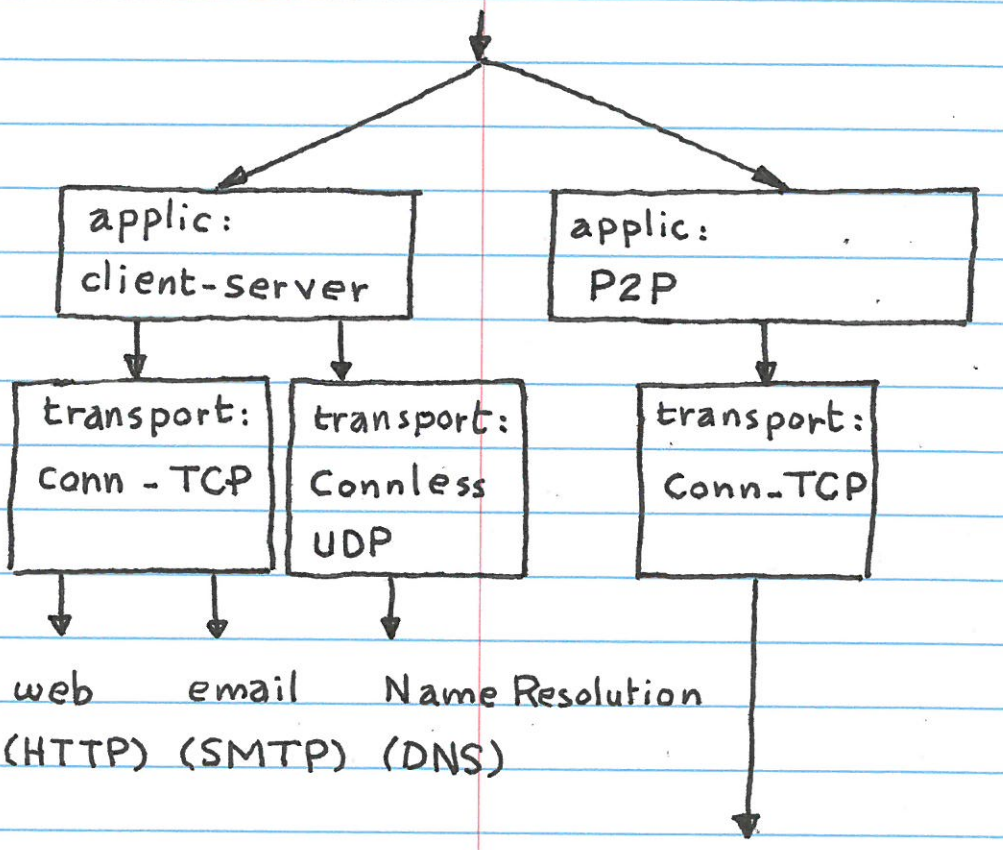
Taxanomy of Applications



P2P : Peer_to_Peer

HTTP : Hyper Text Transfer Protocol

SMTP : Simple Mail Transfer Protocol

DNS : Domain Name System

Application Examples

- most applications are connection client_servers:

HTTP	web
SMTP	email
Telnet	remote terminal access
FTP	file transfer protocol
YouTube	streaming multimedia
SIP	Internet telephony

- few applications are connectionless client_servers:

DNS	name resolution
SNMP	network management prot.
RIP	routing information prot.

- rare applications are connection P2P:

Bit Torrent	file distribution

Why Connection Client-Server Application?

- connection allows client and server to exchange more than one application msg

- connection allows client and server to exchange control information before exchanging application msgs

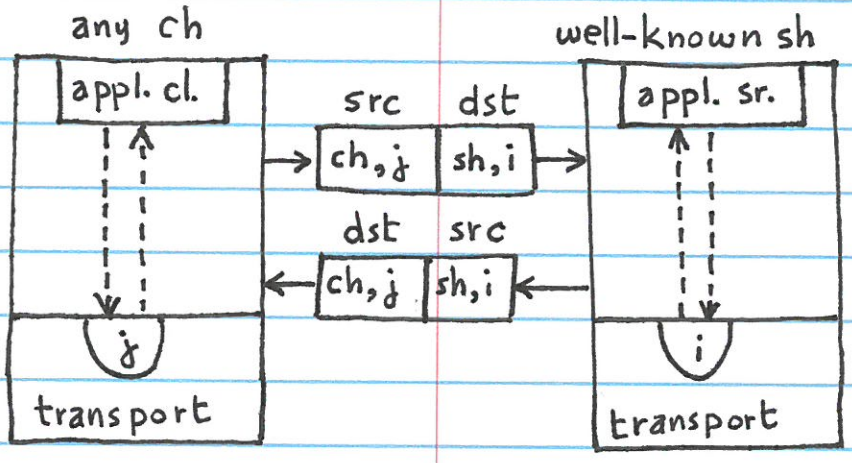
- connection supports reliable data transfer, flow control, and congestion control

- one disadvantage: connection is expensive

Client_Server Applications

- appl. server is always running on well_known low_numbered port i in well-known server host sh. (Port for web is 80, and for email is 25.)

