# DNS (con't) & Email

**EE 122: Intro to Communication Networks**

Fall 2006 (MW 4-5:30 in Donner 155)

Vern Paxson

TAs: Dilip Antony Joseph and Sukun Kim

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, and colleagues at Princeton and UC Berkeley



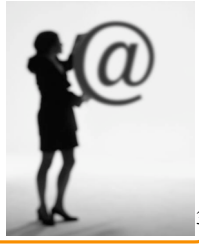
## Announcements

- Homework #1 solutions are out.
  - See announcement page for URL.
- Your graded homework #1 will be returned in your discussion section
  - Except for a few that are still pending (e.g., possible copying)
- FTP to be covered in Section rather than lecture
  - Starting Friday

## Goals of Today's Lecture

- Finish Discussion of DNS (Domain Name System)
  - Iterative and recursive queries
  - TTL-based caching
  - Use of the `dig` utility
  - Security analysis

- A look at an application: Email
  - Actually involves 4 application-layer protocols
  - Simple Mail Transfer Protocol (SMTP)
    - Does the work of moving email around
  - Various protocols used to read email
    - Post Office Protocol (POP)
    - IMAP
    - Hypertext Transfer Protocol (HTTP)



3

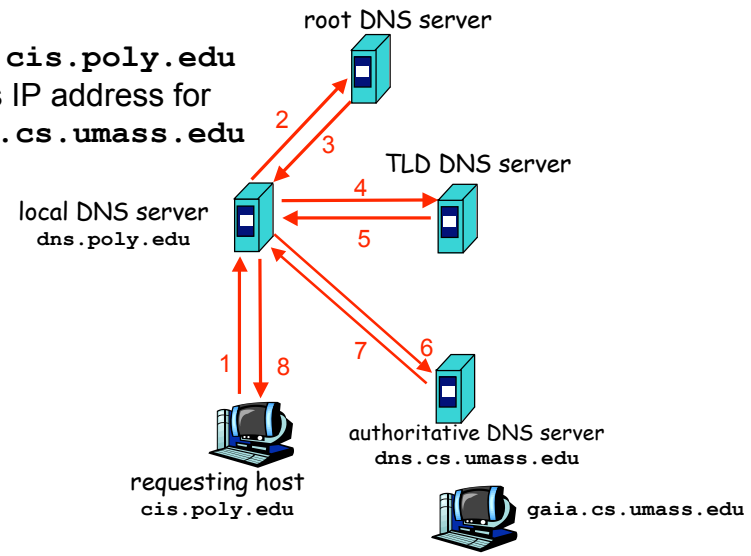
## Using DNS

- Local DNS server ("default name server")
  - Usually near the endhosts that use it
  - Local hosts configured with local server (e.g., `/etc/resolv.conf`) or learn server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do *gethostbyname()* to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional *gethostbyaddr()* to translate into name

4

## Example

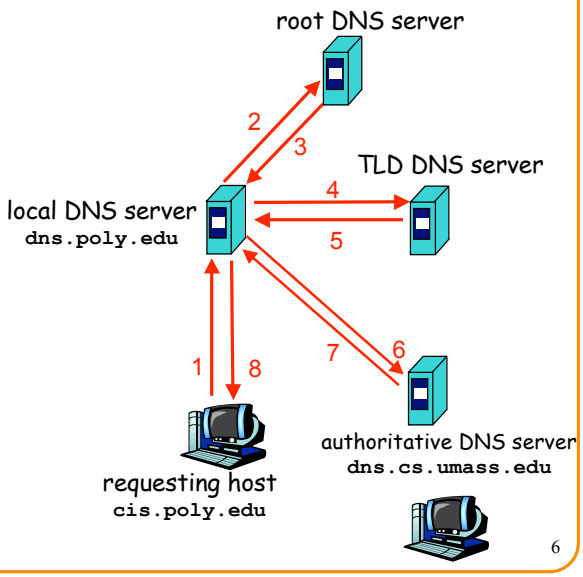
Host at **cis.poly.edu** wants IP address for **gaia.cs.umass.edu**




