Name Services

Nicola Dragoni
Embedded Systems Engineering
DTU Informatics

9.1 Introduction
9.2 Name services
    Case Study: Domain Name System (DNS)
9.3 Directory Services
9.4 Case Study: X.500 Directory Service
Naming

• In a distributed system, names are used to refer to a wide variety of resources (computers, services, remote objects and files, ...).

• Names facilitate communication and resource sharing.

• Processes cannot share particular resources managed by a computer system unless they can name them consistently.

• Users cannot communicate with one another via a distributed system unless they can name one another, for example, with email addresses.
Names and Addresses

• Any process that requires access to a specific resource must possess a name for it.
  ‣ file names: /etc/passwd
  ‣ URLs: http://www.cdk4.net
  ‣ Internet domain names: www.cdk4.net

• **Name**: an identifier permanently associated with an object, independent of its location within the distributed system.

• **Address**: an identifier associated with the current location of the object.
  ‣ **Example**: Email addresses usually have to change when they move between organizations or ISPs; they are not enough in themselves to refer to a specific individual over time.
Names VS Addresses

• An address is how you get to an endpoint.
  
  ‣ Typically, **hierarchical** (for scaling):
    
    - 950 Charter Street, Redwood City CA, 94063
    
    - 204.152.187.11, +1-650-381-6003
  
• A “name” is how an endpoint is referenced.
  
  ‣ Typically, **no structurally significant hierarchy**
    
    - “David”, “Tokyo”, “itu.int”
Binding and Attributes

• We say that a name is resolved when it is translated into data about the named resource or object, often in order to invoke an action upon it.

• The association between a name and an object is called a binding.

• In general, names are bound to attributes of the named objects, rather than the implementation of the object themselves.

• An attribute is the value of a property associated with an object (for instance, its address).

• Example: “www.imm.dtu.dk” (name) is bound to the IP address (attribute) “192.38.82.230” of the IMM Web server (resource).
Name Services

- A name service stores a collection of one or more naming contexts.

- Naming contexts: sets of bindings between textual names and attributes for objects (such as users, computers, services and remote objects).

- Major operation of a name service: name resolution, that is to look up attributes from a given name.

- Operations are also required for creating new bindings, deleting bindings and listing bound names and adding and deleting contexts.
Name Services