- appl. client runs only when needed on any high_numbered port j in any client host ch



- one or more application clients can communicate with the same appl. server

Sockets

- associated with each allocated port (i or j) is an input buffer called socket such that any msg that is sent to the allocated port is stored in the associated socket until the application receives the msg from the associated socket

- **socket table:**
  determines for each allocated port the socked that is associated with this port

Socket Tables in Connless Appl. 15

- Server host : sh
  - " port : i in sh
  - " socket: ss in sh

- client hosts : ch1 , ch2
  - " ports : j1 in ch1, j2 in ch2
  - " sockets: cs1 in ch1, cs2 in ch2

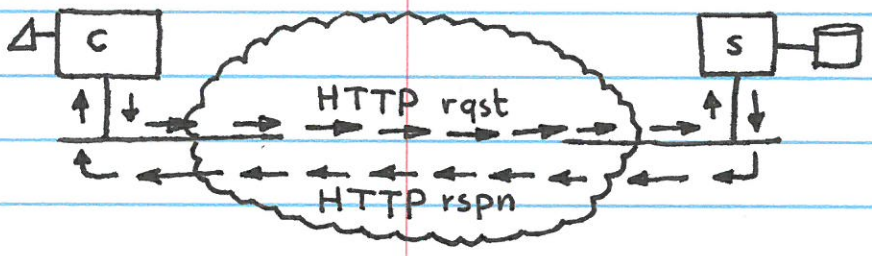
- socket table in sh : (port = i, socket = ss)
  - " " " ch1: (port = j1, socket = cs1)
  - " " " ch2: (port = j2, socket = cs2)

Socket Tables in Conn Appl.

• server host :	sh
" port :	i in sh
" sockets:	ss in sh, ss1 in sh, ss2 in sh
• client hosts :	ch1, ch2
" ports :	j1 in ch1, j2 in ch2
" sockets:	cs1 in sh1, cs2 in sh2
• socket table in sh:	((any), (sh, i), ss), ((ch1, j1), (sh, i), ss1), ((ch2, j2), (sh, i), ss2)
" " " ch1:	((sh, i), (ch1, j1), cs1)
" " " ch2:	((sh, i), (ch2, j2), cs2)

HTTP

- this application is used to transfer web pages from servers to clients



- each web page consists of HTML base file that refers to other objects: HTML files, JPEG images, video clips

- each web page and each referenced object in it is named by a URL:

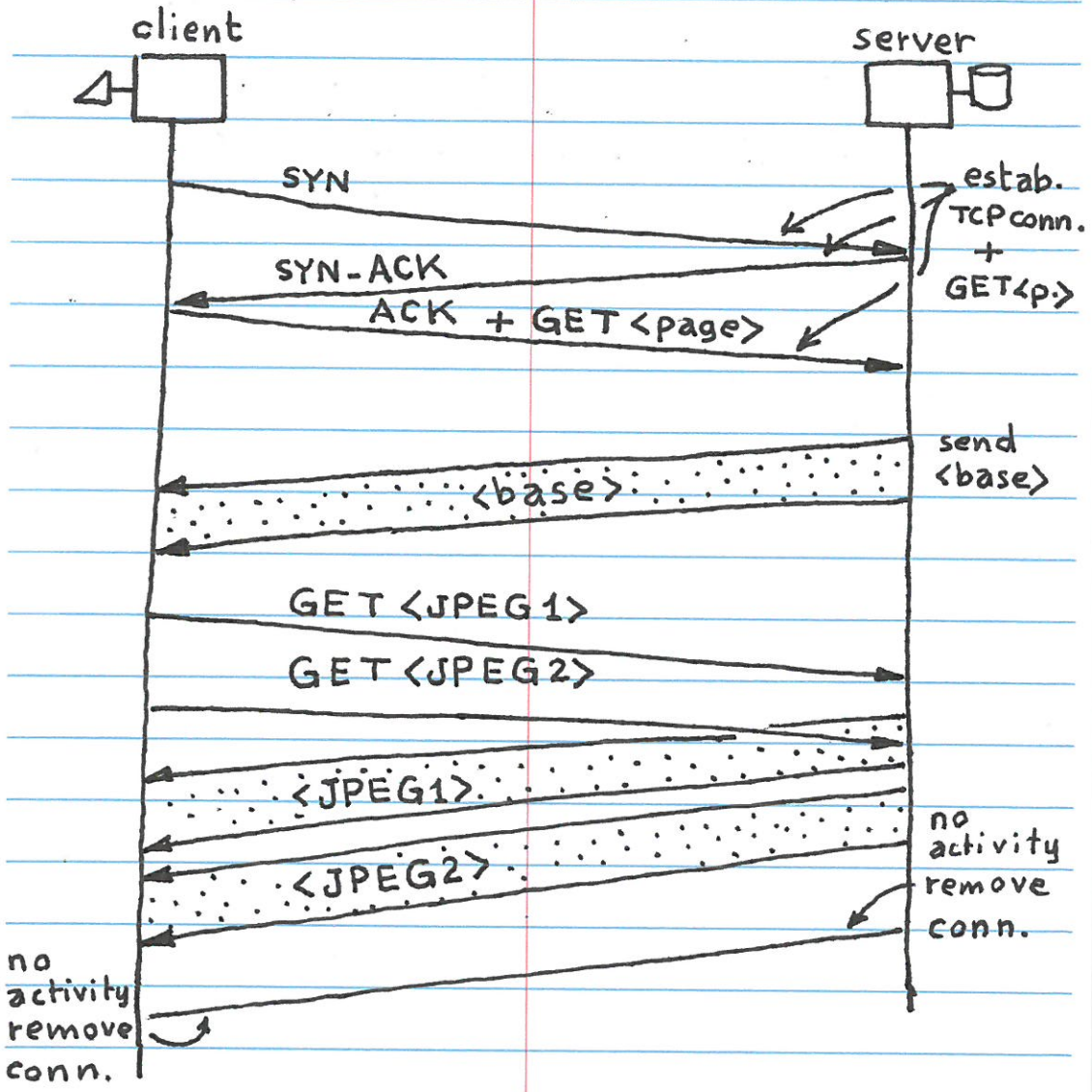
http://www.cs.utexas.edu/users/g

host name | pathname

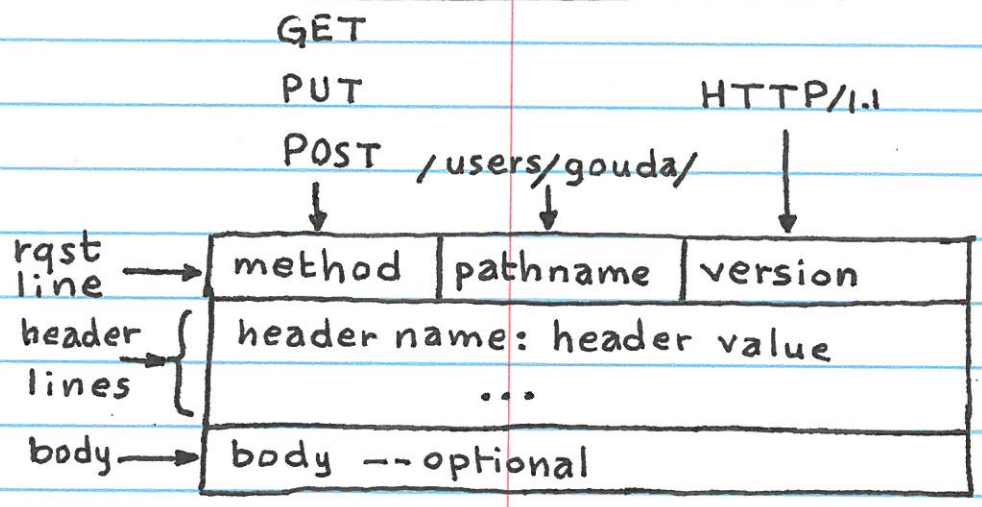
- the hostname names the host where the web page or object are stored

An HTTP Scenario

- A client gets a web <page>, consisting of a <base> file and two <JPEG> images from same server:



Format of HTTP Rqst Msgs



- examples of methods:

  GET get and display specified web page; msg has no body

  PUT add body to specified web page; msg has body

  POST get and display web page specified by pathname and body; msg has body

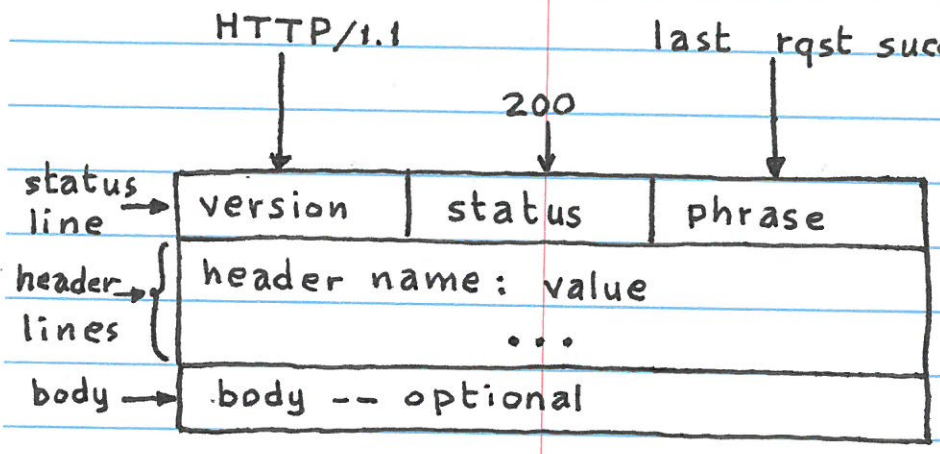
- examples of headers:

  host: www.cs.s.edu {where web page is}

  cookie: 1225 {see next}

  if_modif_since: T {see next}

Format of HTTP Rspn Msgs



- examples of status phrases:

  200 last rqst succeeded

  301 moved permanently and new host is specified in "location" header below

  400 last rqst cant be understood

- examples of headers:

