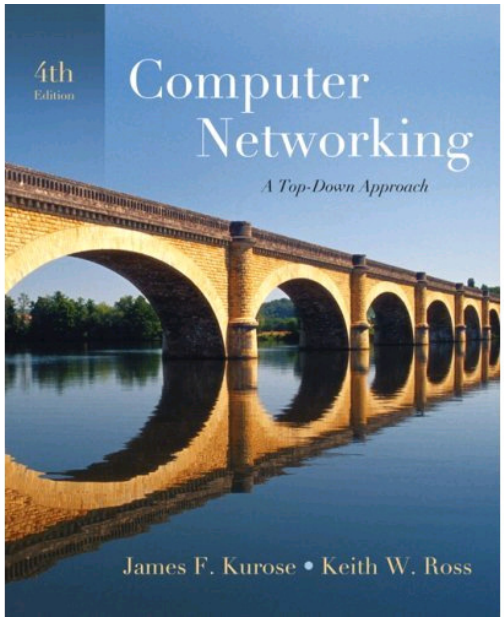
Services and Application Layer

Instructor: Dr. Eng. Abdulhalim Dandoush

Adandoush at gmail.com
Adandoush.com
Tishreen University & Arab Academy for Science and Technology

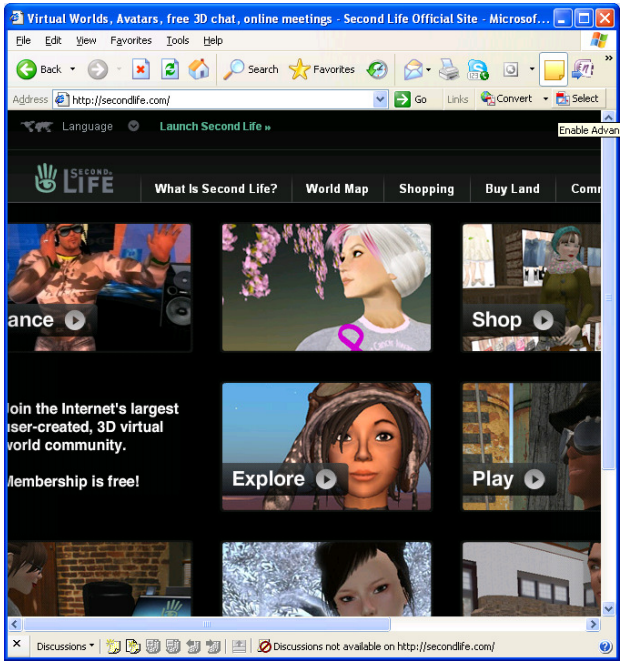
A special acknowledge goes to Joanna Moulierac, from the INRIA of Sophia Antipolis and IUT de Nice.
Some of the slides used in this lecture are adapted from their original slides
An other Ref: “Computer Networking, A Top-Down Approach”

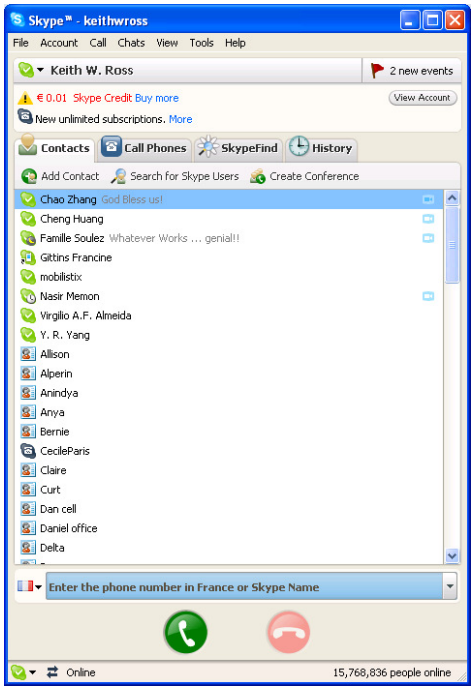


Application Layer

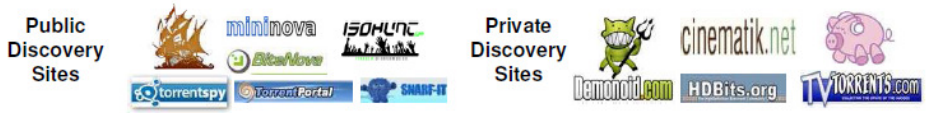
Our goals:

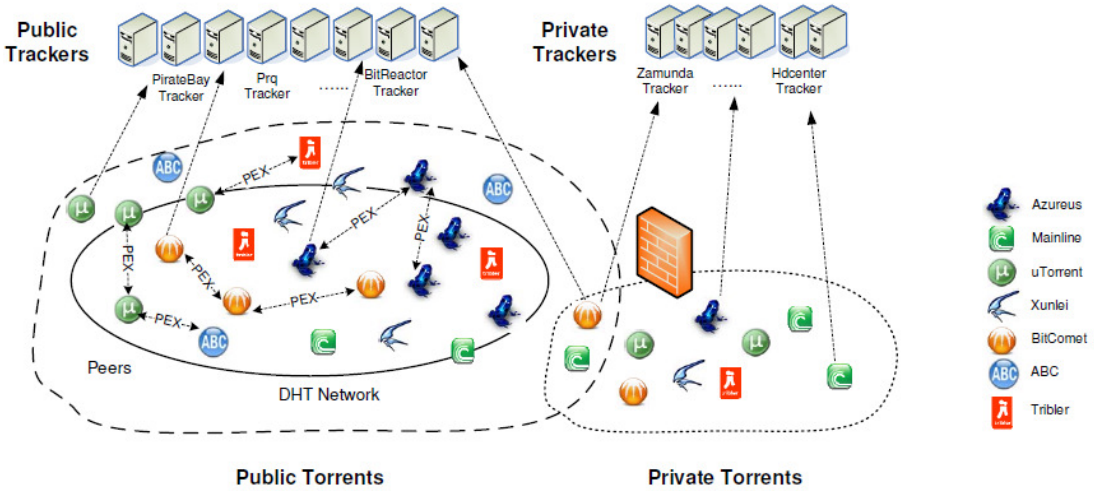
- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
  - File Sharing
  - Terminal Service
  - LDAP

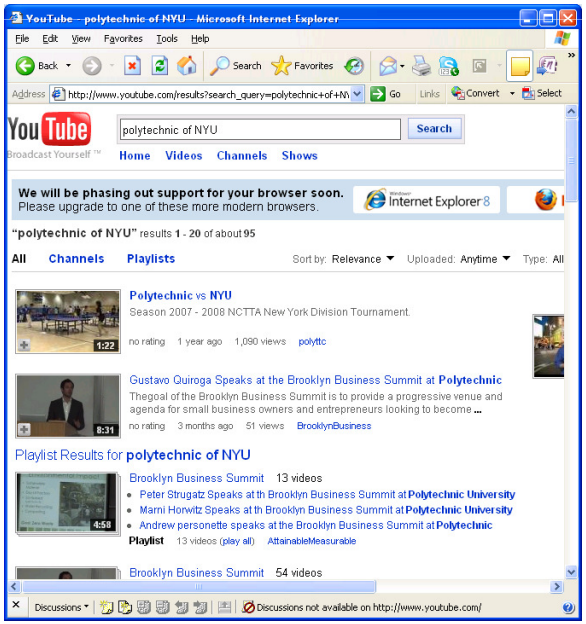

Virtual Worlds, Avatars, free 3D chat, online meetings - Second Life Official Site - Microsof...
What Is Second Life?
World Map
Shopping
Buy Land
Shop
Explore
Play
Discussions not available on http://secondlife.com/


Skype™ - keithwross
Keith W. Ross
Contacts
Call Phones
SkypeFind
History
Enter the phone number in France or Skype Name
Online
15,768,836 people online


Facebook | Keith Ross - Microsoft Internet Explorer
facebook
Keith Ross
Wall
Info
Photos
Boxes
Video
Discussions not available on http://www.facebook.com/


Public Discovery Sites
Private Discovery Sites
torrentspy
TorrentPortal
SNARF-IT
Demonoid.com
HDBits.org
TVTORRENTS.com


Public Trackers
PirateBay Tracker
Prq Tracker
BitReactor Tracker
Private Trackers
Zamunda Tracker
Hdcenter Tracker
PEX
Peers
DHT Network
Azureus
Mainline
uTorrent
Xunlei
BitComet
ABC
Tribler
Public Torrents
Private Torrents


YouTube - polytechnic of NYU - Microsoft Internet Explorer
YouTube
polytechnic of NYU
Search
Home
Videos
Channels
Shows
We will be phasing out support for your browser soon.
Please upgrade to one of these more modern browsers.
"polytechnic of NYU" results 1 - 20 of about 95
Polytechnic vs NYU
Season 2007 - 2008 NCTTA New York Division Tournament.
Gustavo Quiroga Speaks at the Brooklyn Business Summit at Polytechnic
Playlist Results for polytechnic of NYU
Brooklyn Business Summit 13 videos
Brooklyn Business Summit 54 videos
Discussions not available on http://www.youtube.com/

Some network apps

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- social networks
- voice over IP
- real-time video conferencing
- cloud computing

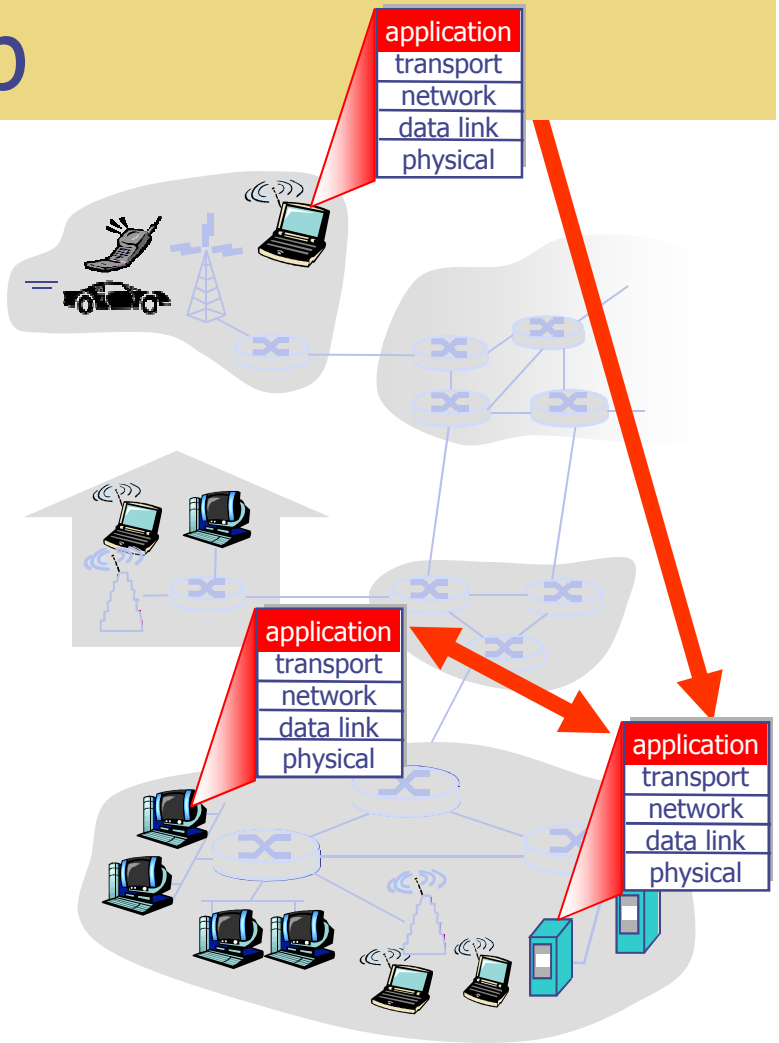
Creating a network app

Write programs that

- run on (different) *end systems*
- communicate over network
- e.g., web server software communicates with browser software

No need to write software for network-core devices

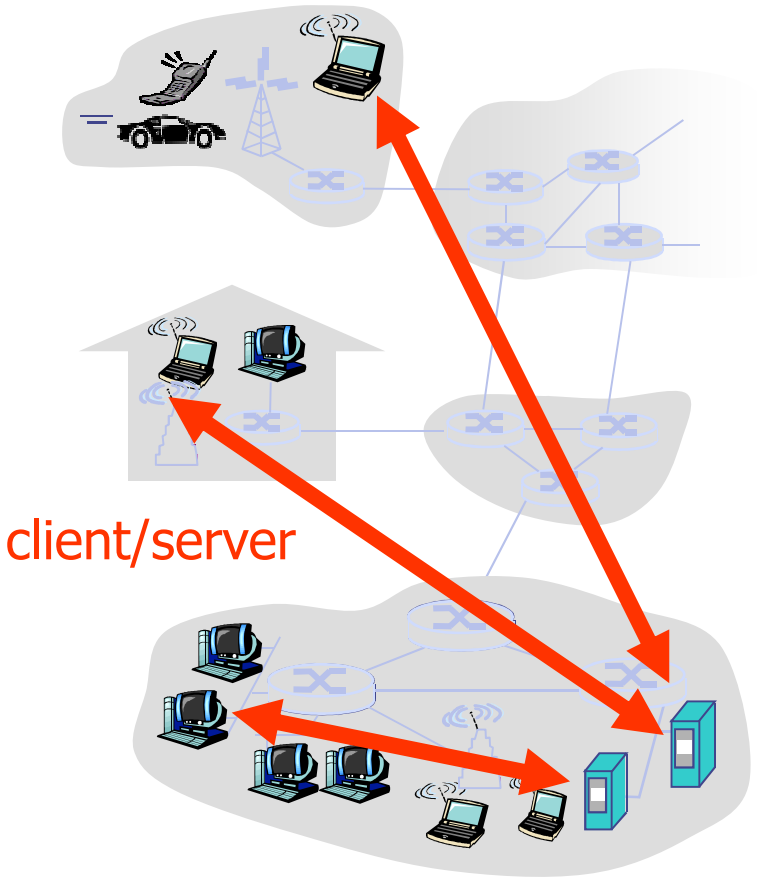
- Network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation



Application architectures

- Client-server
  - Including data centers / cloud computing
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-server architecture



server:

- always-on host
- permanent IP address
- server farms for scaling

clients:

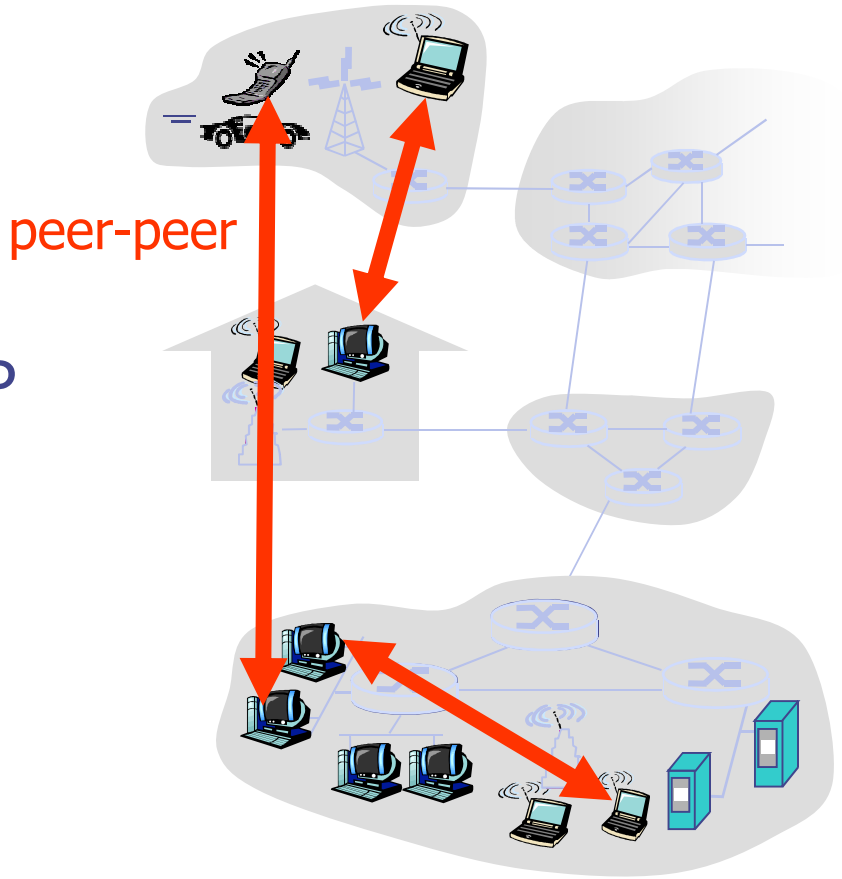
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Google Data Centers

- Estimated cost of data center: $600M
- Google spent $2.4B in 2007 on new data centers
- Each data center uses 50-100 megawatts of power

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses



Highly scalable but difficult to manage

Hybrid of client-server and P2P

Skype

- voice-over-IP P2P application
- centralized server to find address of remote party
- client-client connection direct (not through server)

Instant messaging

- chatting between two users is P2P
- centralized service: client presence detection/location
  - user registers its IP address with central server when it comes online
  - user contacts central server for buddy IP addresses

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using inter-process communication (defined by OS).
- processes in different hosts communicate by exchanging messages

Client process: process that initiates communication

Server process: process that waits to be contacted

Note: applications with P2P architectures also have client processes & server processes

Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- Exercise: use `ipconfig` (Windows) or `ifconfig` (Mac & Linux) from command prompt to get your IP address

- *Q:* does IP address of host on which process runs suffice for identifying the process?

