Lecture (10)
Application layer

Dr. Ahmed M. ElShafee

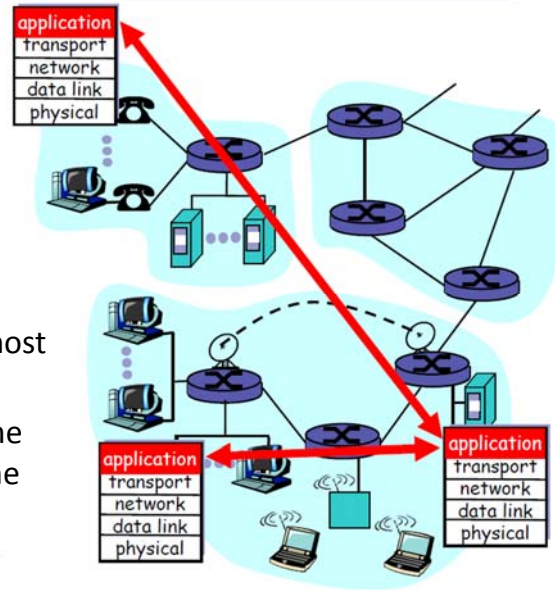
Agenda

- *
- *
- *

Introduction

Application:

- Distributed processes communicating by exchanging messages
- Ex: web, email,…
- application transport network data link physical
- A program running on a host is a process
- Two processes on the same host communicate thru the OS



- Processes running on different machines communicate by exchanging messages (of an application layer protocol).

Network Application architecture

Application Architectures:

- Client/server
- Peer-to-peer (p2p)
- Hybrid

Processes communication

- How processes running on different machines communicate (use of sockets)

Application protocols

- Define format and order of messages, actions taken
- Use services provided by lower layer protocols
- Ex: http, smtp,..

Client/Server Architecture

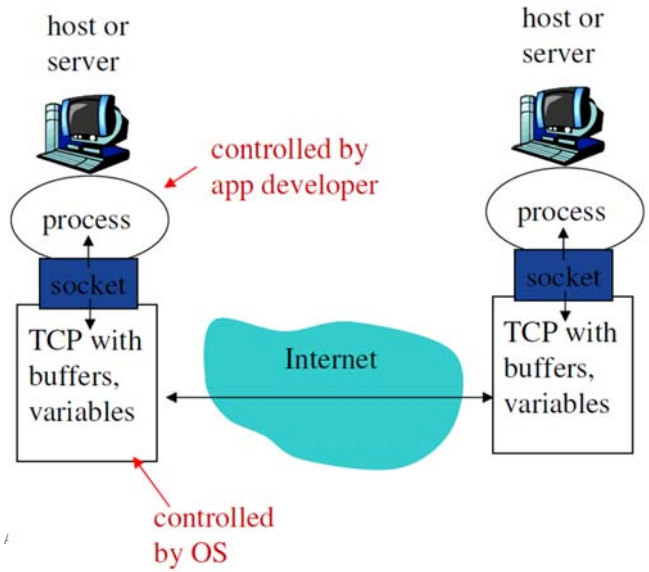
Client:

- Speaks first (communicate with server)
- Doesn't have to be always ON
- No direct communication between other clients
- Examples: web client, email reader

Server:

- Always ON
- Permanent IP address
- Examples: web server, mail server

- **Network API: Sockets**
- Processes communicate by sending/receiving data thru sockets
- Socket is a door between the process and the underlying transport protocol



How does a process identify another process on the net?

- IP address (alone is not enough)
- Port number (associated with the process. E.g. HTTP server: 80)
- 2 processes communicate by exchanging messages thru their sockets.