5

## Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs



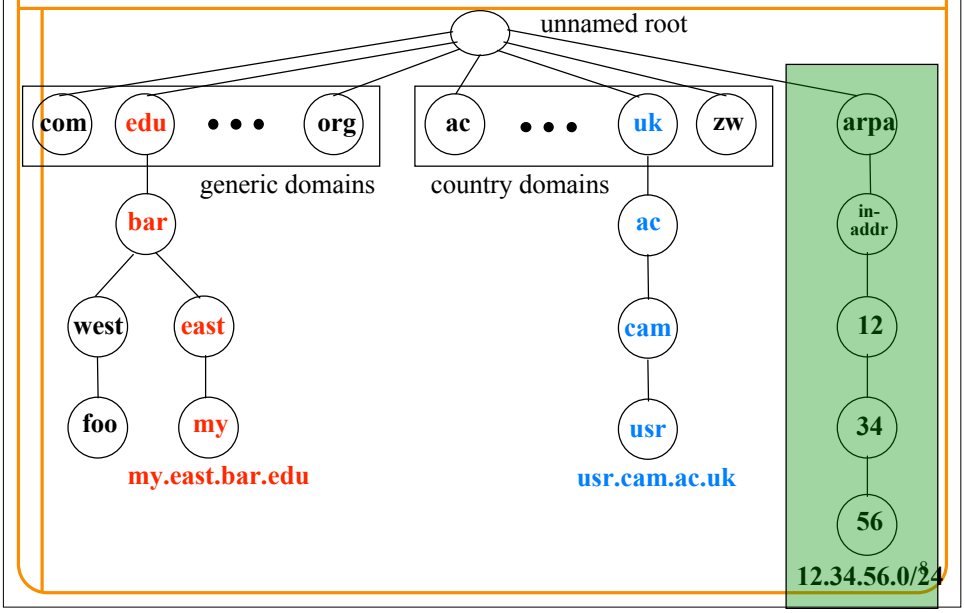

6

## Reverse Mapping (Address → Host)

- How do we go the other direction, from an IP address to the corresponding hostname?
- Addresses already have natural "quad" hierarchy:
  - 12.34.56.78
- But: quad notation has most-sig. hierarchy element on left, while www.cnn.com has it on the right
- Idea: reverse the quads = 78.56.34.12 …
  - … and look that up in the DNS
- Under what TLD?
  - Convention: **in-addr.arpa**
  - So lookup is for `78.56.34.12.in-addr.arpa`

7

## Distributed Hierarchical Database




8

## DNS Caching

- Performing all these queries takes time
  - And all this before actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can greatly reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes cached entry after TTL expires

9

## DNS Resource 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=PTR
  - **`name`** is reversed IP quads
    - E.g. 78.56.34.12.in-addr.arpa
  - **`value`** is corresponding hostname
- Type=CNAME
  - **`name`** is alias name for some "canonical" name

    E.g., **`www.cs.mit.edu`** is really **`eecsweb.mit.edu`**
  - **`value`** is canonical name
- Type=MX
  - **`value`** is name of mailserver associated with **`name`**
  - Also includes a weight/preference

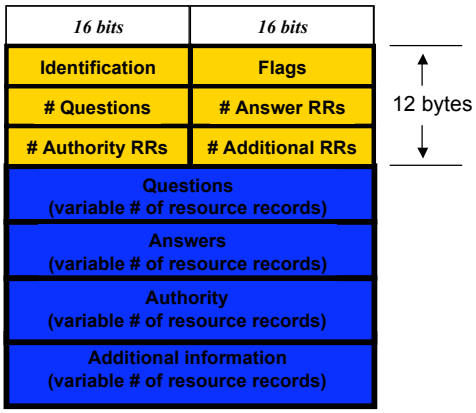
10

## DNS Protocol

**DNS protocol**: *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




11

## Interactive DNS lookups using `dig`

- `dig` program on Unix
  - Allows querying of DNS system
  - Dumps each field in DNS responses
  - By default, executes recursive queries
    - Disable via `+norecurse` so that operates one step at a time

