Name & Directory Services

- Naming concepts
- Service function and goals
- Name resolution
- Server distribution and replication
- Domain Name Service
- X500
- Service Discovery

---

Name Concepts 1

**Names**: are strings used to identify objects such as people, computers, processes, files, communication links.

**Textual names**: are human readable & identify services, people etc.
- `a.person@doc.ic.ac.uk`
- `swan.doc.ic.ac.uk`

**Numeric addresses**: are location dependent eg IP `146.169.14.20`

**Route**: directed path from source to destination
- `smtp!doc-gw!cc-core!janet!xyz`

There is no real distinction between a name, address and route – they are all names within different namespaces which are resolved in different ways

**Object identifiers**: pure names, usually numeric, large context eg `5021334679`
- Never reuse ➔ include timestamp component
- Location independent
- Used to compare identities
- Example unique identifiers?

---

Name Concepts 2

**Name context**: a set of bindings between textual names and attributes for objects (eg directory). Name must be unique within context.

**Name Space**: a set of homogeneous contexts with a syntax for specifying names and rules for resolving names e.g. right to left or left to right.
- A name may specify a path through a namespace
- Namespace name may be used as prefix to name to indicate to the system which name service to use for name resolution
  - Uniform Resource Name (URN) eg `urn:ISBN:0-201-62433` or URLs
- Other examples

---

Name Space

- Naming authority may control the assignment of names to objects within a namespace or within a context e.g.
  - university or department assigning host names
  - X500 Administration domain
  - Mercedes assigning engine & chassis number to a car.

Objects may be registered more than once within a namespace.

- **Alias**: alternative name for an object which either maps directly to the object or onto a path name in the name space, which has to be resolved to locate the object.

- **Group names**: identify a set of objects
  - Multicast name ➔ single name/address which maps to a group of objects
  - Broadcast name ➔ single name/address which maps to all objects in a context (eg nodes on a LAN)
Hierarchic names

- A sequence of name tokens resolved in different contexts
  Syntax: name token (text string) + delimiter eg “/” or “.”
  e.g.: /usr/ic/dse/staff/mss or mss-pc.doc.ic.ac.uk
- Explicit identification of context as in X500
  e.g. C=uk, O=ac, OUN=ic, LN=doc, CN=mss-pc
- Structure has semantic significance i.e. it reflects organisational structure or physical location
  ➔ name changes if object migrates
- Name may be absolute – relative to a root
  e.g.: /usr/ic/dse/staff/mss or relative to a particular current context e.g. ~/public_html/ds-course
- Distributed authorities control local contexts – Why is this good?
- Names may be rather “verbose” e.g. X.500

Flat Names

- Single global context and naming authority for all names
  e.g. National Security numbers, object identifier, computer serial number, Ethernet address
- No semantic information in name
  ➔ Difficult to resolve: which directory to use and how to access directory – hashing?
  ➔ Easy to create – local information
  ➔ How would you create a unique flat object ID?

Name Binding

- An association between a name and an object.
- Name is bound to a set of attributes of an object, one of which may be an address. Other attributes include room number, telephone number for people; type information, quality of service information for services; OS, memory, Ethernet address for workstations.
- Name may be a placeholder which can be bound to different objects at different times e.g. reference – name identifies a “variable” which holds reference or address values
  eg pointer, RPC interfaceref, object-reference,socket
- Sometimes can translate from one namespace to another if there is a binding between names in the different namespaces.
  e.g. IP address to Ethernet address

First Party Binding

- User of a name queries a name service to resolve name or service specification to an address. Then binds object-ref to this address before using it for invocations
  i.e. explicit access to logically centralised name service to find object address is required before object invocation

Third Party Binding

- A third party configuration manager (possibly via a configuration language) binds a required interface to a provided interface on components.
  cf wiring between integrated circuit pins.
Directory Service Functions

- The directory service translates (resolves) names for objects (people, computers, servers, files, software components) into a set of attributes (one of which is usually an address) i.e. lookup
- Complex searches based on attribute values or ranges
  - **Yellow pages or Trading service**: attribute values → list of objects
    - The attribute could be based on type of service offered by a server
    - e.g. lookup type:laser → a list of printer names
  - **Browsing** when user cannot specify query
- Basic Operations: insert, modify or delete entries.
- Manage internal, possibly replicated tables & propagate updates to replicas
- Cache results of queries
- Access control for both reads and updates

Multicast Lookup

- Send single Lookup message with full pathname to group of servers
  - Servers hold full pathname to local directories
  - Can also be used for flat names
  - Fast lookup, but requires multicast support
  - Does not scale to very large systems