• Example: Domain Name System (DNS) names objects across the Internet (maps human-friendly hostnames into IP addresses).

```
HOST(1) BIND9 HOST

NAME
host - DNS lookup utility

SYNOPSIS
host [-aCdlnsTwv] [-c class] [-N ndots] [-R number] [-t type]
       [-W wait] [-m flag] [-4] [-6] {name} {server}

DESCRIPTION
host is a simple utility for performing DNS lookups. It is
normally used to convert names to IP addresses and vice versa.
When no arguments or options are given, host prints a short
```
Nicola-Dragoni-MacBook-Pro:~ nic$ host imm.dtu.dk
imm.dtu.dk has address 192.38.82.230
imm.dtu.dk mail is handled by 10 smtpgw1.imm.dtu.dk.
Nicola-Dragoni-MacBook-Pro:~ nic$  
```
Name Spaces

- **Name space**: collection of all valid names recognized by a particular service.

- Names may have an internal structure that represents their position in a hierarchic name space.

- Most important advantage of hierarchic name spaces: each part of a name is resolved relative to a separate context, and the same name may be used with different meanings in different contexts.

- Example: **file systems** (/etc/passwd is different from /oldetc/passwd).
Example: DNS

- DNS names are called **domain names**.
  - Example: www.cdk4.net (computer), net, com, ac.uk (domains).

- The **DNS name space** has a **hierarchic structure**: a domain name consists of one or more strings called **name components** or **labels**, separated by the delimiter “.”.

- The **name components** are non-null printable strings that do not contain “.”.

- In general, a **prefix** of a name is an initial section of the name that contains only zero or more entire components.
  - Example: www and www.cdk4 are both prefixes of www.cdk4.net.

- DNS names are **not case-sensitive** (www.cdk4.net == WWW.CDK4.NET)
Naming Domains

- **Naming domain**: name space for which there exists a single overall administrative authority for assigning names within it.

- This authority is in overall control of which names may be bound within the domain, but it is free to delegate this task.
Name Resolution

- In general, resolution is an iterative process whereby a name is repeatedly presented to naming contexts.

- A naming context either maps
  - a given name onto a set of primitive attributes directly
  - or it maps it onto a further naming context and a derived name to be represented to that context.

- To resolve a name:
  - it is first presented to some initial naming context
  - resolution iterates as long as further contexts and derived names are output.
Example: UNIX File System

• In the case of file systems, each directory represents a context.

• /etc/passwd is a hierarchic name with two components.
  ‣ The first, “etc” is resolved relative to the context “/” or root.
  ‣ The second part, “passwd”, is relative to the context “/etc”.

• The name /oldetc/passwd can have a different meaning because its second component is resolved in a different context.

• Similarly, the same name /etc/passwd may resolve to different files in the contexts of two different computers.
Name Servers

• *Any name service*, such as DNS, that stores a very large database and is used by a large population will *not* store all of its naming information on a single server computer.

• Such a server would be a bottleneck and a critical point of failure.

• Any heavily used name services should use *replication* to achieve high availability.

• The partitioning of data implies that the *local name server cannot answer all enquiries without the help of other name servers*.

  ‣ A name server in the *dcs.qmul.ac.uk* domain would not be able to supply the IP address of a computer in the domain *cs.purdue.edu* unless it was cached (certainly not the first time it is asked)
• **Navigation**: the process of locating naming data from among more than one name server in order to resolve a name.
Client-Controlled Iterative Negotiation

- DNS supports the model known as **ITERATIVE NAVIGATION**:
  - To resolve a name, a client presents the name to the local name server, which attempts to resolve it;
    - if the local name server has the name, it returns the result immediately;
    - if it does not, it will suggest another server that will be able to help.
  - Resolution proceeds at the new server, with further navigation as necessary until the name is located or is discovered to be unbound.
Multicast Negotiation

- An alternative to iterative negotiation.

- A client multicasts the name to be resolved to a group of name servers.

- Only the server that holds the named attributes responds to the request.

- Unfortunately, however, if the name proves to be unbound, then the request is greeted with silence.
Another Alternative: Server-Controlled Navigation

- **Idea**: a name server coordinates the resolution of the name and passes the result back to the user agent.

- **Non-recursive server-controlled navigation:**
  - any name server can be chosen by the client;
  - this server communicates by multicast or iteratively with its peers.
Another Alternative: Server-Controlled Navigation

- Idea: a name server coordinates the resolution of the name and passes the result back to the user agent.

- Recursive server-controlled navigation:
  - the client once more contacts a single server;
  - if this server does not store the name, the server contacts a peer storing a (larger) prefix of the name, which in turn attempts to resolve it.
  - This procedure continues recursively until the name is resolved.
Which Navigation Scheme?

