Distributed Systems

3. Naming
Names, Addresses, Identifiers

- Naming is about mapping between names, addresses, identifiers and the referred entities

- Names (a bit- or character-string referring to an entity)
  - E.g. John Smith or ftp-server
  - Can be *human-friendly* (or not) and *location dependent* (or not)

- Addresses (define access points)
  - Entities can be operated through an *access point*
  - The name of an access point is an address
  - E.g. phone number, or IP-address + port for a service

- Identifiers (*unique* identifiers)
  - A (true) identifier is a name with the following properties
    1. Each identifier refers to at most 1 entity and
    2. Each entity is referred to by at most 1 identifier
    3. An identifier always refers to the same entity (never reused)
  - E.g. John Smith + social security number, or MAC address
Name Spaces

- A set of related names, typically represented as a directed graph (maybe restricted to a DAG or a tree, or even a list)
  - *Path names* (sequence of edges) can be *absolute* and *relative*
  - *Name resolution* looks up the content of a referred node
  - Names can be *global* in a whole system (e.g. names of commands, like “cd”) or *local* (relative to an implicitly known node, e.g. home directory)
Alias names

- The same entity may have several *alias* names
  - *Hard links*: The same node is referred via several path names (see the 2 links leading to n5 on the previous slide)
  - *Symbolic links*: The referred node contains a further (abs.) reference, which can be used instead of the original path name
Linking and Mounting

- Mounting remote name spaces ("remote symbolic link") needs
  - A specific protocol (e.g. NFS, see later)
  - At a certain mount point of a given server
  - A name, containing access protocol, remote server, foreign mounting point

```
REMOTE/vu/mbox
```

```
"nfs://flits.cs.vu.nl/home/steen"
```
Name Space Distribution (1)

- Partitioning of the DNS name space into three layers
- More details of DNS see at “computer networks”
A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global, an administrative layer, and a managerial layer

<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>
Implementation of Name Resolution (1)

- The principle of iterative name resolution
  - The address of the root server must be well known
Implementation of Name Resolution (2)

• The principle of recursive name resolution
  + Enables efficient caching and reduces communication
  - Causes higher performance demand on the name servers
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;cs,ftp&gt;</td>
<td>#&lt;vu&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&gt; #&lt;nl,vu&gt; #&lt;nl,vu,cs,ftp&gt;</td>
<td>#&lt;nl&gt; #&lt;nl,vu&gt; #&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
The comparison between recursive and iterative name resolution with respect to communication costs.
The X.500 Name Space (1)

• More than a naming service
  ➢ A directory service with search
  ➢ Items can be found based on properties (not only full names)
  ➢ X.500 defines attribute-value pairs, such as:

<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>L</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>
The X.500 Name Space (2)

- The collection of directory entries
  - Form the Directory Information Base (DIB)
  - Naming attributes (Country etc.) are called Relative Distinguished Names (RDN)
  - Each record is uniquely named, by a sequence of RDNs

- Directory Information Tree (DIT)
  - Naming graph
  - Each node represents a directory entry and an X.500 record
  - With read we can read a certain record
  - With list we can read all outgoing edges of the node

- A simplified version of X.500 is generally used
  - Known as Lightweight Directory Access Protocol (LDAP)
The X.500 Name Space (3)

- Part of the directory information tree
  - The X.500 name /C=NL/O=Vrije Universiteit/OU=Math. & Comp. Sc.
  - is analog to the DNS name nl.vu.cs
The X.500 Name Space (4)

- Two directory entries having Host_Name as Relative Distinguished Name (RDN)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>NL</td>
</tr>
<tr>
<td>Locality</td>
<td>Amsterdam</td>
</tr>
<tr>
<td>Organization</td>
<td>Vrije Universiteit</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>Math. &amp; Comp. Sc.</td>
</tr>
<tr>
<td>CommonName</td>
<td>Main server</td>
</tr>
<tr>
<td>Host_Name</td>
<td>star</td>
</tr>
<tr>
<td>Host_Address</td>
<td>192.31.231.42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>NL</td>
</tr>
<tr>
<td>Locality</td>
<td>Amsterdam</td>
</tr>
<tr>
<td>Organization</td>
<td>Vrije Universiteit</td>
</tr>
<tr>
<td>OrganizationalUnit</td>
<td>Math. &amp; Comp. Sc.</td>
</tr>
<tr>
<td>CommonName</td>
<td>Main server</td>
</tr>
<tr>
<td>Host_Name</td>
<td>zephyr</td>
</tr>
<tr>
<td>Host_Address</td>
<td>192.31.231.66</td>
</tr>
</tbody>
</table>
Naming versus Locating Entities

- DNS and X.500 assume restricted change in the system
- For highly mobile entities additional location services are needed
  - A simple example is ARP (IP $\rightarrow$ MAC addr. based on LAN-broadcast)

\[\text{a) Direct mapping between user-friendly names and addresses}\]
\[\text{b) 2-level mapping using (non-user-friendly) identities}\]
Forwarding Pointers (1)

- A moving entity leaves behind a reference (proxy or stub) at a client pointing to the place at the server (skeleton or scion)
- Long SSP chains can be built (stub → skeleton → stub → skeleton …)
Forwarding Pointers (2)

- Redirecting a forwarding pointer, by storing a shortcut in a proxy
- The response can be sent directly, or along the whole reverse path, thus updating all intermediate proxies
- Pointer forwarding is fully transparent, but vulnerable to errors
Home-Based Approaches

- This is also the principle of Mobile IP (see “Computer Networks”)

László Böszörményi
Distributed Systems
Naming - 19
Hierarchical Approaches (1)

- A generalization of the 2-tired home based approach
- Hierarchical organization of a location service into domains (like DNS)
- The leaf level corresponds e.g. to LANs or mobile phone cells
Hierarchical Approaches (2)

- A replicated entity having two addresses in different leaf domains
- The higher level directories store only pointers
Hierarchical Approaches (3)

- Looking up a location in a hierarchically organized location service
- Lookup searches in an increasing ring, thus exploiting *locality*
Pointer Caches

- Caching is only effective if changes are rare
- It is better to cache addresses to directories than to individual entities

Cached pointers to node \( \text{dir}(D) \)

E moves regularly between the two subdomains
Scalability Issues

• The root node must be partitioned if the system is big
  ➢ Its sub-nodes could be placed uniformly distributed
The Problem of Unreferenced Objects

- Unreachable entities should be removed
- Distributed garbage collection is hard
  - Reference counting is simple, but cannot handle cycles