Name Servers

Jonathan Walpole
CSE515 Distributed Computing Systems
Domain Name System (DNS)

- Internet hosts, routers like to use fixed length addresses (numbers)
  - IP address (32 bit) - used for addressing datagrams
- Humans like to use names
  - www.cse.ogi.edu
- Name servers, such as DNS
  - Map from names to IP addresses
  - Map from IP addresses to names
Early Internet Naming

• Flat namespace
  – /etc/hosts
  – SRI kept main copy
  – Downloaded regularly

• Problems
  – Count of hosts was increasing: machine per domain → machine per user
    • Many more downloads
    • Many more updates
Goals for a New Naming System

• Implement a wide area distributed database
  – Scalability
  – Decentralized maintenance
  – Robustness, fault-tolerance
  – Global scope
    • Names mean the same thing everywhere
  – Don’t need
    • Atomicity
    • Strong consistency
Why Not Centralize DNS?

- Single server with all name-to-IP address mappings
  - single point of failure
  - traffic volume
  - distant centralized database (performance)
  - maintenance
  - doesn’t scale!
DNS (Domain Name System)

- distributed database implemented in hierarchy of many name servers
- decentralized control and management of data
- application-layer protocol used by hosts and name servers
  - communicate to resolve names (name/address translation)
  - core Internet function implemented as application-layer protocol
- complexity at network’s “edge”
DNS Name Space

- Hierarchical canonical name space
  - www.cse.ogi.edu
DNS Name Servers

• Authoritative name servers store parts of the database

• Names assigned to authoritative name servers
  – For a host, authority stores that host’s IP address, name
  – Responds to queries for host’s IP address
  – Perform name/address translation for that host’s name

• Root name server knows authoritative servers for particular subdomains
  – Hierarchy organizes authoritative name servers
  – Reserving a domain gives you control of entry in root name server for particular names
DNS Name Lookup

• hierarchical lookup
  – Each host has a pointer to a local name server to query for unknown names
  – Each local name server knows root of hierarchy
  – Root points to sub-levels, sub-levels point to deeper sub-levels, … , deeper sub-levels point to leaf name server representing authority for unknown name
DNS Name Lookup Example

Root name servers:
- may not know authoritative name server
- may know intermediate name server: who to contact to find authoritative name server
- multiple root name servers for fault-tolerance

Example:
- surf.eurecom.fr wants to talk to gaia.cs.umass.edu
  - contact local dns server
  - local dns contacts root
  - root contacts authoritative (or next level to authoritative)
DNS Root Name Servers

- contacted by local name server that can not resolve any part of the 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 (all that fit in a 512 octet SOA record)
DNS Server Name Database

- DB contains tuples called resource records (RRs)
  - RR contains type, class and application data
    - Before types added, only one record type (A)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines types, e.g. for IN:
    - A = address
    - NS = name server
    - CNAME = canonical name (for aliasing)
    - HINFO = CPU/OS info
    - MX = mail exchange
    - PTR = pointer for reverse mapping of address to name

- nslookup example
DNS Record Types

Resource records (RR) and their types

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 IP address 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 mailserver associated with name
**DNS Database**

<table>
<thead>
<tr>
<th>Type of record</th>
<th>Associated entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Zone</td>
<td>Holds information on the represented zone</td>
</tr>
<tr>
<td>A</td>
<td>Host</td>
<td>Contains an IP address of the host this node represents</td>
</tr>
<tr>
<td>MX</td>
<td>Domain</td>
<td>Refers to a mail server to handle mail addressed to this node</td>
</tr>
<tr>
<td>SRV</td>
<td>Domain</td>
<td>Refers to a server handling a specific service</td>
</tr>
<tr>
<td>NS</td>
<td>Zone</td>
<td>Refers to a name server that implements the represented zone</td>
</tr>
<tr>
<td>CNAME</td>
<td>Node</td>
<td>Symbolic link with the primary name of the represented node</td>
</tr>
<tr>
<td>PTR</td>
<td>Host</td>
<td>Contains the canonical name of a host</td>
</tr>
<tr>
<td>HINFO</td>
<td>Host</td>
<td>Holds information on the host this node represents</td>
</tr>
<tr>
<td>TXT</td>
<td>Any kind</td>
<td>Contains any entity-specific information considered useful</td>
</tr>
</tbody>
</table>

- The most important types of resource records forming the contents of nodes in the DNS name space.
DNS DB Example

- An excerpt from the DNS database for the zone *cs.vu.nl*.