12

```
unix> dig +norecurse @a.root-servers.net www.cnn.com

; <<>> DiG 9.2.2 <<>> +norecurse @a.root-servers.net www.cnn.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21041
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14

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

;; AUTHORITY SECTION:
com.                    172800  IN      NS      A.GTLD-SERVERS.NET.
com.                    172800  IN      NS      G.GTLD-SERVERS.NET.
com.                    172800  IN      NS      H.GTLD-SERVERS.NET.
com.                    172800  IN      NS      C.GTLD-SERVERS.NET.
com.                    172800  IN      NS      I.GTLD-SERVERS.NET.
com.                    172800  IN      NS      B.GTLD-SERVERS.NET.
com.                    172800  IN      NS      D.GTLD-SERVERS.NET.
com.                    172800  IN      NS      L.GTLD-SERVERS.NET.
com.                    172800  IN      NS      F.GTLD-SERVERS.NET.
com.                    172800  IN      NS      J.GTLD-SERVERS.NET.
com.                    172800  IN      NS      K.GTLD-SERVERS.NET.
com.                    172800  IN      NS      E.GTLD-SERVERS.NET.
com.                    172800  IN      NS      M.GTLD-SERVERS.NET.
```

Note, no "**ANSWER**" section

```
;; ADDITIONAL SECTION:
A.GTLD-SERVERS.NET.     172800  IN      AAAA    2001:503:a83e::2:30
A.GTLD-SERVERS.NET.     172800  IN      A       192.5.6.30
G.GTLD-SERVERS.NET.     172800  IN      A       192.42.93.30
H.GTLD-SERVERS.NET.     172800  IN      A       192.54.112.30
C.GTLD-SERVERS.NET.     172800  IN      A       192.26.92.30
I.GTLD-SERVERS.NET.     172800  IN      A       192.43.172.30
B.GTLD-SERVERS.NET.     172800  IN      AAAA    2001:503:231d::2:30
B.GTLD-SERVERS.NET.     172800  IN      A       192.33.14.30
D.GTLD-SERVERS.NET.     172800  IN      A       192.31.80.30
L.GTLD-SERVERS.NET.     172800  IN      A       192.41.162.30
F.GTLD-SERVERS.NET.     172800  IN      A       192.35.51.30
J.GTLD-SERVERS.NET.     172800  IN      A       192.48.79.30
K.GTLD-SERVERS.NET.     172800  IN      A       192.52.178.30
E.GTLD-SERVERS.NET.     172800  IN      A       192.12.94.30

;; Query time: 117 msec
;; SERVER: 198.41.0.4#53(a.root-servers.net)
;; WHEN: Mon Sep 25 11:13:15 2006
;; MSG SIZE  rcvd: 501
```

**dig +norecurse @g.gtld-servers.net www.cnn.com**

```
; <<>> DiG 9.2.4 <<>> +norecurse @g.gtld-servers.net www.cnn.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 74
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 4

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

;; AUTHORITY SECTION:
cnn.com.                172800  IN      NS      twdns-01.ns.aol.com.
cnn.com.                172800  IN      NS      twdns-02.ns.aol.com.
cnn.com.                172800  IN      NS      twdns-03.ns.aol.com.
cnn.com.                172800  IN      NS      twdns-04.ns.aol.com.

;; ADDITIONAL SECTION:
twdns-01.ns.aol.com.    172800  IN      A       149.174.213.151
twdns-02.ns.aol.com.    172800  IN      A       152.163.239.216
twdns-03.ns.aol.com.    172800  IN      A       207.200.73.85
twdns-04.ns.aol.com.    172800  IN      A       64.12.147.120
```

**dig +norecurse @twdns-01.ns.aol.com www.cnn.com**

```
…
;; QUESTION SECTION:
;www.cnn.com.                   IN      A

;; ANSWER SECTION:
www.cnn.com.            300     IN      CNAME   cnn.com.
cnn.com.                300     IN      A       64.236.24.12
cnn.com.                300     IN      A       64.236.24.20
cnn.com.                300     IN      A       64.236.24.28
cnn.com.                300     IN      A       64.236.29.120
cnn.com.                300     IN      A       64.236.16.20
cnn.com.                300     IN      A       64.236.16.52
cnn.com.                300     IN      A       64.236.16.84
cnn.com.                300     IN      A       64.236.16.116

;; AUTHORITY SECTION:
cnn.com.                600     IN      NS      twdns-02.ns.aol.com.
cnn.com.                600     IN      NS      twdns-03.ns.aol.com.
cnn.com.                600     IN      NS      twdns-04.ns.aol.com.
cnn.com.                600     IN      NS      twdns-01.ns.aol.com.
```

## Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas
- Usually, UDP used for queries
  - Need reliability: must implement this on top of UDP
  - Spec supports TCP too, but not always implemented
- Try alternate servers on timeout
  - Exponential backoff when retrying same server