  A: No, *many* processes can be running on same

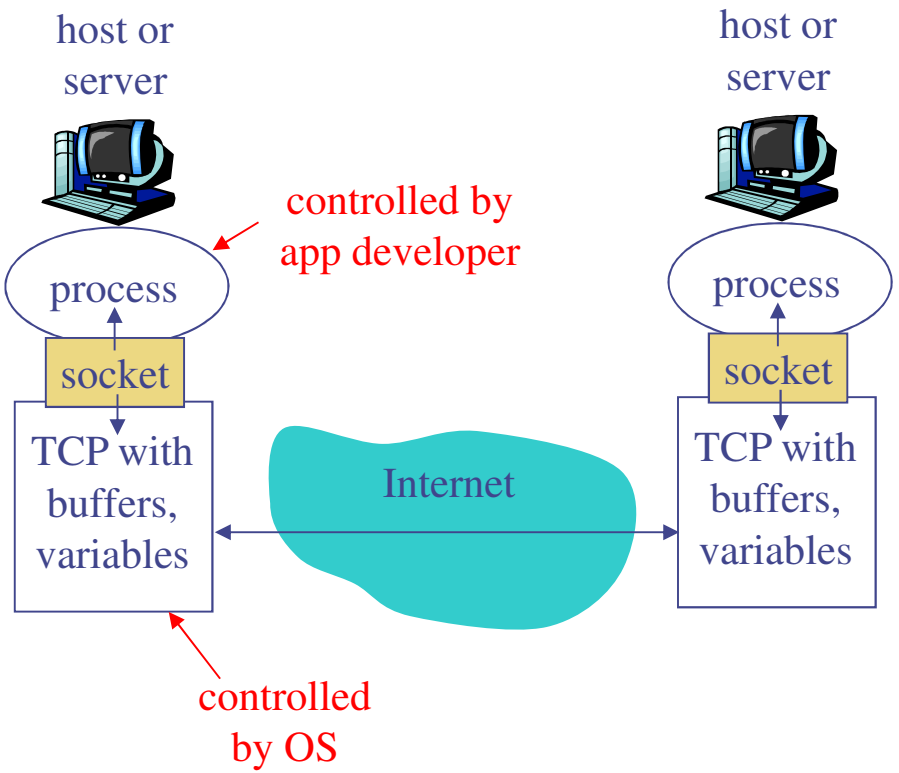
Identifier includes both IP address and port numbers associated with process on host.

Example port numbers:

- HTTP server: 80
- Mail server: 25

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process



API: (1) choice of transport protocol;

(2) ability to fix a few parameters

App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response
- Message syntax:
  - what fields in messages & how fields are delineated
- Message semantics
  - meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP, BitTorrent

Proprietary protocols:

- e.g., Skype, PPLive

What transport service does an app need?

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

Security

- Encryption, data integrity, …

Transport service requirements of common apps

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet transport protocols services

TCP service:

- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantees, security

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

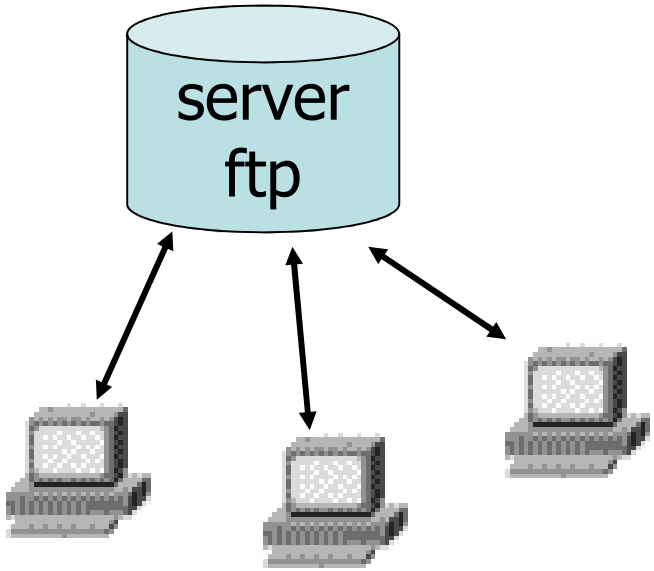
Services and applications design

ports

- 65536 ports (16 bits) :
  - **from 0 to 1023 :** («**Well Known Ports**»).
- **21 :** FTP
- **22 :** SSH
- **23 :** Telnet
- **25 :** SMTP
- **53 :** Domain Name System
- **68 :** DHCP
- **80 :** HTTP
- **110 :** POP3
- …

FTP (*File Transfer Protocol*)

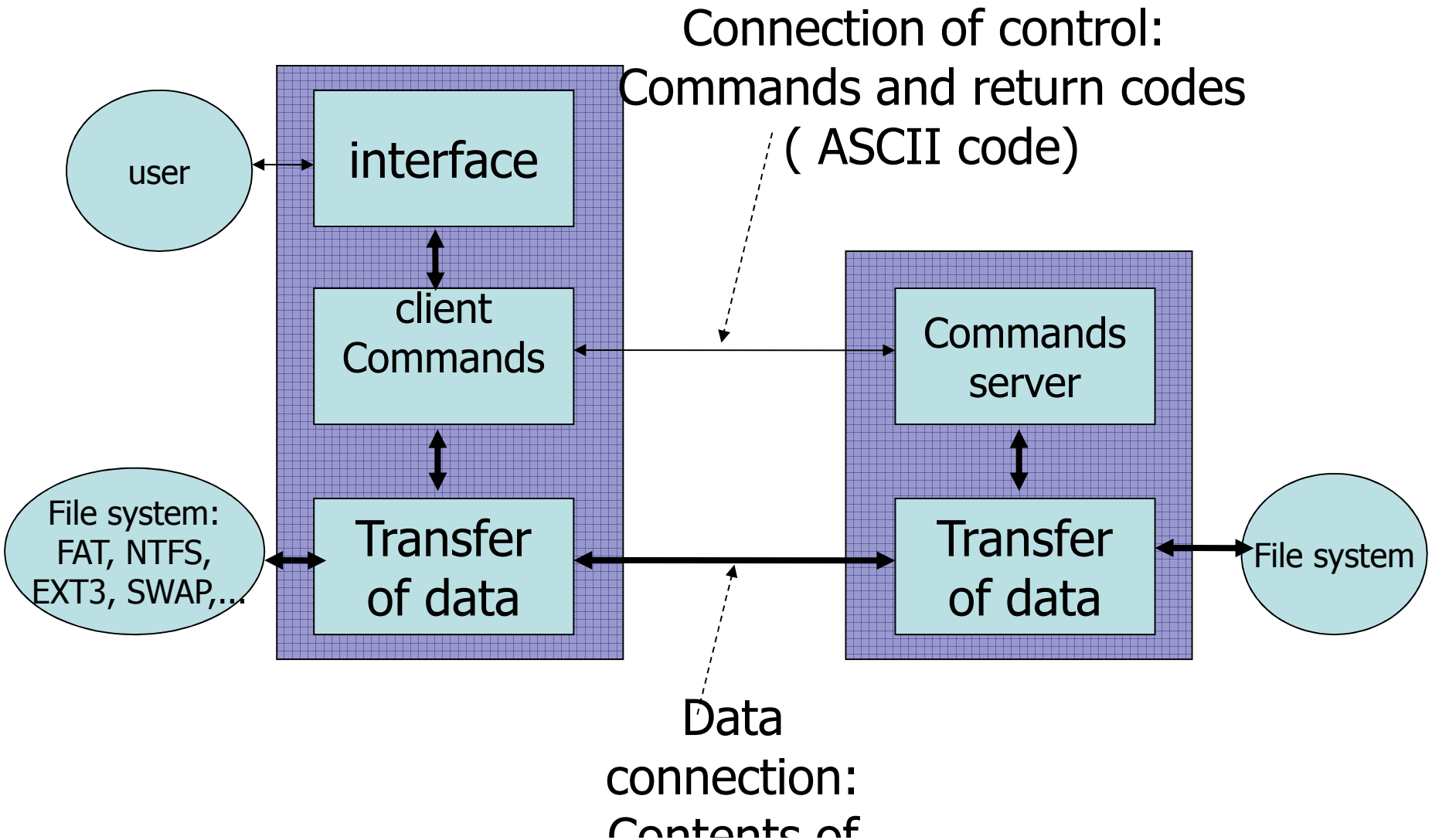
- Retrieve stored files by Net users



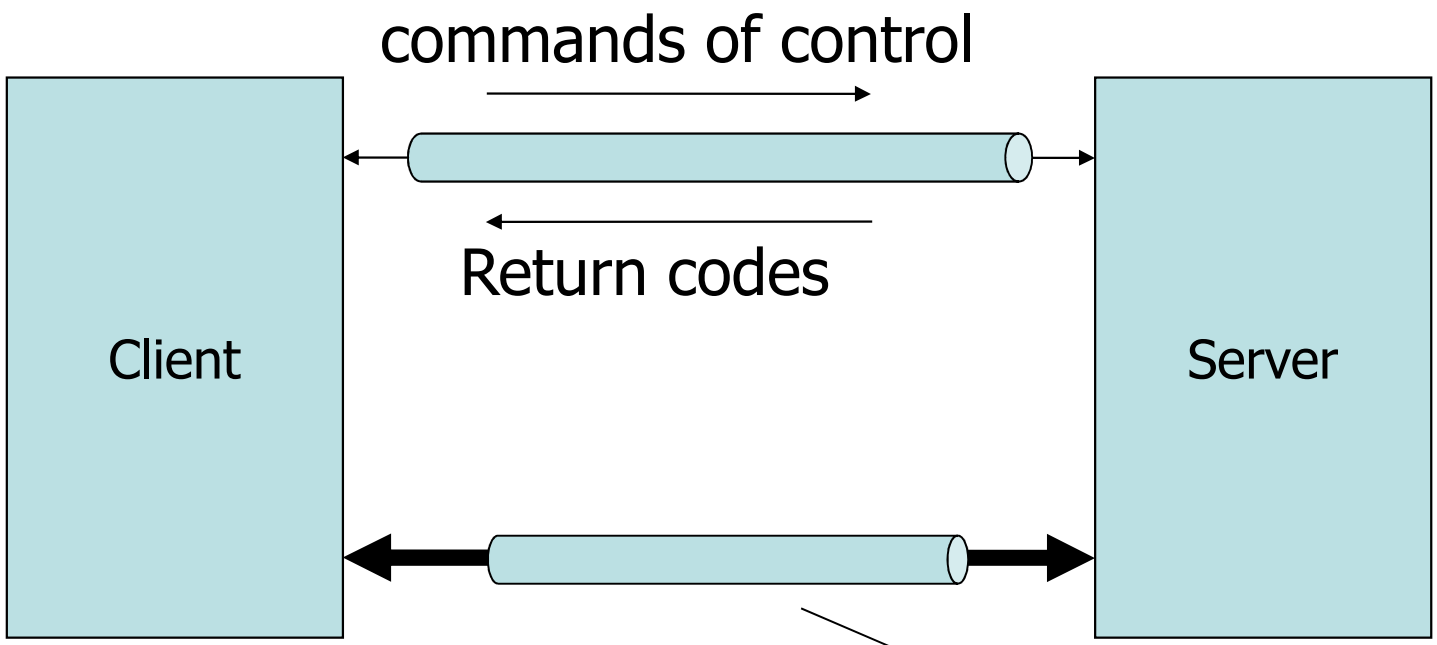
Commands

- get, mget
- put, mput
- lcd, cd
- bin, ascii
- quit, bye
- ls
- help
- …

Functionality model



Functionality model



Commands of controls
Access : OPEN, USER, PASS, CWD, QUIT
Transfer : PORT, PASSV, MODE
Service : STOR, RETR, LIST

each transfer :
new TCP connection

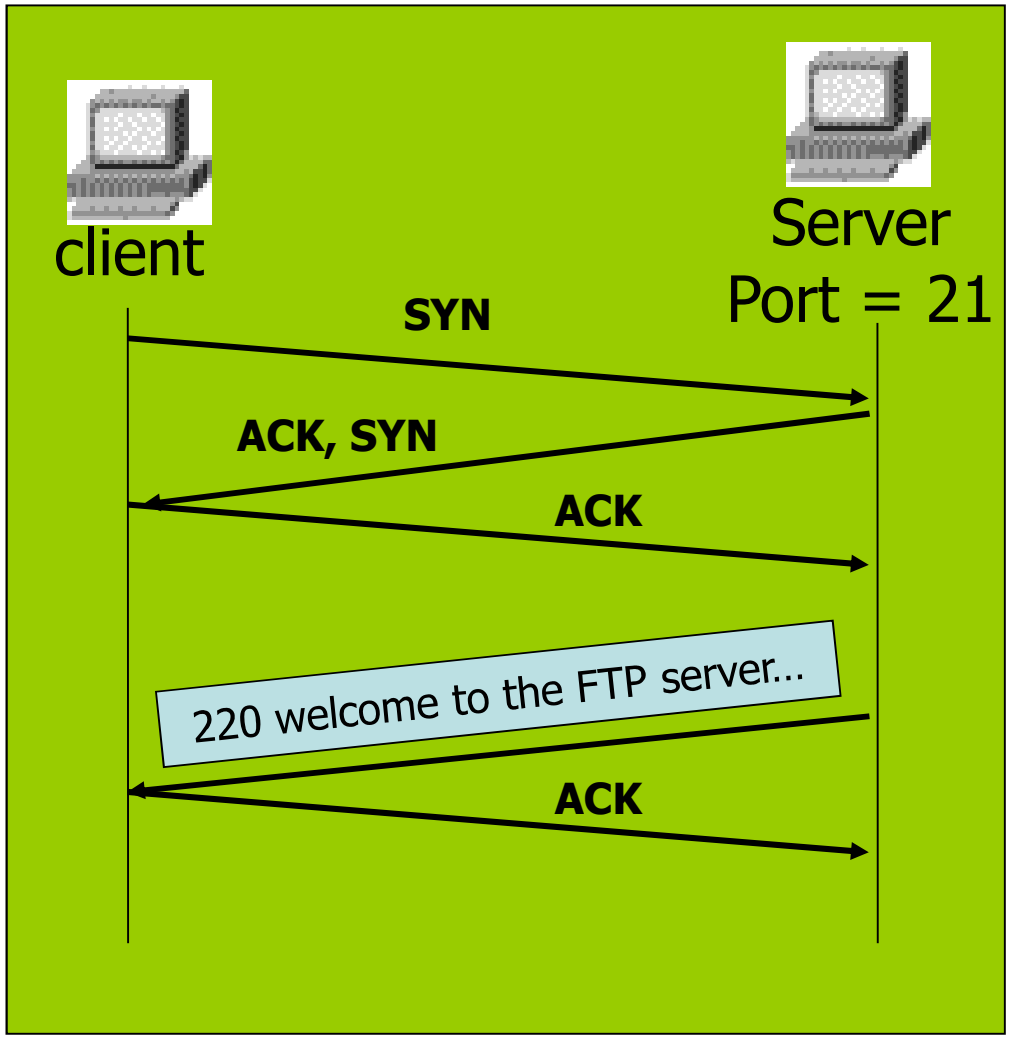
```
adandoush@Ubuntu-server:~$ ftp
ftp> help
Commands may be abbreviated.  Commands are:

!           debug       mdir        qc          send
$           dir         mget        sendport    site
account     disconnect  mkdir       put         size
append      exit        mls         pwd         status
ascii       form        mode        quit        struct
bell        get         modtime     quote       system
binary      glob        mput        recv        sunique
bye         hash        newer       reget       tenex
case        help        nmap        rstatus     tick
cd          idle        nlist       rhelp       trace
cdup        image       ntrans      rename      type
chmod       lcd         open        reset       user
close       ls          prompt      restart     umask
cr          macdef      passive     rmdir       verbose
delete      mdelete     proxy       runique     ?
ftp>
```