  location: new server of moved page

  set_cookie: 1225 {see next}

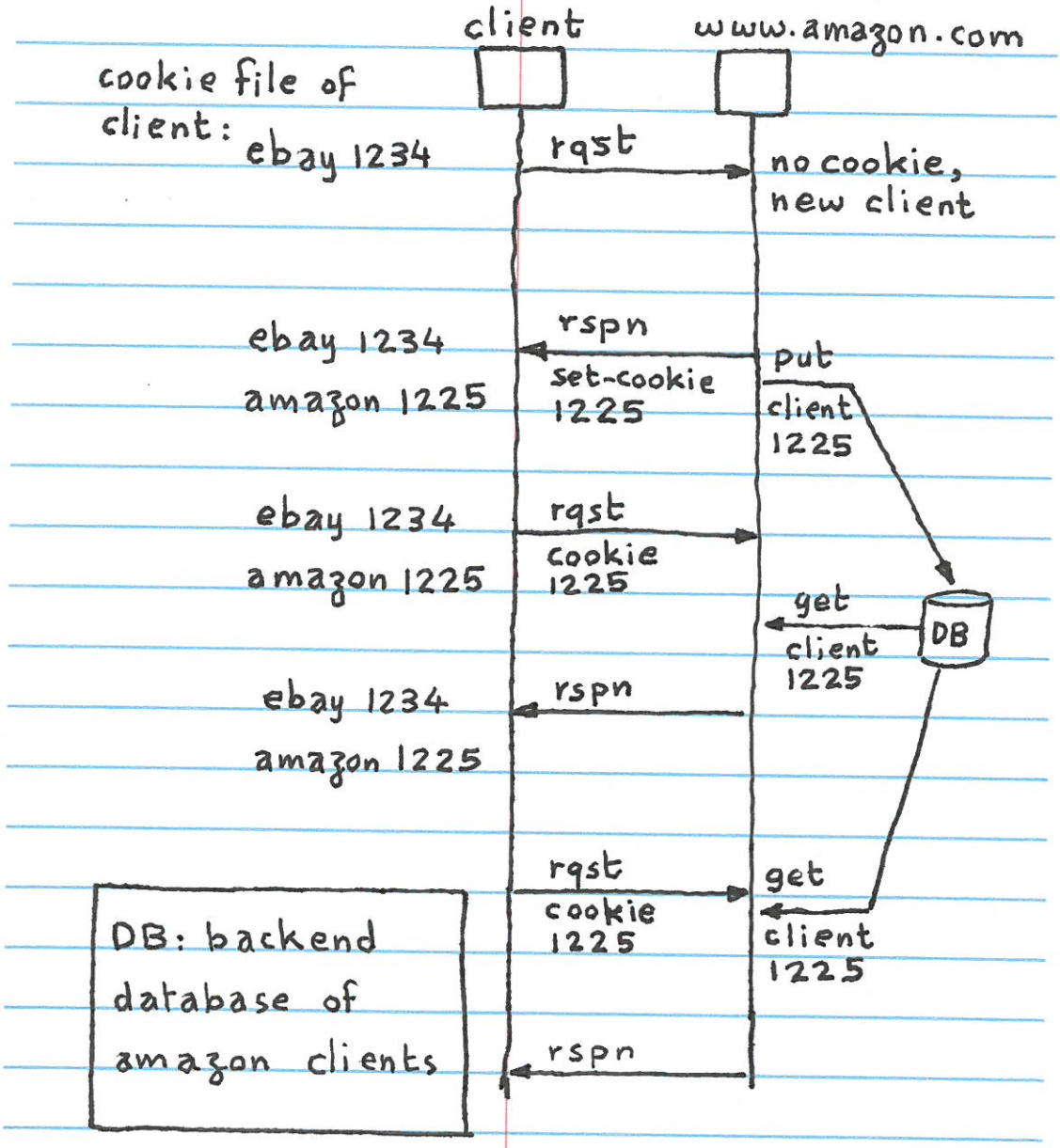
  last_modified: T {see next}

- body:

  the rqstd object, if any, is sent in the rspn msg

Cookies

- Inform server that the current rqst came from same client of a previous rqst msg



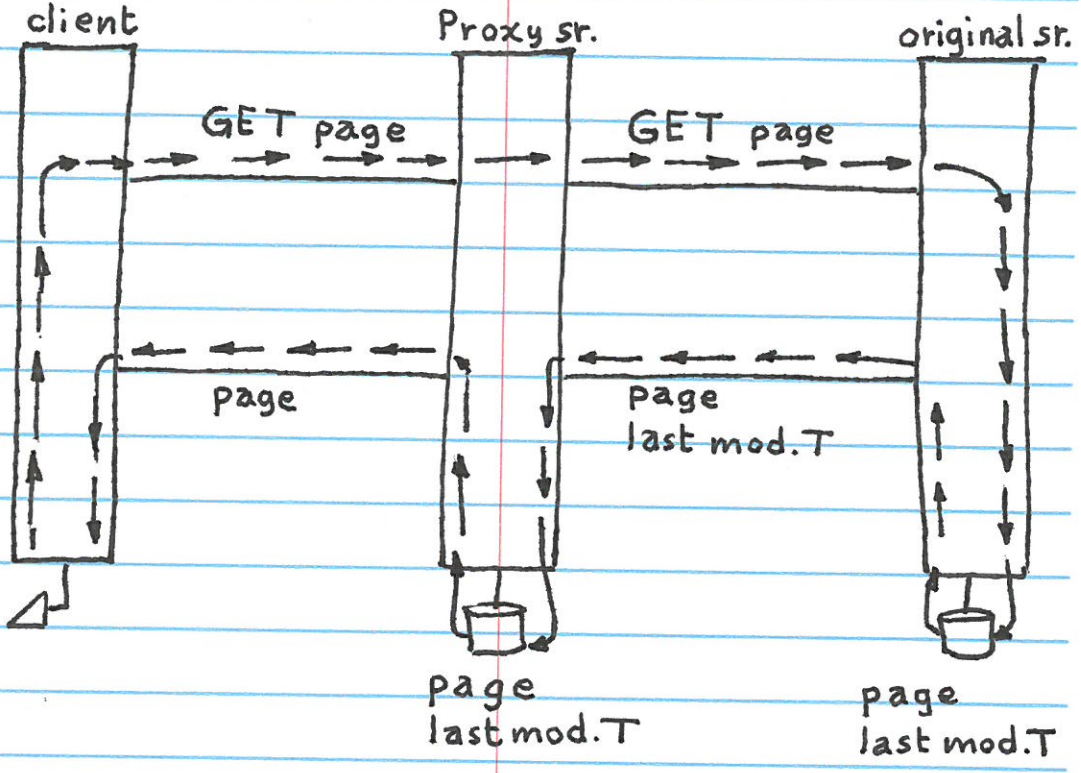
Web Caching

- A proxy server is a server that replies to HTTP rqst msgs on behalf of original server

