```
Dish-> search -object "c=FI@o=Tampere University of Technology@ou=Software Systems
Laboratory" -filter surname=Virtanen -type telephoneNumber
```

Other useful extensions could include shadowing of cached entries, maintaining a position in the DIT, and distributing attributes of entries among several DSAs. The latter capability might be used, for example, to store telephone numbers in separate DSAs operated by telephone companies.

References

[Hui88] Huitema, C., The X.500 Directory Services. Computer Networks and ISDN Systems 16, 1988/1989, 161-166.

[ISO88a] ISO, The Directory, Part 1: Overview of Concepts, Models, and Services. ISO DIS 9594-1, 1988.

[ISO88b] ISO, The Directory, Part 2: Information Framework. ISO DIS 9594-2, 1988.

[ISO88c] ISO, The Directory, Part 3: Access and System Services Definition. ISO DIS 9594-3, 1988.

[ISO88d] ISO, The Directory, Part 4: Procedures for Distributed Operation. ISO DIS 9594-4, 1988.

[ISO88e] ISO, The Directory, Part 5: Access and System Protocols Specification. ISO DIS 9594-5, 1988.

[ISO88f] ISO, The Directory, Part 6: Selected Attribute Types. ISO DIS 9594-6, 1988.

[ISO88g] ISO, The Directory, Part 7: Selected Object Classes. ISO DIS 9594-7, 1988.

[ISO88h] ISO, The Directory, Part 1: Authentication Framework. ISO DIS 9594-8, 1988.

[Kil88] Kille, S.E., The QUIPU directory service. Proc. IFIP WG 6.5 Conference on Message Handling Systems and Distributed Applications, North Holland Publishing, October 1988.

[Kil89] Kille, S.E. et al., Proposal for Co-ordination of a RARE OSI Directory Project. Department of Computer Science, University College London, 1989.

[Lun88] Lundberg, Johan, Introduction till Directory Systems X.500. TeleDelta Report M 88 029, 1988.

[Ros89] Rose, M.T., The ISO Development Environment. User's Manual (version 5.0), March 1989.

5. Future Work

The current directory standard leaves many areas beyond its scope as a local matter or topics for future extensions. Among the most important such issues are access and alias control, schema and DIT management, as well as use of descriptive names.

The standard covers extensively user identification via various authentication mechanisms but doesn't define how to control users' access to data. Such facility is, however, essential for any real directory implementation.

The QUIPU directory handles access control by associating an Access Control List (ACL) attribute for each directory entry. The ACL specifies what rights (e.g. none, detect, compare, read, and write) each DU (e.g. owner of the entry, member of a group, and others) has over the attributes of the entry. For example, an entry containing a userPassword attribute should at the very least have the following ACL:

```
acl= others # compare # attributes # userPassword & self # write # attributes # userPass-
word
```

It gives the owner of the entry the right to modify the userPassword attribute value and others only the right to compare it.

The current standard allows creating alias entries that should refer to object entries. The Directory doesn't, however, guarantee the integrity of alias entries, for example, in case the aliased object entry is removed. As the result, the alias entry may point to a nonexisting or wrong object entry.

Any DSA owner may create new object classes and attribute types specific to his/her directory applications and register them with a naming authority. Currently there is, however, no automatic way to distribute knowledge about such local additions among DSAs. So remote DUAs may not be aware of the structure of the locally defined entries or the syntax of the locally defined attribute values.

As explained in Section 2, the Directory is a distributed data base where each DSA represents some portion of the DIB. If a DSA can't process a query by itself, it must somehow know to which DSA it should chain or refer the request. The standard doesn't specify how the knowledge about these external DSAs is stored in a DSA or distributed among participating DSAs.

Many directory operations have DNs as arguments. Since the RDNs forming a DN have to be unique at each branch of the DIT, they tend to be long and cumbersome to use. The RDN requirement should be relaxed in most operations so that any of the possible multiple values of the RDN attributes could be used in the request. For example, to search the telephone number of every Virtanen at the Software Systems Laboratory of the Tampere University of Technology, one should be able to type

```
Dish-> search -object c=FI@o=TUT@ou=SSL -filter surname=Virtanen -type telephoneNum-
ber
```

instead of the full form

Option "-subtree" indicates that all subtrees of the DIT below the object are searched. Filter predicate "~=" means approximate match and "*" is a wild card.

