Distributed Naming

EECS 591
Farnam Jahanian
University of Michigan

Reading List

• Tanenbaum Chapter 4.1-4.2, 4.3(optional)

“Any problem in computer science can be solved with another layer of indirection”
Distributed Naming

• Name: string of bits or characters referring to an entity (resources, services, mailboxes, newsgroups, web pages, network connections, processors, …)

• Identifier is the unique name associated with an entity.
  – Domain names and URLs are good identifiers –
    • barney.eecs.umich.edu identifies a host
    • www.msn.com identifies a service

• An entity has an access point, and the name associated with an access point is an address.
  – e.g. IP address

• Location independence: a name for an entity (i.e. identifier) is independent from its addresses (i.e. name of access points).

Naming: What problem does it solve?

• Naming is a layer of indirection
• Makes objects human readable
• Hides complexity and dynamics
  • Multiple lower-layer objects can have one name
  • Changes in lower-layer objects hidden
• Allows an object to be found in different ways
  • One object can have multiple names
Name Spaces (1)

- Names are organized into a name space – typically represented as a labeled directed graph with leaf nodes and directory nodes.
- Relative vs. absolute path names

![Diagram of a general naming graph with a single root node.](image1)

Linking and Mounting (1)

- The concept of a symbolic link explained in a naming graph.

![Diagram showing linking and mounting concept.](image2)
• Mounting remote name spaces through a specific process protocol.
Name Space Distribution (1)

- An example partitioning of the DNS name space, including Internet-accessible files, into three layers.
- The name space is divided into non-overlapping parts, called zones—each zone is implemented by a separate name server.

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 administrational layer, and a managerial layer.
Name Resolution

- Name spaces offer a mechanism for storing and retrieving info about entities by means of names.
- The process of looking up a name is called name resolution.
- Name resolution service maps names to addresses.
- Iterative vs. recursive name resolution.

Implementation of Name Resolution (1)

Iterative Name Resolution
Recursive Name Resolution

Implementation of Name Resolution (3)

<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>&lt;ftp&gt;</td>
<td>#&lt;ftp&gt;</td>
<td>--</td>
<td>--</td>
<td>#&lt;ftp&gt;</td>
</tr>
<tr>
<td>vu</td>
<td>&lt;cs, ftp&gt;</td>
<td>#&lt;cs&gt;</td>
<td>&lt;ftp&gt;</td>
<td>#&lt;ftp&gt;</td>
<td>#&lt;cs&gt; #&lt;cs, ftp&gt;</td>
</tr>
<tr>
<td>nl</td>
<td>&lt;vu, cs, ftp&gt;</td>
<td>#&lt;vu&gt;</td>
<td>&lt;cs, ftp&gt;</td>
<td>#&lt;vu&gt; #&lt;vu, cs&gt; #&lt;vu, cs, ftp&gt;</td>
<td>#&lt;vu&gt; #&lt;vu, cs&gt; #&lt;vu, cs, ftp&gt;</td>
</tr>
<tr>
<td>root</td>
<td>&lt;nl, vu, cs, ftp&gt;</td>
<td>#&lt;nl&gt;</td>
<td>&lt;vu, cs, ftp&gt;</td>
<td>#&lt;nl, vu, cs, ftp&gt;</td>
<td>#&lt;nl, vu, cs, ftp&gt;</td>
</tr>
</tbody>
</table>

- Recursive name resolution of <nl, vu, cs, ftp>. Name servers cache intermediate results for subsequent lookups.
Implementation of Name Resolution (4)

- The comparison between recursive and iterative name resolution with respect to communication costs.

Domain Name System (DNS)

- Distributed directory service; DNS is simple but powerful
- Hierarchical name space
- Each level separated by ‘.’
  - Analogous to ‘/’ separator in file systems
- One global root
  - Replicated across <20 root servers!
  - There have been Denial of Service (DoS) attacks on
  - these root servers, none real successful
  - Because of caching, queries to root servers relatively rare
Domain Name System (DNS)

- DNS is a Hierarchical name space
  - A subtree is called a domain
  - A path name to its root is called a domain name
  - A domain name can be relative or absolute
- The name space is divided into non-overlapping parts, called zones – each zone is implemented by a separate name server
- The contents of a node is defined by a collection of resource records
- Only one type of query: Query(domain name, RR type)
  - Resource Record (RR) type is like an attribute type
- Answer(values, additional RRs)
- Limited number of RR types
- Hard to make new RR types, but not for technical reasons, because each requires global agreement

The DNS Name Space

<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 Implementation (1)

- An excerpt from the DNS database for the zone cs.vu.nl.
- cs.vu.nl domain implemented as a single zone.

