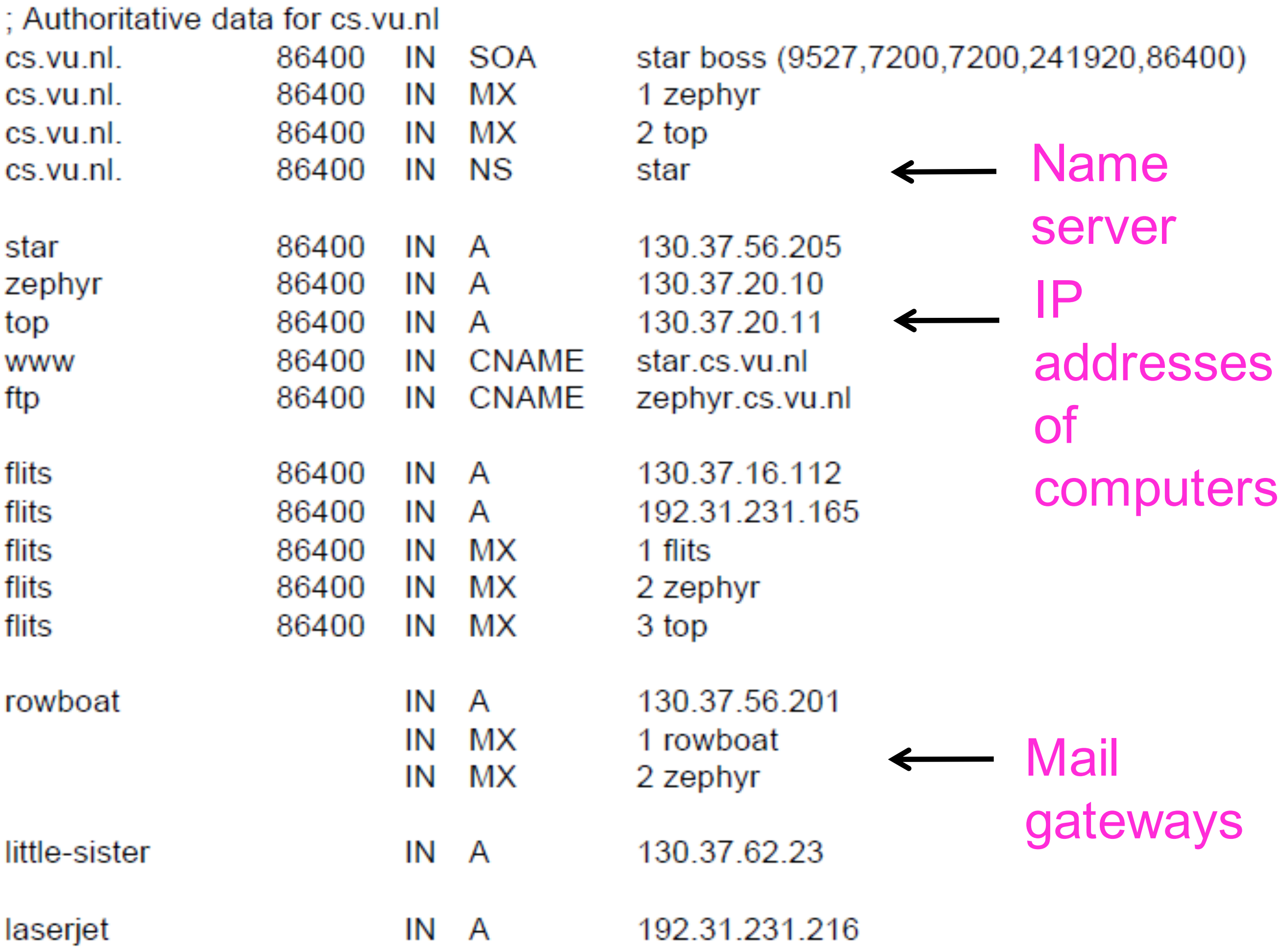
# Domain Resource Records

The key resource records in the namespace are IP addresses (A/AAAA) and name servers (NS), but there are others too (e.g., MX)