17

## Inserting Resource Records into DNS

- Example: just created startup "FooBar"
- Register `foobar.com` at Network Solutions (say)
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RR pairs into the `com` TLD server:
    - (`foobar.com`, `dns1.foobar.com`, `NS`)
    - (`dns1.foobar.com`, `212.212.212.1`, `A`)
- Put in authoritative server `dns1.foobar.com`
  - Type A record for `www.foobar.com`
  - Type MX record for `foobar.com` (email server)

18

## Security Analysis of DNS

- What security issues does the design & operation of the Domain Name System raise?
- Degrees of freedom:

| *16 bits* | *16 bits* |
|---|---|
| Identification | Flags |
| # Questions | # Answer RRs |
| # Authority RRs | # Additional RRs |
| Questions (variable # of resource records) | |
| Answers (variable # of resource records) | |
| Authority (variable # of resource records) | |
| Additional information (variable # of resource records) | |

19

## Security Problem #1: Starbucks

- As you sip your latte and surf the Web, how does your laptop find google.com?
- Answer: it asks the local name server per Dynamic Host Configuration Protocol (DHCP) …
  - … which is run by Starbucks or their contractor
  - … and can return to you any answer they please
  - … including a "man in the middle" site that forwards your query to Google, gets the reply to forward back to you, yet can change anything they wish in either direction
- How can you know you're getting correct data?
  - Today, you can't. (Though if site is HTTPS, that helps)
  - One day, hopefully: DNSSEC extensions to DNS

20

## Security Problem #2: Cache Poisoning

- Suppose you are a Bad Guy and you control the name server for foobar.com. You receive a request to resolve www.foobar.com and reply:

```
;; QUESTION SECTION:
;www.foobar.com.                IN      A

;; ANSWER SECTION:
www.foobar.com.         300     IN      A       212.212.212.44

;; AUTHORITY SECTION:
foobar.com.             600     IN      NS      dns1.foobar.com.
foobar.com.             600     IN      NS      google.com.

;; ADDITIONAL SECTION:
google.com.             600     IN      A       212.212.212.55
```

**A foobar.com machine, *not* google.com**

21

## Cache Poisoning, con't

- Okay, but how do you get the victim to look up www.foobar.com in the first place?
- Perhaps you connect to their mail server and send
  - **`HELO www.foobar.com`**
  - Which their mail server then looks up to see if it corresponds to your source address (anti-spam measure)
- Note, with compromised name server we can also lie about PTR records (address → name mapping)
  - E.g., for 212.212.212.55 = 55.212.212.212.in-addr.arpa return google.com (or whitehouse.gov, or whatever)

22

## Cache Poisoning, con't

- Suppose Bad Guy is at Starbuck's and they can guess the identification field the local server will use in its next request:

| *16 bits* | *16 bits* |
|---|---|
| **Identification** | **Flags** |

- They:
  - Issue a (recursive) lookup of google.com
  - Locally spoof subsequent reply from correct name server using the identification field
  - Bogus reply arrives sooner than legit one
- Local server duly caches the bogus reply!
  - Now: every future Starbuck's is served the bogus answer out of the local server's cache

23

## 5 Minute Break

Questions Before We Proceed?

24

## Application Protocols

25

## Application-Layer Protocols

- Network applications run on end systems
  - They depend on the network to provide a service
  - … but cannot run software on the network elements
- Network applications run on multiple machines
  - Different end systems communicate with each other
  - Software is often written by multiple parties
- Leading to a need to explicitly define a protocol
  - Types of messages (e.g., requests and responses)
  - Message syntax (e.g., fields, and how to delineate)
  - Semantics of the fields (i.e., meaning of the information)
  - Rules for when and how a process sends messages

26

## Applications vs. Application-Layer Protocols

- Application-layer protocol is just one piece
  - Defining how the end hosts communicate
- Example: World Wide Web
  - HyperText Transfer Protocol is the protocol
  - But the Web includes other components, such as document formats (HTML), Web browsers, servers, …
- Example: electronic mail
  - Simple Mail Transfer Protocol (SMTP) is one of the protocols
  - But e-mail includes other components, such as mail servers, user mailboxes, mail readers

