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 file sharing
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server
Imagine a world without DNS

- You would have to remember the IP addresses of
  - Every website you want to visit
  - Your bookmarks will be a list of IP addresses

You will speak like

“I went to 167.33.24.10, and there was an awesome link to 153.11.35.81...“
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 and alias names
    - tokyo.ibm.com
    - ibm.com
- 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

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 web server

Client queries same Amazon.com DNS server to get IP address of mail.amazon.com mail server.
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
- c Cogent, Herndon, VA (also Los Angeles)
- d U Maryland College Park, MD
- g US DoD Vienna, VA
- h ARL Aberdeen, MD
- i Verisign, (11 locations)
- b USC-ISI Marina del Rey, CA
- l ICANN Los Angeles, CA
- e NASA Mt View, CA
- f Internet Software C. Palo Alto, CA (and 17 other locations)
- k RIPE London (also Amsterdam, Frankfurt)
- j Verisign, (11 locations)
- m WIDE Tokyo
TLD and Authoritative Servers

- **Top-level domain (TLD) servers:**
  - responsible for com, org, net, edu, etc.
  - all top-level country domains uk, fr, ca, jp.
  - Network solutions maintains servers for com TLD
  - Educause for edu TLD

- **Authoritative DNS servers:**
  - An organization’s DNS servers,
    - providing authoritative hostname to IP mappings for organization’s servers (e.g., Web and mail).
  - Can be maintained by organization or service provider
Local Name Server

- Does not strictly belong to hierarchy

- Each ISP (residential, company, univ) has one.
  - Also called “default name server”

- When a host makes a DNS query
  - query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy.
Example

- **Iterative Querying**
  Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
Recursive queries

recursive query:
- puts burden of name resolution on contacted name server
- heavy load?

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

Which is a better design choice?
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)

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

- **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
  - value is canonical name

- **Type=MX**
  - value is name of mailserver associated with name

```
Auth \rightarrow (\text{cnn.com}, 166.34.33.11, A, 10)
TLD \rightarrow (\text{cnn.com}, 166.34.23.18, NS, 10)
```

4 categories: A, NS, CNAME, MX

- alias \rightarrow cannonical

- time to live

- 4 categories.
Local DNS

Local → Everest.

ibm.com

D & (ibm.com, servereast.ibm.com, CNAME, 10)

serverwest.ibm.com

servereast.ibm.com

113.63.11.12

(Servercast, 113.63.11.12, A, 10)
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</td>
<td>(variable number of questions)</td>
</tr>
<tr>
<td>answers</td>
<td>(variable number of resource records)</td>
</tr>
<tr>
<td>authority</td>
<td>(variable number of resource records)</td>
</tr>
<tr>
<td>additional information</td>
<td>(variable number of resource records)</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>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identification</td>
<td>16-bit sequence identifier</td>
</tr>
<tr>
<td>flags</td>
<td>16-bit flags</td>
</tr>
<tr>
<td>number of questions</td>
<td>16-bit field</td>
</tr>
<tr>
<td>number of answer RRs</td>
<td>16-bit field</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>16-bit field</td>
</tr>
<tr>
<td>number of additional RRs</td>
<td>16-bit field</td>
</tr>
<tr>
<td>questions</td>
<td>Variable number of resource records</td>
</tr>
<tr>
<td>answers</td>
<td>Variable number of resource records</td>
</tr>
<tr>
<td>authority</td>
<td>Variable number of resource records</td>
</tr>
<tr>
<td>additional information</td>
<td>Variable number of resource records</td>
</tr>
</tbody>
</table>
Inserting records into DNS

- Example: just created startup “Network Utopia”
- Register name networkuptopia.com at a registrar (e.g., Network Solutions)
  - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (networkutopia.com, dns1.networkutopia.com, NS)
    - (dns1.networkutopia.com, 212.212.212.1, A)

- Also, in the startup’s Auth server, put Type A record for www.networkuptopia.com and Type MX record for networkutopia.com
- How do people get the IP address of your Web site?
Questions ?
Chapter 2: Application layer

- 2.1 Principles of network applications
  - app architectures
  - app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P file sharing
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server
P2P file sharing

Example

- Alice runs P2P client application on her notebook computer.
- Intermittently connects to Internet; gets new IP address for each connection.
- Asks for “Hey-Jude.mp3”.
- Application displays other peers that have copy of Hey Jude.

- Alice chooses one of the peers, Bob.
- File is copied from Bob’s PC to Alice’s notebook: HTTP.
- While Alice downloads, other users uploading from Alice.
- Alice’s peer is both a Web client and a transient Web server.

All peers are servers = highly scalable!
P2P: centralized directory

original “Napster” design

1) when peer connects, it informs central server:
   - IP address
   - content

2) Alice queries for “Hey Jude”. Server informs that Bob has it.

3) Alice requests file from Bob
P2P: problems with centralized directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

file transfer is decentralized, but locating content is highly centralized
Query flooding: Gnutella

- fully distributed
  - no central server
- public domain protocol
- many Gnutella clients implementing protocol

Overlay network: graph

- edge between peer X and Y if there’s a TCP connection
- all active peers and edges is overlay net
- Edge is not a physical link
- Given peer will typically be connected with < 10 overlay neighbors
Gnutella: protocol

- Query message sent over existing TCP connections
- Peers forward Query message
- QueryHit sent over reverse path

Scalability: limited scope flooding

File transfer: HTTP

Query message sent over existing TCP connections, peers forward Query message, QueryHit sent over reverse path.
Gnutella: Peer joining

1. Joining peer X must find some other peer in Gnutella network: use list of candidate peers
2. X sequentially attempts to make TCP with peers on list until connection setup with Y
3. X sends Ping message to Y; Y forwards Ping message
4. All peers receiving Ping message respond with Pong message
5. X receives many Pong messages. It can then setup additional TCP connections

What happens when peer leaves: find out as an exercise!
Exploiting heterogeneity: KaZaA

- Each peer is either a group leader or assigned to a group leader.
  - TCP connection between peer and its group leader.
  - TCP connections between some pairs of group leaders.

- Group leader tracks the content in all its children.
KaZaA: Querying

- Each file has a hash and a descriptor
- Client sends keyword query to its group leader
- Group leader responds with matches:
  - For each match: metadata, hash, IP address
- If group leader forwards query to other group leaders, they respond with matches
- Client then selects files for downloading
  - HTTP requests using hash as identifier sent to peers holding desired file
KaZaA tricks

- Limitations on simultaneous uploads
- Request queuing
- Incentive priorities
- Parallel downloading

For more info:
- J. Liang, R. Kumar, K. Ross, “Understanding KaZaA,”
  (available via cis.poly.edu/~ross)
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 file sharing
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server