6. Naming and Directory Services
Outline

- Definitions and requirements
- Basic Terms & Name Structures
- Implementation Techniques
- System Examples
Definitions – Naming & Directory Services

- **Naming Service**
  - Mapping of logical names to physical addresses or object references
  - Logical names given by user (location independent)
  - Service returns the addresses (location dependent)

- **Directory Service**
  - Management of attributes of the named instances
  - Attribute based search for named instances
  - Similar to the “Yellow Pages” search model

- **Examples of name usage:**
  - Selection of a printer server via logical name
  - Mapping of e-mail-address to mail server

→ **Need:**
  - Mapping of logical names/attributes to addresses
  - Registration of logical names and addresses
  - Special System Support
  → Distributed Directory Service
Requirements

- Requirements to the name structure:
  - Unique names within the system
  - Multi-stage names ("rechnernetze@tu-dresden.de")
  - Attributed names ("host_x:CPU=Dual, PRT=lpr, LOC=rz")
  - Group names ("Group_1” ➔ “meier, müller, schmidt”)
  - Flexible binding with roles and groups
    - E.g. Role: Customer Services Representative. Name bound to currently available employee
  - Alias names

- Requirements to the Directory Service:
  - Scalability, reliability, fault tolerance:
    - Distributed, decentralized name management (e.g. one server per department of a company)
    - Distributed protocol for name interpretation
    - Replication of name tables (efficiency, fault tolerance)
  - Performance:
    - Result caching ➔ reduce amount of queries needed
Basic Terms & Name Structures

- **Address**: explicit, physical object notation (“#346”)

- **Name**: logical object notation, mostly location independent (printer)

- **Interpretation**: mapping of name to address

- **Context**:
  - Interpretation of a component of a multi-part name, e.g. user@hostX: user is interpreted in the context of hostX
  - Disambiguation from the same name in different namespaces possible

- **Namespace**: set of contexts, e.g. “<user>@<computer-node>”

- **Relative names**: interpretation dependent on specific context

- **Absolute names**: context independent
Hierarchical name space:

Example (Unix, NFS): 
```
"/Customer/PrivateCustomer/CustomerX"
```

Flat name space:

Example (local operating system): “Customer X”
Basic Terms & Name Structures

Context & Name Interpretation

• Example: interpretation of hierarchical name “Customer_X@host_X”

```
“Customer_X@host_X”
```

```
Context R (computer name)
```

```
Address of host_X
```

```
Context B
```

```
“Customer_X”
```

```
host_X
```

```
Context B (user names in the space of host_X)
```

```
User identification
```
Context & Name Interpretation

- **Approach to interpretation:**
  - Interpretation of each name component via related context
  - Result: mapping of components onto an address + a new context
  - Assignment of different contexts to different name servers
  - decentralized, distributed interpretation

- **Junction: Combination of Namespaces**
  - Combination of several namespaces into one
  - Management of individual namespace by dedicated directory service
  - Further directory service called with transition to a subordinate namespace
  - E.g. Interpretation of global file name
    - Firstly, internet domain name server finds appropriate fileserver
    - Then further name interpretation takes place within the fileserver
Hierarchical name space with contexts and related name servers, doubly linked:

- Name Server S1: Context “Company Names”
  - Name Server S2: Context “Online Trader”
  - Name Server S4: Context “Warehouse Management”
  
- Name Server S3: Context “Supplier”
  - Name Server S5: Context “Customer Management”
  - Name Server S6: Context “Ordering”

- Interpretation of name dependent on contacted name server
- e.g. name in context of warehouse management on S4
Name query examples:

- **Chaining**: E.g. $S_4 \rightarrow S_2 \rightarrow S_1 \rightarrow S_3 \rightarrow S_6$ then result sent back along chain
- **Referral**: Client receives address of next server in each query step; if client already knows servers it can improve response time with multicast

Query processing: query $S_4$ for “ordering.supplier.com” (S6)
Caching - goals and approaches:

- **Performance improvement via re-use of precedent query results**
  - complete names (specific, fast resolution)
  - partial name (more general, more requests)

- **Caching via clients of the Directory Service:**
  - parts of an interpreted name and address of the interpreting name server of subordinate level
  - example of a cache-record: (“OnlineTrader: CustomerMngt”, “S5”) → direct querying of the lowest subordinate server possible (here “S5”)