27

## A Wide Range of Application Styles

- Underlying transport:
  - TCP (Web, Email), UDP (DNS, Network Time), both (NFS, Skype)
- Connection structure:
  - Single connection (SSH), series of connections (Email, HTTP), concurrent connections (HTTP, FTP)
  - Control separate from data (FTP) or intermingled (HTTP, SMTP, Telnet)
  - Long stateful sessions (FTP, Telnet), stateless servers (HTTP)
- Application protocol format:
  - Binary (DNS, NFS), Text (SMTP, HTTP), both (SSH)
- Data encoding:
  - None (Telnet), MIME (SMTP, HTTP), XDR (RPC), custom compression (DNS)

28

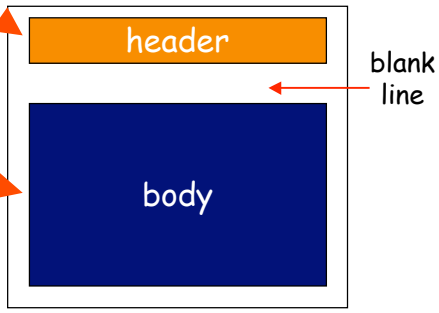
## Example of An Application Protocol:

## Electronic Mail

29

## E-Mail Message Format (RFC 822)

- E-mail messages have two parts
  - A header, in 7-bit U.S. ASCII text
  - A body, also represented in 7-bit U.S. ASCII text
- Header
  - Lines with “type: value”
  - “To: vern@icir.org”
  - “Subject: Go Bears!”
- Body
  - The text message
  - No particular structure or meaning
  - Need ways to deal with 8-bit text, binaries, multiple items



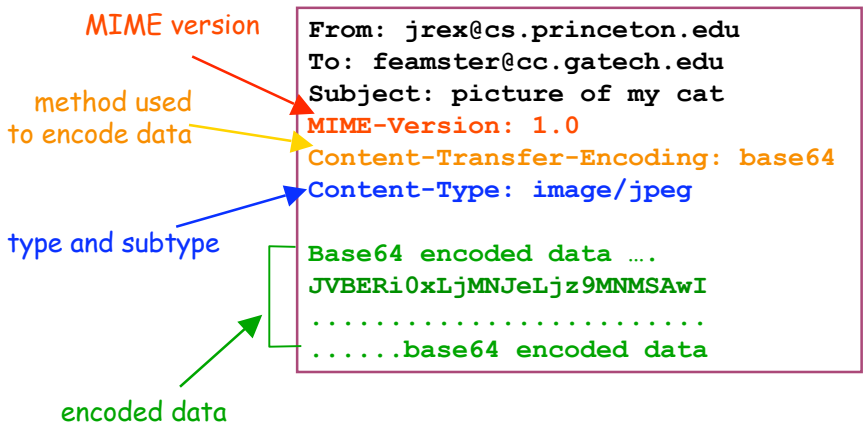

30

## Multipurpose Internet Mail Extensions

- Additional headers to describe the message body
  - MIME-Version: the version of MIME being used
  - Content-Type: the type of data contained in the message
  - Content-Transfer-Encoding: how the data are encoded
- Definitions for a set of content types and subtypes
  - E.g., image with subtypes gif and jpeg
  - E.g., text with subtypes plain, html, and richtext
  - E.g., application with subtypes postscript and msword
  - E.g., multipart for messages with multiple data types
- A way to encode the data in ASCII format
  - Base64 encoding (64-character printable alphabet)

31

## Example: E-Mail Message Using MIME



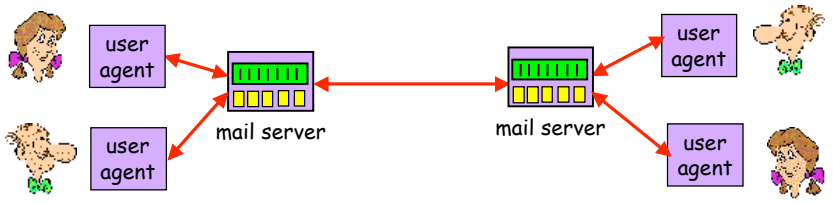

32

## E-Mail Addresses

- Components of an e-mail address
  - Local mailbox (e.g., vern or dilip.antony.joseph)
  - Domain name (e.g., icir.org or gmail.com)
- Identifying the mail server for a domain
  - DNS query asking for MX records
    - Mail eXchange
  - E.g., dig eecs.berkeley.edu mx
  - Yields:
    - `mx1.eecs.berkeley.edu, mx2.eecs.berkeley.edu, bc1.eecs.berkeley.edu, bc2.eecs.berkeley.edu`
  - Then, regular DNS query to learn the IP address