| Type | Meaning | Value |
|---|---|---|
| SOA | Start of authority | Parameters for this zone |
| A | IPv4 address of a host | 32-Bit integer |
| AAAA | IPv6 address of a host | 128-Bit integer |
| MX | Mail exchange | Priority, domain willing to accept email |
| NS | Name server | Name of a server for this domain |
| CNAME | Canonical name | Domain name |
| PTR | Pointer | Alias for an IP address |
| SPF | Sender policy framework | Text encoding of mail sending policy |
| SRV | Service | Host that provides it |
| TXT | Text | Descriptive ASCII text |

# Domain Resource Records



```
; Authoritative data for cs.vu.nl
cs.vu.nl.        86400   IN   SOA     star boss (9527,7200,7200,241920,86400)
cs.vu.nl.        86400   IN   MX      1 zephyr
cs.vu.nl.        86400   IN   MX      2 top
cs.vu.nl.        86400   IN   NS      star

star             86400   IN   A       130.37.56.205
zephyr           86400   IN   A       130.37.20.10
top              86400   IN   A       130.37.20.11
www              86400   IN   CNAME   star.cs.vu.nl
ftp              86400   IN   CNAME   zephyr.cs.vu.nl

flits            86400   IN   A       130.37.16.112
flits            86400   IN   A       192.31.231.165
flits            86400   IN   MX      1 flits
flits            86400   IN   MX      2 zephyr
flits            86400   IN   MX      3 top

rowboat                  IN   A       130.37.56.201
                         IN   MX      1 rowboat
                         IN   MX      2 zephyr

little-sister            IN   A       130.37.62.23

laserjet                 IN   A       192.31.231.216
```