- **Caching via name server directly:**
  - records: name context and addresses of all servers on the same level
  - reduction of one level

- **Time stamps to recognize and discard obsolete cache entries**
Caching - example

Name Server S1: Context “Firm_names”

Name Server S2: Context “OnlineTrader”

Name Server S3: Context “Supplier”

Name Server S4: Context “WarehouseMngt”

Name Server S5: Context “CustomerMngt”

Name Server S6: Context “Ordering”

Server Cache: “Supplier”, “S3”

Server Cache: “OnlineTrader”, “S2”

Client Cache: “OnlineTrader: CustomerMngt”, “S5”
Context Replication

- **Goals:**
  - Fault tolerance, Locality of queries, Load Balancing

- **Approach:**
  - Implementation of a context via several name servers
  - Selection of an alternative server after timeout by a query

- **Consistency between replicas**
  - Assumption: Name modifications occur rarely → Toleration of temporary inconsistencies between replicas → simple update process
  - Modifications performed on Primary Server → then notification of replica
  - Convergence to consistence state within reasonable timeframe
    - No strict consistence → possibility of obsolete result provided by replica
    + Update process does not require concurrent availability of all replicas
Context Replication - Example

- Name Server S1: Context “Firm_names”
- Name Server S2: Contexts: “OnlineTrader”, “Supplier”
- Name Server S3: Contexts: “Supplier”, “OnlineTrader”
- Name Server S4: Context “WarehouseMngt”
- Name Server S5: Context “CustomerMngt”
- Name Server S6: Context “Ordering”

- S2 and S3 replicate each others’ contexts
- Reduce need for forwarding
  - E.g. If query for OnlineTrader sent to S3 - processed without further forwarding
  - I.e. No need to go S3->S1->S2 because OnlineTrader replicated on S3 and S2.
Relatively extensive administration tasks
- Installation and monitoring of server processes
- Definition and reconfiguration of name space structures
- Setting of access control mechanisms
- Replication control of name records; example for fine-grained replication:

Tools
- Simple command interface
- Vendor-specific tools
- Few standards

Examples
- create replica /Firm_names clearinghouse /Server1
- set directory /Firm_names convergence = high

<table>
<thead>
<tr>
<th>Server 1</th>
<th>Server 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Firm_names</td>
<td>/Firm_names</td>
</tr>
<tr>
<td>/...</td>
<td>/Firm_names/OnlineTrader</td>
</tr>
<tr>
<td>/Firm_names/OnlineTrader</td>
<td>/Firm_names/Supplier/...</td>
</tr>
<tr>
<td>/CustomerMgmt/...</td>
<td>/Firm_names/Supplier/...</td>
</tr>
</tbody>
</table>
Internet Domain Name System (DNS)

- **Naming service** of the Internet
  - Mapping of domain names to IP addresses
  - Queries for services such as SIP server or RTP gateway

- Hierarchical namespace: one or more contexts (domains) managed by replicated servers of a zone:
  - Domain name – path along composition of contexts
  - Top-level entry via one out of 13 root servers, all of them replicated worldwide
  - Name assignment by Internet Corporation for Assigned Names & Attributes (ICANN) or authorized companies
Example domain name: order.supplier.com.

Internet Domain Name System
Hierarchical Naming Structure

Cached Secure DNS (DNSsec) Zone
(TLD DNS Servers all know each other)

Special: Infrastr.
Top Level Domain

General Top-Level Domains (gTLD)
.onlinetrader

Country Top-Level Domains (ccTLD)
.supplier

.iTLD
.arpa

( iTLD)
.firm ...
.edu
.org
.com
.net
.de
.uk ...
.it

warehouse-
management
customer-
management

.order
customer-
management

Root "."
Resource Records: Storing information with names
- typically assignment host -> IP address

\(<\text{Domain-Name}, \text{Lifespan}, \text{Class}, \text{Type}, \text{Value}>\)

- Lifespan: validity of entry expires after lifespan

- Class: separation of records into different databases
  - e.g. “IN” for Internet Information