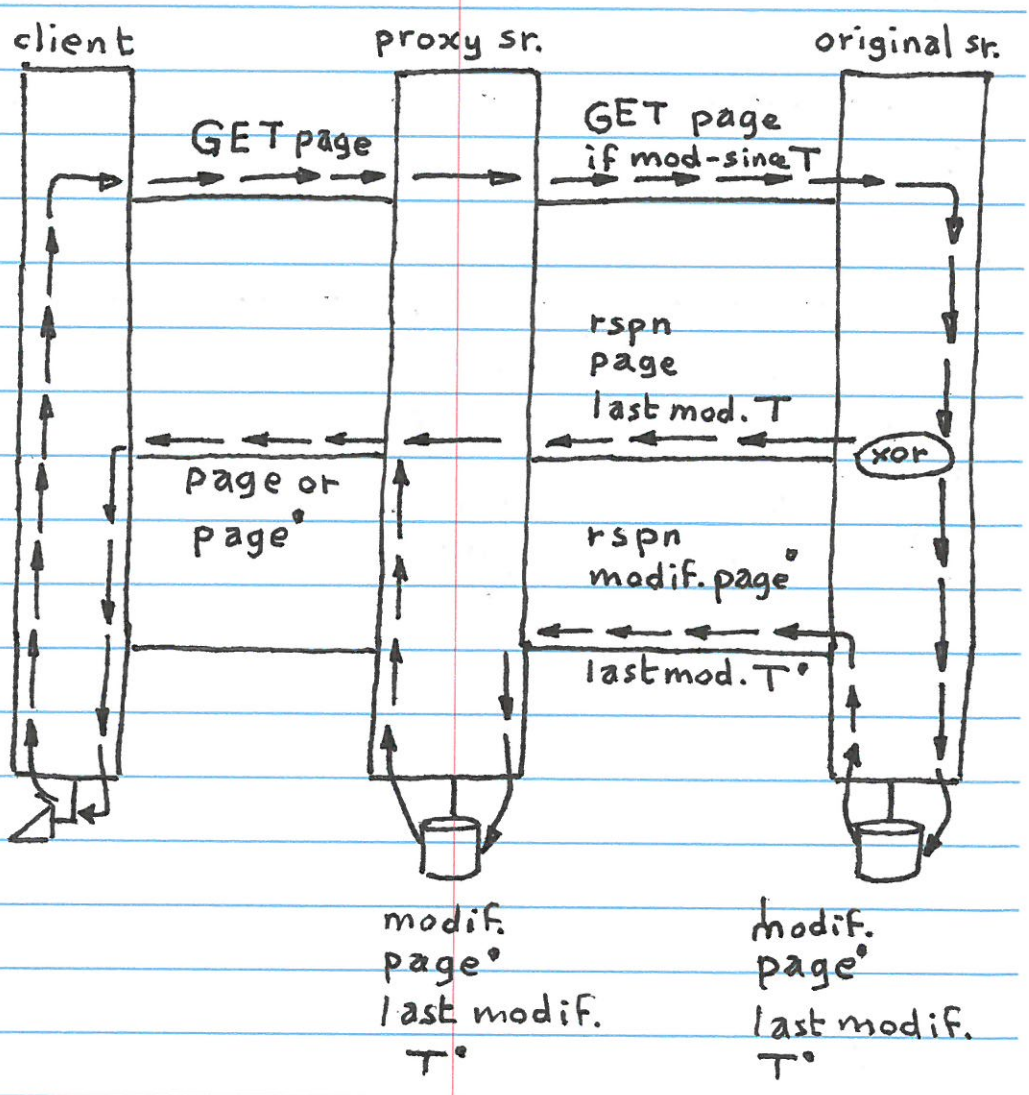
- When a client sends a GET rqst to an original server, the Internet directs the rqst to a proxy server

- What the proxy does (to reply to the GET rqst) depends on
  - whether the proxy does not have rqstd page
  - or the proxy does have rqstd page