33

## Mail Servers and User Agents




- Mail servers
  - Always on and always accessible
  - Transferring e-mail to and from other servers
- User agents
  - Sometimes on and sometimes accessible
  - Intuitive interface for the user

34

## SMTP *Store-and-Forward* Protocol


- Messages sent through a series of servers
  - A server stores incoming messages in a queue
  - … to await attempts to transmit them to the next hop
- If the next hop is not reachable
  - The server stores the message and tries again later
- Each hop adds its identity to the message
  - By adding a "Received" header with its identity
  - Helpful for diagnosing problems with e-mail

35

## Example With Received Header

```
Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
      by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04M5R7Y023164
      for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from bluebox.CS.Princeton.EDU ([128.112.136.38])
      by ribavirin.CS.Princeton.EDU (SMSSMTP 4.1.0.19) with SMTP id M2006010417053607946
      for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
      by bluebox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04M5XNQ005204
      for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from [192.168.1.101] (adsl-69-107-78-147.dsl.pltn13.pacbell.net [69.107.78.147])
      (authenticated bits=0)
      by smtp-roam.Stanford.EDU (8.12.11/8.12.11) with ESMTP id k04M5W92018875
      (version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
      Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC46AF.3030306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit
```

36

## Multiple Server Hops

- Typically at least two mail servers
  - Sending and receiving sides
- May be more
  - Separate servers for key functions
    - Spam filtering
    - Virus scanning
  - Servers that redirect the message
    - From vern@ee.lbl.gov to vern@icir.org
    - Messages to ee.lbl.gov go through extra hops
  - Electronic mailing lists
    - Mail delivered to the mailing list's server
    - … and then the list is expanded to each recipient

37

## Simple Mail Transfer Protocol


- Client-server protocol
  - Client is the sending mail server (a form of peer-to-peer)
  - Server is the receiving mail server
- Reliable data transfer
  - Built on top of TCP (on port 25)
- Push protocol
  - Sending server pushes the file to the receiving server
  - … rather than waiting for the receiver to request it

38

## Simple Mail Transfer Protocol (Cont.)

- Command/response interaction
  - Commands: ASCII text
  - Response: three-digit status code and phrase
- Synchronous
  - Sender awaits response from a command
  - … before issuing the next command
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure

39

## Scenario: Alice Sends Message to Bob

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

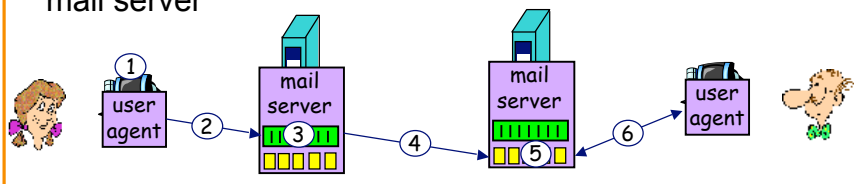
2) Alice's UA sends message to her mail server; message placed in message queue

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




40

## 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: From: alice@crepes.fr
C: To: hamburger-list@burger-king.com
C: Subject: Do you like ketchup?
C:
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Message header

Message body

Lone period marks end of message

41

## Retrieving E-Mail From the Server

- Server stores incoming e-mail by mailbox
  - Based on the "From" field in the message
- Users need to retrieve e-mail
- Variety of ways to do this:
  - Directly by same-machine access to the mailbox
  - Via Interactive Mail Access Protocol (IMAP)
    - Supports concurrent access by multiple clients, server-side searchers, partial MIME fetches, multiple mailboxes
  - Via HTTP (Web)
    - E.g., GMail
  - Via Post Office Protocol (POP)

42

## 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 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

43

## Summary

- Domain Name System (DNS)
  - Distributed, hierarchical database
  - Distributed collection of servers
  - Caching to improve performance
  - Examine using **`dig`** utility
- Application-layer protocols
  - Rich and constantly evolving area
  - Tailoring communication to the application
- Electronic-mail protocols
  - SMTP to transfer e-mail messages
  - MIME to represent a wide variety of data formats
  - Several retrieval techniques (POP, IMAP, and Web)

44

## Next Lecture

- The Web
- Reading:
  - Sections 9.2.2
- Project #1 **due this Thursday** by 11PM PDT
- Reminder: feedback solicited

45