Naming and Directory Services

1. General Concepts

2. Examples
   - Domain Name Service (DNS)
   - X.500/LDAP
   - Corba Naming Service
1. General Concepts

Structure:

- name/attributes may refer to a person, device, object, service, etc.
- types of resolution:
  - search by name (white pages)
  - search by attributes (yellow pages)
Naming versus Locating Entities

(a) Direct, single level mapping between names and addresses.
(b) Two-level mapping using identities.
Design Issues

- **Binding information is relatively static**
  - Optimized for high-volume reading

- **Principal operations are query/change**
  - Need not support complex database operations
  - May provide a simple query API

- **Information tailored for a specific purpose**
  - Predefined/fixed schema
Design Issues

scalability → partitioning

reliability → replication

performance → caching
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>

Nodes are managed by the same server
Recursive Name Resolution

1. <nl,vu,cs,ftp>

2. <vu,cs,ftp>

3. <cs,ftp>

4. <ftp>

5. #<ftp>

6. #<cs,ftp>

7. #<vu,cs,ftp>

8. #<nl,vu,cs,ftp>

Client's name resolver

Root name server

Name server nl node

Name server vu node

Name server cs node
Information Views

Directory Service

User
(access, update)

Administrator
(define/update structure)

Security
(access control)

Deployment
(distribution, replication)
2. Examples

• Domain Name Service (DNS)
  www.cs.vt.edu   →   IP address

• X.500/Lightweight Directory Access Protocol (LDAP)
  C=US, OU=Virginia Tech, CN=Dennis Kafura   →   email address

• Corba Naming Service
  name1.kind1/name2.kind2/name3.kind3   →   object
DNS Organization

- An example partitioning of the DNS name space, including Internet-accessible files, into three layers.
## DNS Organization

<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.
# The DNS Name Space - Records

<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>
DNS 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>vuchs-das.cs.vu.nl</td>
<td>PTR</td>
<td>0.26.37.130.in-addr.arpa</td>
</tr>
<tr>
<td>vuchs-das.cs.vu.nl</td>
<td>A</td>
<td>130.37.26.0</td>
</tr>
</tbody>
</table>
LDAP: Evolution
LDAP/X.500 - Models

- Information (what is stored in an entry)
- Naming (how are entries identified/organized)
- Functional (what operations are provided)
- Security (how is authentication and authorization provided)
### LDAP Information Model

**entry**

- **attribute**
  - Mandatory objectClass attribute defining the schemas for other mandatory and permitted attributes

- **type**
  - Defines syntax of value(s)

- **value**

<table>
<thead>
<tr>
<th>attribute</th>
<th>syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>commonName (CN)</td>
<td>cis: case ignore string</td>
</tr>
<tr>
<td>telephoneNumber</td>
<td>tel: text, ignoring blanks and dashes</td>
</tr>
</tbody>
</table>
LDAP Naming Model

Directory Information Tree (DIT)

- relative distinguished name (RDN) represented as a \textit{name} = \textit{value} pair (e.g. OU = Computer Science)

- distinguished name (DN) represented as a comma-separated sequence of RDNs (e.g., CN=Dennis Kafura, OU=Computer Science, O=Virginia Tech, C= US)
The “root DSE” stored at a server is an empty (zero-length) distinguished name that is used to store as its attributes:

- suffixes provided by this server
- object classes and attribute schema
LDAP Functional Model

- **Operation Categories**
  - Query (search, compare)
  - Update (add, delete, modify)
  - Authentication (bind, unbind, abandon)

- **Search parameters**
  - base (DN of where to begin search)
  - scope (extent of subtree examined)
  - filter (criteria for an entry to be matched)
  - attributes (list of attributes returned from each matched entry)
  - alias (whether aliases are followed)
  - limits (in size/time)

- **Filters**
  - boolean combination of attribute-value comparisons
  - example: ( | (SN = Smith) ( & (OU=Autstin) (SN=Miller)))
LDAP Security Model

- **Authentication**
  - None (anonymous session)
  - Basic (DN (i.e., username) and password)
  - Simple Authentication and Security Layer (SASL)
    - GSSAPI
    - Kerberos
    - External (e.g., SSL/TLS)

- **Access control**
  - vendor specific
  - access control list is typical
Corba Naming Service - Names

• Implemented as a CORBA service; operations described as an IDL interface.
• Names: each name component can refer to
  – naming context (an object for resolving names)
  – application object
• Naming contexts can be structured to create naming graphs
• Compound names locate an object in a naming graph (a sequence of names defining a path in the naming graph)

```c
struct NameComponent {
    String id;
    String kind;
};
typedef sequence<NameComponent> Name;
```
Operations:

```plaintext
void bind (in Name n, m in Object o);
Object resolve (in Name n);
void undind (Name n);
NamingContext bind_new_context (in Name n);
void destroy ();
```
Corba Naming Service - Federation

A federated naming graph can span multiple naming services.