Example of FTP scenario

User Interface (shell)

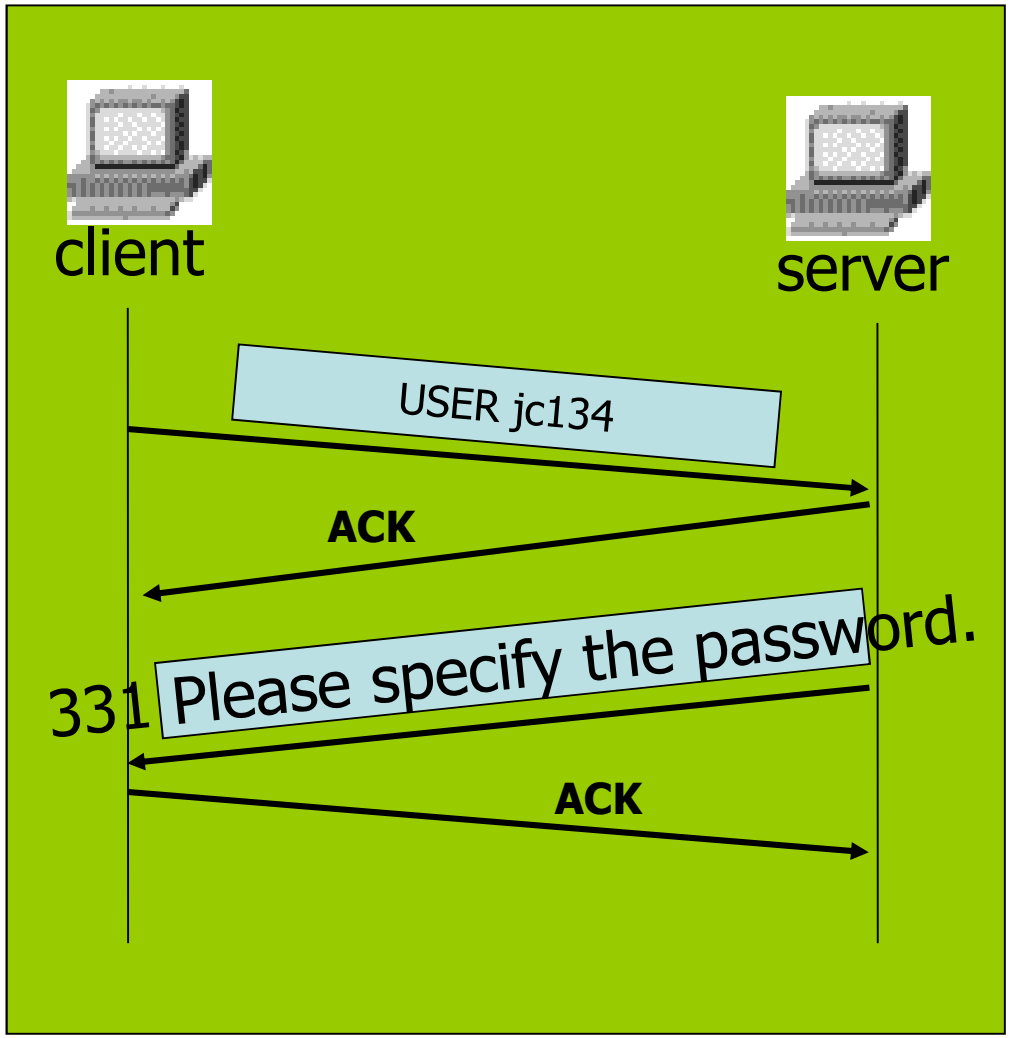
```
ftp
>open server_name
```



Example of FTP scenario

User Interface

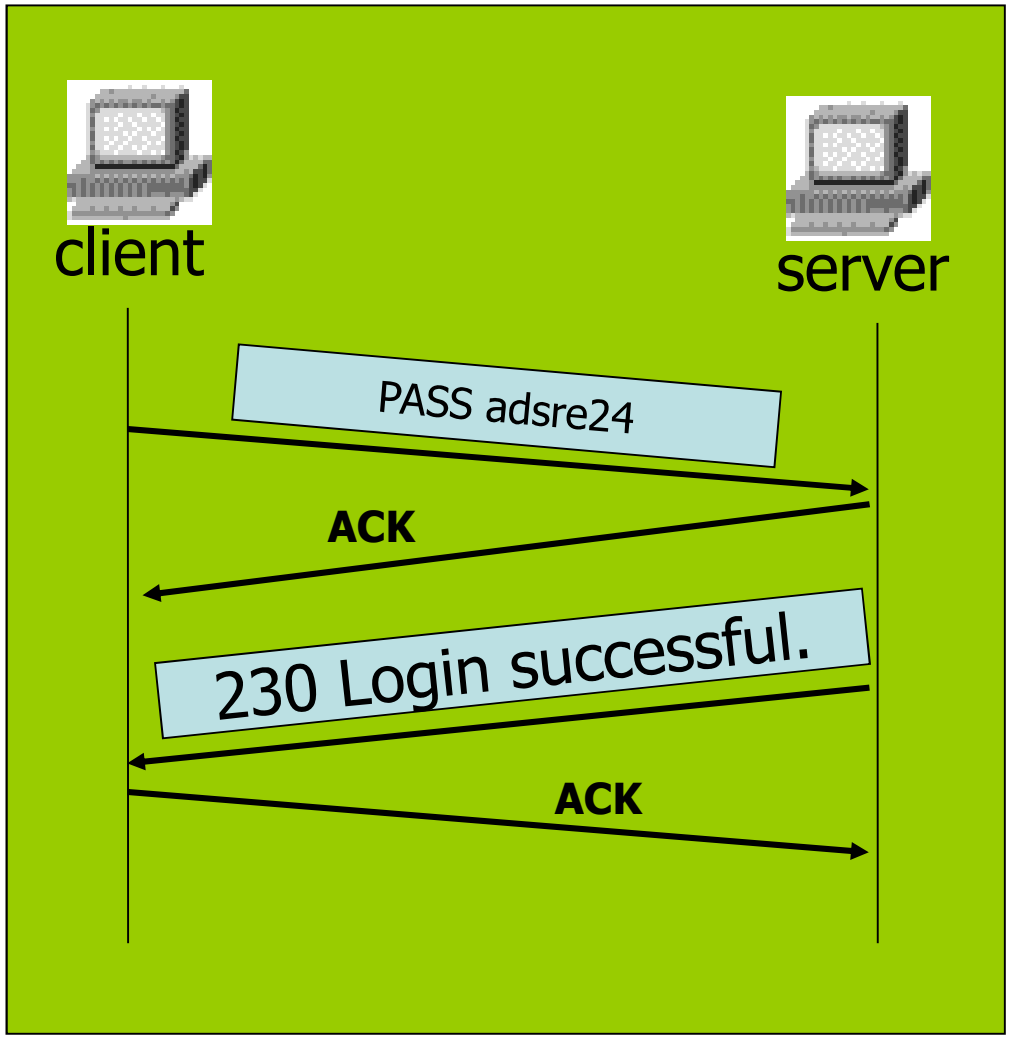
ftp
>open servername
welcome to the
FTP server…
>user jc1234



Example of FTP scenario

User Interface

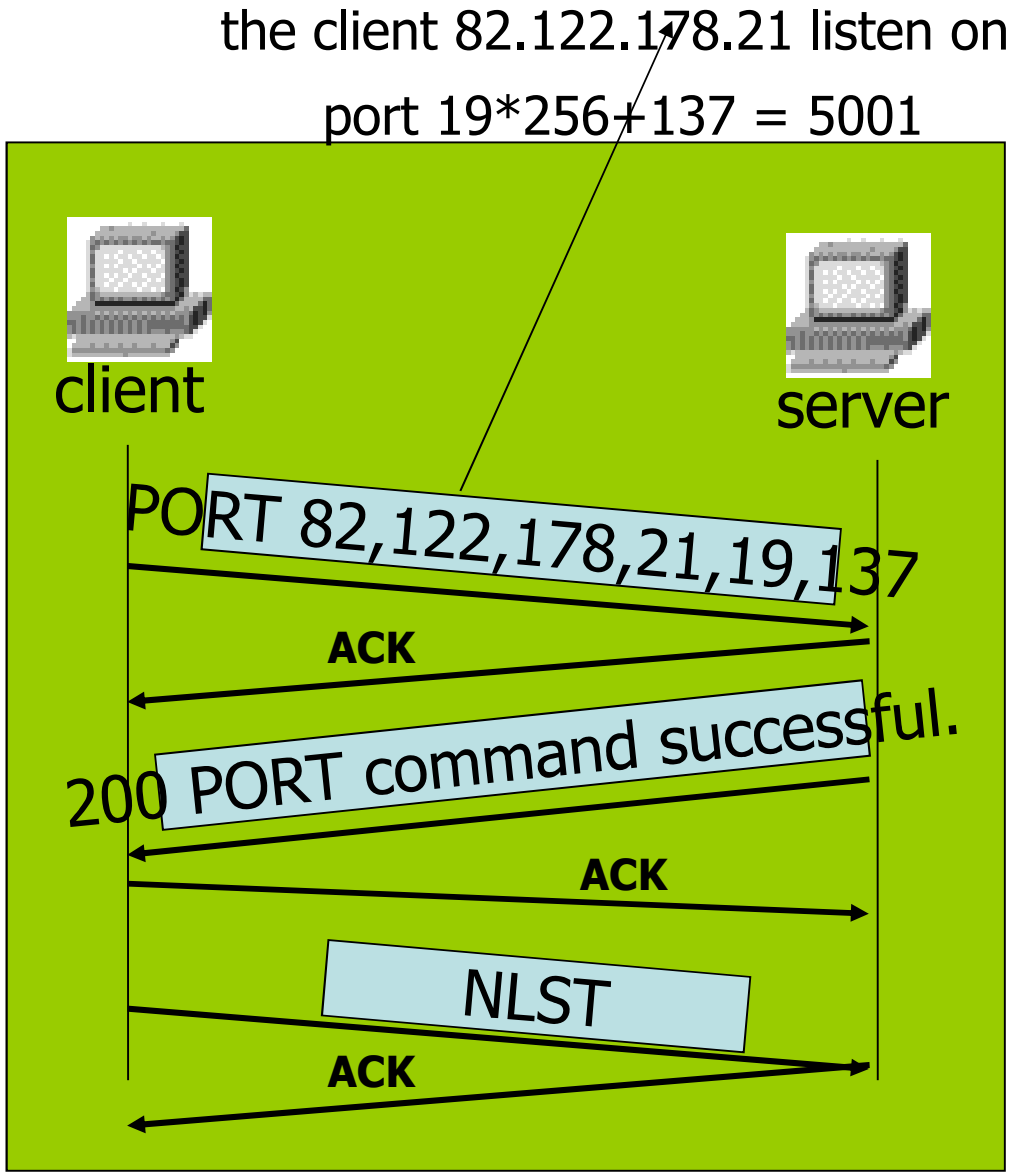
```
ftp
>open servername
welcome to the
FTP server…
>user jc1234
Password: adsre24
Login successful.
>
```



Example of FTP scenario

User Interface

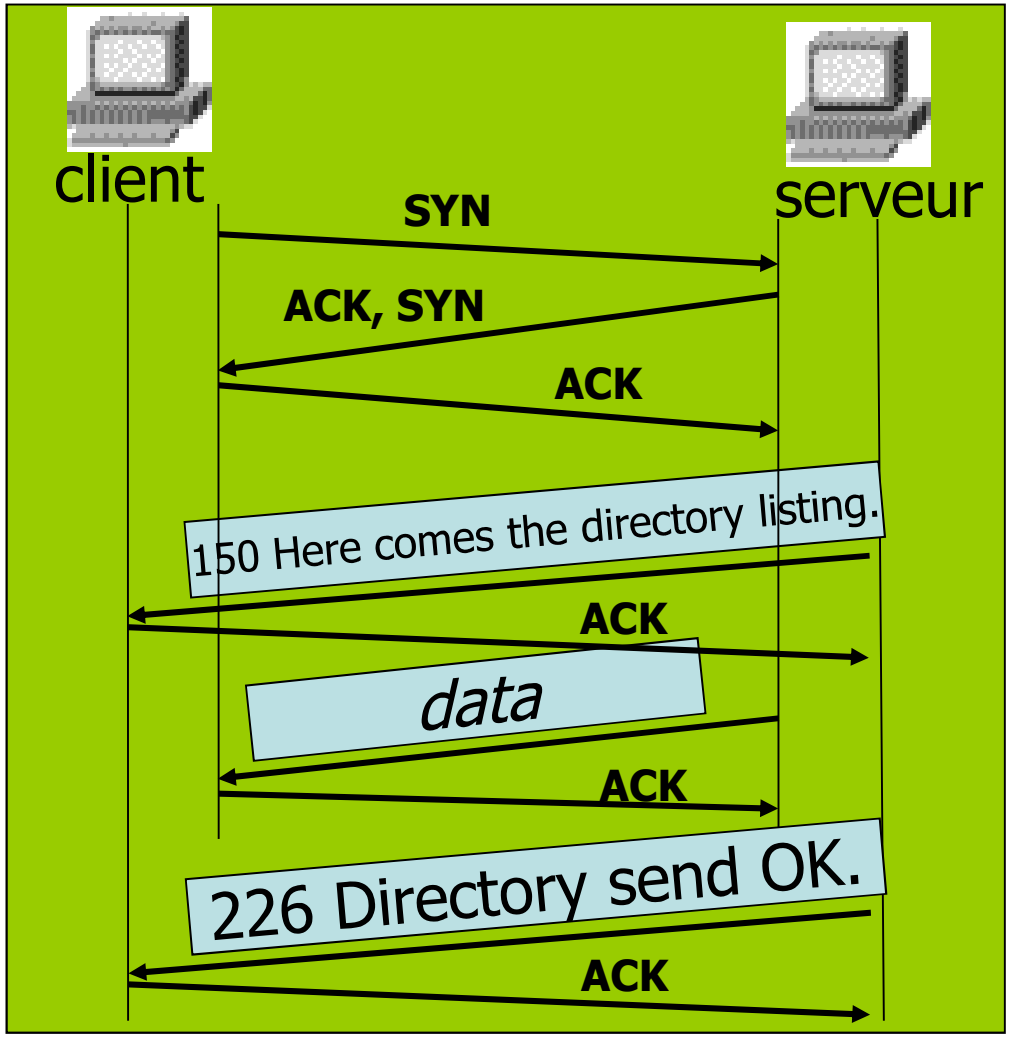
ftp
>open servername
welcome to the
FTP server…
>user jc1234
Password: adsre24
Login successful.
>ls