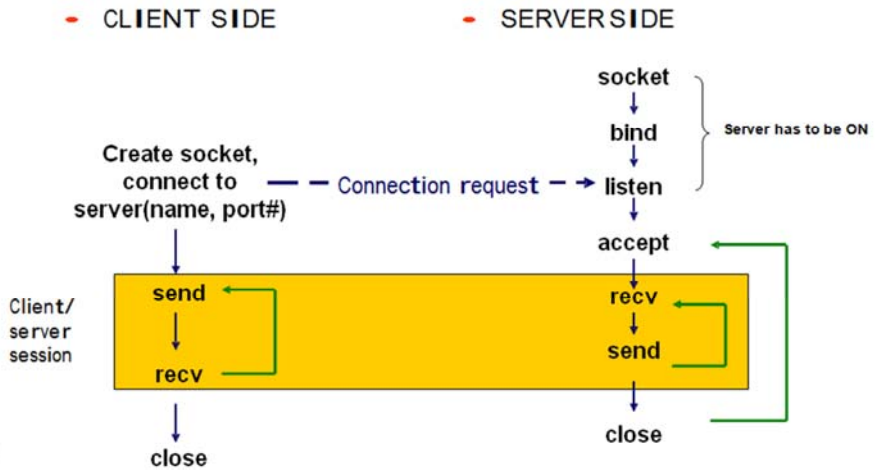
To create a server socket:

- Perform a passive open (ready to accept connection but not really establishing the connection) as follows:
- i.Bind
- ii.Listen
- iii.Accept

To create a client socket

- Perform an active open using “connect” command
- Use “send” and “recv” commands to send/receive messages on that connection

TCP Sockets in Client/Server Model

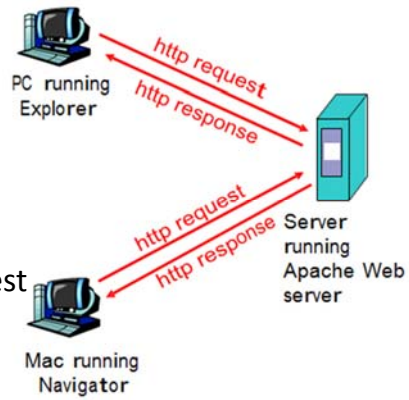


Internet Applications

- Hyper Text Transfer Protocol (HTTP)
- File Transfer Protocol (FTP)
- Simple Mail Transport Protocol (SMTP)
- Domain Name System (DNS)

Hyper Text Transfer Protocol (HTTP)

- Application layer protocol for www (major application on the Internet)
- Follows client/server model
  - Client: browser that requests, receives, displays object
  - Server: receives requests and responds to them
- HTTP: defines how web clients request web pages from the server and how web servers transfer web pages to clients
- Stateless protocol: Server doesn’t maintain information on clients



- A web page consists of a base HTML-file which includes several referenced objects (HTML file, JPEG image, Java applet, audio file,.)
- A web page consists of a base HTML-file which includes several referenced objects (HTML file, JPEG image, Java applet, audio file,.)Each object is addressable by a URL. A URL is composed of host name of the server and object s path name
- Example URL:

www.acu.edu.eg/images/aculogo.gif

host name path name

- Client: initiates TCP connection (creates socket) to server, port 80 (default port # for HTTP)
- Server: accepts TCP connection from client
- http messages exchanged between browser (http client) and Web server (http server)
- TCP connection closed

You enter the following : www.aelshafee.net/fall2012.html that contains text, references to fall2012.html objects

1a. http client initiates TCP connection to http server (process) at aelshafee.net

1b. http server at host www.guc.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. http client sends http request message (containing URL) into TCP connection socket (wants object aelshafee.net/fall2012.html)