Abandon. Applies to an outstanding interrogation and informs the Directory that the request can be cancelled. The directory may then cease processing the request and discard any results so far achieved. Abandon has not been implemented in the current version of QUIPU DAP.

The remaining four directory operations – Add Entry, Remove Entry, Modify Entry, and Modify Relative Distinguished Name – are available for adding and removing entries, modifying the attributes of an entry, and changing the distinguished name of an entry:

Add Entry. Causes a new leaf entry to be added to the DIT.

Example: Adding an entry for a person.

```
Dish-> add "c=FI@o=Tampere University of Technology@ou=Software Systems Laboratory@cn=Matti Virtanen" -objectclass organizationalperson
```

At this point the DUA invokes an editor on a draft entry of the specified object class. After the user has filled the entry and exited the editor, the entry it added to the DIT.

Remove Entry. Allows removing a leaf entry from the DIT.

Example: Removing a leaf entry.

```
Dish-> delete "c=FI@o=Tampere University of Technology@ou=Software Systems Laboratory@cn=Matti Virtanen"
```

Modify Entry. Modifies existing entries in the DIT. The sequence of changes may include addition, removal, and replacement of attributes or attribute values of a particular entry. Modify Entry is atomic in that if any of the changes fails then the whole operation will fail.

Example: Modifying an entry.

```
Dish-> modify "c=FI@o=Tampere University of Technology@ou=Software Systems Laboratory@cn=Maija Meikalainen"
```

The DUA again invokes an editor on the specified entry. The actual modification takes place after the user has made his/her changes and exited the editor.

Modify Relative Distinguished Name. Modifies the RDN of a leaf entry.

Example: Modifying a RDN.

```
Dish-> modifyrdn "c=FI@o=Tampere University of Technology@ou=Software Systems Laboratory@cn=Maija Meikalainen" -name "cn=Maija-Liisa Meikalainen"
```

Note that in current version of the Directory service it is only possible to add or remove object entries that remain as leaves, eg. entries for persons. So it is not possible to construct the DIT from scratch by using the Add Entry operation or totally remove it by using the Remove Entry operation.

Compare. Allows comparing whether a supplied value matches an attribute value of a particular named DIB entry. For example, a password in the Directory might be inaccessible for read but accessible for compare.

Example: Comparing a person's directory password.

```
Dish-> compare c=FI@o=Tampere University of Technology@ou=Software Systems Labo-
ratory@cn=Juha Heinanen -attribute userPassword=foo -print

FALSE
```

List. Causes the Directory to return the list of immediate subordinates of a particular named entry in the DIT.

Example: Listing at most five subordinates of Norway.

```
Dish-> list c=NO -sizelimit 5

localityName=More og Romsdal fylke
localityName=Sor-Troendelag fylke
localityName=Nord-Trondelag fylke
localityName=Nordland fylke
localityName=Finnmark fylke
```

Option "sizelimit" is an example of a *service control*. Service controls can be included in most directory operations. In addition to "sizelimit", there exist service controls for time limit, priority, chaining, caching, alias dereferencing as well as scope for the operation and the result.

Search. Causes the Directory to return information from all the entries within a given portion of the DIT that satisfy some *filter*. Filter is a logical expression of one or more conditions. As with read, the information returned from each entry consists of some or all attributes of that entry.