Name server

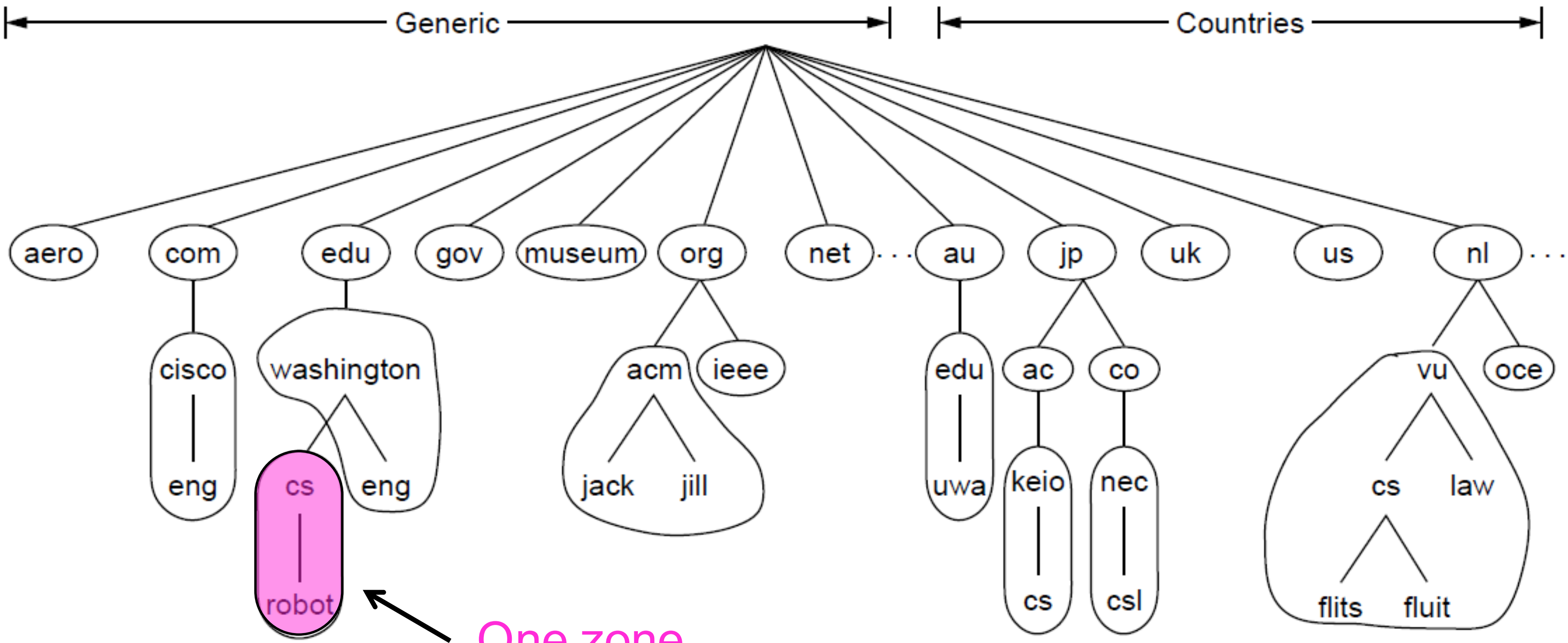
IP addresses of computers

Mail gateways

- A portion of a possible DNS database for cs.vu.nl.

# Name Servers

Name servers contain data for portions of the name space called zones (circled).