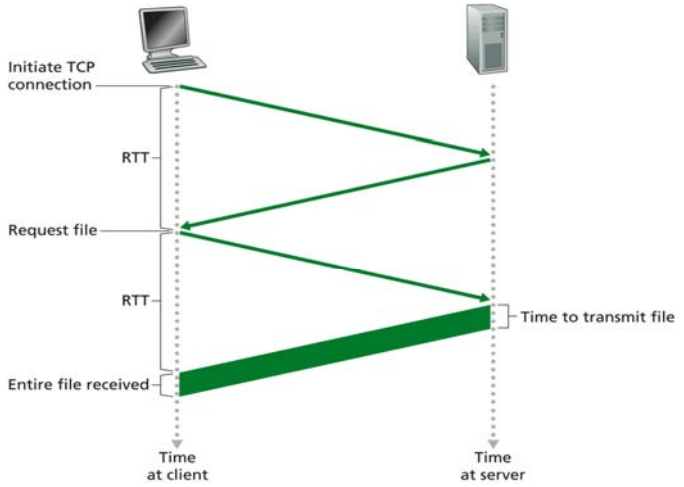
3. http server receives request, forms response msg containing requested object (aelshafee.net/fall2012.html), sends message into socket

4. http server closes TCP conn.

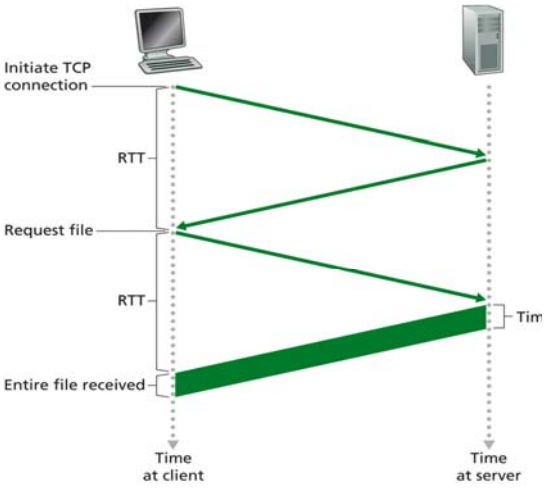
5. http client receives response message containing html file, displays html. Parsing html file, finds I0 referenced jpeg objects

6. Steps I-4 repeated for each of fall2012.html objects

Non-Persistent HTTP Connections

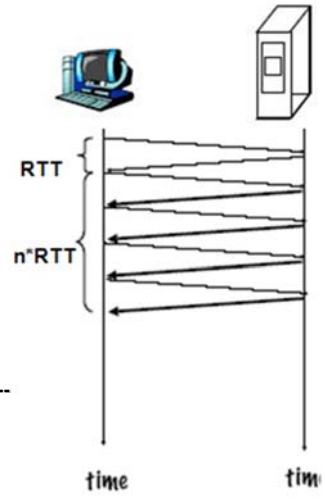


- http/1.0 : server parses request, responds, closes connection (1 object per connection)
- Request time for a single object = 2*RTT + file transmission time
- Request time for n object = 2*n*RTT + files transmission time
- Browsers often open parallel connections



Persistent HTTP Connections

- Default for http/l.l
- TCP connection is initiated only once
- All objects (files) are transmitted)
- TCP connection is closed
- Request time for n objects = (n+l)*RTT + transmission time
- If pipelining is used (persistent with pipelining), then minimum time can be RT + files transmission
- time (in addition to l RTT to open connection)



HTTP messages

Two types: Request, Response

Request message:

- written in ASCII
- Methods: get, post, head, (also put and delete in version 1.1 only)

request line (GET, POST, HEAD commands)

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

header lines

Extra carriage return, line feed indicates end of message

(extra carriage return, line feed)

body

Response Message

status line (protocol, status code, status phrase)

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
```

header lines

data, e.g., requested html file