Proxy Doesn't Have Rqstd Page 23



Proxy Has Rqstd Page, Last Modif. at T



Email Users

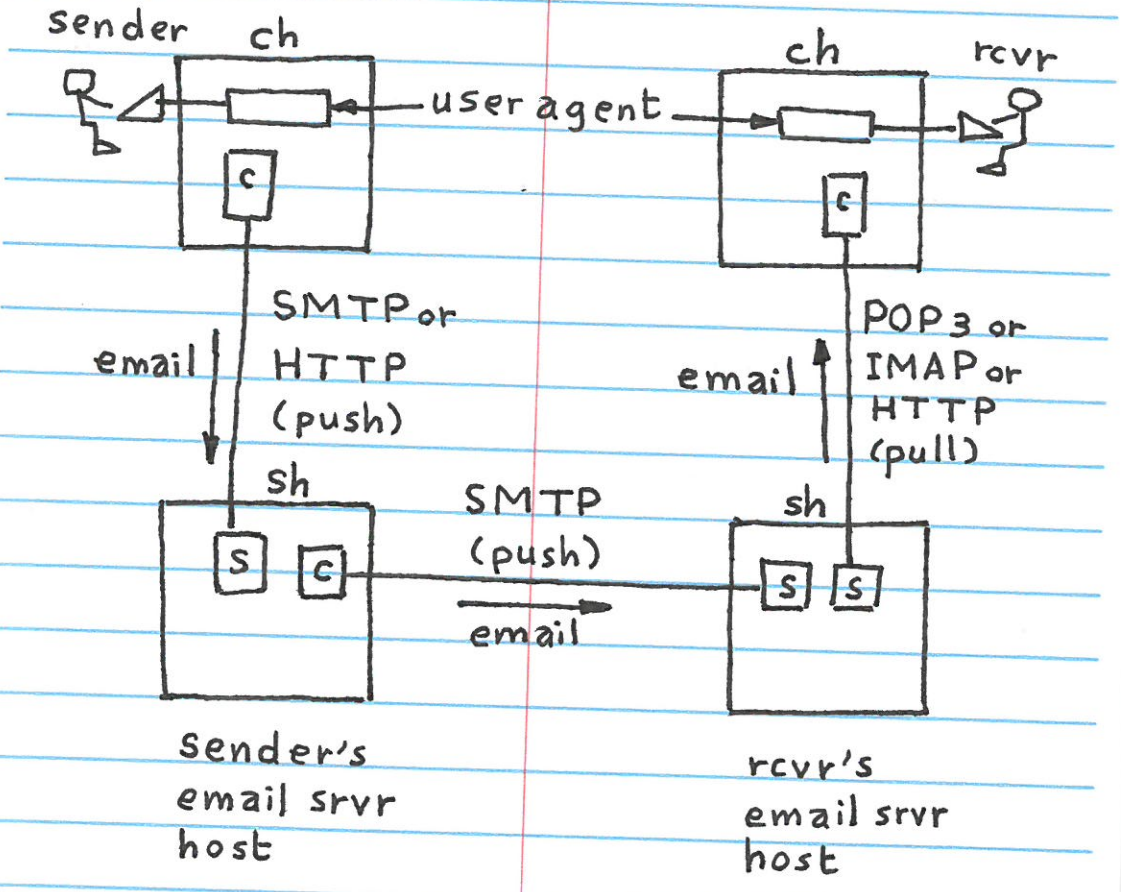
- each email user is registered in an email server with a unique hostname:
  gmail.com , cs.utexas.edu , ...

- each email user has a unique name:
  gouda @ cs.utexas.edu
  id — hostname of email server

  where id of email user is unique in the email server where the user is registered

- each email user has a user agent, e.g. "outlook" or "browser", that implements the user's client. The email user informs its agent of the user's name.

Email Protocols



SMTP : Simple Mail Transfer Protocol

HTTP : Hyper Text Transfer Protocol

POP3 : Post Office Protocol version3

IMAP: Internet Mail Access Prot.

Objectives of DNS

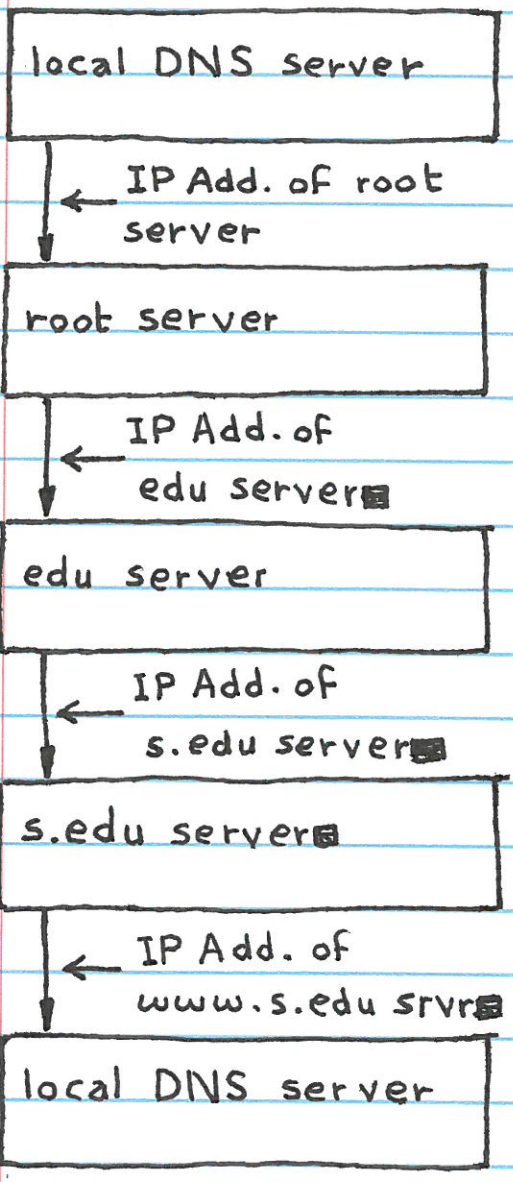
- DNS translates hostname of a server (e.g. web, email) into an IP address of this server

- if server is busy, then DNS translates its hostname into several IP addresses (ordered at random to balance load among several copies of server)

- a server host can have one long canonical name and several short alias names. In this case, DNS translates any alias name of server into its canonical name

- web and email servers can have same alias name but different canonical names. In this case, DNS translates the common alias name to the correct canonical name

DNS Servers to Resolve (www.s.edu)



- Root servers there are 13!
- TLD (Top Level Domain) servers
- "A" (Authoritative) servers

DNS Msgs to Resolve (www.s.edu) 29

- msg 1: (from local DNS servr to root srvr)
  (IP address of www.s.edu srvr ?)

