7. Naming and Directory Services
Outline

- Definitions
- Requirements
- Basic Terms & Name Structures
- Implementation Techniques
- System Examples
- Summary
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
  - selection of a computer node with special CPU-capacity (attribute)

  ==> mapping of logical names or attributes to the addresses necessary
  ==> registration of logic name and address/object reference
  ==> special system support required
  => distributed Directory Services
Requirements to the name structure:

- Names have to be unique within the system
- Multi-stage names ("schill@inf.tu-dresden.de")
- Attributed names ("host_x:CPU=Pentium, PRT=lpr, LOC=rz")
- Group names ("Arbeitsgruppe_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**: name mapping to an address
- **Context**:
  - scope within which a component of some multi-part name can be interpreted
  - e.g. set of user names at host_X for "schill@host_X"
  - Allows disambiguation from a component with the same name in a different namespace.
- **Namespace**:
  - set of contexts
  - e.g. "<user>@<computer-node>"
- **Relative names**:
  - interpretation depends on specific context
  - e.g. interpretation of "schill" within context "host_X" compared to that within "host_Y"
- **Absolute names**: context independent
Hierarchical name space:

Example (Unix, NFS):
"Customer/PrivateCustomer/Customer X"

Routing-oriented name space:

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
```

Basic Terms & Name Structures

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 separate contexts to different name servers possible
  
  $$\Rightarrow$$ decentralized, distributed interpretation

- **Junctions: Combination of Namespaces**
  - Several namespaces can be combined into one via junctions
  - Each Individual namespace managed 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
Basic conceptions:

- hierarchical name spaces
- assignment of one or several contexts to each name server
- each name server knows the sub-ordinate and super-ordinate servers
- distributed interaction of the name servers for processing of a name interpretation
  - no expensive broadcast necessary
  - name servers are relatively autonomous
Example of a name space with related name servers:

- Name Server S1: Context “Company Names”
- Name Server S2: Context “Online Trader”
- Name Server S3: Context “Supplier”
- Name Server S4: Context “Warehouse Management”
- 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
Example of a name space with related name servers:

Query processing: query S4 for “ordering.supplier.com” (S6)
- Direct – Query sent directly to S6
- Indirect – Query sent first to different server
  - Chaining: E.g. S4 \rightarrow S2 \rightarrow S1 \rightarrow S3 \rightarrow S6 then result sent back along chain
  - Referral: Client given address of next server to query
    - If client already knows servers can improve response time with multicast
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”) then 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 stamping to recognize and discard obsolete cache entries
Implementation Techniques: Optimization

Caching - example

Server Cache: "Supplier", "S3"

Name Server S1: Context "Firm_names"

Name Server S2: Context "OnlineTrader"

Server Cache: "OnlineTrader", "S2"

Name Server S3: Context "Supplier"

Name Server S4: Context "WarehouseMngt"

Name Server S5: Context "CustomerMngt"

Clients Cache: "OnlineTrader: CustomerMngt", "S5"

Name Server S6: Context "Ordering"
Context Replication

- **Goals:**
  - Fault tolerance
  - Locality of queries -> shorten interpretation route
  - Load Balancing – especially for higher levels which receive many queries

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

- **Consistency between replicas**
  - Assumed: Name modifications occur only rarely
  - Temporary inconsistencies between replicas tolerated
    - Allows simple update process
  - Modifications performed first on a pre-determined Primary Server
  - Replicas are then notified and updated periodically
  - Convergence to consistence state ensured within reasonable timeframe
  - No strict consistence -> possibility of obsolete result provided by replica
  - Update process does not require concurrent availability of all replicas
Implementation Techniques: Optimization

Context Replication - Example

- S2 and S3 cache 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.
Context Replication - Fine Grained Replication

- Selection of contexts to be replicated
- Many can be replicated on one server
- Or fewer may be replicated due to:
  - Expense
  - Performance
  - Effort for maintaining consistency
Relatively extensive administration tasks