```
data data data data data ...
```

- Status Code: examples
- **2XX Success**

  Ex: 200 OK: request succeeded, requested object later in this msg
- **3XX Redirection**

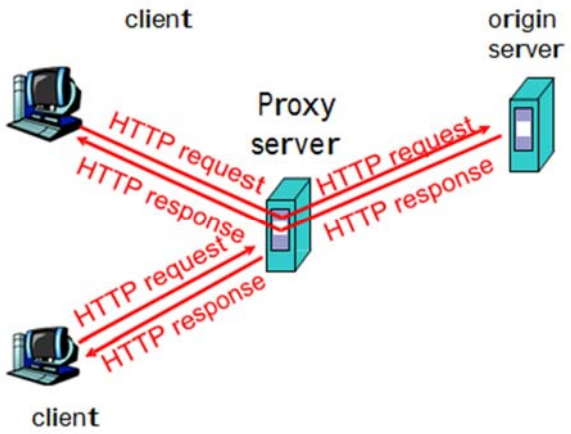
  Ex: 301 moved permanently: requested object moved, new location specified later in this message (Location)
- **4XX Client error**

  Ex: 400 bad request: request message not understood by server; 404 not found: requested document not found on this server

- **5XX Server error**

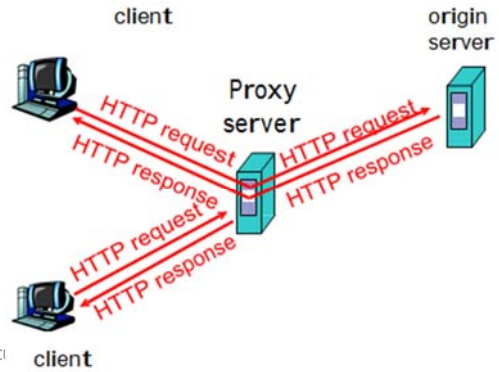
  Ex: 505 HTTP Version Not Supported

- User sets browser: Web accesses via cache
- Browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



Why web caching?

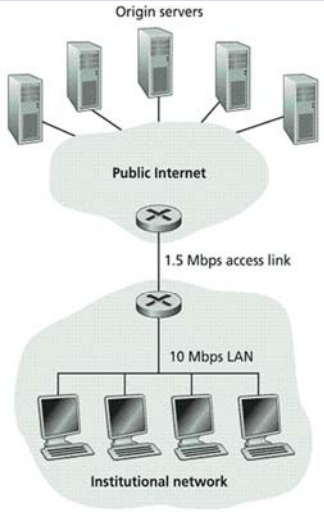
- Reduce response time for client request.
- Reduce traffic on an institution's access link.



Cache Example

Assumptions:

- average object size = l00,000 bits
- avg. request rate from institution's browsers to origin servers = 15 req/sec
- delay from institutional router to any origin server and back to router = 2 sec



Consequences:

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay = 2 sec + minutes + milliseconds

Possible solution:

- Increase bandwidth of access link to, say, 10 Mbps

Consequences

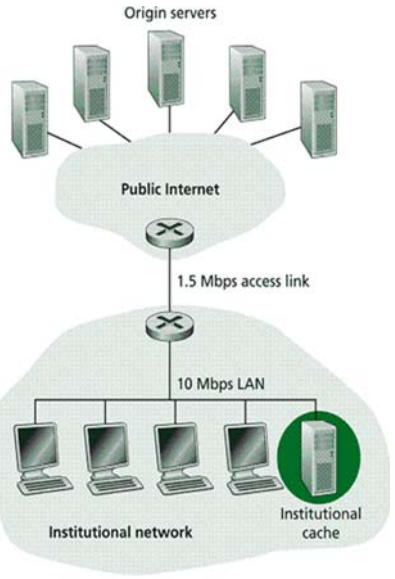
- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay = 2 sec + msecs + msecs
- often a costly upgrade

Another solution:

- Use a cache, assume hit rate = 0.4

Consequences

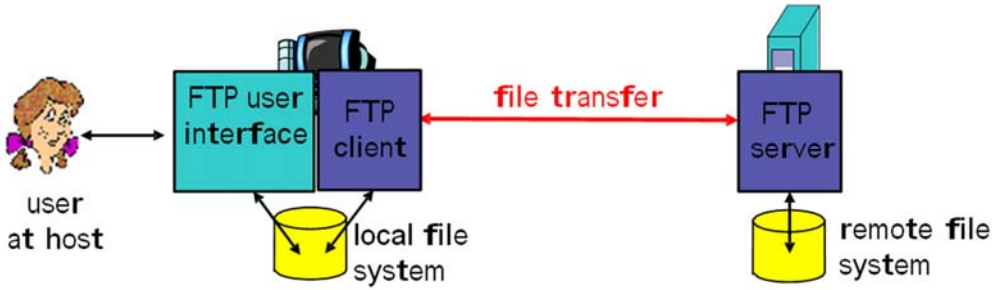
- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to
- 60%, resulting in negligible delays (say l0 msec)