Recursive Lookup

- Cannot block S1 during time taken to resolve complete path name
  - threads within server to do lookup
- Increase loading and complexity of server.
- If message passing used, Server 3 can reply directly to client.
Iterative Lookup Using Referrals

- Simpler servers – off load work (referrals) to clients, but there are multiple clients anyway.
- Used with a name server agent in client to do lookup

Replication

- Replicate some directories for performance & availability reasons
  - **Updates:**
    - Write to single master
      - Master is responsible for propagating updates to all replicas
      - Should it maintain strong consistency?
    - Write to any replica – and later merge updates
      - Time stamp updates and latest update wins if updates occur in different servers
  - **Lookups:**
    - Try any local server – it can always get higher or to root
    - How?
      - May have to go up tree to find root directory and then go down tree until reach end token in name.
  - **Caching:**
    - Clients typically cache names & addresses of recently used objects, particularly directories – need validity time

Replica Consistency

- **Strong Consistency:**
  - Lock all copies while updating
    - no lookups while updating
  - Do you block update if all copies are unreachable ie atomic actions? impractical

- **Weak Consistency:**
  - Eventually propagate updates to all copies
    - entries may be out of date in some copies.
    - Address can be considered a hint, which may be incorrect – detected at invocation time.
  - **What to do if address is incorrect?**

Internet Domain Name Service (DNS)

- Used mainly for urls, host names and email addresses.
- Extensible number of fields and each field is variable length text delimited by "."
- eg a.smith@doc.ic.ac.uk      dse-mail.doc.ic.ac.uk
  schmidt@atm.aeg.kn.daimlerbenz.com

- Top level: USA domains + other countries
  - com, edu, gov, mil, org, fr, uk, jp, kr, de
- Second Level
  - e.g. in UK – co, ac, mod
- Third level
  - Institutions or companies e.g. ic, lancs, bt
- Additional Levels
  - Depends on institutions – usually departments.
  - Hierarchy no longer meaningful e.g. dot coms.
**DNS Implementation**

- Naming data divided into Zones which contain:
  - Attribute data for local names, less sub-domains
  - Name & address of at least 2 authoritative name servers for the zone
  - Names and IP addresses of sub-domain name servers
  - Zone management parameters governing caching and replication of zone data.
  - Replicated and partitioned information – update master.
  - Secondary servers download data periodically from master and can cache entries from any other servers.
  - Secondary servers always hold address of one or more masters a numbers of levels above
  - Higher level directories do not change frequently and are cached by many servers.
  - Uses recursive or iterative lookup

**DNS Record Types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA</td>
<td>Start of authority – zone parameters</td>
</tr>
<tr>
<td>A</td>
<td>IP address for a host</td>
</tr>
<tr>
<td>NS</td>
<td>Authoritative nameserver for domain</td>
</tr>
<tr>
<td>Cname</td>
<td>Canonical name for alias</td>
</tr>
<tr>
<td>WKS</td>
<td>Well known service – service names and protocols</td>
</tr>
<tr>
<td>PTR</td>
<td>Domain name for IP address for reverse lookup</td>
</tr>
<tr>
<td>MX</td>
<td>Mail exchange – list of hosts for receiving mail for domain name</td>
</tr>
<tr>
<td>HINFO</td>
<td>Host information – machine architecture and OS</td>
</tr>
<tr>
<td>TXT</td>
<td>Arbitrary text string</td>
</tr>
</tbody>
</table>

**DNS Resource Records**

<table>
<thead>
<tr>
<th>Domain name</th>
<th>Time to live</th>
<th>Class</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc.ic.ac.uk</td>
<td>86400</td>
<td>IN</td>
<td>NS</td>
<td>hawk.doc</td>
<td>Name server for doc</td>
</tr>
<tr>
<td>hawk</td>
<td>86400</td>
<td>IN</td>
<td>A</td>
<td>146.169.1.24</td>
<td>IP address for hawk</td>
</tr>
<tr>
<td>doc.ic.ac.uk</td>
<td>86400</td>
<td>IN</td>
<td>MX</td>
<td>1 pop.doc.ic.ac.uk</td>
<td>Prime mail server for doc</td>
</tr>
<tr>
<td>doc.ic.ac.uk</td>
<td>86400</td>
<td>IN</td>
<td>MX</td>
<td>2 popug.doc.ic.ac.uk</td>
<td>Secondary mail server for doc</td>
</tr>
<tr>
<td>pop</td>
<td>86400</td>
<td>IN</td>
<td>A</td>
<td>146.169.1.54</td>
<td>IP address for pop</td>
</tr>
<tr>
<td>www-dse</td>
<td>86400</td>
<td>IN</td>
<td>CNAME</td>
<td>book.doc</td>
<td>Alias for www-dse</td>
</tr>
</tbody>
</table>