Example: Searching entries and attributes from the Directory.

```
Dish-> search -object "c=NO@o=University of Oslo" -subtree -filter "surname~=pedersen
& cn=g*"

ou=Institutt for Informatikk@cn=Geir Pedersen
ou=Institutt for Informatikk@cn=Geir Kenneth Pedersen
ou=Institutt for Informatikk@cn=Gisli O Petursson
ou=Institutt for Informatikk@cn=Geir Foss-Pedersen

Dish-> search -object "c=NO@o=University of Oslo@ou=Institutt for Informatikk" -filter
username=geir* -type rfc822mailbox

cn=Geir Pedersen
rfc822Mailbox       - geirp@ifi.uio.no
cn=Geir Hardt
rfc822Mailbox       - geirh@ifi.uio.no
cn=Geir Ove Bjoerkedal
rfc822Mailbox       - geirove@ifi.uio.no
cn=Geir Ivar Thorud
rfc822Mailbox       - geirt@ifi.uio.no
```

plied parameters, violations of security policy, schema rules, etc. A request may also fail because of a referral to another DSA.

Before a DUA can start issuing requests to a DSA, the DUA must set up an association with a Directory access point using a *Bind* operation. Bind has arguments for user authentication and for selecting a common version of the Directory implementation.

In its simplest form the authentication parameters consist of the DN of the user and (optionally) a password that is matched against the userPassword attribute in the user's directory entry. The Directory will then either confirm or deny its service without requiring further authentication for subsequent requests.

Example: Binding a user to a DSA called Jaguar.

```
Dish-> bind -user c=FI@o=Tampere University of Technology@ou=Software Systems Labo-
ratory@cn=Juha Heinanen -password secret -call Jaguar
```

The version information is supplied automatically by the Dish program which also knows the presentation address of the Jaguar DSA.

Stronger authentication schemes have also been defined (but not usually implemented yet) that are based upon public key cryptography. They allow each request to contain a digital signature of the requestor together with information to assist the correct party to verify it. See [ISO88h] for a detailed description of the Directory authentication framework.

When the DUA has done with its requests, it closes the association with the DSA by issuing an *Unbind* operation. Unbind has no parameters.

After an association to a Directory access point is established, the Directory may be accessed by nine different operations. Five of them – read, compare, list, search, and abandon – deal with directory interrogation:

Read. Allows reading information from a particular named entry in the DIB. Either all or a selected set of attributes of that entry is returned.

Example: Reading entries and attributes from the Directory.

```
Dish-> showentry c=FI@o=Tampere University of Technology

telephoneNumber    - +358-31-162111
postalCode         - SF-33101
organizationName   - TUT & TTKK & Tampereen teknillinen korkeakoulu & Tampere Univer-
sity of Technology
streetAddress      - PO Box 527
localityName       - Tampere

Dish-> showentry c=FI@o=Tampere University of Technology@ou=Software Systems Lab-
oratory@cn=Juha Heinanen -type rfc822Mailbox telephoneNumber

rfc822Mailbox      - jh@tut.fi
telephoneNumber    - +358-31-162578
```

mandatory and optional attributes in an entry of that class (e.g. a "telephoneNumber" may appear in a person's entry but a "serialNumber" may not).

- *Attribute Type Definitions* that (1) identify the object identifier and (2) syntax of an attribute as well as (3) tell whether the attribute can have multiple values. For example, an "organizationName" attribute type is defined to be multivalued and of syntax "caseIgnoreStringSyntax".
- *Attribute Syntax Definitions* that (1) may assign an object identifier to an attribute syntax, (2) define for each attribute syntax the underlying ASN.1 data type (e.g. "caseIgnoreStringSyntax" is either a "T61String" or "PrintableString"), and (3) specify matching rules for comparing a given value with a value in the DIB (e.g. "caseIgnoreString" is matched for "equality" and "substrings").

The relationship between the schema definitions and the DIT elements is summarized in Figure 8.