- total avg delay = Internet delay + access delay + LAN delay = .6*(2.01) secs + 0.4*(0.01) secs < 1.4 sec



Summery

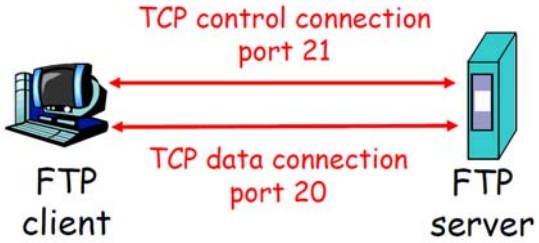
- HTTP: Runs on top of TCP (TCP connection must be established first, requiring 1 RTT)
- Connections are
  - Non-persistent (http 1.0)
  - Persistent (http 1.1)
    - Without pipelining
    - With pipelining
- Proxy server can be used to reduce delay and decrease load on original server

FTP: File Transfer Protocol



- Client/server model
  - Client: side that initiates the transfer
  - Server: remote host

- FTP client contacts FTP server on port 21, establishing a TCP connection.
- Client provides user name and password which are sent to server
- Server authorizes client
- Client browses remote directory (commands sent on control connection)



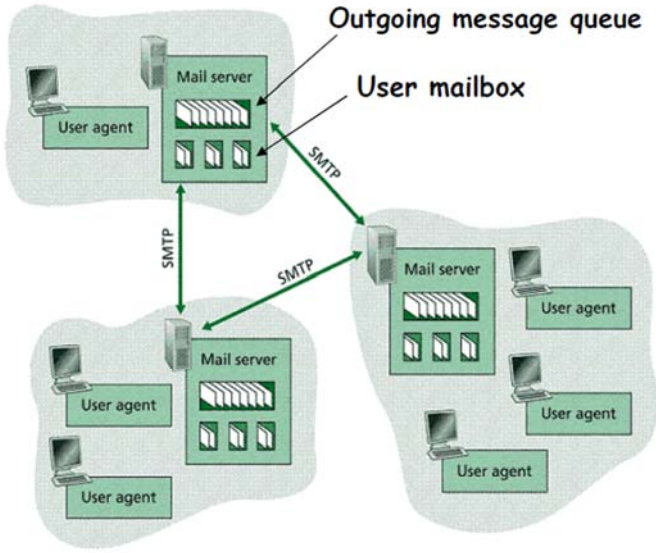
- Server receiving a command for a file transfer, opens a TCP data connection to client, transfers the file, then closes connection
- Transferring another file needs another data connection
- Control information is sent “out of band”
- FTP server maintains “state” current directory, earlier authentication
- Commands and replies are sent as ASCII text over control connection

- Some of the most common commands
  - USER username
  - PASS password
  - LIST
  - RETR filename
  - STOR filename
- Each command is followed by a reply
- Replies are similar to status code and phrase in HTTP
- Some typical replies
  - 331 Username OK, password required
  - 125 Data connection already open; transfer starting
  - 452 error writing file

SMTP: Email in the Interne

3 major components:

- User agent (mail reader)
- Mail server (port 25)
- SMTP protocol



- **User agent (mail reader)**

  compose, read, reply to, and forward messages. Ex outlook
- **Mail server (port 25)**

  stores incoming and outgoing messages for each of its users in mailbox and message queue
- **SMTP protocol:**

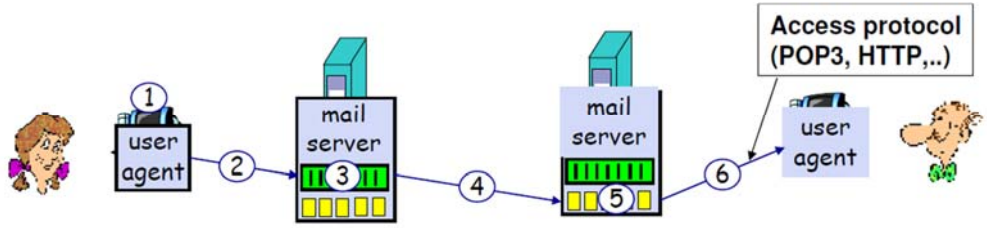
  Application layer protocol for sending messages between mail servers

- Application layer protocol for sending messages between mail servers
- Uses TCP connection (between sender and receiver servers, no intermediate servers)
- Has 2 sides client and server.
- Both sides run on every mail server
- Command/response interaction
  - Commands ASCII text (e.g. HELO, MAIL FROM, RCPT TO, DATA)
  - Response status code and phrase (e.g. 220 server name, 250 hello client name,…)

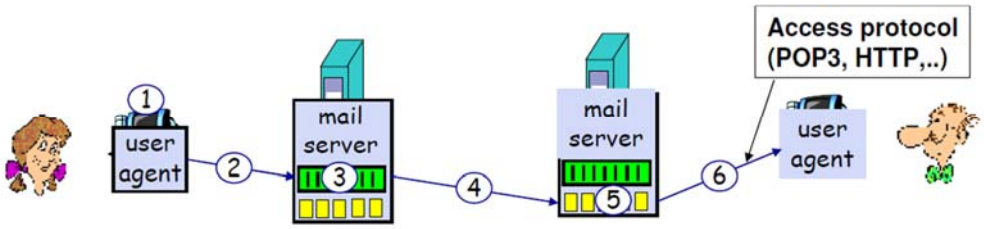
- Messages must be in 7-bit ASCII (problems when attachment is multimedia data)
- SMTP uses persistent connections

Example: Salwa and Ali

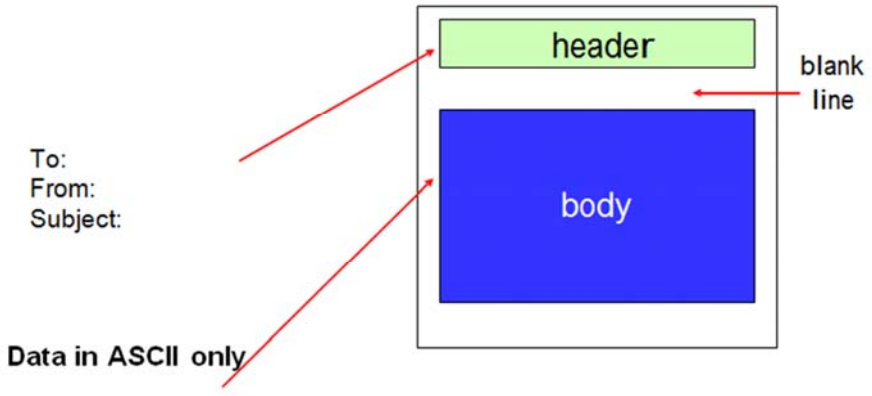
- Salwa composes an email to Ali.
- When done, her user agent sends the message to her mail server
- Message is placed in the server's outgoing message queue (in green)



- SMTP on Salwa's server opens a TCP connection with the SMTP on Ali's server, then sends the message, closes connection
- Ali's server places message in Ali's mailbox (in yellow)
- Ali invokes his user agent to read his mailbox

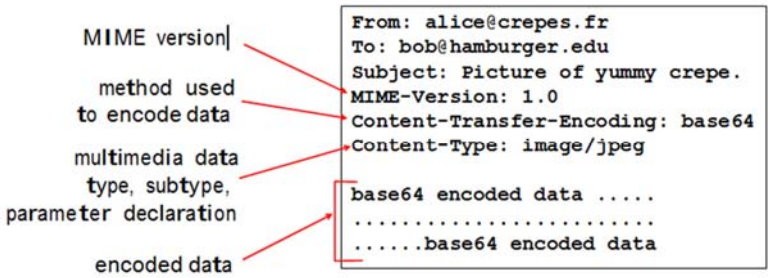


Mail Message Format



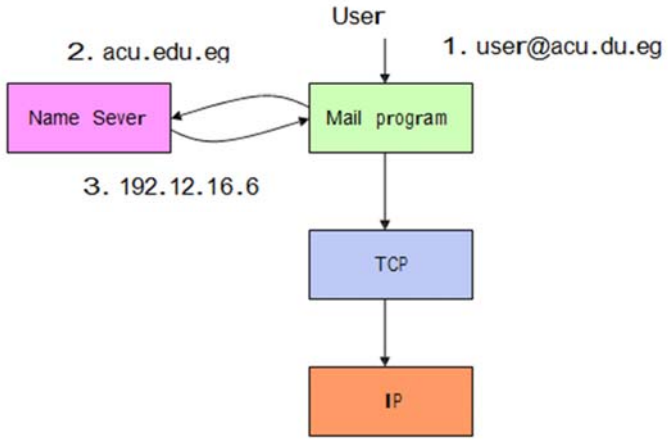
MIME: Multipurpose Internet