<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 (1999121502,7200,3600,24199200,86400)</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 torrado.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 torrado.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>PG 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 Implementation (2)

- Part of the description for the vu.nl domain which contains the cs.vu.nl domain.

<table>
<thead>
<tr>
<th>Name</th>
<th>Record type</th>
<th>Record value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.vu.nl</td>
<td>NS</td>
<td>solo.cs.vu.nl</td>
</tr>
<tr>
<td>solo.cs.vu.nl</td>
<td>A</td>
<td>130.37.21.1</td>
</tr>
</tbody>
</table>
Primary and Secondary Servers

- Each zone is implemented by a name server, replicated for availability.
- Updates are handled by primary server, by modifying local DNS DB.
- Secondary requests the primary server to transfer its content.
- RR for all the nodes in a zone are kept in the local DNS DB.

DNS Cache Management:

- All RRs have Time-to-live (TTL) values
- When TTL expires, cache entries are removed
- NS RRs tend to have long TTLs (reduces load on higher-level servers)
- A RRs may have very short TTLs (1 min for web services, 1 day for typical hosts)
  - What if you want a quick failover for web servers? Keep TTL for web server’s A RRs very short.
- Why is DNS iterative and not recursive?

Locating Mobile Entities (Section 4.2)

- Mobile entity:
  - An entity whose address often changes, laptop running DHCP
  - Not necessarily physical mobility, e.g. dial-up

- Is mobility an issue for DNS?
  - NOT REALLY!
  - Mobility (in practice) affects leaf DNS servers
    - A RR TTL is short, but NS RR TTL is long

- What is the problem?
  - Most mobile nodes are clients; servers are rarely mobile
  - Clients initiate connects, not receive it!
  - Special cases: email, instant msg’ing, VoIP
    - Application-specific registration
    - Clients connects to email server, IM server, SIP server, …
Naming versus Locating Entities

(a) Direct, single level mapping between names and addresses.
(b) Two-level mapping using identities:
   Separation of naming (name $\rightarrow$ identifier) from locating (identifier $\rightarrow$ current location)

Location Service

- Simple solutions for LANs: NOT Scalable
  - Broadcasting & multicasting (similar to ARP)
    - Drawback: in practice for large networks, hardware support needed
  - Forwarding pointers: when moving from A to B, leave a forward reference behind
    - Many drawbacks: long chains, reliance on intermediate nodes, broken links

- Mobile IP approach: (home-based approach)
  - Introduce a home location that keeps track of current location of an entity
  - Mobile nodes has a stable home address at its home network (home agent with fixed IP address)
  - When a mobile node moves, it gets a “care-of-address” … This address is registered at the home agent
  - When the home agent receives a packet for the mobile node:
    - If it’s still at “home” (local network), just forward the packet
    - Otherwise, the packet is tunneled to the current location of the mobile node, wrapped in an IP packet and sent to the “care-of-address”
    - Sender may be informed of the mobile node’s current location (optional)
Location Service

• Issues with Mobile IP approach:
  – Increased communication latency (a problem in large-scale networks)
  • Two-tier or hierarchical
  – Fixed home location must be highly available
  – What if the nodes moves permanently?
  • Use traditional name service

• Two-tier or hierarchical approaches
  – Scalability
  – PCS

Forwarding Pointers (1)

• The principle of forwarding pointers using (proxy, skeleton) pairs.
Forwarding Pointers (2)

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

Home-Based Approaches (Mobile IP)

- The principle of Mobile IP.
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)

- Looking up a location in a hierarchically organized location service.

Hierarchical Approaches (4)

a) An insert request is forwarded to the first node that knows about entity E.

b) 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.

Pointer Caches (2)

- A cache entry that needs to be invalidated because it returns a nonlocal address, while such an address is available.
Scalability Issues

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