Example of FTP scenario

User Interface

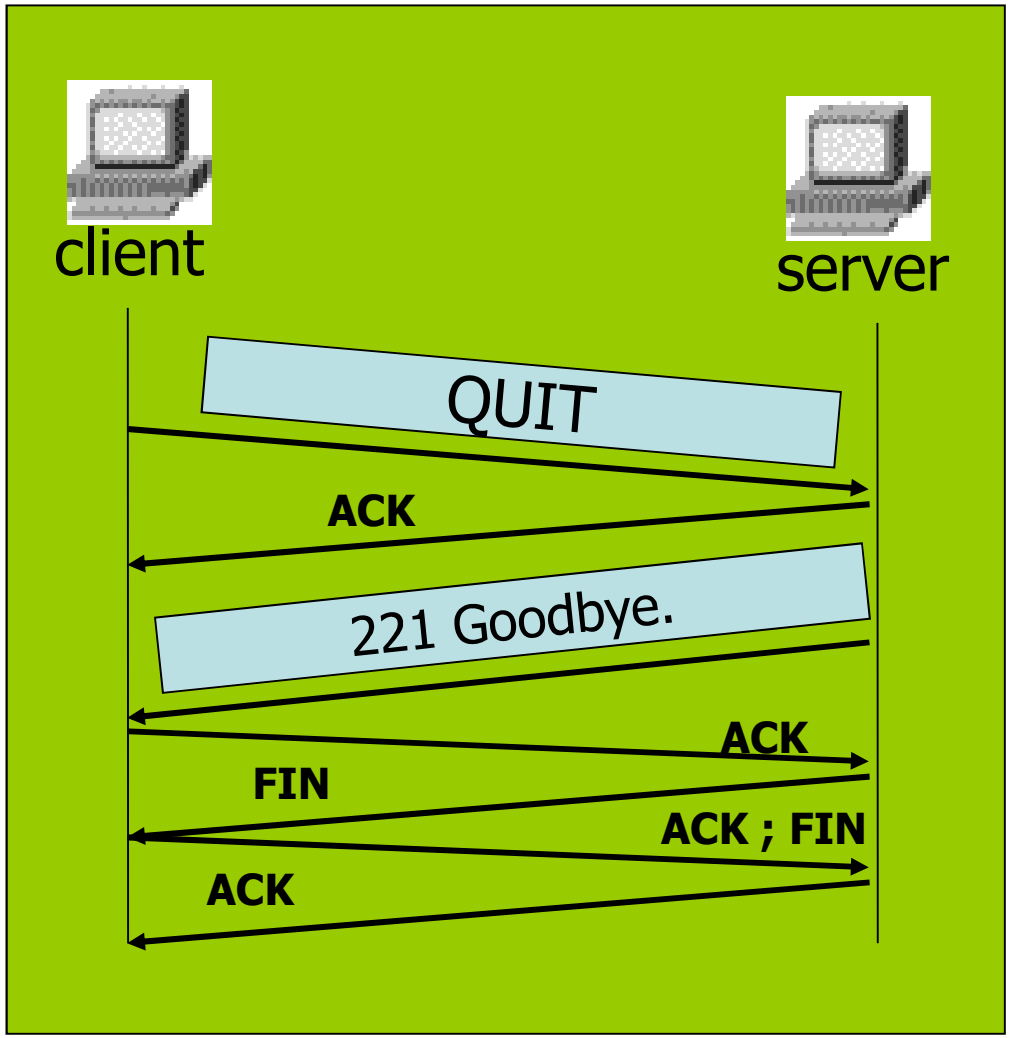
```
...
>ls
```



Example of FTP scenario

User Interface

```
...
>bye
```



HTTP (*HyperText Transfer Protocol)*

- Standard communication protocol for the Web
- Port 80
- Two version:
  - http 1.0
  - http 1.1
- References :
  - HTTP 1.0 : http://www.faqs.org/rfcs/rfc1945.html
  - HTTP 1.1 : http://www.faqs.org/rfcs/rfc2616.html

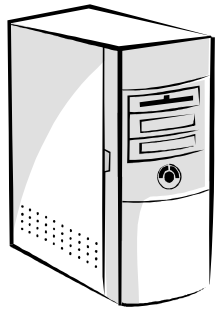
Functionality Model

Client HTTP
(Web browser)

request

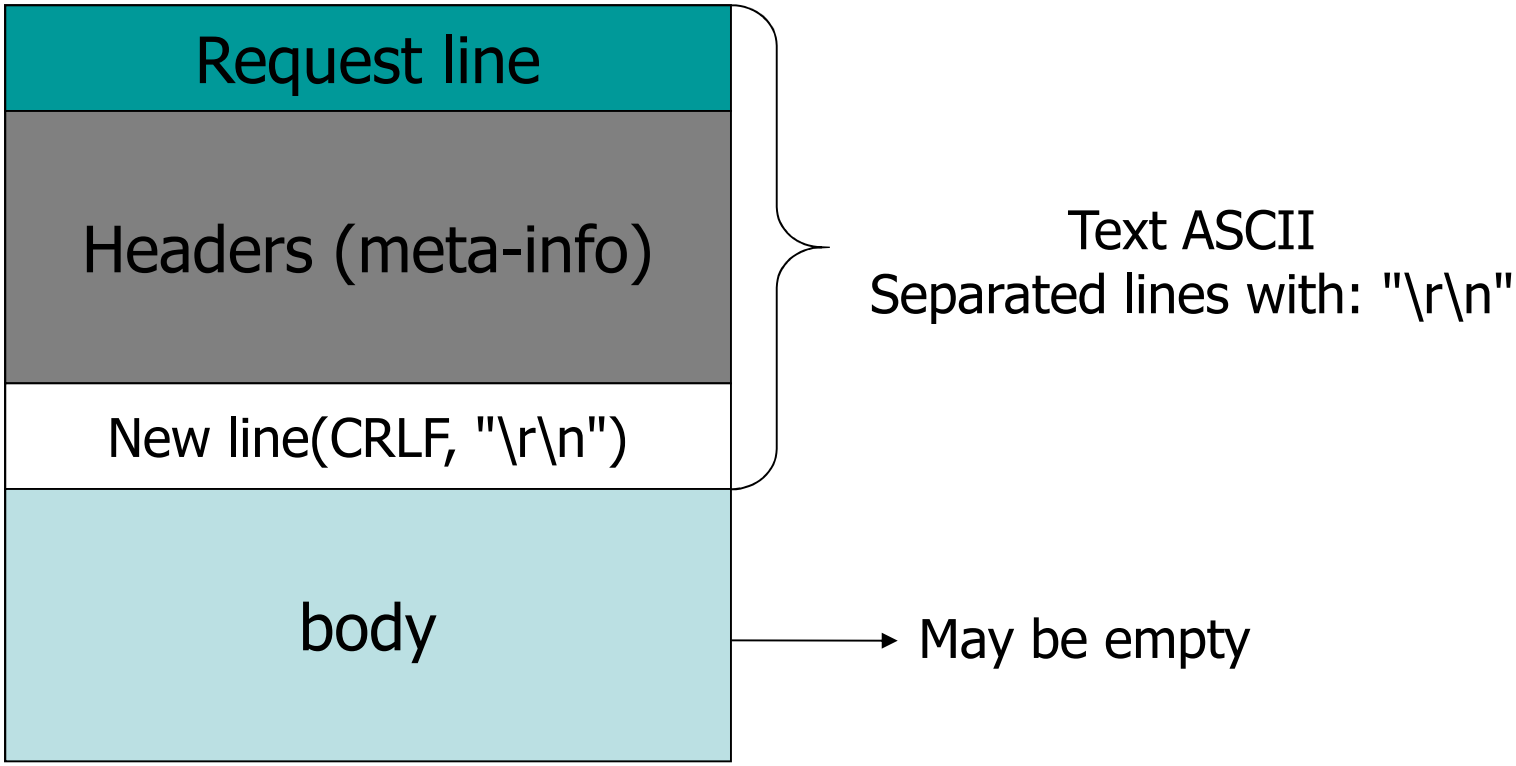
response

Server HTTP
(Web server)



Transport
(usually TCP)
HTTP 1.0 : one request/response per connection
HTTP 1.1 several ..

HTTP Request



HTTP request – general format

- The request message consists of the following:
- Request line, such as
  - GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from server
- Headers, such as Accept-Language: en
- An empty line
- An optional message body

- HTTP defines nine methods

Request methods

- **GET** Requests a representation of the specified resource
- **HEAD** Asks for the response identical to the one that would correspond to a GET request, but without the response body “meta-information”
- **POST** Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request.
- **CONNECT** Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy
- **PUT, DELETE, TRACE, OPTIONS, PATCH**

Status codes

- In HTTP/1.0 and since, the first line of the HTTP response is called the *status line*
- includes a numeric *status code* (such as "404") and a textual *reason phrase* (such as "Not Found").
- The way the user agent handles the response primarily depends on the code and secondarily on the response headers
- Ex:
  - 426 Upgrade Required
  - 200 OK

HTTP request- example

GET /index.htm HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*

Accept-Language: fr

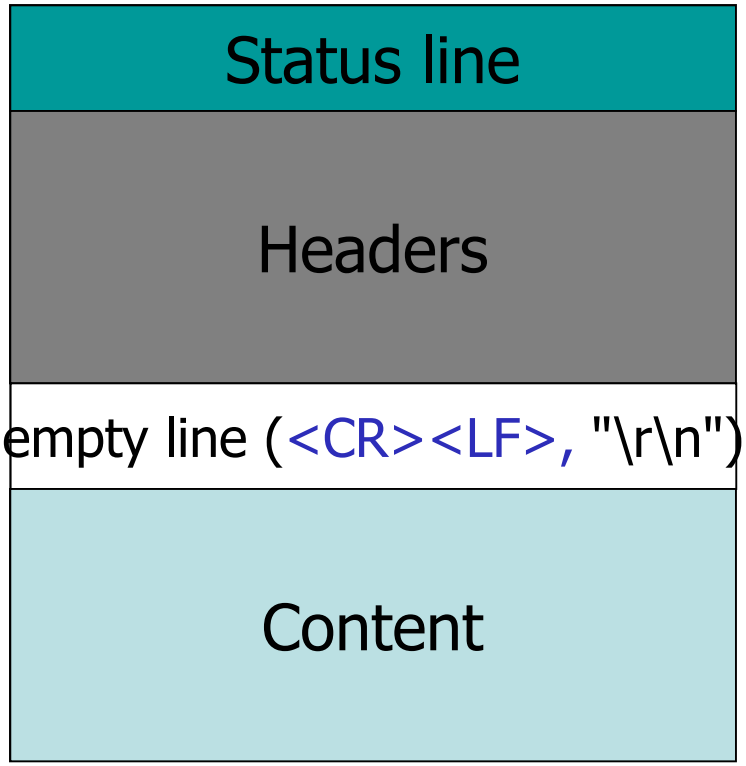
Accept-Encoding: gzip

User-Agent: Mozilla/4.0(compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 1.0.3705)