- Types: certain number defined for each class
  - E.g. “IN”-class:
    - A (IPv4 address for given domain name, 32 Bit)
    - AAAA (IPv6 address for given domain name, 128Bit)
    - NS (Domain name of name server for a particular domain)
    - CNAME (Mapping of alias names to domain names)
    - PTR (Alias for an IP address)
    - SRV (Host provides certain service (e.g. WWW, or SIP) for zone)
<table>
<thead>
<tr>
<th>No.</th>
<th>Domain Names</th>
<th>Lifespan</th>
<th>Class</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1T</td>
<td>IN</td>
<td>SOA</td>
<td>Zone data</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1T</td>
<td>IN</td>
<td>NS</td>
<td>ns1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1T</td>
<td>IN</td>
<td>NS</td>
<td>ns2</td>
</tr>
<tr>
<td>4</td>
<td>warehousemngt</td>
<td>1T</td>
<td>IN</td>
<td>NS</td>
<td>ns3.warehousemngt</td>
</tr>
<tr>
<td>5</td>
<td>ns3.warehousemngt</td>
<td>1T</td>
<td>IN</td>
<td>A</td>
<td>141.76.40.2</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1T</td>
<td>IN</td>
<td>MX</td>
<td>1, server17</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1T</td>
<td>IN</td>
<td>MX</td>
<td>2, server18</td>
</tr>
<tr>
<td>8</td>
<td>www</td>
<td>1T</td>
<td>IN</td>
<td>A</td>
<td>141.76.40.3</td>
</tr>
<tr>
<td>9</td>
<td>mail1</td>
<td>1T</td>
<td>IN</td>
<td>CNAME</td>
<td>server17</td>
</tr>
<tr>
<td>10</td>
<td>mail2</td>
<td>1T</td>
<td>IN</td>
<td>CNAME</td>
<td>server18</td>
</tr>
<tr>
<td>11</td>
<td>server17</td>
<td>1T</td>
<td>IN</td>
<td>A</td>
<td>141.76.40.4</td>
</tr>
<tr>
<td>12</td>
<td>server18</td>
<td>1T</td>
<td>IN</td>
<td>A</td>
<td>141.76.40.5</td>
</tr>
</tbody>
</table>
Domain name to be resolved: ubicomplab.cs.washington.edu

1: request

10: 128.208.5.125

configured name server 141.30.66.135

2: request

3: edu

4: request

5: washington.edu

6: request

7: cs.washington.edu

Recursive query

8: request

9: 128.208.5.125

Root name server a.root-servers.net

EDU name server g.edu-servers.net

uw name server marge.cac.washington.edu.

UWCS name server marge.cac.washington.edu.
X.500 Directory Service Standard

- Standard of International Telecommunications Union (ITU) for an **directory service**

- Many products based on X.500 for corporate directory management

- Hierarchical name structure (RFC 4512)
  - The information held in the Directory is collectively known as the **Directory Information Base (DIB)**.
  - The basic unit of information is called **Directory Entry**
  - The set of entries representing the DIB are organized hierarchically in a tree structure known as the **Directory Information Tree (DIT)**

- **Attribute based** description and attribute search for entries, defined by object classes with inheritance; attributes are typed name/value pairs
Directory Information Tree (DIT)

- **C=uk**
  - **O=onlinetrader**
    - **OU=warehousemngt**
      - **SN=Customer_X**
    - **OU=customermngt**
      - **CN=Firm_Y**
  - **O=supplier**
    - **OU=ordering**
    - **OU=customermngt**

- **C=de**
  - **L=Dresden**
    - **OU=Informatik**
    - **OU=inf**

- **O=TU-Dresden**
  - **L=Frankfurt**
  - **L=Dresden**

Legend:
- **Country (C)**
- **Organization (O)**
- **OrganizationalUnit (OU)**
- **Location (L)**
- **Alias**
- **organizationalPerson**
- **residentialPerson**
- **Device**
X.500 Directory Service Standard

Directory Information Tree (DIT)

Specification of attribute types and object classes in RFC 4519

Directory Entry: Customer_X

<table>
<thead>
<tr>
<th>object class</th>
<th>attribute type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>country</td>
<td>c</td>
<td>DE</td>
</tr>
<tr>
<td>organization</td>
<td>o</td>
<td>onlinetrader</td>
</tr>
<tr>
<td>organizationalUnit</td>
<td>ou</td>
<td>customermngt</td>
</tr>
<tr>
<td>residentialPerson</td>
<td>sn (person)</td>
<td>Customer_X</td>
</tr>
<tr>
<td>residentialPerson</td>
<td>street</td>
<td>Hauptstrasse 23</td>
</tr>
</tbody>
</table>