- For sending non-ASCII content (images, video, arabic characters,..), converts it first to ASCII (encoding method to be used)
- Additional headers declare MIME content type
- Two key MIME headers Content-type, Content-transfer-encoding



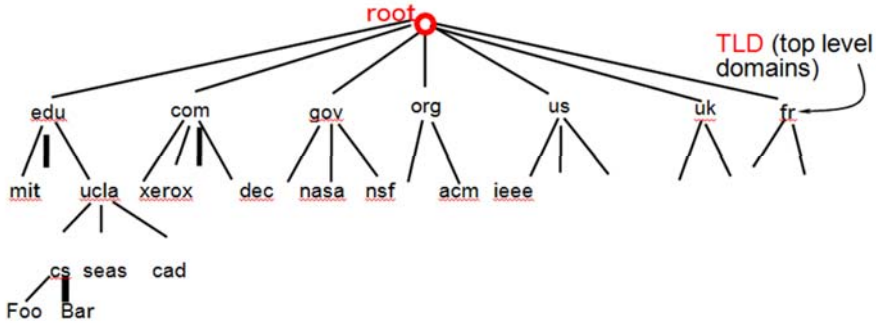
DNS: Domain name System

- Map between names and IP addresses



- Distributed database implemented in a hierarchy of DNS servers
- Application layer protocol allowing hosts to query such distributed database
- Often used by other application layer protocols (http, smtp,…)
- Example: a browser (HTML client) requesting w w w .acu.edu.eg/index .htm l
- DNS runs over UDP (why?) and uses port 53

DNS: Decentralized Approach



- Each non-leaf node in the tree is a domain
- Any domain (zone) can have its own sub-domains, no limit on the depth of any branch
- Each domain belongs to an administrative authority
  - DNS name hierarchy is completely independent from the Internet's topological structure
- Root and TLD servers contain mapping for authoritative servers, they don't contain name-IP address mapping

DNS Servers

Three classes of DNS servers

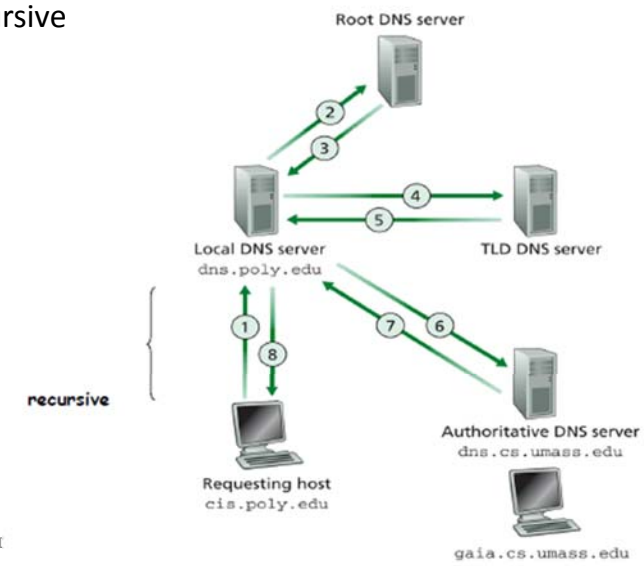
- Root DNS server:
  - There are 13 root DNS servers worldwide
  - When local name server fails, it contacts the root name server
  - Root either knows the mapping (reply directly) or knows IP of an "authoritative" name server that has the mapping
- Top-Level Domain DNS server:
  - Responsible for each of the TLD, e.g .com, .edu,.eg,…

- Authoritative DNS server: (at each organization or ISP)
  - Holds the mapping (names-IP) of all hosts within same organization