Host: www.reuters.com

Connection: Keep-Alive

HTTP Response



HTTP Response - example

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8 …
```

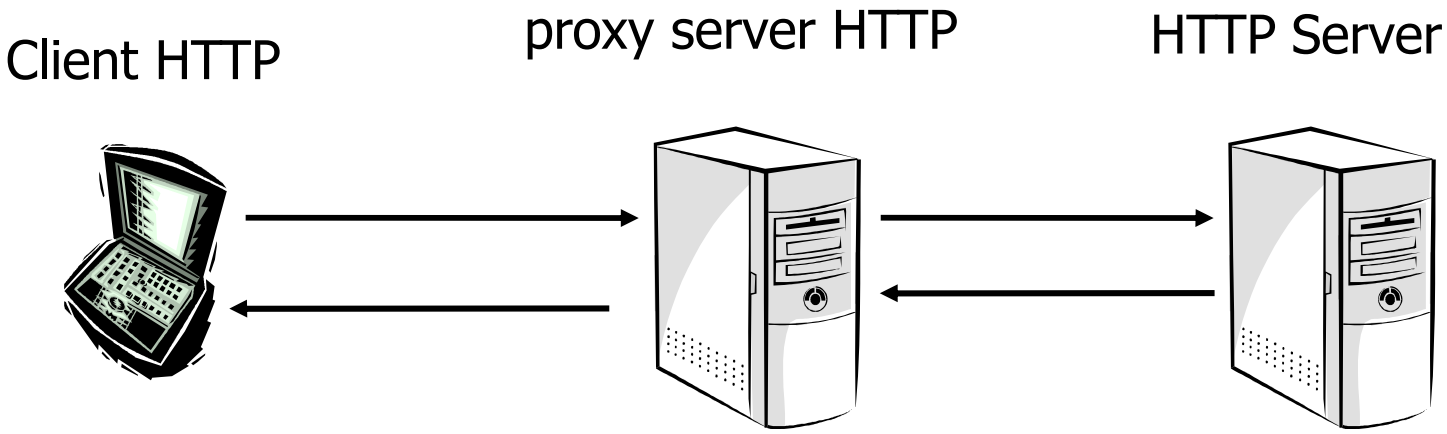
ETag header is to determine if a cached version of the requested resource is identical to the current version on the server

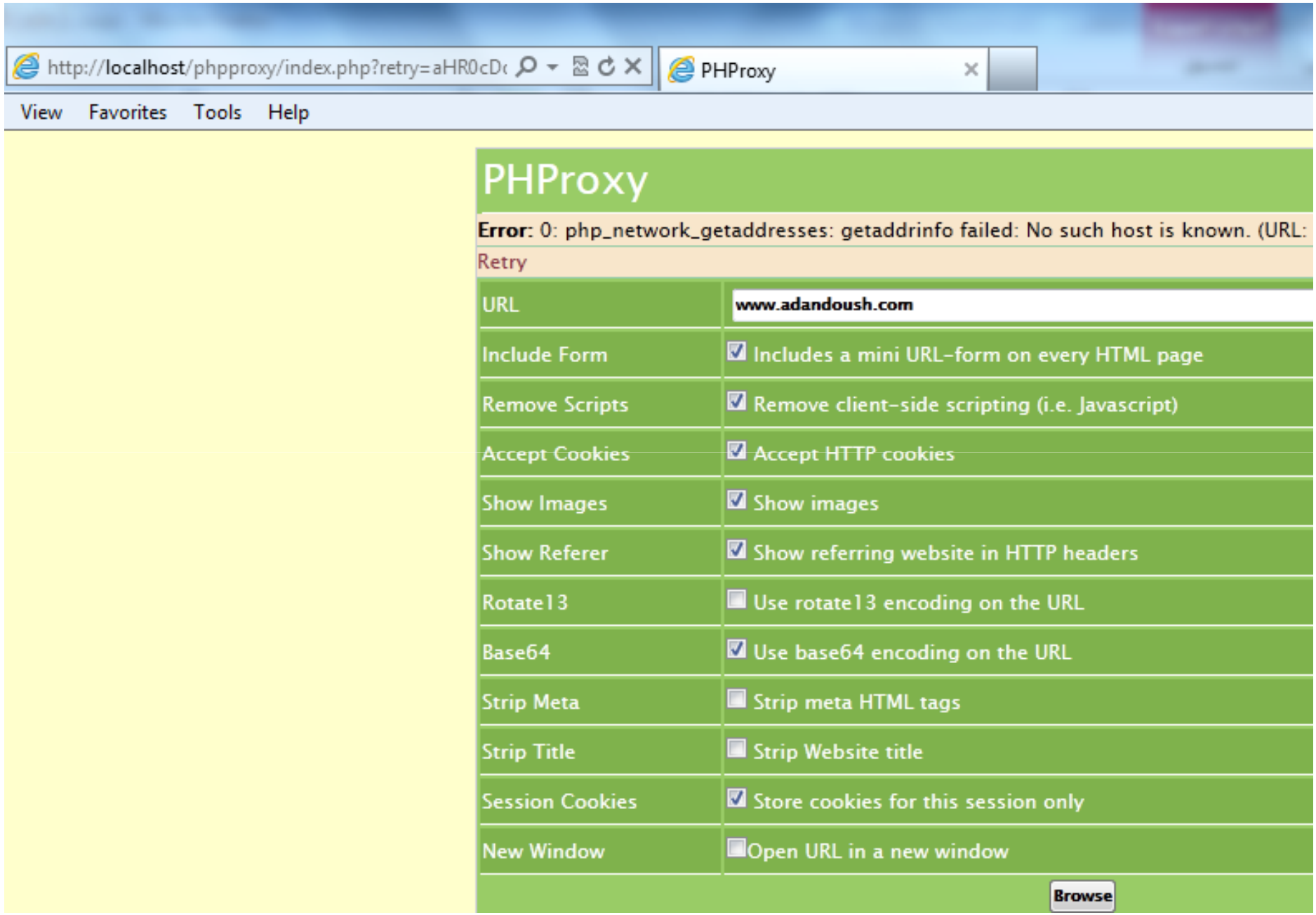
Content-Type specifies the Internet media type of the data conveyed by the http message, while ***Content-Length*** indicates its length in bytes

Connection*: close,* means that the web server will close the TCP connection immediately after the transfer of this response.

HTTP Proxy

- Functionalities :
  - facilitate communication when clients without a globally routable address that are located in private networks.
  - Cache most visited pages to improve response time
  - Filters




http://localhost/phpproxy/index.php?retry=aHR0cD
PHProxy
View
Favorites
Tools
Help
PHProxy
Error: 0: php_network_getaddresses: getaddrinfo failed: No such host is known. (URL:
Retry
URL
www.adandoush.com
Include Form
Includes a mini URL-form on every HTML page
Remove Scripts
Remove client-side scripting (i.e. Javascript)
Accept Cookies
Accept HTTP cookies
Show Images
Show images
Show Referer
Show referring website in HTTP headers
Rotate13
Use rotate13 encoding on the URL
Base64
Use base64 encoding on the URL
Strip Meta
Strip meta HTML tags
Strip Title
Strip Website title
Session Cookies
Store cookies for this session only
New Window
Open URL in a new window
Browse


http://localhost/phpproxy/index.php?q=d3d3LmFkY
A. dandoush blog's
View Favorites Tools Help
A. dandoush blog's
Subscribe
Home
About me
Contact Me
Courses & marking schemes
Private
Register
CATEGORIES
research
Teaching
Android App
Network Managment
Network Programming
Projects
System Administration
RECENT POSTS
Endianness & Byte/Network Order Conversion
How to restore Grub after formating windows from a live Ubuntu cd
Android – Apps can easily embed the web
Endianness & Byte/Network Order Conversion
October 16, 2011 By: admin Category: Network Programming, Teaching
For my students in the 5th year of CS in Tishreen University: Refere to these two articles in order to understand the importance of the APIs presented in P17 of the network programming lect3.
http://en.wikipedia.org/wiki/Endianness
http://www.gnu.org/s/hello/manual/libc/Byte-Order.html
Cheers,
Share / Save
Comment (1)
How to restore Grub after formating windows from a live Ubuntu cd
META
Register
Log in
Entries RSS
Comments RSS
WordPress.org
WHAT'S ON YOUR MIND?
BLOGROLL
code Ghar
Complete FreeBSD
install ns 2.35 in ubuntu
International Teletraffic

Proxy Caching Algorithms

Web Caching Benefits

Web Caching

web caching provides an efficient remedy to the latency problem and network traffic by bringing documents closer to clients.

There are many benefit of proxy caching. It reduces network traffic, average latency of fetching Web documents, and the load on busy Web servers.

Web Caching Location

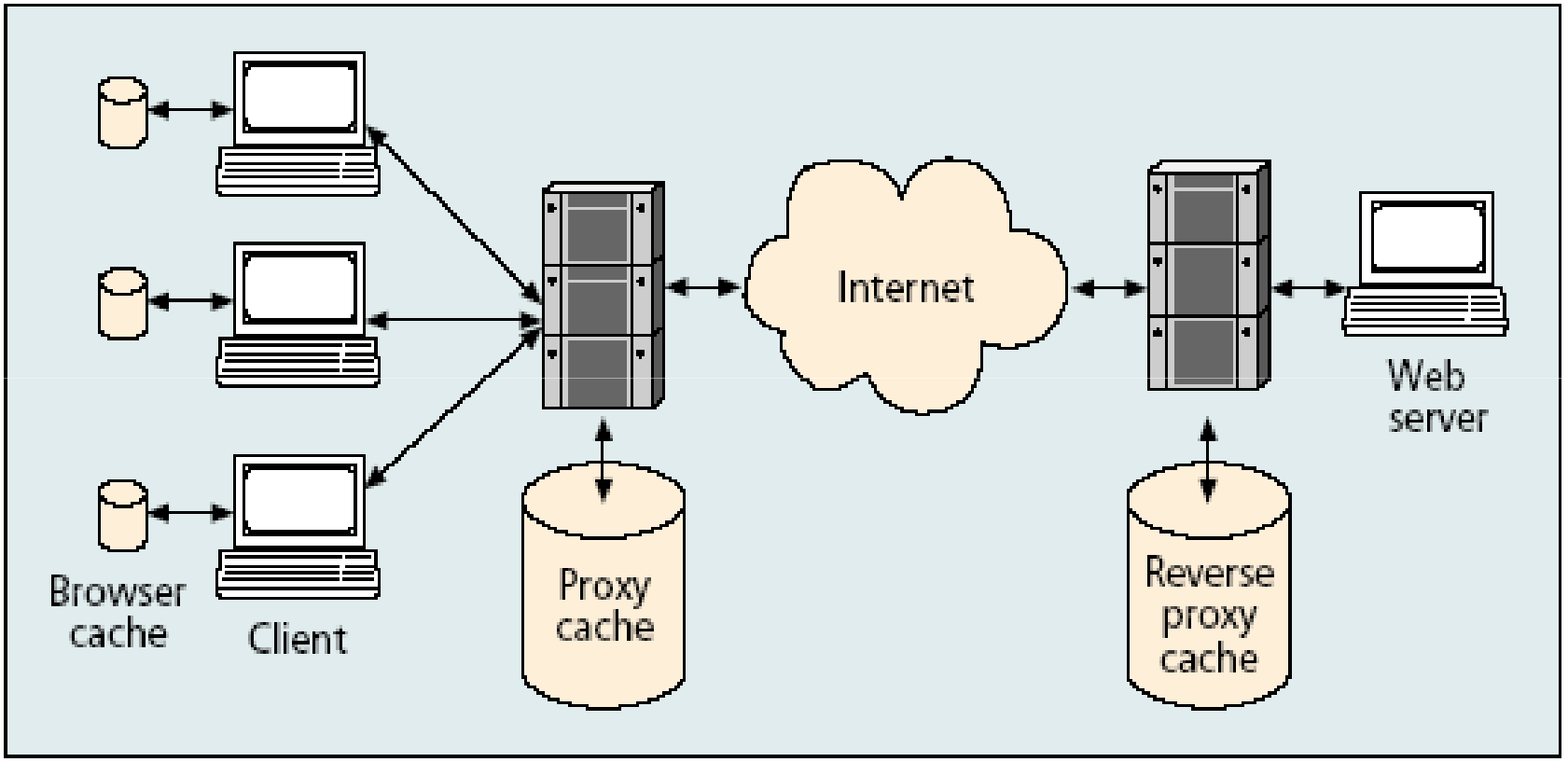


FIGURE 1. *Possible locations for deploying WWW caching.*

Web Caching Replacement Algorithm

- effective use of caching, an informative decision has to be made to evict document from the cache in case of cache saturation.
- key to the effectiveness of proxy caches that can yield high hit ratio.
- differ to page replacement. Why?

Characteristic

- Web caching is variable-size caching
- The cost of retrieving missed Web documents from their original servers depends on many factors.
- Web documents are frequently updated
- Zipf-like popularity of web documents

Key Parameters

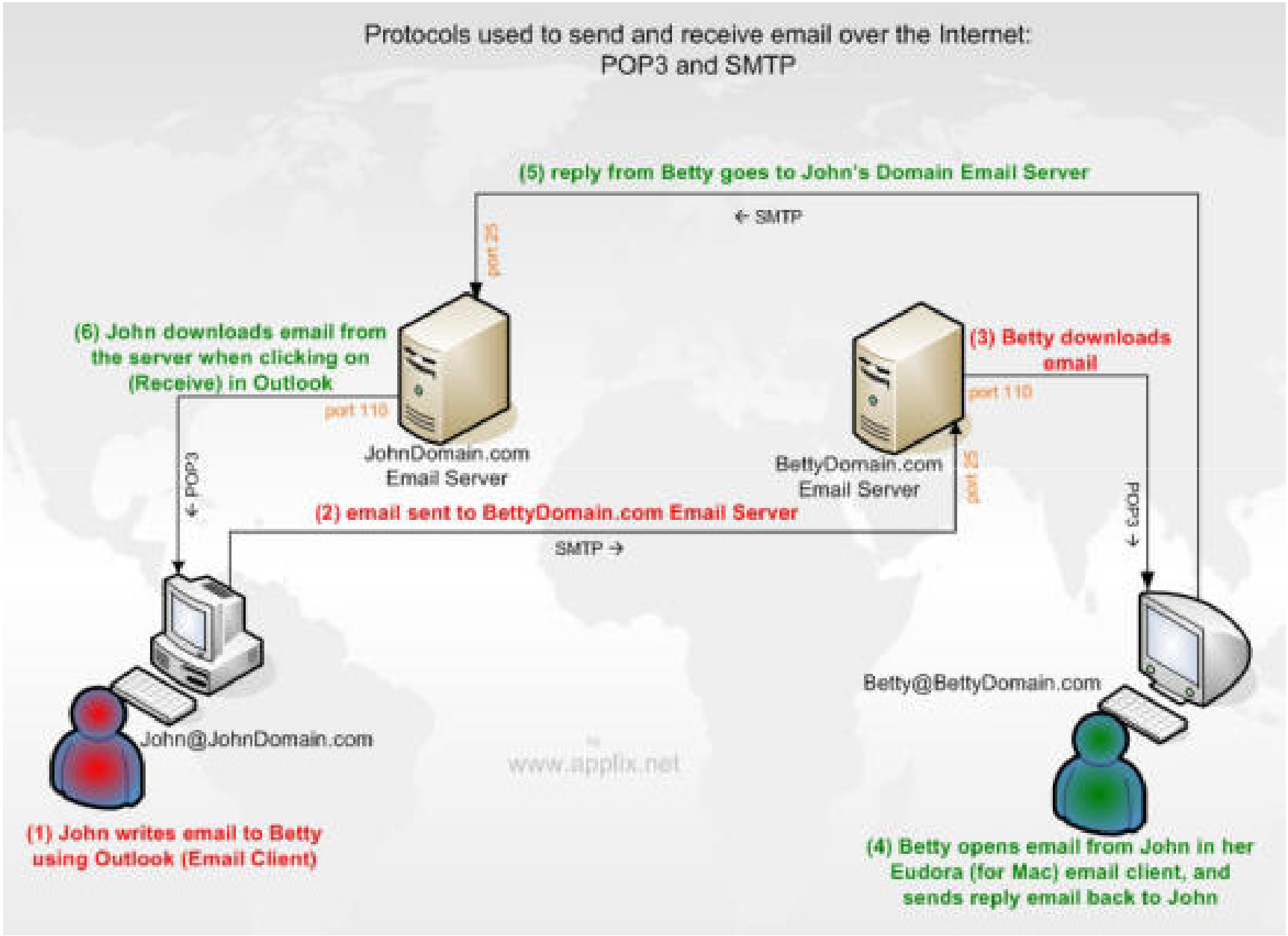
There are four key parameters that most proxy replacement policies considering in design

1. Frequency Information
2. Recency Information
3. Document size
4. Network cost

SMTP and POP3

- Simple Mail Transfer Protocol (port 25)
  - deliver email from our Email Client (send)
  - RFC 5321 (2008) includes the extended SMTP (ESMTP) on port 587
- Post Office Protocol v3 (port 110)
  - handle email between Email Server and our Local Email Client (like Outlook or Eudora)
  - authenticate our credentials on the server and download emails
    - Internet Message Access Protocol (IMAP) instead
- Very simple and general example shown in the next figure:

John@JohnDomain.com to Betty@BettyDomain.com


Protocols used to send and receive email over the Internet:
POP3 and SMTP
(5) reply from Betty goes to John's Domain Email Server
← SMTP
port 25
(6) John downloads email from the server when clicking on (Receive) in Outlook
port 110
← POP3
JohnDomain.com
Email Server
(3) Betty downloads email
port 110
port 25
BettyDomain.com
Email Server
(2) email sent to BettyDomain.com Email Server
SMTP →
POP3 →
John@JohnDomain.com
Betty@BettyDomain.com
(1) John writes email to Betty using Outlook (Email Client)
(4) Betty opens email from John in her Eudora (for Mac) email client, and sends reply email back to John

SMTP and POP3

- In details:
- John’s e-mail ID is John and he has account on JohnDomain.com
- John wats to send email using e-mail client like Outlook Express to Betty who has account on BettyDomain.com and who uses
- mail server config for John is: mail.JohnDomain.com
- When John compose a message and press the Send button, here's what happens:

SMTP and POP3

1. Outlook Express connects to the SMTP server at mail.JohnDomain.com using port 25.
2. Outlook Express has a conversation with the SMTP server, telling the SMTP server the address of the sender and the address of the recipient, as well as the body of the message.
3. The SMTP server takes the "to" address (Betty@BettyDomain.com) and breaks it into two parts: the recipient name (Betty) and the domain name (BettyDomain.com).

SMTP and POP3

- If the "to" address had been another user at JohnDomain.com, the SMTP server would simply hand the message to the POP3 server for JohnDomain.com (using a little program called the delivery agent).
- Since the recipient is at another domain, SMTP needs to communicate with that domain.

1. The SMTP server has a conversation with a DNS.

- It says, "Can you give me the IP address of the SMTP server for BettyDomain.com?" The DNS replies with the one or more IP addresses for the SMTP server(s) that BettyDomain operates.

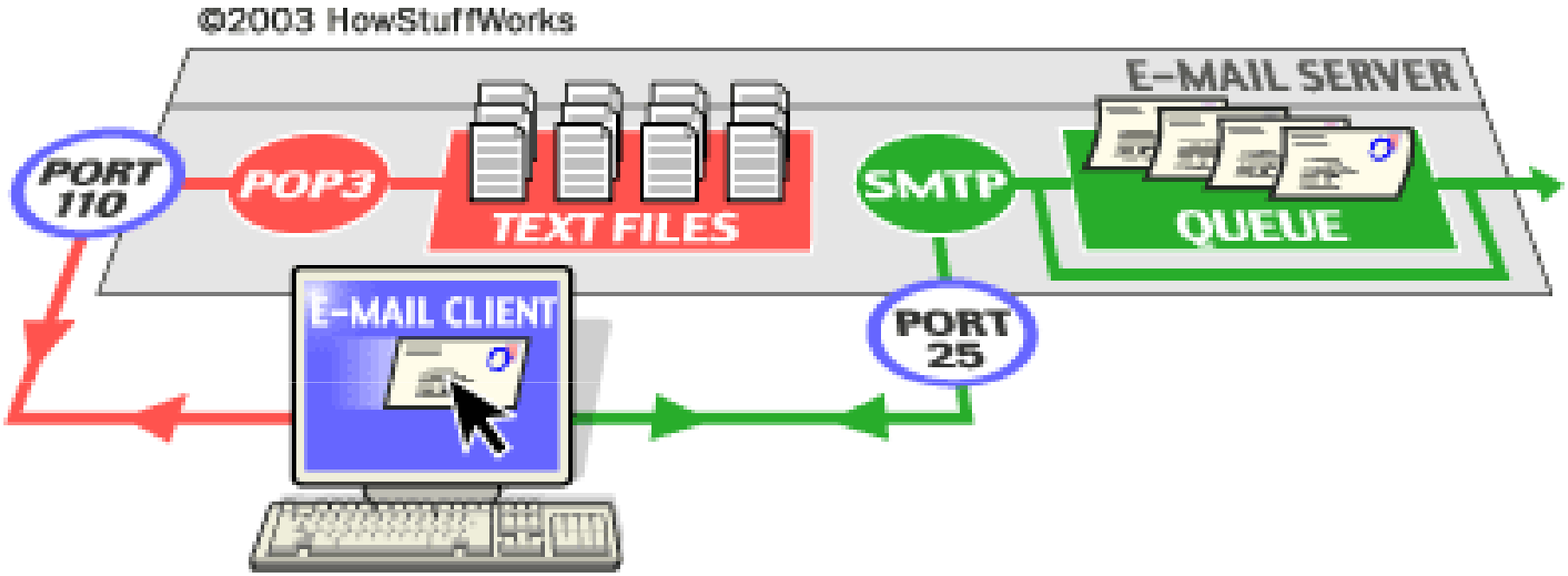
SMTP and POP3

1. The SMTP server at JohnDomain.com connects with the SMTP server at BettyDomain.com using port 25 (through TCP session that starts with a greeting by the server using “HELO CMD ”). It has the same simple text conversation that John e-mail client had with the SMTP server for JohnDomain, and gives the message to the BettyDomain server. The BettyDomain server recognizes that the domain name for Betty is at BettyDomain, so it hands the message to BettyDomain's POP3 server, which puts the message in Betty's mailbox.

SMTP and POP3

- If, for some reason, the SMTP server at JohnDomain cannot connect with the SMTP server at BettyDomain, then the message goes into a queue. The SMTP server on most machines uses a program called sendmail to do the actual sending, so this queue is called the sendmail queue

- See the next figure


©2003 HowStuffWorks
E-MAIL SERVER
PORT 110
POP3
TEXT FILES
SMTP
QUEUE
E-MAIL CLIENT
PORT 25

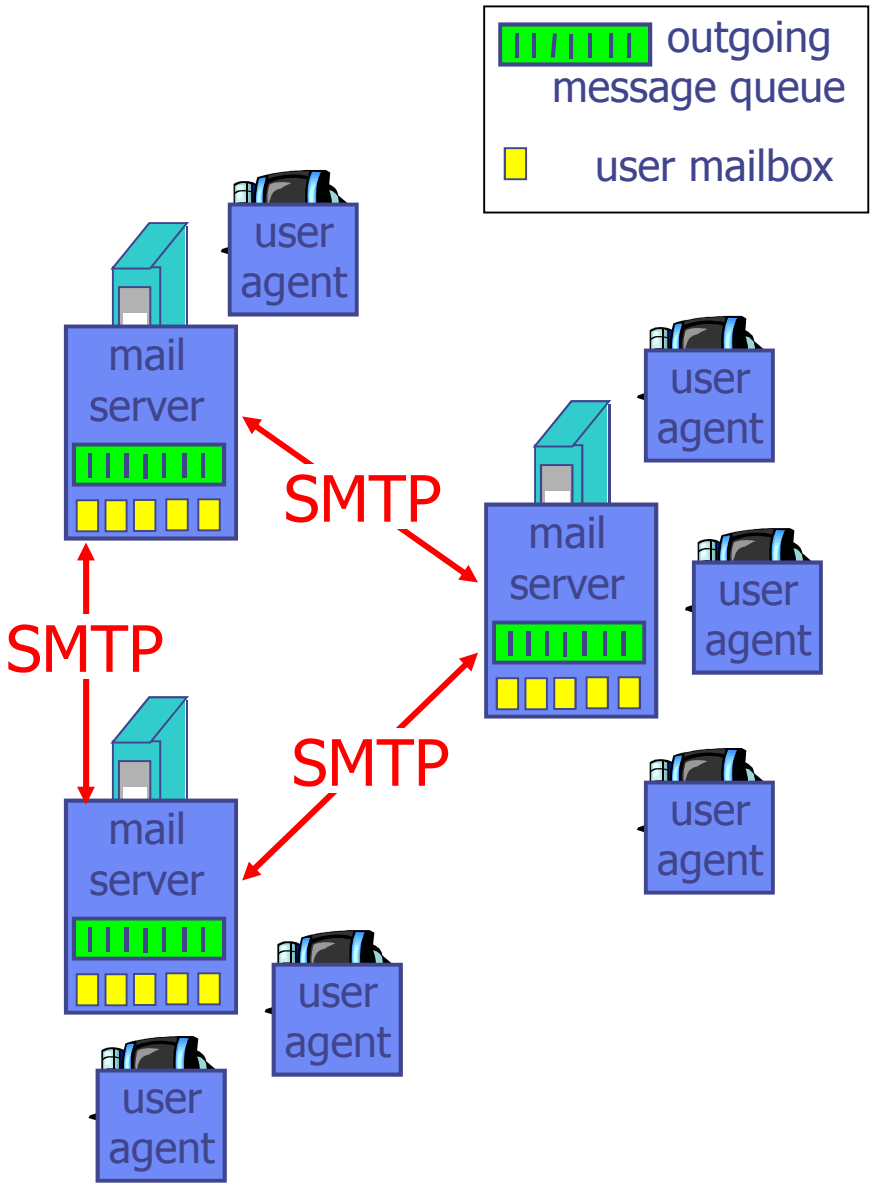
Electronic Mail

Three major components:

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

User Agent

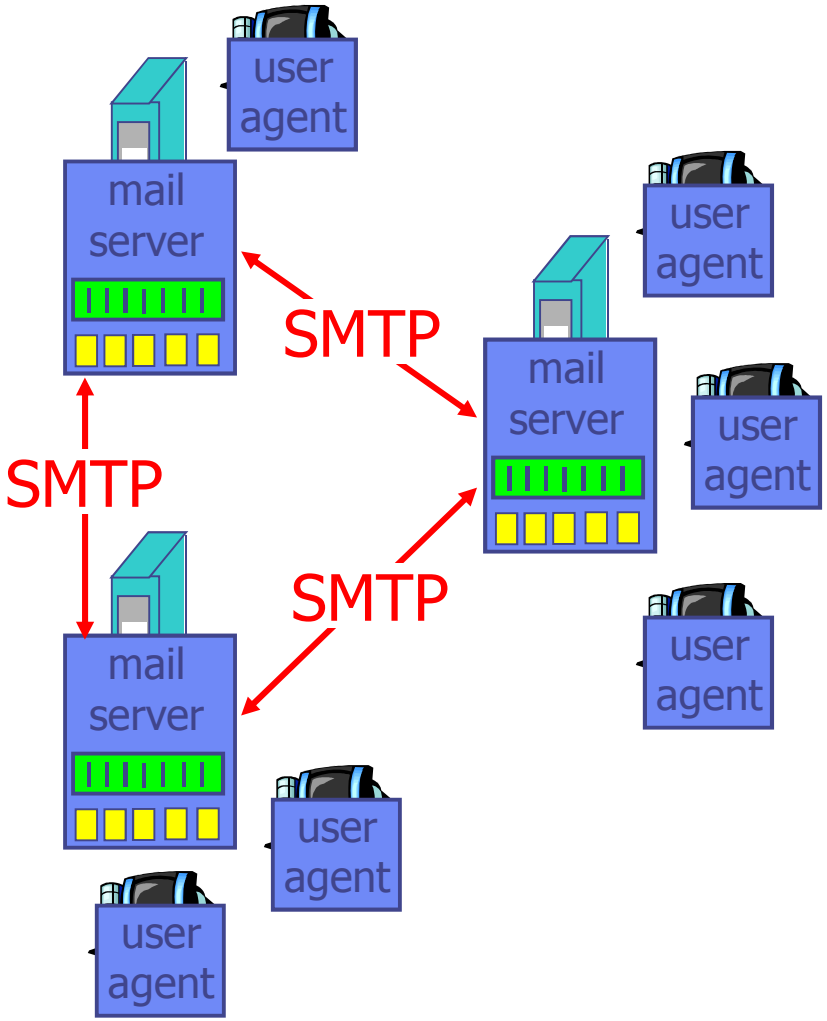
- "mail reader"
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- outgoing, incoming messages stored on server



Electronic Mail: mail servers

Mail Servers

- mailbox contains incoming messages for user
- messagequeue 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]

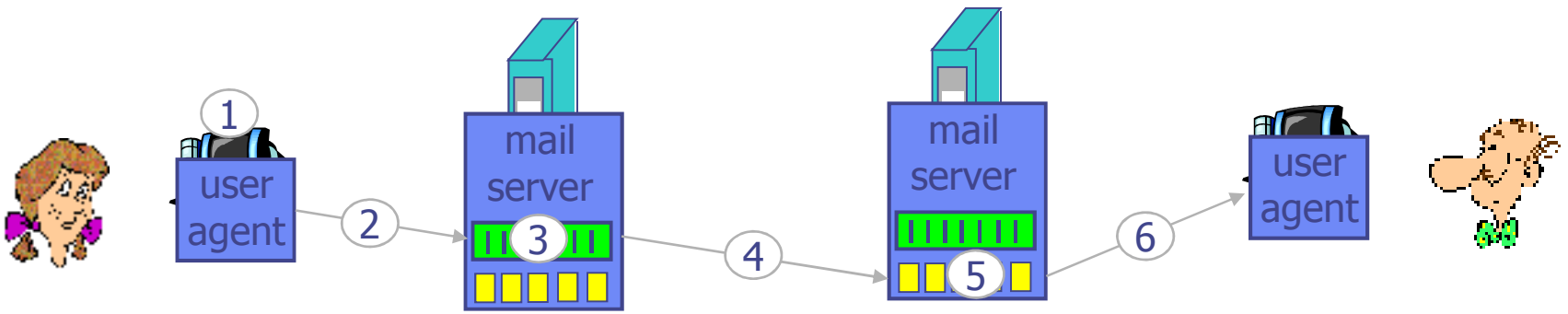
- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

SMTP and POP3

- The most common commands are:
  - HELO - introduce yourself
  - EHLO - introduce yourself and request extended mode
  - MAIL FROM: - specify the sender
  - RCPT TO: - specify the recipient
  - DATA - specify the body of the message (To, From and Subject should be the first three lines.)
  - QUIT - quit the session
  - VRFY - verify an address
- A typical example of sending a message via SMTP to two mailboxes (*alice* and *theboss*) located in the same mail domain (*example.com*) is reproduced in the following session exchange.

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
```