<table>
<thead>
<tr>
<th>Name</th>
<th>Record type</th>
<th>Record value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.vu.nl</td>
<td>SOA</td>
<td>star (1999121502,7200,3600,2419200,86400)</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td>star.cs.vu.nl</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td>top.cs.vu.nl</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td>solo.cs.vu.nl</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>TXT</td>
<td>&quot;Vrije Universiteit - Math. &amp; Comp. Sc.&quot;</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td>1 zephyr.cs.vu.nl</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td>2 tornado.cs.vu.nl</td>
</tr>
<tr>
<td>cs.vu.nl</td>
<td>MX</td>
<td>3 star.cs.vu.nl</td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>HINFO</td>
<td>Sun Unix</td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>MX</td>
<td>1 star.cs.vu.nl</td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>MX</td>
<td>10 zephyr.cs.vu.nl</td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>A</td>
<td>130.37.24.6</td>
</tr>
<tr>
<td>star.cs.vu.nl</td>
<td>A</td>
<td>192.31.231.42</td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>HINFO</td>
<td>Sun Unix</td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>MX</td>
<td>1 zephyr.cs.vu.nl</td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>MX</td>
<td>2 tornado.cs.vu.nl</td>
</tr>
<tr>
<td>zephyr.cs.vu.nl</td>
<td>A</td>
<td>192.31.231.66</td>
</tr>
<tr>
<td><a href="http://www.cs.vu.nl">www.cs.vu.nl</a></td>
<td>CNAME</td>
<td>soling.cs.vu.nl</td>
</tr>
<tr>
<td>ftp.cs.vu.nl</td>
<td>CNAME</td>
<td>soling.cs.vu.nl</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>HINFO</td>
<td>Sun Unix</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>MX</td>
<td>1 soling.cs.vu.nl</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>MX</td>
<td>10 zephyr.cs.vu.nl</td>
</tr>
<tr>
<td>soling.cs.vu.nl</td>
<td>A</td>
<td>130.37.24.11</td>
</tr>
<tr>
<td>laser.cs.vu.nl</td>
<td>HINFO</td>
<td>PC MS-DOS</td>
</tr>
<tr>
<td>laser.cs.vu.nl</td>
<td>A</td>
<td>130.37.30.32</td>
</tr>
<tr>
<td>vucs-das.cs.vu.nl</td>
<td>PTR</td>
<td>0.26.37.130.in-addr.arpa</td>
</tr>
<tr>
<td>vucs-das.cs.vu.nl</td>
<td>A</td>
<td>130.37.26.0</td>
</tr>
</tbody>
</table>
DNS MX Record Type

• MX records point to mail exchanger for a name
  – E.g. mail.acm.org is MX for acm.org

• Addition of MX record type proved to be a challenge
  – How to get mail programs to lookup MX record for mail delivery rather than A record?
  – Needed critical mass of such mailers
DNS Server Database Distribution

- **Administrative hierarchy**
  - Organized into regions known as “zones”
  - “.” as separator
  - zone = contiguous section of name space

- **Zones created by convincing owner node to delegate a subzone**
  - umass.edu zone delegates cs.umass.edu to a different set of authoritative name servers
  - Each zone contains multiple redundant servers
    - Primary (master) name server updated manually
    - Secondary (redundant) servers updated by zone transfer of name space
    - Provides fault-tolerance within zone

- **Host name to address section**
  - Top-level domains → edu, gov, ca, us, etc.
  - Sub-domains = subtrees
  - Human readable name = leaf → root path
DNS Client Lookups

- Each host has a resolver
  - Typically a library that applications can link gethostbyname()
  - Local name servers hand-configured (e.g. /etc/resolv.conf) or automatically configured (DHCP)
    - Can specify a file /etc/hosts
    - Can specify a name server by its IP address (i.e. 129.95.50.2)
  - Host queries local name server for unknown names

```bash
state > more /etc/resolv.conf
domain cse.ogi.edu
nameserver 129.95.40.4
nameserver 129.95.90.19
nameserver 129.95.40.2
state >
```
DNS Name Servers

• Configured with well-known root servers
  – Currently \{a-m\}.root-servers.net

• Local servers
  – Typically answer queries about local zone
  – Typically do a lookup of distant host names for local hosts
Lookup Methods

• Recursive queries
  – Server goes out and searches for more info on behalf of the client (recursive)
  – Only returns final answer or “not found”
  – Puts burden of name resolution on contacted name server
  – Heavy load?
    • Root server implosion

