**The Road Ahead**

- DNS Design
- DNS Today

**Naming**

- Need naming to identify resources
- Once identified, resource must be located
- How to name resource?
  - Naming hierarchy
- How do we efficiently locate resources?
  - DNS: name → location (IP address)
- Challenge: How do we scale these to the wide area?
Obvious Solutions (1)
Lookup a Central DNS?
- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update
- Doesn't scale!

Obvious Solutions (2)
Why not use /etc/hosts?
- Original Name to Address Mapping
  - Flat namespace
  - Lookup mapping in /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Many more updates

Domain Name System Goals
- Basically a wide-area distributed database of name to IP mappings
- Goals:
  - Scalability
  - Decentralized maintenance
  - Robustness
  - Global scope
  - Names mean the same thing everywhere
  - Don't need
    - Atomicity
    - Strong consistency
Programmer's View of DNS

- Conceptually, programmers can view the DNS database as a collection of millions of host entry structures:

```
/* DNS host entry structure */
struct hostent {
    char   *h_name; /* official domain name of host */
    char   **h_aliases; /* null-terminated array of domain names */
    int h_addrtype; /* host address type (AF_INET) */
    int h_length; /* length of an address, in bytes */
    char   **h_addr_list; /* null-terminated array of in_addr structs */
};
```

- `in_addr` is a struct consisting of 4-byte IP address

- Functions for retrieving host entries from DNS:
  - `gethostbyname`: query key is a DNS host name.
  - `gethostbyaddr`: query key is an IP address.

DNS Message Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Sub-fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>No. of Questions, No. of Authority RRs</td>
</tr>
<tr>
<td>Flags</td>
<td>No. of Answer RRs, No. of Additional RRs</td>
</tr>
<tr>
<td>Name, type fields</td>
<td>Questions (variable number of answers)</td>
</tr>
<tr>
<td>for a query</td>
<td>Answers (variable number of resource records)</td>
</tr>
<tr>
<td>Flags in response to query</td>
<td>Authority (variable number of resource records)</td>
</tr>
<tr>
<td>Records for authoritative servers</td>
<td>Additional Info (variable number of resource records)</td>
</tr>
<tr>
<td>Additional info that may be used</td>
<td></td>
</tr>
</tbody>
</table>

DNS Header Fields

- **Identification**
  - Used to match up request/response

- **Flags**
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution
DNS Records

- Database contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

**RR format:** (class, name, value, type, ttl)

**FOR IN class:**
- **Type=A**
  - name is hostname
  - value is IP address
- **Type=NS**
  - name is domain (e.g., foo.com)
  - value is name of authoritative name server for this domain
- **Type=CNAME**
  - name is an alias name for some "canonical" (the real) name
  - value is canonical name
- **Type=MX**
  - value is hostname of mail server associated with name

Properties of DNS Host Entries

- Different kinds of mappings are possible:
  - Simple case: 1-1 mapping between domain name and IP addr:
    - kittyhawk.cs.cmu.edu maps to 128.2.194.242
  - Multiple domain names maps to the same IP address:
    - www.mit.edu and cs.mit.edu both map to 18.62.1.6
  - Single domain name maps to multiple IP addresses:
    - sdc.com and www.sdc.com map to multiple IP addrs
  - Some valid domain names don’t map to any IP address:
    - for example: cs.wisc.edu

DNS Design: Hierarchy Definitions

- Each node in hierarchy stores a list of names that end with same suffix
  - Suffix = path up tree
- E.g., given this tree, where would following be stored:
  - Fred.com
  - Fred.edu
  - Fred.wisc.edu
  - Fred.cs.wisc.edu
  - Fred.cs.cmu.edu
DNS Design: Zone Definitions

- Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree

- A zone has an associated set of name servers
  - Must store list of names and tree links

- Complete Tree
- Subtree
- Single node

DNS Design: Cont.