Legend:
- Country (C)
- Organization (O)
- OrganizationalUnit (OU)
- Location (L)
- Alias
- OrganizationalPerson
- ResidentialPerson
- Device
Object class specification from RFC 4519:

The 'residentialPerson' object class is the basis of an entry that includes a person's residence in the representation of the person. (Source: X.521 [X.521])

\[
(2.5.6.10 \text{NAME} \text{'residentialPerson'} \\
\text{SUP} \text{person} (\text{superclass is person with attribute surname and commonName}) \\
\text{STRUCTURAL} \\
\text{MUST} \text{I} \\
\text{MAY} (\text{businessCategory} \text{ x121Address} \text{ registeredAddress} \text{ destinationIndicator} \text{ preferredDeliveryMethod} \text{ telexNumber} \text{ teletexTerminalIdentifier} \text{ telephoneNumber} \text{ internationalISDNNumber} \text{ facsimileTelephoneNumber} \text{ preferredDeliveryMethod} \text{ street} \text{ postOfficeBox} \text{ postalCode} \text{ postalAddress} \text{ physicalDeliveryOfficeName} \text{ st} \text{ I}))
\]
Lightweight Directory Access Protocol (LDAP)
- Simplified standard protocol to access X.500 directories via TCP/IP
- Widespread vendor support, widely used in practice, optimized
- Typical operations for name search and name manipulation:
  - Bind, Search, Compare, Add, Delete, Modify
- Automatic forwarding of queries between Directory Servers with distributed information repositories
- Authentication, encryption, and integrity for Directory Access
Example: Standalone LDAP Server

**Binding:**

- Session Establishment with LDAP-Server → Client specifies host and TCP/IP-Port, where Server is listening
- Authentication: anonymous session or authentication with username/password

**Search Request**

- **Specify base:** part that is searched
- **Specify search filter:** 
  \((&(C=de)\text{ }\&(OU=\text{customermgmt}))\)

**Search Result:**

- SearchResultEntry 1: DN: \text{SN=Customer_X,OU=customermgmt,O=onlinetrader,C=de}
- SearchResultEntry 2: DN: \text{OU=customermgmt,L=Dresden,O=supplier,C=de}

**Unbinding:**

- Session with server is closed
Directory System Protocol (DSP) – query forwarding (chaining, referral, multicast)
Directory Information Shadowing Protocol (DISP) – replication management
Directory Operational Binding Management Protocol (DOP) – administrative communication
(Lightweight) Directory Access Protocol ((L)DAP) – client access
Universal directory service interface for Java applications, compatible with LDAP and many directory service implementations (via JNDI Service Provider Interface (SPI))

JNDI Context Interface: Basic operations for name contexts:

```java
public interface Context {
    public Object lookup(Name name) throws NamingException;
    public void bind(Name name, Object obj) throws NamingException;
    public void rebind(Name name, Object obj) throws NamingException;
    public void unbind(Name name) throws NamingException;
    public void rename(Name old, Name new) throws NamingException;
    public NamingEnumeration listBindings(Name name) throws NamingException;
    ...
    public Context createSubcontext(Name name) throws NamingException;
    public void destroySubcontext(Name name) throws NamingException;
    ...
}
```
Java Naming & Directory Interface

**JNDI Directory Interface:** Attribute management and search

```java
public interface DirContext extends Context {
    public Attributes getAttributes(Name name)
        throws NamingException;
    public Attributes getAttributes(Name name, String[] attrIds)
        throws NamingException;

    ... public void modifyAttributes(Name name, int modOp, Attributes attrs)
        throws NamingException;
    public void modifyAttributes(Name name, ModificationItem[] mods)
        throws NamingException;

    ...
}
```
Distributed Naming & Directory Services:

- Important for search and management of resources
- Mapping of names to addresses
- Hierarchically structured name spaces
- Multi-level mapping of names
- Distributed name servers – fault tolerant and scalable

Important Standards and Quasi-Standards:

- X.500: directory service with additional attribute management
- Internet Domain Name System: global naming service
- LDAP/JNDI: interfaces for portable implementations
References

- Albitz, P.: DNS and BIND. O’Reilly 2006
- Mockapetris, P.: Domain names - implementation and specification IETF RFC1035, 1987