- msg 2: (from root srvr to local DNS srvr)
  (IP address of www.s.edu srvr ?)
  (IP address of .edu srvr)

- msg 3: (from local DNS srvr to edu srvr)
  (IP address of www.s.edu srvr?)

- msg 4: (from edu servr to local DNS srvr)
  (IP address of www.s.edusrvr?)
  (IP address of s.edu srvr)

- msg 5: (from local DNS srvr to s.edu srvr)
  (IP address of www.s.edusrvr?)

- msg 6: (from s.edu srvr to local DNS srvr)
  (IP address of www.s.edusrvr?)
  (IP address of www.s.edu srvr)

Caching to Speed-up Name Resolution

- resolving the name (www.s.edu) of a web server host into an IP address consists of computing the IP addresses of the following three servers:
  1. edu
  2. s.edu
  3. www.s.edu

- the local DNS server has a cache that may contain the IP addresses of some of these servers (from previous resolutions).

- the IP addresses in this cache can be used to speed-up name resolutions

- see next example.

Name Resolution Caching Example

- assume that the cache of the local DNS server already has (from previous name resolutions) the IP address of edu server

- then resolving the name (www.s.edu) consists of computing the IP addresses of only two servers:
  1. s.edu
  2. www.s.edu

(instead of computing the IP addresses of three servers. as discussed in previous slide)

Data in DNS Servers

- Data in DNS servers are stored as Resource Records (RRs):

  (name, value, type, TTL)

  TTL is the time to live for the record, and "name" and "value" depend on "type"

- type = "A":

  name = canonical name of a server host

  value = IP address of this host

- type = NS:

  name = domain name (e.g. s.edu)

  value = canonical name for the "A" DNS server for this domain

- type = CNAME:

  name = alias name for a host, not mail, server

  value = canonical name for this host

- type = MX:

  name = alias name for a mail server host

  value = canonical name for this host

Format of DNS Msgs

- every DNS msg, query or reply, has following format:

  (identifier, flag,
  set of questions,
  set of answers)

- when DNS client sends query msg (to local DNS server), it assigns the msg an identifier. This identifier remains unchanged until msg returns as a reply to the DNS client

- initially, flag is "0" to indicate that the query has not yet been fully resolved. The flag is "1" when query is fully resolved

- each question is of form (name, type). Set of questions is never empty

- each answer is an RR. Initially, set of answers is empty. Then one or more RRs are added to answer each question

How to Register a Company in DNS 34

- choose a domain name, ab.com, for company

- deploy an "A" DNS server for the company. For this server, choose

  canonical name : dns.west.ab.com
  IP address : 212.83.211.5

- publish two RRs in every com DNS server in the world

  (ab.com, dns.west.ab.com, NS, TTL)
  (dns.west.ab.com, 212.83.211.5, A, TTL)

- deploy a web server for the company. For this server, choose

  alias name : ab.com
  canonical name : www.ab.com
  IP address : 212.83.211.7

- publish two RRs in the "A" DNS server of the company

  (ab.com, www.ab.com, CNAME, TTL)
  (www.ab.com, 212.83.211.7, A, TTL)

Bit Torrent

- It is a P2P application (on top of TCP) for file distribution

- this application consists of a server, called tracker, that keeps track of all clients, called peers, that are currently registered in the system, called torrent

- at each instant, there are ~1000 peers registered in a torrent

- a torrent is used to distribute files from some peers to other peers in the torrent

- each file is partitioned into pieces of 256 Bytes each, called chunks. At each instant, each peer can have none, some, or all chunks of each file in the torrent

Operations in Bit Torrent

- **Register:**
  when peer P needs to join torrent T and download file File F, P registers with the tracker of T and rcvs IP addresses of some 50 peers from 1000 current peers in T

- **Request:**
  First, P requests from each one of 50 peers to list File F chunks that the peer has. Second, P sends a sequence of requests asking for File F chunks (starting with the rarest chunks) from the 50 peers

- **Reply:**
  while P is in T, P rcvs many requests to send file F chunks (that P has rcvd earlier) to any peer of the current 1000 peers in T. However, P gives priority (for sending file F chunks) to those peers that are currently sending most chunks to P