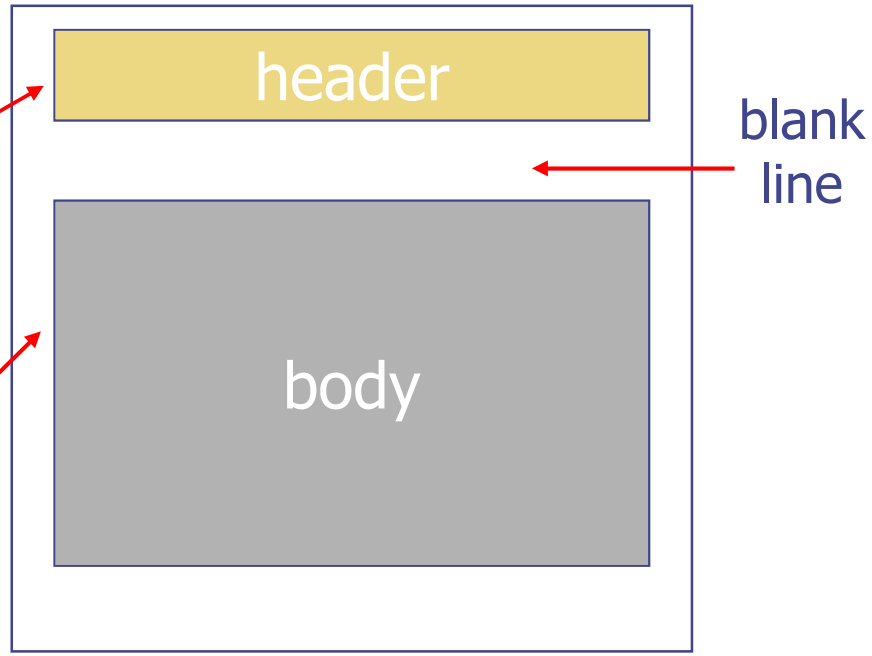
Mail message format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

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

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



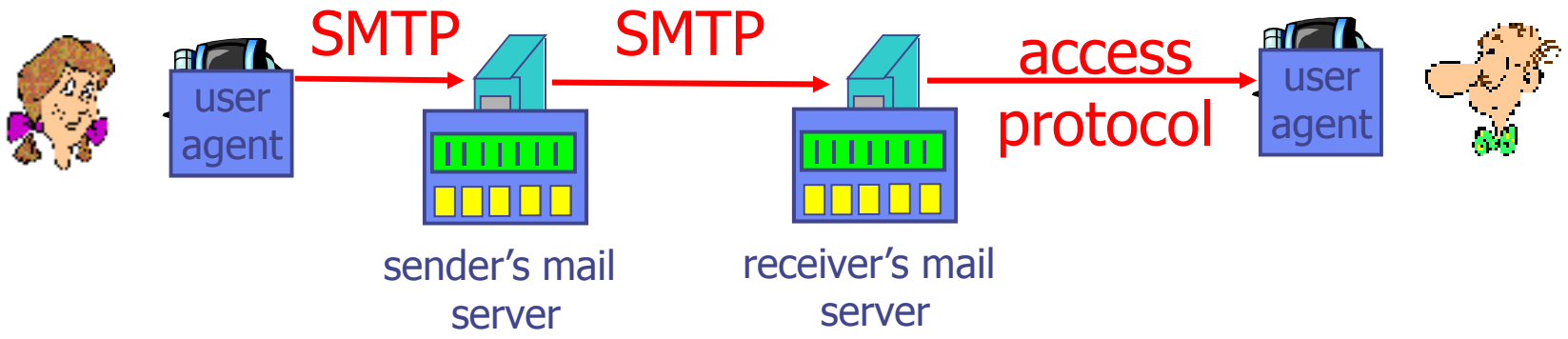
SMTP: final words

- 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 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 1730]
    - 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 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more) and IMAP

More about POP3

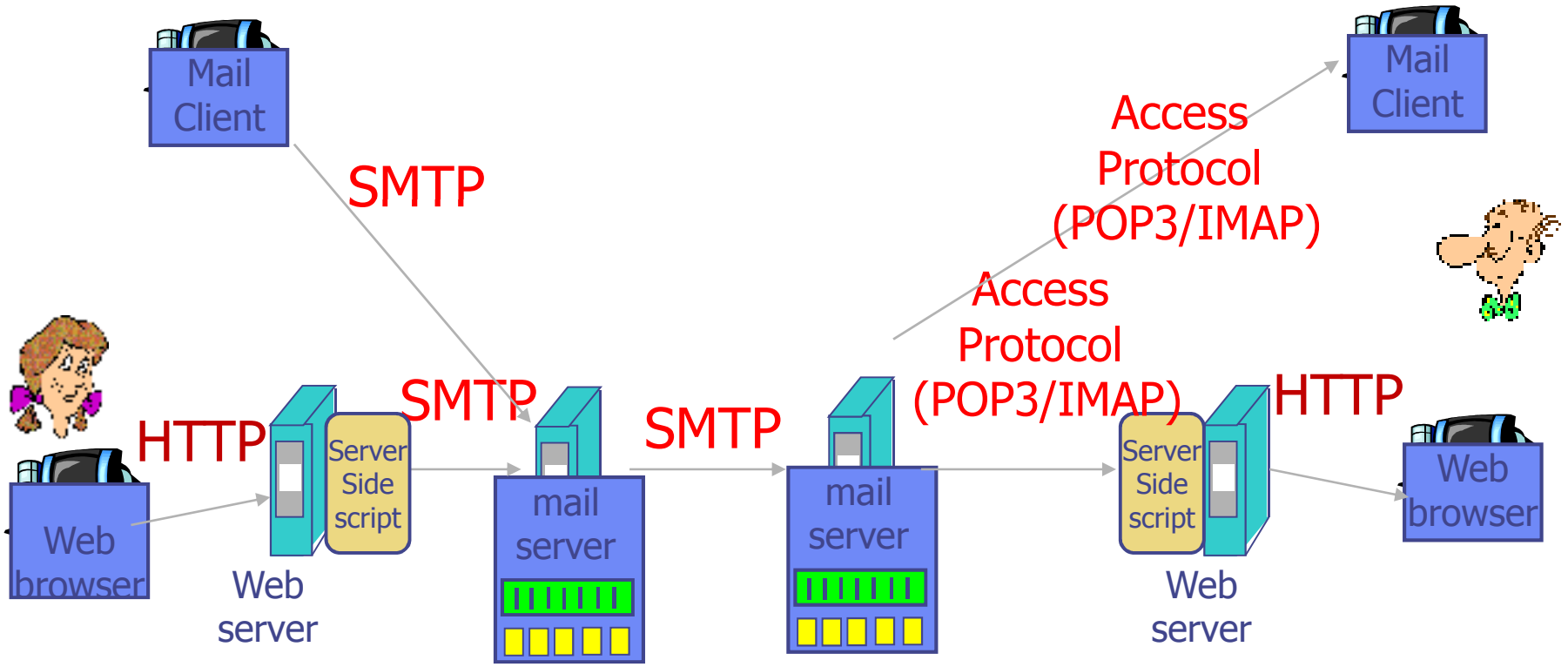
- Previous example uses "download and delete" mode.
- Bob cannot re-read e-mail if he changes client
- "Download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions

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

Web-Based Mail Access

- Comparison of webmail providers
  - http://en.wikipedia.org/wiki/Comparison_of_webmail_providers



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)

Check out the following link for a list of mail servers from major email service providers

http://www.emailaddressmanager.com/tips/mail-settings.html

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.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

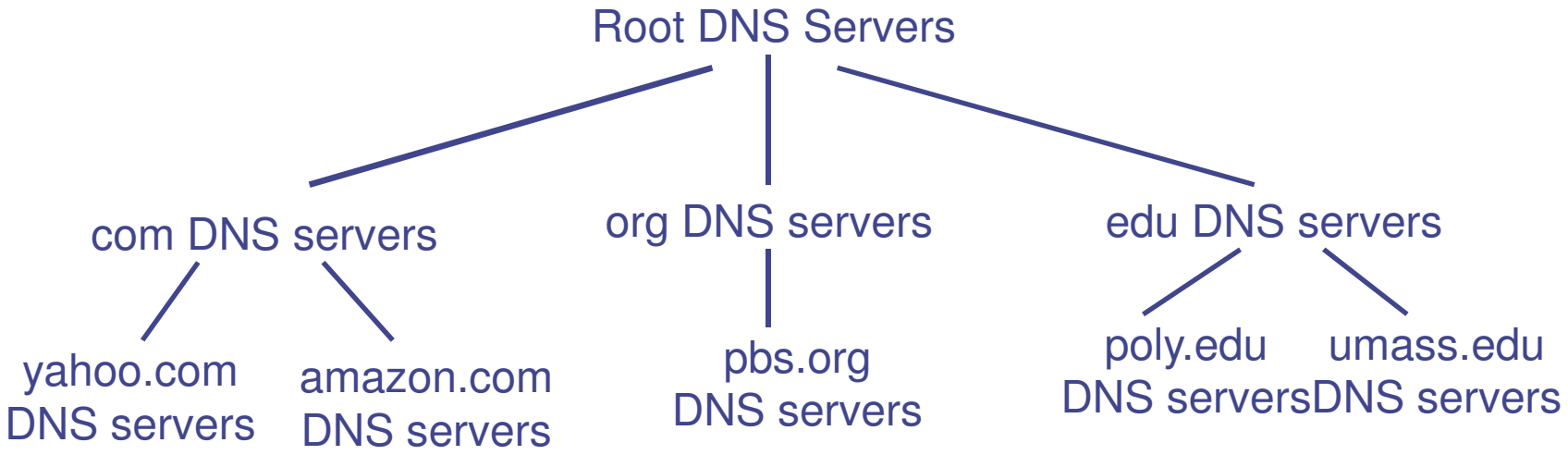
- hostname to IP address translation
- host aliasing
  - Canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