• Iterative
  – Server responds with as much as it knows (i.e. name of server to contact next)
  – “I don’t know this name, but ask this server”
  – Client iteratively queries additional servers
Recursive Name Resolution
Iterative Name Resolution

1. ⟨nl, vu, cs, ftp⟩
2. #⟨nl⟩, ⟨vu, cs, ftp⟩
3. ⟨vu, cs, ftp⟩
4. #⟨vu⟩, ⟨cs, ftp⟩
5. ⟨cs, ftp⟩
6. #⟨cs⟩, ⟨ftp⟩
7. ⟨ftp⟩
8. #⟨ftp⟩

Client's name resolver

Root name server

Name server nl node

Name server vu node

Name server cs node

Nodes are managed by the same server
Recursive DNS Example

host **surf.eurecom.fr** wants IP address of **gaia.cs.umass.edu**

1. Contacts its local DNS server, **dns.eurecom.fr**

2. **dns.eurecom.fr** contacts root name server, if necessary

3. root name server contacts authoritative name server, **dns.umass.edu**, if necessary
DNS Iterated Queries

recursive query:
- puts burden of name resolution on contacted name server
- heavy load?
- root servers now disable recursive queries (RFC 2010)

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

- Client does recursive request to local name server
- Local name server does iterative requests to find name
- Local name server has knowledge of root servers
- Steps for resolving www.ogi.edu
  - Application calls gethostbyname()
  - Resolver contacts local name server (S₁)
  - S₁ queries root server (S₂) for (www.ogi.edu)
  - S₂ returns NS record for ogi.edu (S₃)
  - S₁ queries S₃ for www.ogi.edu
  - S₃ returns A record for www.ogi.edu
- Can return multiple addresses → what does this mean?
DNS Caching

- 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 also cached
  - Don’t have to repeat past mistakes
  - E.g. misspellings
- Cached data periodically times out
  - Soft state
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
  - TTL affects DNS-based load balancing techniques
- update/notify mechanisms under design by IETF
  - RFC 2136
DNS Lookup Example
Subsequent Lookup Example

Client

ftp.cse.ogi.edu

cse.ogi.edu entry cached

Local DNS server

root & edu DNS server

ugi.edu DNS server

cse.ogi.edu DNS server
A word about iterated queries and caching

- Why not do iterative queries from host and recursive queries from local DNS server?
  - Win2k client
    - Does iterative queries from host
  - Caching implications?
### Implications of Lookup Methods on Caching

<table>
<thead>
<tr>
<th>Server for node</th>
<th>Should resolve</th>
<th>Looks up</th>
<th>Passes to child</th>
<th>Receives and caches</th>
<th>Returns to requester</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td><code>&lt;ftp&gt;</code></td>
<td><code>#&lt;ftp&gt;</code></td>
<td><code>--</code></td>
<td><code>--</code></td>
<td><code>#&lt;ftp&gt;</code></td>
</tr>
<tr>
<td>vu</td>
<td><code>&lt;cs,ftp&gt;</code></td>
<td><code>#&lt;cs&gt;</code></td>
<td><code>&lt;ftp&gt;</code></td>
<td><code>#&lt;ftp&gt;</code></td>
<td><code>#&lt;cs&gt;</code></td>
</tr>
<tr>
<td><code>cs,ftp&gt;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><code>#&lt;cs&gt;</code></td>
</tr>
<tr>
<td>nl</td>
<td><code>&lt;vu,cs,ftp&gt;</code></td>
<td><code>#&lt;vu&gt;</code></td>
<td><code>&lt;cs,ftp&gt;</code></td>
<td><code>#&lt;cs&gt;</code></td>
<td><code>#&lt;vu&gt;</code></td>
</tr>
<tr>
<td><code>vu,cs,ftp&gt;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><code>#&lt;vu&gt;</code></td>
</tr>
<tr>
<td>root</td>
<td><code>&lt;nl,vu,cs,ftp&gt;</code></td>
<td><code>#&lt;nl&gt;</code></td>
<td><code>&lt;vu,cs,ftp&gt;</code></td>
<td><code>#&lt;vu&gt;</code></td>
<td><code>#&lt;nl&gt;</code></td>
</tr>
<tr>
<td><code>nl,vu,cs,ftp&gt;</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><code>#&lt;nl&gt;</code></td>
</tr>
</tbody>
</table>

- Recursive name resolution of `<nl, vu, cs, ftp>`. Name servers cache intermediate results for subsequent lookups.
DNS issues

- Static configuration (root server list)
- Inflexible name space
- Lack of support for mobile entities
- Lack of exponential back-off / congestion control
- UDP used for queries
  - Need reliability → Why not TCP?
- Vulnerability of 13 TLD servers
General Naming Issues
Name Spaces (1)