**X500 Directory Service**

- A object-oriented database in which objects represent people, services, servers routers etc.
- Each entry is a set of attributes for an object, one can be tagged as distinguished and is used to identify the object.
- Parts of the information in the Global directory can be stored and controlled by independent administrations
- Differs from normal database:
  - Object-oriented, attribute-based information
  - Optimised for reads rather than updates ie static information
  - Distributed & some directories may be highly replicated
  - Hierarchical naming of objects
  - Potentially large (>1M) numbers of objects
  - No transaction support
- Understanding X500, David Chadwick, Chapman & Hall, 1994
X500 Information Model

- Attributes: eg name, email, phone, fax and room number for a person, sets - members of a group
- Alias: alternative name for object + symbolic link to object
- Collective Attributes: apply to a set of entries or subtree eg switchboard phone no.
- Operational Attributes: set by administrators or system eg date & time of modification, access control information & roles

Directory Information Tree (DIT)

- Relative Distinguished Name (RDN): name within a context e.g. ON=ic
- Distinguished name: determined by position in tree eg C=uk, ON=ibm, LN=ny, CN=Payroll

Directory Schema

- Controls what information can be stored in directory by means of object & attribute class specifications. Uses Abstract Syntax Notation One (ASN.1)
- Predefined Standards – extendable by users
- Standard objects: Top, Alias, Country, Locality, Organization, Organization Unit, Person …
  Uses object class inheritance
- Standard attributes: Name, Description, Fax number, Street address, Postal Code, Country, Owner, Member, …
  Attributes definitions include rules for matching – eg equality, ordering, substrings, ignore case

Example Attribute & Object

```plaintext
name ATTRIBUTE ::= {
  WITH SYNTAX DirectoryString {MAX}
  EQUALITY MATCHING RULE caselgnoreMatch
  SUBSTRING MATCHING RULE caselgnoreSubstringMatch
  ID {id-at-name} }

person OBJECT-CLASS ::= {
  SUBCLASS OF top
  MUST CONTAIN { commonName | surname }
  MAY CONTAIN { description | seeAlso | telephoneNumber | userPassword }
  ID { id-oc-person } }
```
Jini Components

- Jini enables distributed components to locate resources and interact to form a named community (group) to enable "plug and play" distributed applications. Group could be Engineering, home_video etc.

- **Discovery Service** – finds lookup services that keep track of shared resources of a community and hence finds communities

- **Lookup Service** – finds services within a community. Similar to a nameserver or trader that understands Java type hierarchy. A community may be registered with multiple lookup services. Could be used by a hotel guest with laptop to find a nearby printer.

- **Leasing** – supports time-based resource reservation and hence caters for failures. Leases can be renewed or delegated.

- **Remote Events** – allows services to notify each other of change of state.

- **Transaction Service** – two phase commit based atomic transactions to enable reliable interactions.

Jini Discovery

- Multicast request protocol – when application first becomes active it multicasts a request (with a limited time to live) to find a nearby lookup service. Request could be for a known named community or a default one. Lookup service replies with proxy object to be used by client to make requests on lookup service.

- Lookup services announce presence using multicast announce protocol at startup and periodically.

- Unicast Discovery Protocol is used to interact with a known lookup service or one on a remote network to which multicast will not work. Eg `jini://video-service.doc.ic.ac.uk`

Lookup

- Name service maintaining a list of service items.

- A service item contains a proxy object (c.f. stub) which can be downloaded to enable its use, and a set of attributes used to describe the service. Proxy may be more complex than a stub and can act as a driver for a device service.

- A service eg printer can publish service items to one or more lookup services: `resourceClass=printer, type=laser, colour=yes, resolution=600dpi, location=room101, url=http://www.hotelDuLac.com/services/printer57`

Leasing

- Ensures communities are stable, self-healing and resilient to failures

- Resources are reserved for a time period. If client fails, resource is released after period expires.

- Lease can be renewed before expiry or cancelled.

- Can batch lease operations for multiple resources

- Lease can be acquired by a third party on behalf of a client or the lease can be delegated to a third party to manage renewals eg a mobile device with limited processing power can use an agent on a server to manage and renew its leases.
Summary

- Names can be textual or numbers; can represent addresses, routes, identifiers, can be flat or hierarchical.
- Directory service maps a set of attributes to an object – one attribute is usually an address and another a name.
- Logically centralised but implemented as a set of co-operating servers
- Partitioned to reflect application distribution
- Replicated for availability and performance
  - weak or strong consistency?
- Finding initial name server
  - Well known address or configuration file
  - Use discovery protocol based on multicast
- Use local service agent in client node
  - Cache recently used information to improve performance
  - Hold knowledge of at least one server, which can be used to find other servers