- If a name service spans distinct administrative domains, than clients executing in one administrative domain may be prohibited from accessing name servers belonging to another such domain.

- Moreover, even name servers may be prohibited from discovering the disposition of naming data across name servers in another administrative domain.

- Then, both client-controlled and non-recursive server-controlled navigation are inappropriate.

- Recursive server-controlled navigation must be used: authorized name servers request name service data from designated name servers managed by different administrations, which return the attributes without revealing where the different parts of the naming database are stored.
Caching

• In DNS and other name services, client name resolution software and servers maintain a cache of the results of precious name resolutions.

• When a client requests a name lookup,
  
  ‣ the name resolution software consults its cache;
  
  ‣ if it holds a recent result from a precious lookup for the name, it returns it to the client;
  
  ‣ otherwise, it sets about finding it from a server.
  
  ‣ That server, in turn, may return data cached from other servers.
Case Study: Domain Name System (DNS)

Based on slides (by David Conrad) downloaded from the Web (http://www.itu.int/osg/spu/enum/workshopjan01/annex2-conrad.ppt)
The DNS is...

• The “Domain Name System”
  ‣ Created in 1983 by Paul Mockapetris (RFCs 1034 and 1035), modified, updated, and enhanced by a myriad of subsequent RFCs

• An Internet Name Service: what Internet users use to reference anything by name on the Internet.

• The mechanism by which Internet software translates names to addresses and vice versa.

• A globally distributed, loosely coherent, scalable, reliable, dynamic database.

• Comprised of three components: a “name space”, servers making that name space available, resolvers (clients) which query the servers about the name space.
DNS as a Lookup Mechanism

• **Users** generally prefer **names** to numbers.

• **Computers** prefer **numbers** to names.

• **DNS provides the mapping between the two.**

  ▸ I have “x”, give me “y”

• **DNS is ** NOT **a directory service.**

  ▸ No way to search the database.

  - No easy way to add this functionality.
DNS as a Database

- Keys to the database are “domain names” (www.foo.com, 18.in-addr.arpa, 6.4.e164.arpa, ...).

- Very large database (over 100,000,000 domain names stored in 2001)

- Each domain name contains one or more attributes, known as “resource records”.
  
  ‣ Each attribute individually retrievable

- Distributed:
  
  ‣ Data is maintained locally, but retrievable globally.
  
  ‣ No single computer has all DNS data.
  
  ‣ DNS lookups can be performed by any device.
  
  ‣ Remote DNS data is locally cachable to improve performance.
Loose Coherency

• The database is always internally consistent.

  ‣ Each version of a subset of the database (a zone) has a serial number.

    - The serial number is incremented on each database change.

• Changes to the master copy of the database are replicated according to timing set by the zone administrator.

• Cached data expires according to timeout set by zone administrator.
Reliability

• Data is replicated.
  ‣ Data from master is copied to multiple slaves

• Clients can query
  ‣ Master server
  ‣ Any of the copies at slave servers

• Clients will typically query local caches.

• DNS protocols can use either UDP or TCP
  • If UDP, DNS protocol handles retransmission, sequencing, etc.
The DNS Components: the Name Space

- The name space is the **structure of the DNS database**.
  - An inverted tree with the root node at the top.
- Each node has a **label**.
  - The root node has a null label, written as “".".
(DNS) Domain Name

- A domain name is the sequence of labels from a node to the root, separated by dots ("."s), read left to right.
- The name space has a maximum depth of 127 levels.
- Domain names are limited to 255 characters in length.
- A node’s domain name identifies its position in the name space.
Subdomains and Delegation

• One domain is a subdomain of another if its domain name ends in the other’s domain name.
  ‣ \textit{sales.nominum.com} is a subdomain of \textit{nominum.com} and \textit{com}.
  ‣ \textit{nominum.com} is a subdomain of \textit{com}.

• Administrators can create subdomains to group hosts.
  ‣ According to \textit{geography}, \textit{organizational affiliation} or any other criterion.
  ‣ An administrator of a domain can delegate responsibility for managing a subdomain to someone else.
    • \textit{But this isn’t required}.

• The parent domain retains links to the delegated subdomain.
  ‣ \textit{The parent domain “remembers” who it delegated the subdomain to}. 
Delegation Creates Zones

- Each time an administrator delegates a subdomain, a new unit of administration is created.
  - The subdomain and its parent domain can now be administered independently.
  - These units are called zones.
  - The boundary between zones is a point of delegation in the name space.

- Delegation is good: it is the key to scalability.
Dividing a Domain into Zones

nominum.com domain
nominum.com zone
ams.nominum.com zone

rwc.nominum.com zone

rwc
www
ftp
ams
molokai
skye
gouda
cheddar

acmebw
nominum
netsol

.com
.edu
.arpa
The DNS Components: Name Servers

- Name servers store information about the name space in units called “zones”.
  - The name servers that load a complete zone are said to “have authority for” or “be authoritative for” the zone.