- A general naming graph with a single root node.
Linking and Mounting (1)

- The concept of a symbolic link explained in a naming graph.
Linking and Mounting (2)

- Mounting remote name spaces through a specific process protocol.
Linking and Mounting (3)

- Organization of the DEC Global Name Service
Name Space Distribution (1)

- An example partitioning of the DNS name space, including Internet-accessible files, into three layers.
Name Space Distribution (2)

<table>
<thead>
<tr>
<th>Item</th>
<th>Global</th>
<th>Administrative</th>
<th>Managerial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographical scale of network</td>
<td>Worldwide</td>
<td>Organization</td>
<td>Department</td>
</tr>
<tr>
<td>Total number of nodes</td>
<td>Few</td>
<td>Many</td>
<td>Vast numbers</td>
</tr>
<tr>
<td>Responsiveness to lookups</td>
<td>Seconds</td>
<td>Milliseconds</td>
<td>Immediate</td>
</tr>
<tr>
<td>Update propagation</td>
<td>Lazy</td>
<td>Immediate</td>
<td>Immediate</td>
</tr>
<tr>
<td>Number of replicas</td>
<td>Many</td>
<td>None or few</td>
<td>None</td>
</tr>
<tr>
<td>Is client-side caching applied?</td>
<td>Yes</td>
<td>Yes</td>
<td>Sometimes</td>
</tr>
</tbody>
</table>

- A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, as an administrative layer, and a managerial layer.
Directory Services
The X.500 Name Space (1)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Abbr.</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>C</td>
<td>NL</td>
</tr>
<tr>
<td>Locality</td>
<td>L</td>
<td>Amsterdam</td>
</tr>
<tr>
<td>Organization</td>
<td>O</td>
<td>Vrije Universiteit</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>OU</td>
<td>Math. &amp; Comp. Sc.</td>
</tr>
<tr>
<td>CommonName</td>
<td>CN</td>
<td>Main server</td>
</tr>
<tr>
<td>Mail_Servers</td>
<td></td>
<td>130.37.24.6, 192.31.231,192.31.231.66</td>
</tr>
<tr>
<td>FTP_Server</td>
<td></td>
<td>130.37.21.11</td>
</tr>
<tr>
<td>WWW_Server</td>
<td></td>
<td>130.37.21.11</td>
</tr>
</tbody>
</table>

- A simple example of a X.500 directory entry using X.500 naming conventions.
The X.500 Name Space (2)

- Part of the directory information tree.
The X.500 Name Space (3)

- Two directory entries having *Host_Name* as RDN.
Naming versus Locating Entities

- Direct, single level mapping between names and addresses.
- Two-level mapping using identities.
Forwarding Pointers (1)

- The principle of forwarding pointers using (proxy, skeleton) pairs.

Diagram:
- Process P1
  - Proxy p
  - Interprocess communication

- Process P2
  - Proxy p
  - Skeleton
  - Proxy p' refers to same skeleton as proxy p

- Process P3
  - Skeleton
  - Identicalproxy
  - Local invocation
  - Identical skeleton

- Process P4
  - Object
Forwarding Pointers (2)

- Redirecting a forwarding pointer, by storing a shortcut in a proxy.
Home-Based Approaches

1. Send packet to host at its home
2. Return address of current location
3. Tunnel packet to current location
4. Send successive packets to current location

Host's home location
Client's location
Host's present location
Hierarchical Approaches (1)

- Hierarchical organization of a location service into domains, each having an associated directory node.
Hierarchical Approaches (2)

- An example of storing information of an entity having two addresses in different leaf domains.
Hierarchical Approaches (3)

- Node has no record for E, so that request is forwarded to parent
- Node knows about E, so request is forwarded to child

Look-up request

Domain D
Hierarchical Approaches (4)

- An insert request is forwarded to the first node that knows about entity $E$.
- A chain of forwarding pointers to the leaf node is created.
Pointer Caches (1)

- Caching a reference to a directory node of the lowest-level domain in which an entity will reside most of the time.

Diagram:
- cached pointers to node dir(D)
- E moves regularly between the two subdomains
- Domain D
Pointer Caches (2)

- A cache entry that needs to be invalidated because it returns a non-local address. Cached pointer to node dir(D) which should be invalidated.

Original address (is still valid)

New address
Scalability Issues

- The scalability issues related to uniformly placing subnodes of a partitioned root node across the network covered by a location service.