- Installation and monitoring of server processes
- Definition of name space structure
- Setting of access control mechanisms
- Replication control of name records
- Re-configuration of name spaces

Tools

- Simple command interface
- Vendor-specific tools (also Remote Management)
- Lack of orientation towards standards

Examples

- `create replica /hosts clearinghouse /t500m0_ch`
- `set directory /hosts Convergence = high`
Existing systems

- Internet Domain Name System
- X.500 Standard – Products:
  - Novell eDirectory
  - Microsoft Active Directory
- Java Naming & Directory Interface
Internet Domain Name System (DNS)

- Naming service of the internet
- Maps domain names to IP Addresses
- No implementation of attribute based queries
- Hierarchical namespace – organised into contexts and zones
- Contexts called Domains
- Domain name – path along/composition of contexts
  - e.g. ordering.supplier.com.
  - Root - highest order context – denoted by “.” but omitted.
- Top level domains managed by Internet Corporation for Assigned Names & Attributes (ICANN)

- Numerous name servers distributed worldwide
- Every server manages a subset of names and information
- Resource Records – collection of attributes associated with names
Internet Domain Name System

Hierarchical Naming Structure

Example domain name: ordering.supplier.com.

Root „.“

General Top-Level Domains:
- tv
- firm
- ... eduard
- org
- com
- net

Country Specific Top-Level Domains:
- de
- uk
- ...
- it

- OnlineTrader
  - Warehouse Management
  - Customer Management

- Supplier
  - Ordering
  - Customer Management
Zones and Relationships

- Zones enclose names of a particular hierarchy level of a domain. E.g. names included in “onlinetrader.com” zone:
  - mail.onlinetrader.com
  - www.onlinetrader.com

- Name servers (NS) manage data for one or more zones
- Every NS knows further NSs for:
  - Updating and caching
  - Resolution of names in other zones

- Authoritative Name Servers (ANS) exist in every zone
  - Hold the up-to-date data for the zone
  - Minimum of 2 for every zone – i.e. data always replicated
- Every NS must know at least two ANSs for its zone as well as ANSs for subordinate zones
- Every NS knows at least one superordinate NS to which it can forward queries
Internet Domain Name System

Resource Records

- for storing attributes with names

- `<Domain-Name, Lifespan, Class, Type, Value>`
- Lifespan – control of adjustment between replications and cache entries
- Class – separation of records into different databases
  - e.g. “IN” for Internet Information
- Certain number of types defined for each class
- Types defined for “IN” class:
  - A – Holds IP address for given domain name
  - NS – Domain name of name server for a particular domain
  - MX – Domain name and priority for email servers
  - SOA – Start of data for particular zone and zone information
  - CNAME – Mapping of alias names to domain names
  - PTR – Alias for an IP address
  - HINFO – Host information such as machine architecture and OS
### Resource Records - Example

<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>141.76.40.2</td>
</tr>
<tr>
<td>9</td>
<td>mail1</td>
<td>1T</td>
<td>IN</td>
<td>CNAME</td>
<td>1 server17</td>
</tr>
<tr>
<td>10</td>
<td>mail2</td>
<td>1T</td>
<td>IN</td>
<td>CNAME</td>
<td>2 server18</td>
</tr>
<tr>
<td>11</td>
<td>server17</td>
<td>1T</td>
<td>IN</td>
<td>A</td>
<td>141.76.40.3</td>
</tr>
<tr>
<td>12</td>
<td>server18</td>
<td>1T</td>
<td>IN</td>
<td>A</td>
<td>141.76.40.4</td>
</tr>
</tbody>
</table>
X.500 Directory Service Standard

- Standard of International Telecommunications Union (ITU)
- Various products based on standard
- Hierarchical name structure
- Attribute based description and searching for entries

Directory Entries described object orientated
- Entries are instances of one or more object classes
- Classes determine permissible attributes
- Standard and freely definable attributes
- Extensive use of standard attributes encouraged to maximise usability of the directory
- Inheritance and derivation of classes
Naming

- **Distinguished Name (DN) (comparable with DNS domain name)**
  - Composite name for an individual entry
  - Composed from *relative distinguished names*
  - Put together in order starting with the directory tree root
  - Unique within whole directory tree (absolute name)

  - E.g. DN: 
    `/C=de/O=onlinetrader/OU=customermngt/CN=Customer_X+Street=Hauptstrasse 23`

- **Relative Distinguished Name (RDN)**
  - Individual component of a DN
  - Relative to super ordinate entry
  - Formed from attributes of the entry
  - Must be unique within level of directory tree

  - E.g. RDN: `Customer_X+Street=Hauptstrasse 23`
X.500 Directory Service Standard

Naming (Directory Information Tree)