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

doesn't *scale!*

Distributed, Hierarchical Database

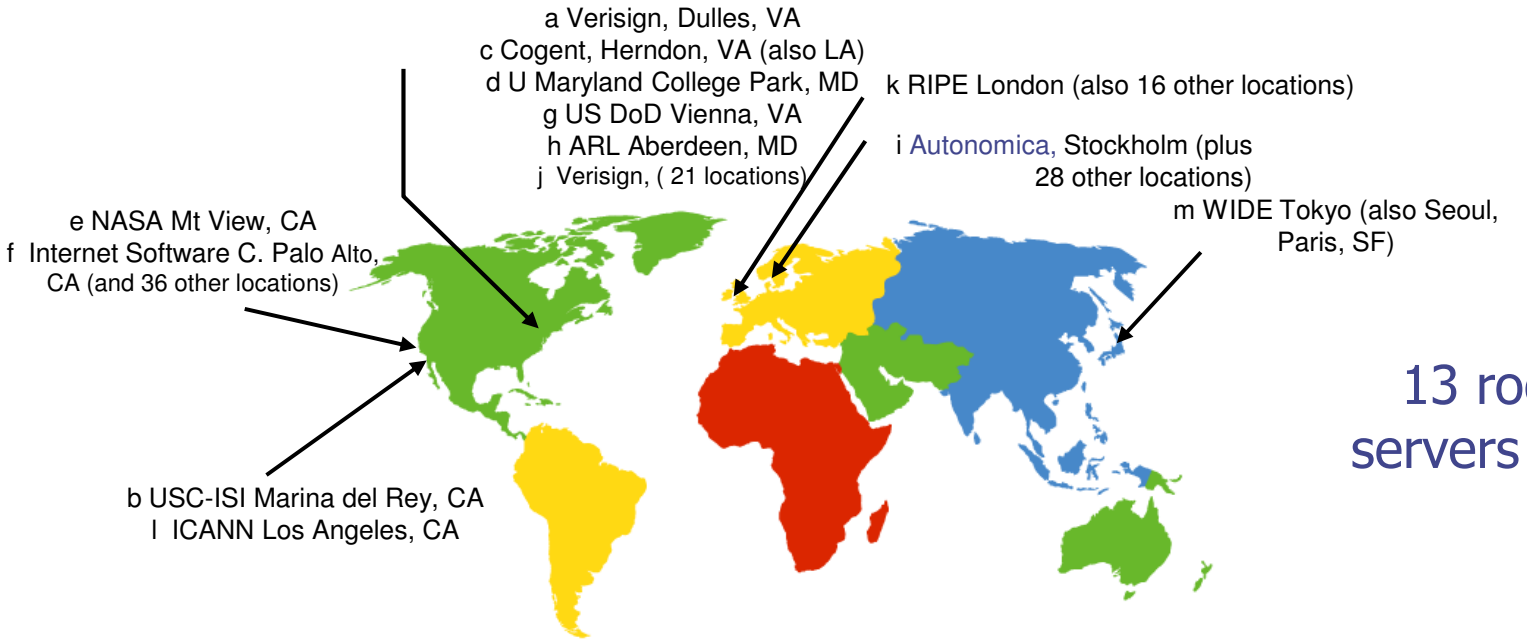


Client wants IP for www.amazon.com; 1st approx:

- 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

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.
  - Network Solutions 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, mail).
  - can be maintained by organization or service provider

Local Name Server

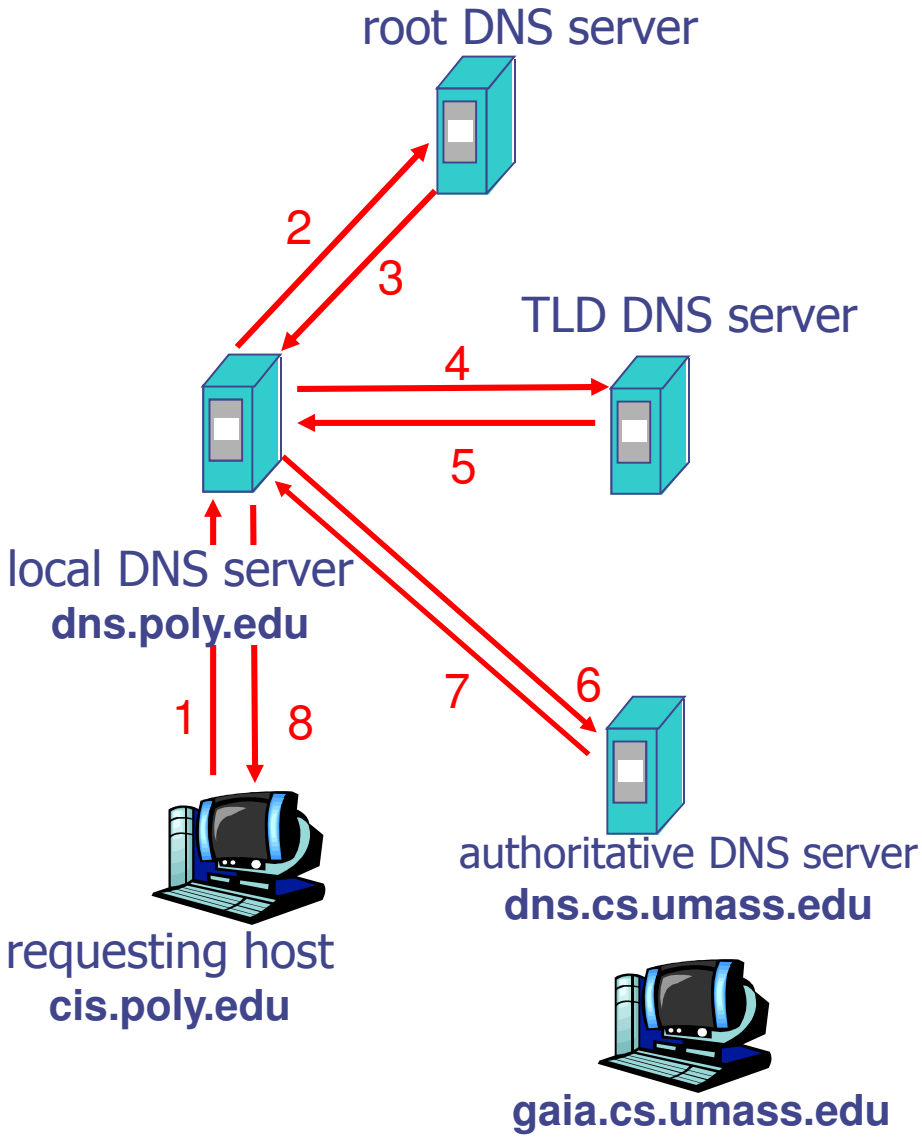
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one.
  - also called “default name server”
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

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

iterative query:

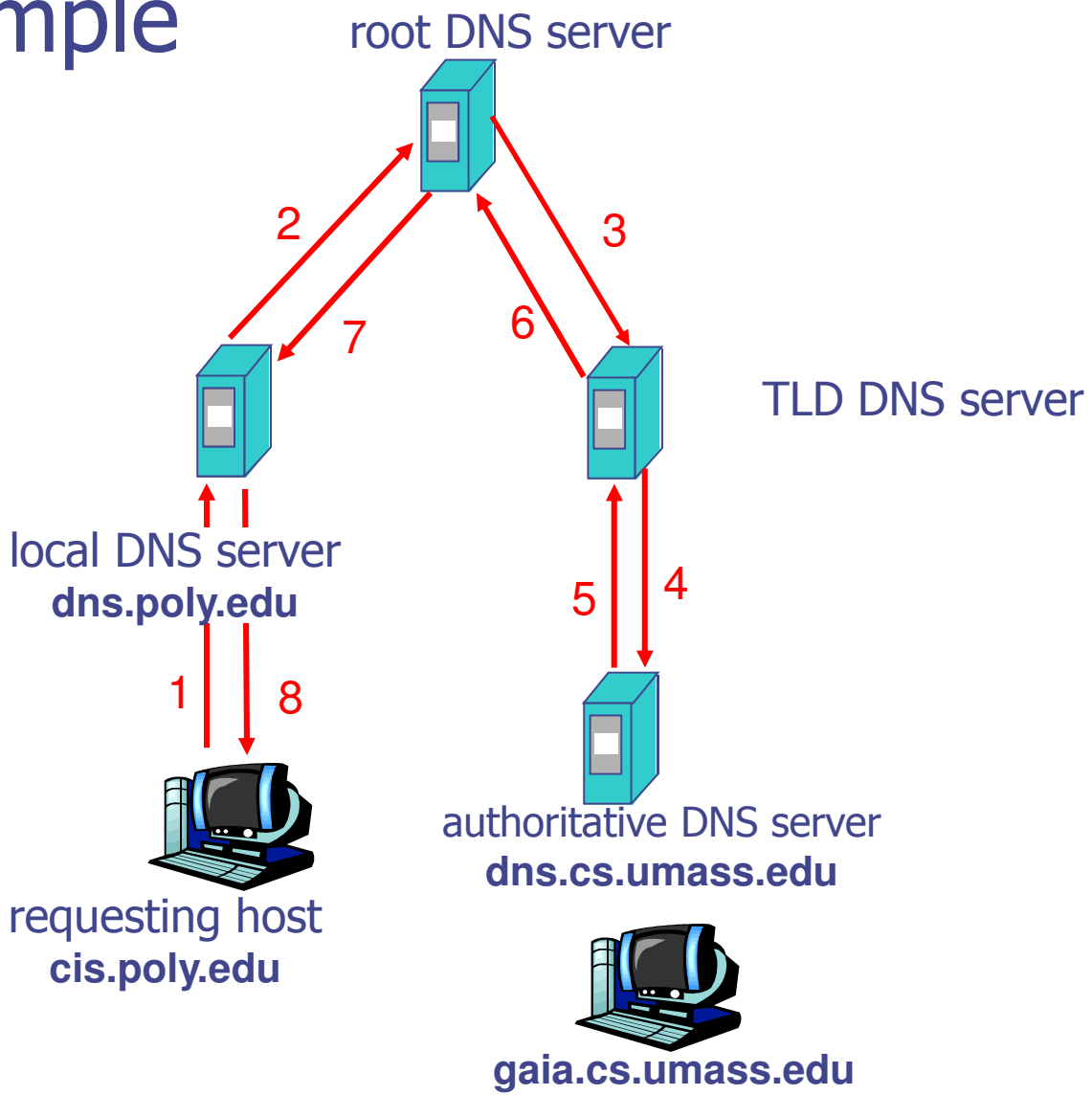
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server



DNS: caching and updating records

- once (any) name server learns mapping, it *caches* 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 under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

DNS records

DNS: distributed db storing resource records (RR)

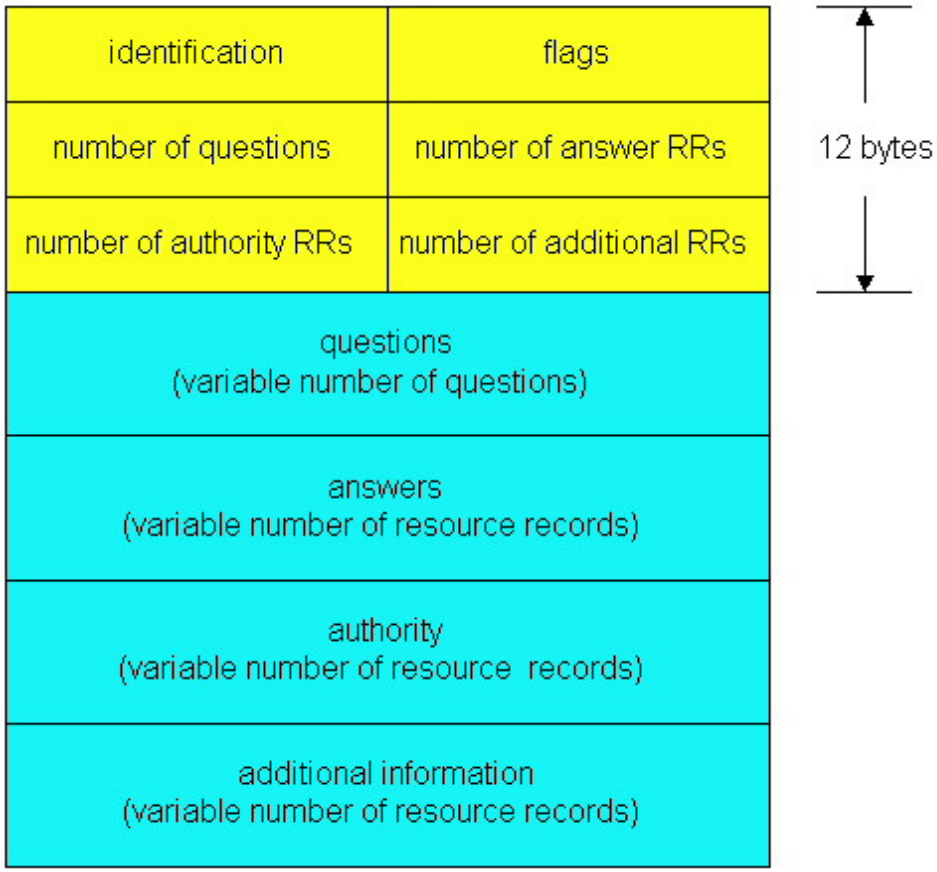
RR format: **`(name, value, type, ttl)`**