  - When host issues a DNS query, its local name server is contacted first.
- Local DNS server
  - Doesn't belong to the DNS hierarchy
  - Exists in every ISP
  - Acts as a default DNS server (contacted first)

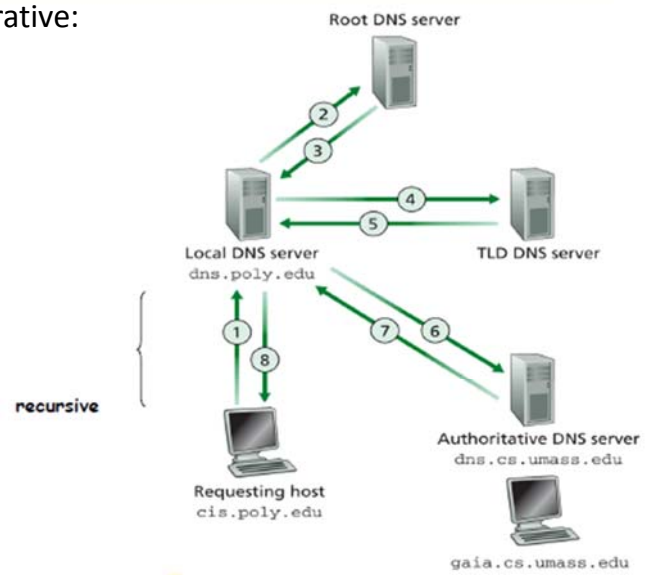
DNS Queries

- 2 types of queries:
  - Recursive
    - Puts burden of name resolution on contacted name server
    - the contacted name server resolves the name completely
  - Iterative:
    - Contacted server replies with name of server to contact
    - "I don't know this name, but ask this server"

- DNS Queries: Recursive



- DNS Queries: Iterative:



DNS Caching

- When any name server learns mapping, it caches mapping
  - cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers (root name servers not often visited)

DNS Records

record format: (name, value, type,ttl)

- Type=A
  - **name** is hostname, **value** is IP address. Ex:

  (machine1.foo.com, 145.37.2.126, A)
- Type=CNAME
  - **name** is an alias name for some "cannonical" (the real) name, **value** is cannonical name

  (foo.com, machine1.foo.com, CNAME)
- Type=NS
  - **name** is domain, **value** is name of authoritative name server for this domain. Ex:

  (foo.com, dns.foo.com, NS)
- Type=MX
  - **value** is canonical name of mailserver associated with alias name **name**

  (foo.com, mail.foo.com, MX)

- Application Layer Protocols: Summary
- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code
- message formats:
  - Headers fields giving info about data
  - data info being communicated
- HTTP, FTP, SMTP: run on top of TCP
- DNS: runs on top of UDP
- Next time: Transport Layer

Thanks, wishing you the best of luck,…