CSEE 4119 Computer Networks

Chapter 2 Application (3/5)
Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
  + Caching!
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
Web caches (proxy server)

**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 to client
More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

why Web caching?
- reduce response time for client request
- reduce traffic on an institution’s access link.
- Internet dense with caches: enables “poor” content providers to effectively deliver content (but so does P2P file sharing)
Caching example

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
Caching example (cont)

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
**Caching example (cont)**

**possible solution:**
- install cache

**consequence**
- suppose hit rate is 0.4
  - 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 = 0.6*(2.01) secs + 0.4*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`
- **server**: response contains object modified after `<date>`:
  - `HTTP/1.0 200 OK `<data>"
Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
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 address and name, and vice versa?

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!
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

- Verisign, Dulles, VA
- Cogent, Herndon, VA (also LA)
- U Maryland College Park, MD
- US DoD Vienna, VA
- ARL Aberdeen, MD
- Verisign, (21 locations)
- NASA Mt View, CA
- Internet Software C. Palo Alto, CA (and 36 other locations)
- USC-ISI Marina del Rey, CA
- ICANN Los Angeles, CA
- RIPE London (also 16 other locations)
- Autonomica, Stockholm (plus 28 other locations)
- WIDE Tokyo (also Seoul, Paris, SF)
- Verisign, (21 locations)
TLD and Authoritative Servers

Top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: 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

**iterated 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
- 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 proposed IETF standard
  - RFC 2136
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

<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>
<tr>
<td>questions (variable number of questions)</td>
<td></td>
</tr>
<tr>
<td>answers (variable number of resource records)</td>
<td></td>
</tr>
<tr>
<td>authority (variable number of resource records)</td>
<td></td>
</tr>
<tr>
<td>additional information (variable number of resource records)</td>
<td></td>
</tr>
</tbody>
</table>
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>

| Questions (variable number of questions) |
| Answers (variable number of resource records) |
| Authority (variable number of resource records) |
| Additional information (variable number of resource records) |

12 bytes
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
- How do people get IP address of your Web site?
Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS

- Bonus:
  - a detour on CDN

- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
Web caches (proxy server)

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 to client

Application 2-24
Content distribution networks (CDNs)

**Content replication**

- challenging to stream large files (e.g., video) from single origin server in real time
- *solution*: replicate content at hundreds of servers throughout Internet
  - content downloaded to CDN servers ahead of time
  - placing content “close” to user avoids impairments (loss, delay) of sending content over long paths
  - CDN server typically in edge/access network
**Content distribution networks (CDNs)**

**Content replication**
- CDN (e.g., Akamai) customer is the content provider (e.g., CNN)
- CDN replicates customers’ content in CDN servers.
- when provider updates content, CDN updates servers
**CDN example**

1. Origin server
   - HTTP request for www.foo.com/sports/sports.html

2. DNS query for www.cdn.com


**Origin server (www.foo.com)**
- Distributes HTML

**CDN company (cdn.com)**
- Distributes gif files
- Uses its authoritative DNS server to route redirect requests

Multimedia Networking
More about CDNs

Routing requests

- CDN creates a “map”, indicating distances from leaf ISPs and CDN nodes

- When query arrives at authoritative DNS server:
  - Server determines ISP from which query originates
  - Uses “map” to determine best CDN server

- CDN nodes create application-layer overlay network