**Goal:** satisfy client request without involving origin server

- 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
Assumptions
• average object size = 100,000 bits
• avg. request rate from institution’s browsers to origin servers = 15/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

consequence

• 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
possible solution: install cache

- suppose hit rate is 0.4

consequence

- 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 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = 6*(2.01) secs + 4*milliseconds < 1.4 secs
**Conditional GET**

- **Goal:** don’t send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
  - `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:
  - HTTP/1.0 304 Not Modified

---

**cache**

- HTTP request msg
  - `If-modified-since: <date>`

**server**

- HTTP response
  - HTTP/1.0 304 Not Modified

---

- HTTP request msg
  - `If-modified-since: <date>`

- HTTP response
  - HTTP/1.0 200 OK
  - `<data>`

5
Three major components:

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

User Agent

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

- **mailbox** contains incoming messages for user
- **message queue** of outgoing (to be sent) mail messages
- **SMTP protocol** between mail servers to send email messages
  - client: sending mail server
  - “server”: receiving mail server
- 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
A Typical Scenario

- 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
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
- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Comparison with HTTP:

HTTP: pull
SMTP: push

both have ASCII command/response interaction, status codes

HTTP: each object encapsulated in its own response msg
SMTP: multiple objects sent in multipart msg
Mail message format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

• header lines, e.g.,
  – To:
  – From:
  – Subject:  
    different from SMTP commands!

• body
  – the “message”, ASCII characters only
Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type

```plaintext
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data ..... 
........................
......base64 encoded data
```
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`

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

People: many identifiers:
- SSN, name, passport #

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - used by humans

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 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!
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

- a Verisign, Dulles, VA
- b USC-ISI Marina del Rey, CA
- c Cogent, Herndon, VA (also LA)
- d U Maryland College Park, MD
- e NASA Mt View, CA
- f Internet Software C. Palo Alto, CA (and 36 other locations)
- g US DoD Vienna, VA
- h ARL Aberdeen, MD
- i Autonomica, Stockholm (plus 28 other locations)
- j Verisign, ( 21 locations)
- k RIPE London (also 16 other locations)
- l ICANN Los Angeles, CA
- m WIDE Tokyo (also Seoul, Paris, SF)
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
Name resolution example

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

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

- puts burden of name resolution on contacted name server
- heavy load?
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
DNS records

**DNS:** distributed db storing resource records

**RR format:** \((\text{name}, \text{value}, \text{type}, \text{ttl})\)

- **Type=A**
  - \text{name} is hostname
  - \text{value} is IP address

- **Type=NS**
  - \text{name} is domain (e.g. foo.com)
  - \text{value} is hostname of authoritative name server for this domain

- **Type=MX**
  - \text{value} is name of mailserver associated with \text{name}

- **Type=CNAME**
  - \text{name} is alias name for some “canonical” (the real) name
  - \text{value} is canonical name

\(\text{www.ibm.com} \text{ is really } \text{rvereast.backup2.ibm.com}\)
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

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of questions</td>
<td>Number of answer RRs</td>
</tr>
<tr>
<td>Number of authority RRs</td>
<td>Number of additional RRs</td>
</tr>
</tbody>
</table>

12 bytes
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

How do people get IP address of your Web site?