- Usually, more than one name server are authoritative for the same zone.
- Also, a single name server may be authoritative for many zones.
- Two main types of servers:
  - Authoritative – maintains the data.
    - Master – where the data is edited.
    - Slave – where data is replicated to.
  - Caching – stores data obtained from an authoritative server.
- The most common name server implementation (BIND) combines these two into a single process.
Name Servers and Zones

128.8.10.5 serves data for both nominum.com and isc.org zones

202.12.28.129 serves data for nominum.com zone only

204.152.187.11 serves data for isc.org zone only
Name Server Architecture

• You can think of a name server as part:
  
  ‣ **database server**, answering queries about the parts of the name space it knows about (i.e., is authoritative for),
  
  ‣ **cache**, temporarily storing data it learns from other name servers, and
  
  ‣ **agent**, helping resolvers and other name servers find data that other name servers know about.
Interaction Scenario 1: Authoritative Data

Name Server Process

Authoritative Data
(primary master and slave zones)

Cache Data
(responses from other name servers)

Agent
(looks up queries on behalf of resolvers)

Resolver
Query
Response
Interaction Scenario 2: Using Other Name Servers

Name Server Process

Authoritative Data
(primary master and slave zones)

Cache Data
(responses from other name servers)

Agent
(looks up queries on behalf of resolvers)
Interaction Scenario 2: Cached Data

Name Server Process

- **Authoritative Data**
  (primary master and slave zones)

- **Cache Data**
  (responses from other name servers)

- **Agent**
  (looks up queries on behalf of resolvers)

Resolver

Response

Query
Name Resolution

- **Name resolution** is the process by which resolvers and name servers cooperate to find data in the name space.

- To find information anywhere in the name space, a name server only needs the names and IP addresses of the name servers for the root zone.

- A DNS query has three parameters:
  - A domain name (e.g., www.nominum.com),
  - A class (e.g., IN), and
  - A type (e.g., A)

- A name server receiving a query from a resolver looks for the answer in its authoritative data and its cache.
  - If the answer isn’t in the cache and the server isn’t authoritative for the answer, the answer must be looked up.
The Resolution Process (Step By Step)

- ping www.nominum.com

The workstation **annie** asks its configured name server, **dakota**, for **www.nominum.com**’s address.
The name server **dakota** asks a root name server, **m**, for **www.nominum.com**’s address.
The root server **m** refers **dakota** to the **com** name servers (this type of response is called a “referral”).
The Resolution Process (Step By Step)

The name server **dakota** asks a **com** name server, **f**, for **www.nominum.com**’s address.
The Resolution Process (Step By Step)

Here’s a list of the nominum.com name servers. Ask one of them.

The com name server f refers dakota to the nominum.com name servers.
The name server dakota asks an nominum.com name server, ns1.sanjose, for www.nominum.com’s address
The nominum.com name server $ns1.sanjose$ responds with www.nominum.com’s address.
The Resolution Process (Step By Step)

The name server dakota responds to annie with www.nominum.com’s address.
The Resolution Process - Caching

• After the previous query, the name server dakota now knows:

  ‣ The names and IP addresses of the com name servers
  ‣ The names and IP addresses of the nominum.com name servers
  ‣ The IP address of www.nominum.com

• Let’s look at the resolution process again...
The workstation *annie* asks its configured name server, *dakota*, for *ftp.nominum.com*’s address.

- ping *ftp.nominum.com*

The workstation *annie* asks its configured name server, *dakota*, for *ftp.nominum.com*’s address.
The Resolution Process (Step By Step) - Caching

**dakota** has cached an NS record indicating **ns1.sanjose** is a **nominum.com** name server, so it asks it for **ftp.nominum.com**'s address.
The Resolution Process (Step By Step) - Caching

The nominum.com name server ns1.sanjose responds with ftp.nominum.com’s address
The name server **dakota** responds to **annie** with **ftp.nominum.com**’s address.
End of Case Study: Domain Name System (DNS)
Directory Services

• “What is the name of the user with telephone number 020-5559980?”

• Sometimes users require wish to find a particular person or resource, but they don’t know its name, only some of its (other) attributes.

• “Where can I print a high-resolution color image?”

• Sometimes users require a service, but they are not concerned with that system entity supplies that service.

• Attributes may be used as values to be looked up.
Directory Services

- **Directory service**: a service that stores collections of bindings between names and attributes and that looks up entries that match attribute-based specifications.

- Sometimes called *yellow pages services* or *attribute-based name services*.

- A directory service returns the sets of attributes of any objects found to match some specified attributes.

  Example: the request “PhoneNumber = 020-5559980” might return \{“Name = John Smith”, “PhoneNumber = 020-5559980”, ...\}

- The client may specify that only a subset of the attributes is of interest.
Case Study: X.500
The X.500 Standard

• X.500 can be used in the same way as a conventional name service, but it is primarily used to satisfy **descriptive queries**.