- 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`**

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



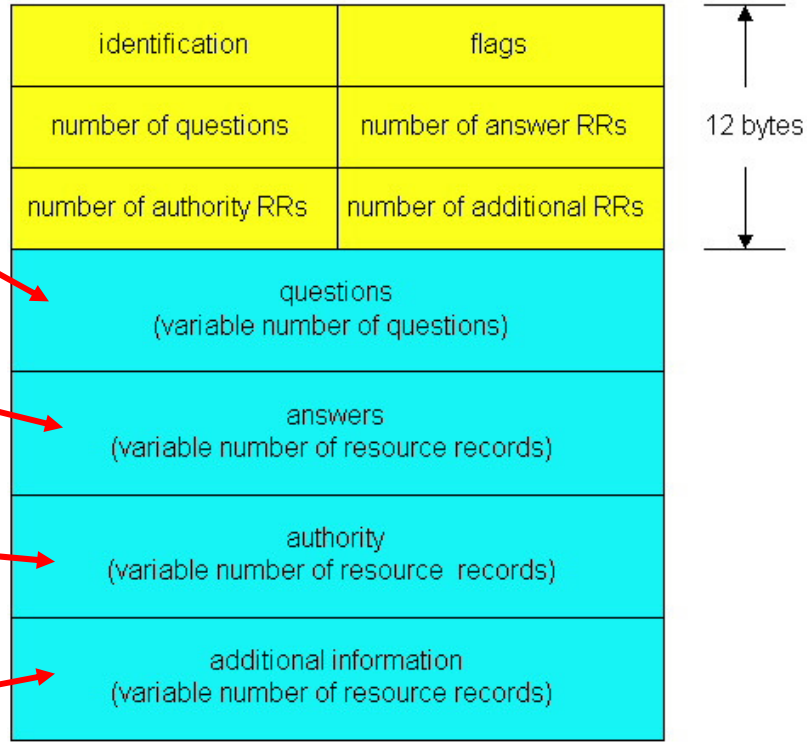
DNS protocol, messages

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



Inserting records into DNS

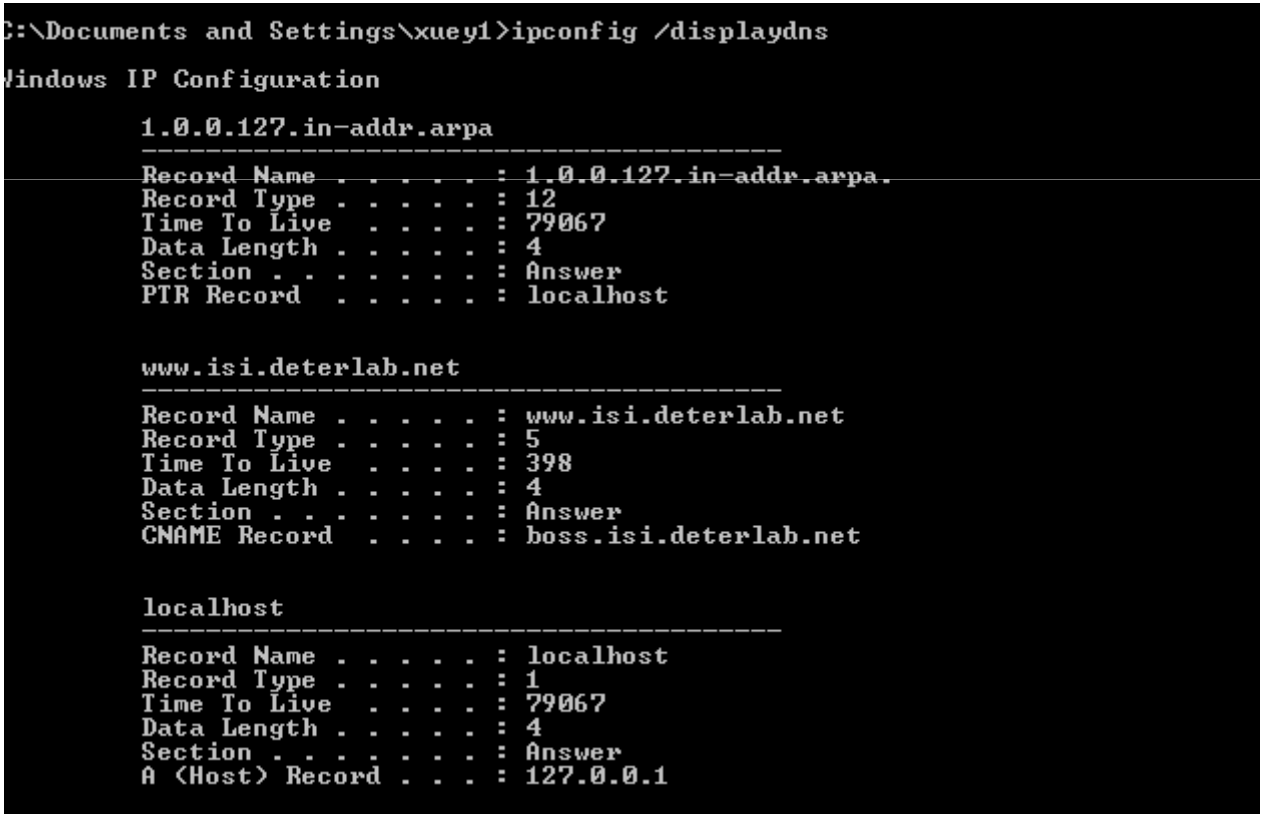
- example: new startup “Network Utopia”
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into com TLD server:

```
(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

From Principle to Practice

- Examine the DNS cache on local machine
  - On Windows, use command "ipcofigure /displaydns"
- Flush the DNS cache "ipcofigure /flushdns"



From Principle to Practice

- Use tool "nslookup" to query the DNS system

```
C:\Documents and Settings\xuey1>nslookup
Default Server:  jffs4.vuse.vanderbilt.edu
Address:  129.59.90.20

> help
Commands:   (identifiers are shown in uppercase, [] means optional)
NAME            - print info about the host/domain NAME using default server
NAME1 NAME2     - as above, but use NAME2 as server
help or ?       - print info on common commands
set OPTION      - set an option
    all                 - print options, current server and host
    [no]debug           - print debugging information
    [no]d2              - print exhaustive debugging information
    [no]defname         - append domain name to each query
    [no]recurse         - ask for recursive answer to query
    [no]search          - use domain search list
    [no]vc              - always use a virtual circuit
    domain=NAME         - set default domain name to NAME
    srchlist=N1[/N2/.../N6] - set domain to N1 and search list to N1,N2, etc.
    root=NAME           - set root server to NAME
    retry=X             - set number of retries to X
    timeout=X           - set initial time-out interval to X seconds
    type=X              - set query type (ex. A,ANY,CNAME,MX,NS,PTR,SOA,SRV)
    querytype=X         - same as type
    class=X             - set query class (ex. IN (Internet), ANY)
    [no]msxfr           - use MS fast zone transfer
    ixfrver=X           - current version to use in IXFR transfer request
server NAME     - set default server to NAME, using current default server
lserver NAME    - set default server to NAME, using initial server
finger [USER]   - finger the optional NAME at the current default host
root            - set current default server to the root
ls [opt] DOMAIN [> FILE] - list addresses in DOMAIN (optional: output to FILE)
    -a          -  list canonical names and aliases
    -d          -  list all records
    -t TYPE     -  list records of the given type (e.g. A,CNAME,MX,NS,PTR etc.)
view FILE           - sort an 'ls' output file and view it with pg
exit            - exit the program

> www.google.com
Server:  jffs4.vuse.vanderbilt.edu
Address:  129.59.90.20

Non-authoritative answer:
Name:    www.l.google.com
Addresses:  74.125.45.103, 74.125.45.104, 74.125.45.105, 74.125.45.106
          74.125.45.147, 74.125.45.99
Aliases:  www.google.com
```

Checkout the DNS query and reply using Wireshark

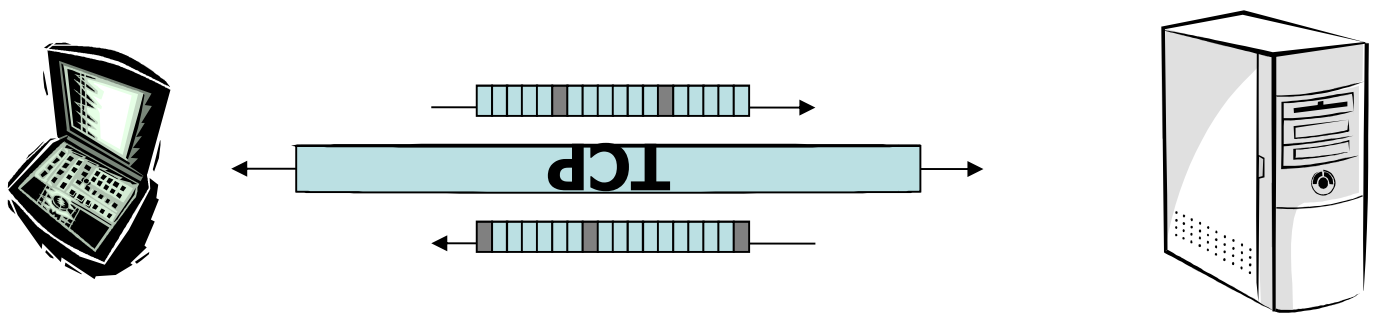
```
⊟ Domain Name System (query)
    [Response In: 136]
    Transaction ID: 0x847c
  ⊞ Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊟ www.google.com: type A, class IN
        Name: www.google.com
        Type: A (Host address)
        Class: IN (0x0001)
```

```
Domain Name System (response)
  [Request In: 135]
  [Time: 0.010080000 seconds]
  Transaction ID: 0x847c
⊞ Flags: 0x8180 (Standard query response, No error)
  Questions: 1
  Answer RRs: 7
  Authority RRs: 4
  Additional RRs: 0
⊟ Queries
  ⊟ www.google.com: type A, class IN
      Name: www.google.com
      Type: A (Host address)
      Class: IN (0x0001)
⊟ Answers
  ⊟ www.google.com: type CNAME, class IN, cname www.l.google.com
      Name: www.google.com
      Type: CNAME (Canonical name for an alias)
      Class: IN (0x0001)
      Time to live: 6 days, 23 hours, 59 minutes, 19 seconds
      Data length: 8
      Primary name: www.l.google.com
  ⊟ www.l.google.com: type A, class IN, addr 74.125.45.99
      Name: www.l.google.com
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 4 minutes, 53 seconds
      Data length: 4
      Addr: 74.125.45.99 (74.125.45.99)
  ⊟ www.l.google.com: type A, class IN, addr 74.125.45.103
      Name: www.l.google.com
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 4 minutes, 53 seconds
      Data length: 4
      Addr: 74.125.45.103 (74.125.45.103)
  ⊞ www.l.google.com: type A, class IN, addr 74.125.45.104
  ⊞ www.l.google.com: type A, class IN, addr 74.125.45.105
  ⊞ www.l.google.com: type A, class IN, addr 74.125.45.106
  ⊞ www.l.google.com: type A, class IN, addr 74.125.45.147
⊟ Authoritative nameservers
  ⊟ google.com: type NS, class IN, ns ns1.google.com
      Name: google.com
      Type: NS (Authoritative name server)
      Class: IN (0x0001)
      Time to live: 1 day, 23 hours, 58 minutes
      Data length: 6
      Name server: ns1.google.com
  ⊞ google.com: type NS, class IN, ns ns3.google.com
  ⊞ google.com: type NS, class IN, ns ns4.google.com
  ⊞ google.com: type NS, class IN, ns ns2.google.com
```

TELNET

- TELNET = a Network protocol
- telnet = a program that uses the TELNET protocol
- TELNET
  - Permit to connect a client (text mode) to a server (commands interpreter)

Characters ASCII + characters of control



Client
(commands interface)

Server
(commands interpreter)

TELNET

- Port 23
- Data + control code & commands over the same TCP connection
- Network Virtual Terminal (NVT)
  - Virtual Representation of a generic terminal (standard keyboard, standard screen size, etc.)
- Negotiation the options between the client and the server
- Code ASCII : 33 char of commands + 95 visualisation =128
- 0xxx xxxx state & code of control
- 1xxx xxxx cmd

Commands TELNET

- Commands :
  - IAC 255 (Interpret As Command) first cmd
  - EL 247 (Erase Line)
  - EC 246 (Erase Character)
  - IP 243 (Interrupt Process)
  - NOP 241 “No Operation”
  - AYT 246 “Are You There”
  - AO 245 "Abort Output”
  - IP 244 “Interrupt Process”
  - BRK 243 “Break”
  - …

Exemple connexion telnet

```
Z:\users>telnet if-4433.insa-lyon.fr
Login: SP1321
Password: ********
*===============================================================
Bienvenue à Microsoft Telnet Server.
*===============================================================
C:\>netstat

Connexions actives

  Proto  Adresse locale         Adresse distante       Etat
  TCP    if-4433:telnet         localhost:4342         TIME_WAIT
  TCP    if-4433:telnet         localhost:4352         ESTABLISHED
  TCP    if-4433:4352           localhost:telnet       ESTABLISHED
  TCP    if-4433:4143           servif-baie.insa-lyon.fr:microsoft-ds  ESTABLISH
ED
  TCP    if-4433:4145           cs27.msg.dcn.yahoo.com:5050  ESTABLISHED
  TCP    if-4433:4146           baym-cs17.msgr.hotmail.com:1863  ESTABLISHED
  TCP    if-4433:4170           servif-impr.insa-lyon.fr:netbios-ssn  ESTABLISHE
D
  TCP    if-4433:4306           csiges9.insa-lyon.fr:993  ESTABLISHED

C:\>exit

Perte de la connexion à l'hôte.

Z:\users>
```

Problems with TELNET

- Servers do not allow users to use TELNET
  - telnet can be used without control cmd
  - Exemple :
    - telnet www.wanadoo.fr 80
- Not secure connection
  - User name+ pass are captured easily (e.g. wireshark)

- adandoush@Ubuntu-server:~$ telnet Ubuntu-server

Trying ::1...

Trying 127.0.1.1...

Trying 192.168.1.100...

telnet: Unable to connect to remote host: Connection refused

- adandoush@Ubuntu-server:~$ telnet www.adandoush.com 80

Trying 66.84.14.67...

telnet: Unable to connect to remote host: Connection timed out

SSH

- Secure Shell: set of programs which employ public/private key (authenticating & encrypting sessions)
- Alternative to TELNET & Berkeley "r" utilities: rlogin, rcp, rsh
- Used as a way to "tunnel" other protocols to improve security against packet sniffing and "man in the middle" attacks
- Port 22
- ssh-keygen -> subdirectory $HOME/.ssh : identity and identity.pub, server side :$HOME/.ssh/authorized_keys

```
adandoush@Ubuntu-server:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adandoush/.ssh/id_rsa):
Created directory '/home/adandoush/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adandoush/.ssh/id_rsa.
Your public key has been saved in /home/adandoush/.ssh/id_rsa.pub.
The key fingerprint is:
c6:4b:70:0e:ff:d2:8e:ad:0d:4a:6b:bf:d7:0c:02:ea adandoush@Ubuntu-
    server
The key's randomart image is:
+--[ RSA 2048]----+
|      o .        |
|      .B         |
|     . .S        |
|    .  o.+.      |
|   .  . +.o+     |
|    E..o B. o    |
|     .o.=++      |
```

- adandoush@adandoush-laptop:~$ ls -l .ssh

-rw------- 1 adandoush adandoush 1679 2011-10-10 22:26 id_rsa

-rw-r--r-- 1 adandoush adandoush 405 2011-10-10 22:26 id_rsa.pub

- adandoush@adandoush-laptop:~$ scp .ssh/id_rsa.pub adandoush@Ubuntu-server:/home/adandoush

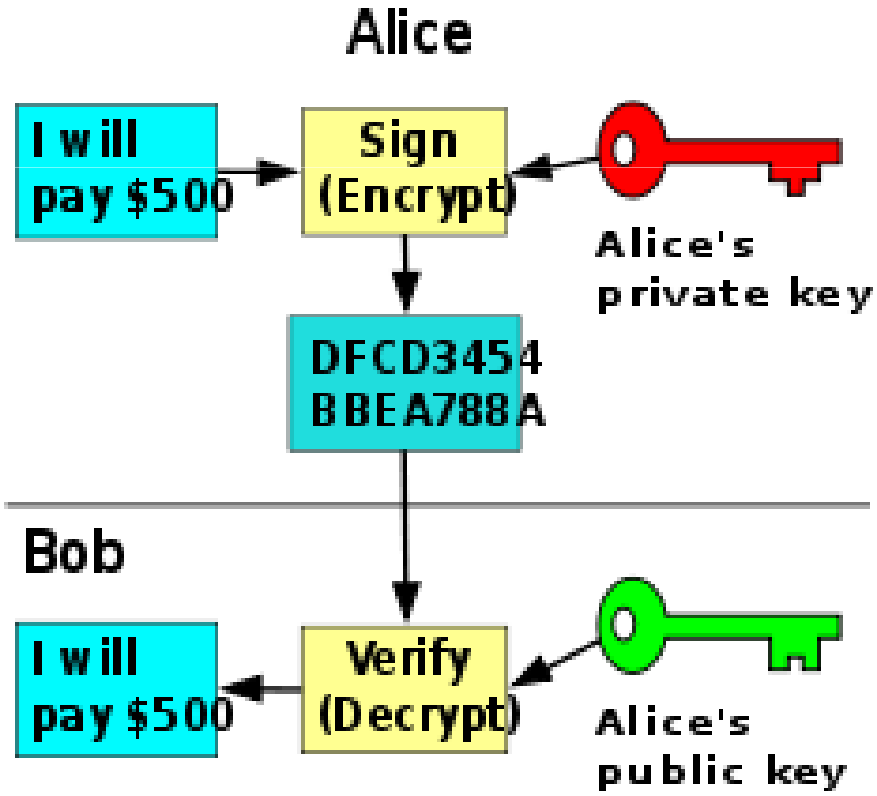
- root@Ubuntu-server:~$ cat /home/adndoush/id_rsa.pub > .ssh/authorized_keys

Public-key cryptography

- asymmetric key algorithms
- does not require a secure initial exchange of one or more secret keys.
- create a mathematically related key pair: a secret private key and a published public key

Public-key cryptography

- protection of the authenticity of a message → digital signature of a message using the private key, which can be verified using the public key



Public-key cryptography

- protection of the confidentiality and integrity of a message, by public key encryption, which can only be decrypted using the private key.