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

**DNS services**
- Hostname to IP address translation
- Host aliasing
  - Canonical and 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

![Root DNS Servers](image)
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

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 and 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 a host makes a DNS query, query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy
- Example
  - 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"
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**: *(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>16-bit identification</th>
<th>16-bit flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>identification</td>
<td>flags</td>
</tr>
<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>

Name, type fields for a query
- questions (variable number of questions)

RRs in response to query
- answers (variable number of resource records)

records for authoritative servers
- authority (variable number of resource records)

additional "helpful" info that may be used
- additional information (variable number of resource records)

---

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)
- Put in authoritative server **Type A record for** www.networkuptopia.com and **Type MX record for** networkuptopia.com
- **How do people get the IP address of your Web site?**
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”
- 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”
3) Alice requests file from Bob

Problems
- Single point of failure
- Performance bottleneck
- Copyright infringement: “target” of lawsuit is obvious

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
**Gnutella: Peer joining**

1. Joining peer Alice must find another peer in Gnutella network: use list of candidate peers
2. Alice sequentially attempts TCP connections with candidate peers until connection setup with Bob
3. **Flooding:** Alice sends Ping message to Bob; Bob forwards Ping message to his overlay neighbors (who then forward to their neighbors....)
   - peers receiving Ping message respond to Alice with Pong message
4. Alice receives many Pong messages, and can then setup additional TCP connections

---

**Hierarchical Overlay**

- between centralized index, query flooding approaches
- 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 content in its children

---

Stan Kurkovsky
Comparing Client-server, P2P architectures

**Question**: How much time to distribute file initially at one server to \( N \) other computers?

\[
\begin{align*}
\text{Server:} & \quad u_s: \text{server upload bandwidth} \\
\text{client/peer i:} & \quad u_i: \text{client/peer i upload bandwidth} \\
\text{client/peer i:} & \quad d_i: \text{client/peer i download bandwidth}
\end{align*}
\]

\[
\begin{align*}
\text{File, size } F & \\
\text{Network (with abundant bandwidth)} & \\
\text{Server} & \quad u_s \\
\text{client i} & \quad d_i \\
\text{client i takes } F/d_i \text{ time to download}
\end{align*}
\]

**Client-server: file distribution time**

- server sequentially sends \( N \) copies of a file of size \( F \):
  - \( NF/u_s \) time
- client i takes \( F/d_i \) time to download

\[
\text{Time to distribute } F \\
to \ N \text{ clients using client/server approach} = \left( \frac{NF}{u_s}, \frac{F}{\min(d_i)} \right) \\
\]

Increases linearly in \( N \) (for large \( N \))
**P2P: file distribution time**

- server must send one copy: $F/u_s$ time
- client $i$ takes $F/d_i$ time to download
- NF bits must be downloaded (aggregate)

- fastest possible upload rate (assuming all nodes sending file chunks to same peer): $u_s + \sum_{i=1,N} u_i$

$$d_{P2P} = \max \left\{ \frac{F}{u_s}, \frac{F}{\min(d_i)}, \frac{NF}{u_s + \sum_{i=1,N} u_i} \right\}$$

---

**Comparing Client-server, P2P architectures**

![Graph comparing P2P and Client-Server architectures](graph.png)
**P2P Case Study: BitTorrent**

- file divided into 256KB *chunks*.
- peer joining torrent:
  - has no chunks, but will accumulate them over time
  - registers with tracker to get list of peers, connects to subset of peers (“neighbors”)
- while downloading, peer uploads chunks to other peers.
- peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain

---

**P2P Case Study: BitTorrent**

**Pulling Chunks**
- at any given time, different peers have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
- Alice issues requests for her missing chunks
  - rarest first

**Sending Chunks: tit-for-tat**
- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
P2P Case study: Skype

- P2P (pc-to-pc, pc-to-phone, phone-to-pc)
  - Voice-Over-IP (VoIP) application
    - also IM
  - proprietary application-layer protocol (inferred via reverse engineering)
  - hierarchical overlay

- Making a call
  - User starts Skype
  - SC registers with SN
    - list of bootstrap SNs
  - SC logs in (authenticate)
  - Call: SC contacts SN with callee ID
    - SN contacts other SNs (unknown protocol, maybe flooding) to find addr of callee; returns addr to SC
    - SC directly contacts callee, over TCP