• **Designed to discover the names and attributes** of other users or system resources.

• Specified [ITU/ISO 1997] as an **application level service** in the Open Systems Interconnection (OSI) set of standards, but
  
  ‣ its design does not depend to any significant extent on the other OSI standards
  
  ‣ it can be viewed as a design for a **general-purpose directory service**.
X.500 Directory Service

- Data stored in servers is organized in a **tree structure with named nodes** (Directory Information Tree (DIT)).

- Wide range of attributes are stored at each node in the tree.

- Entire directory structure (including data associated with the nodes) called **Directory Information Base (DIB)**.

- Access is not just by name but also by **searching for entries with any required combination of attributes**.

- Clients access the directory by establishing a connection to a server and issuing a request.

- If the data required are not in the segment of the DIB held by the contacted server, it will either invoke other servers to resolve the query or redirect the client to another server.
X.500 Architecture: DSAs and DUAs

- In the terminology of the X.500 standard, servers are Directory Service Agents (DSAs), and clients are Directory User Agents (DUAs).

Example:

DUAs interact with a single DSA

The DSA accesses other DSAs as necessary to satisfy requests.
X.500 DIT Example

X.500 Service (root)

... France (country) Great Britain (country) Greece (country) ...

... BT Plc (organization) University of Gormenghast (organization)

... Computing Service (organizationalUnit)

Department of Computer Science (organizationalUnit)

Engineering Department (organizationalUnit)

... Departmental Staff (organizationalUnit)

elv (applicationProcess)

Research Students (organizationalUnit)

... Alice Flintstone (person) ... Pat King (person) James Healey (person) Janet Papworth (person) ...
X.500 Directory Information Base

• Each entry in the DIB consists of a name and a set of attributes.

• The full name of an entry corresponds to a path through the DIT from the root of the tree to the entry.

• The name is determined by selecting one or more of its attributes as distinguished attributes, or entry’s Distinguished Name (DN).

• An attribute has a type and one or more values.

• The type of each attribute is denoted by a type name (for example, countryName, organizationName, commonName, telephoneNumber, ...).

• New attribute types can be defined if they are required.
**X.500 DIB Example**

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>info</strong></td>
<td>Alice Flintstone, Departmental Staff, Department of Computer Science, University of Gormenghast, GB</td>
</tr>
<tr>
<td><strong>commonName</strong></td>
<td>Alice.L.Flintstone</td>
</tr>
<tr>
<td></td>
<td>Alice.Flintstone</td>
</tr>
<tr>
<td></td>
<td>Alice Flintstone</td>
</tr>
<tr>
<td></td>
<td>A. Flintstone</td>
</tr>
<tr>
<td><strong>surname</strong></td>
<td>Flintstone</td>
</tr>
<tr>
<td><strong>telephoneNumber</strong></td>
<td>+44 986 33 4604</td>
</tr>
<tr>
<td><strong>uid</strong></td>
<td><a href="mailto:alice.Flintstone@dcg.gormenghast.ac.uk">alice.Flintstone@dcg.gormenghast.ac.uk</a></td>
</tr>
<tr>
<td><strong>mail</strong></td>
<td><a href="mailto:alf@dcg.gormenghast.ac.uk">alf@dcg.gormenghast.ac.uk</a></td>
</tr>
<tr>
<td><strong>roomNumber</strong></td>
<td>Z42</td>
</tr>
<tr>
<td><strong>userClass</strong></td>
<td>Research Fellow</td>
</tr>
</tbody>
</table>
Access Methods: Read

• A name (a domain name in X.500 terminology) for an entry is given, together with a list of attributes to be read (or an indication that all the attributes are required).

• The Directory Service Agent (DSA) locates the named entry by navigating in the Directory Information Tree (DIT).

• The DSA passes requests to other DSAs where it does not hold relevant parts of the tree.

• Finally, it retrieves the required attributes and returns them to the client.
Access Methods: Search

• Attributed-based access request.

• A base name and a filter expression are supplied as arguments.

• Base name: the node in the DIT from which the search is to commence.

• Filter expression: boolean expression that has to be evaluated for every node below the base node in the DIT.

• The filter specifies a search criterion: a logical combination of tests on the values of any of the attributes in an entry.

• The search command returns a list of names for all of the entries below the base node for which the filter evaluates to TRUE.
Access Methods Example

- **[SEARCH]** A *filter* might be constructed and applied to find the commonNames of members of staff who occupy room Z42 in the Department of Computer Science at the University of Gormenghast.

- **[READ]** A *read* request could then be used to obtain any or all the attributes of those DIB entries.
Administration and Updating of the DIB

• The DSA interface includes operation for adding, deleting and modifying entries.

• Access control is provided for both queries and updating operations.

• The DIB is partitioned, with the expectation that each organization will provide at least one server holding the details of the entries in that organization.

• Portions of the DIB may be replicated in several servers.

• As a standard, X.500 does not address implementation issues.

However, it is quite clear that any implementation involving multiple servers in a wide area network must rely on extensive use of replication and caching techniques to avoid too much redirection of queries.
End of Case Study: X.500