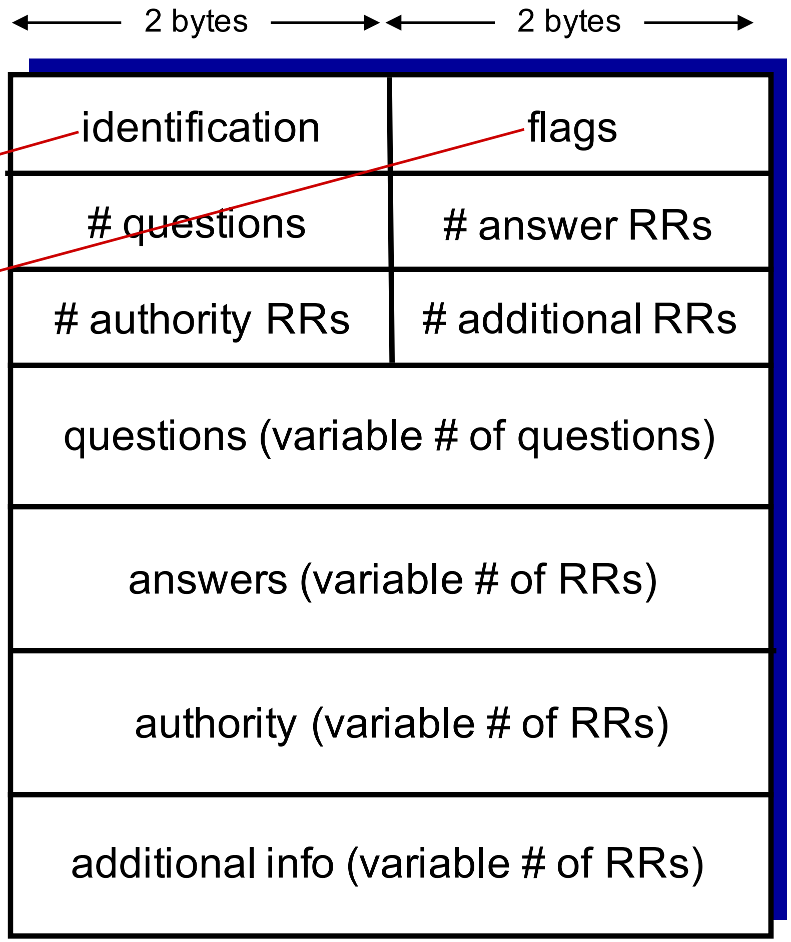
One zone

# DNS protocol, messages

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

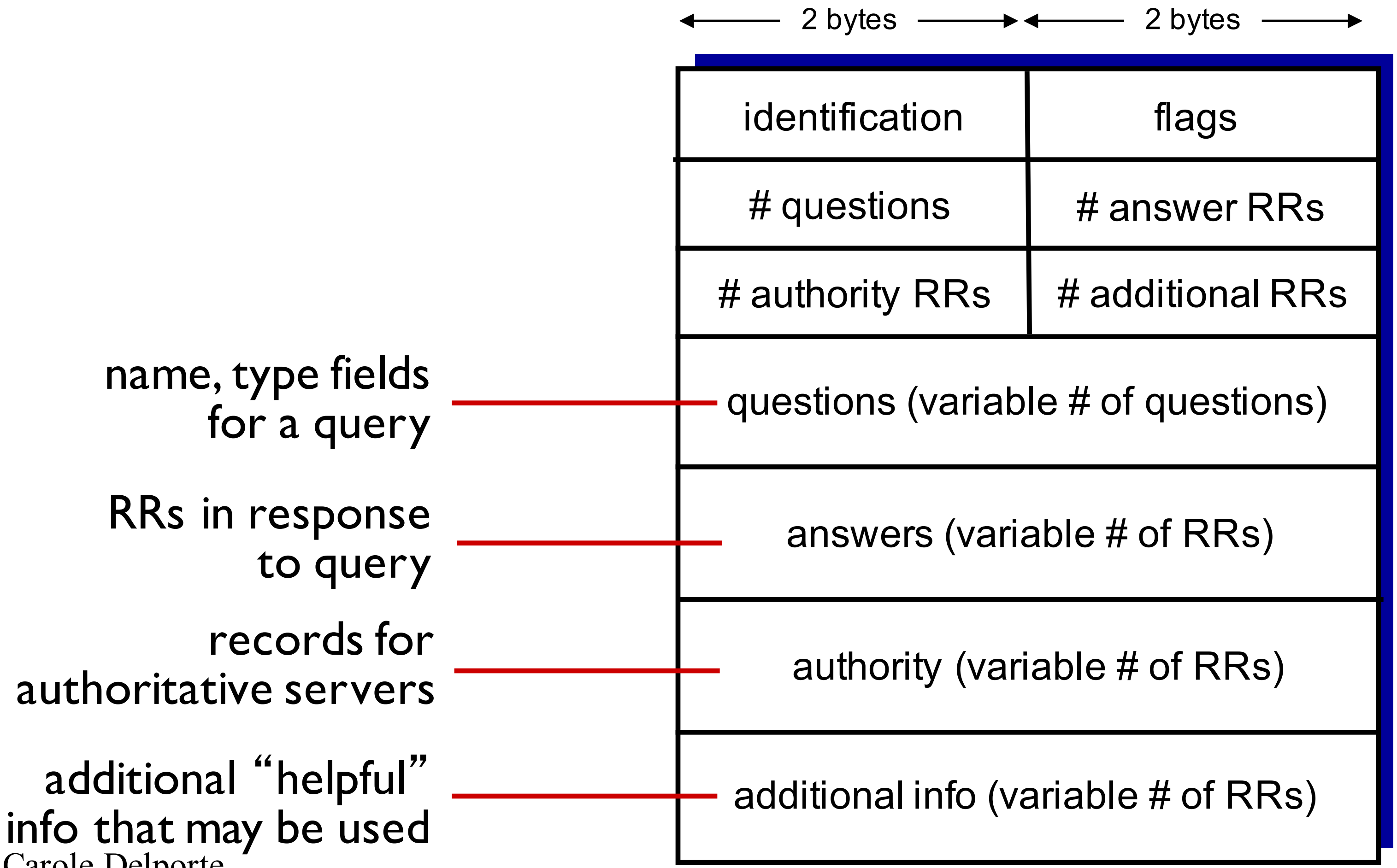
msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



| 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 protocol, messages



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)
  - 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

# 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