```
DIT
  \---- C=uk
  \---- C=de
  \---- L=Dresden
      \---- O=TU-Dresden
          \---- OU=inf

  \---- O=supplier

  \---- O=onlinetrader
      \---- OU=warehousemngt
          \---- SN=Customer_X
              \---- CN=Firm_Y

  \---- OU=customermngt

  \---- L=Frankfurt
      \---- OU=ordering

  \---- L=Dresden
      \---- OU=customermngt
```
## X.500 Directory Service Standard

### Decentralized
- Replication across several directory servers
- Name interpretation: chaining, referral or multicast protocols
- **Communication:**
  - Directory System Protocol (DSP)
    - interaction between directory servers
  - Directory Access Protocol (DAP)
    - client access to directory servers
  - Lightweight Directory Access Protocol (LDAP)
    - Simplified version of DAP
    - Widely used in practice
    - Access based on TCP/IP
LDAP

- de Facto-Standard for access to Directory Services, widespread vendor support
- Simplified variant of X.500 Directory Access Protocol (DAP), implementation via TCP/IP
- Front-End for heterogeneous Directory Servers (for instance, X.500, NIS, MS Active Directory, CDS, Novell eDirectory etc.)
- Typical operations for name search and name manipulation:
  - Bind, Search, Compare, Add, Delete, Modify
- Optimization of access functions
- Automatic forwarding of queries between Directory Servers with distributed information repositories ("referral" without involving of client application)
- Improvements regarding authentication, encryption, and integrity for Directory Accesses
Java Naming & Directory Interface

- Directory service interface for Java applications

- Used in connection with RMI and search for application servers in the frame of the binding process

- Designed independent of particular directory service implementations

- Abstract interfaces for access to different directory services
  - JNDI API – access to directory information for applications
    - Functions for event handling
    - LDAP v3 extensions
  - JNDI SPI (Service Provider Interface) – integration of diverse directory service implementations

- Abstract interfaces can be integrated into composite namespaces
No absolute names – always relate to particular context
Applications acquire "InitialContext" first

Example Interfaces:
- **Name** – operations for manipulation of names
- **javax.naming.directory** – access to directories and their attributes
- **Attributes** – operations for attributes belonging to a directory
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;
    ...
}
```
JNDI Directory Interface

- Examining and updating attributes and for searching the directory

```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;
    ...
}
```
Summary

Distributed Naming & Directory Services:
- Important for search and management of resources
- Mapping of names on addresses
- Hierarchically structured name spaces
- Multi-level mapping of names
- Distributed name servers – fault tolerant and scalable
- DNS – Capable, worldwide available example of naming service
- X.500 directory services offer additional attribute management
- Standard interfaces (LDAP & JNDI) enable portable implementations

Important Standards and Quasi-Standards:
- ISO/OSI 9594 / CCITT X.500
- OSF Distributed Computing Environment
- Internet Domain Name System
- CORBA Naming Service
- Novell eDirectory
- Microsoft Active Directory
References

- Albitz, P.: DNS and BIND. O’Reilly 2001
- Sun Microsystems: Java Naming and Directory Interface (JNDI), http://java.sun.com/products/jndi/