- Zones are created by convincing owner node to create/delegate a subzone
  - Records within zone store multiple redundant name servers
  - Primary/master name server updated manually
  - Secondary/redundant servers updated by zone transfer of name space
  - Zone transfer is a bulk transfer of the "configuration" of a DNS server - uses TCP to ensure reliability

- Example:
  - CS.WISC.EDU created by WISC.EDU administrators
  - Who creates WISC.EDU or .EDU?

DNS: Root Name Servers

- Responsible for "root" zone

- Approx. 13 root name servers worldwide
  - Currently (a-n).root-servers.net

- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers
Servers/Resolvers

- Each host has a resolver
  - Typically a library that applications can link to
  - Resolves contacts name server
  - Local name servers hand-configured (e.g. /etc/resolv.conf)

- Name servers
  - Either responsible for some zone or...
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone

Typical Resolution

- Steps for resolving www.wisc.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server (S₁)
  - S₁ queries root server (S₂) for (www.wisc.edu)
  - S₂ returns NS record for wisc.edu (S₃)
  - What about A record for S₃?
    - This is what the additional information section is for
    - S₁ queries S₃ for www.wisc.edu
    - S₃ returns A record for www.wisc.edu

- Can return multiple A records → what does this mean?

Lookup Methods

Recursive query:
- Server goes out and searches for more info (recursive)
- Only returns final answer or "not found"

Iterative query:
- Server responds with as much as it knows (iterative)
  - "I don't know this name, but ask this server"

Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative

What is a zone?
- A definable part of the internet
- Includes hosts and names

What is a name server?
- Server that responds to queries about host names
Workload and Caching

- Are all servers/names likely to be equally popular?
  - Why might this be a problem? How can we solve this problem?

- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains

- DNS negative queries are cached
  - Don't have to repeat past mistakes
    - E.g., misspellings, search strings in resolv.conf

- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

Typical Resolution

Subsequent Lookup Example
Reliability

- DNS servers are replicated
  - Name service available if ≥ one replica is up
  - Queries can be load balanced between replicas

- UDP used for queries
  - Need reliability → must implement this on top of UDP!
  - Why not just use TCP?

- Try alternate servers on timeout
  - Exponential backoff when retrying same server

- Same identifier for all queries
  - Don’t care which server responds

Reverse DNS

- Task
  - Given IP address, find its name
  - When is this needed?

- Method
  - Maintain separate hierarchy based on IP names
  - Write 128.2.194.242 as 242.194.2.128.in-addr.arpa
  - Why is the address reversed?

- Managing
  - Authority manages IP addresses assigned to it
  - E.g., CMU manages name space 2.128.in-addr.arpa

Prefetching

- Name servers can add additional data to response

- Typically used for prefetching
  - CNAME/MX/NS typically point to another host name
  - Responses include address of host referred to in “additional section”
DNS Today: Root Zone

- Generic Top Level Domains (gTLD) = .com, .net, .org, etc...
- Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc...
- Root server (.root-servers.net) also used to cover gTLD domains
  - Load on root servers was growing quickly!
  - Moving .com, .net, .org off root servers was clearly necessary to reduce load → done Aug 2000

New gTLDs

- .info → general info
- .biz → businesses
- .aero → air-transport industry
- .coop → business cooperatives
- .name → individuals
- .pro → accountants, lawyers, and physicians
- .museum → museums
- Only new one active so far = .info, .biz, .name

DNS Performance

- No centralized caching per site
  - Each machine runs own caching local server
  - Why is this a problem?
  - How many hosts do we need to share cache? → recent studies suggest 10-20 hosts
- Hit rate for DNS = 80% → \(1 - \frac{\text{DNS/connections}}{}\)
  - Is this good or bad?
- Most Internet traffic is Web
  - What does a typical page look like? → average of 4-5 embedded objects → makes 4-5 transfers
  - This alone accounts for 80% hit rate!
- Lower TTLs for A records does not affect performance
- DNS performance really relies more on NS-record caching
Summary

- Motivations → large distributed database
  - Scalability
  - Independent update
  - Robustness

- Hierarchical database structure
  - Zones
  - How is a lookup done

- Caching/prefetching and TTLs

- Reverse name lookup