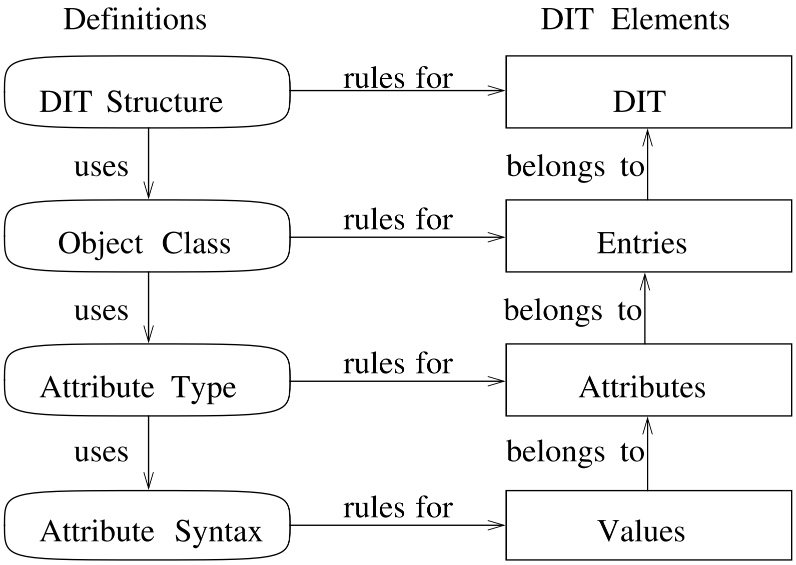


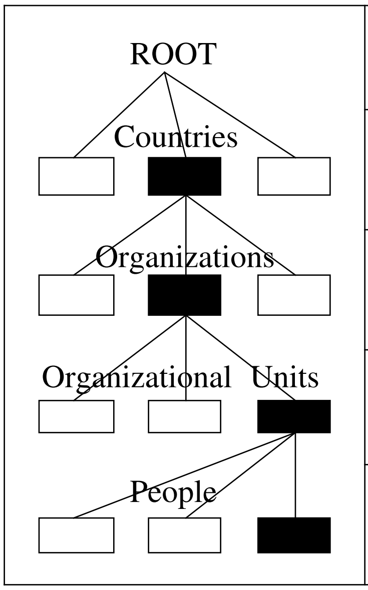
Figure 8. Overview of the Directory Schema

The Directory Schema is distributed like the DIB itself. It is the task of each Administrative Authority to establish the schema for those portions of the DIB that it administers. Current version of the Directory standard doesn't support any automatic means to distribute schema information across different DSAs.

4. Directory Services

Referring back to Section 2, the Directory can be viewed as an abstract object that provides services to its users in response to requests from DUAs. The requests allow interrogation and modifying the Directory.

The Directory gives a response to each request by reporting an outcome. Besides normal outcomes that are specific to each request, there are abnormal outcomes that are common to several requests. Abnormal outcomes may be errors due to problems with user sup-



RDN	Distinguished Name
{}	{}
C=FI	C=FI
O=FUNET	{C=FI, O=FUNET}
OU=TUT	{C=FI, O=FUNET, OU=TUT}
CN=Kaittola	{C=FI, O=FUNET, OU=TUT, CN=Kaittola}

Figure 7. A Naming Example

In addition to its object entry, an object may have one or more so called *alias entries*. Alias entries of an object point to its object entry, and provide a means for referring the object using alternative names. For example, an organization object with RDN "Tampere University of Technology" might have an alias entry with RDN "TUT":

```
commonName= TUT
aliasedObjectName= c=FI@o=Tampere University of Technology
```

The DIB is kept well-formed by enforcing a set of rules called the Directory Schema. The Directory Schema resembles database schemas and its purpose is to capture the time independent properties of the DIB. More precisely, the Directory Schema comprises a set of

- *DIT Structure Rules* that define (1) the permitted hierarchical relationship between entries (e.g. an organization may be subordinate to a country but not vise versa) and (2) their permitted RDNs. If a rule for an entry allows immediate subordinates of a particular class, it also allows subordinates belonging to any subclass of that class.

  In QUIPU the DIT structure is defined using a multivalued attribute called "treeStructure". The "treeStructure" attribute of an "organization" entry might look like

  ```
  treeStructure = organizationalUnit & applicationProcess & dSA & applicationEntity
  ```

  telling that entries of the listed object classes are permitted as immediate subordinates of an "organization" entry.

- *Object Class Definitions* that (1) may assign an *object identifier* for an object class, (2) define which classes are superclasses of the object class (e.g. an "internetPerson" might be a subclass of "person"), and (3) specify which are the

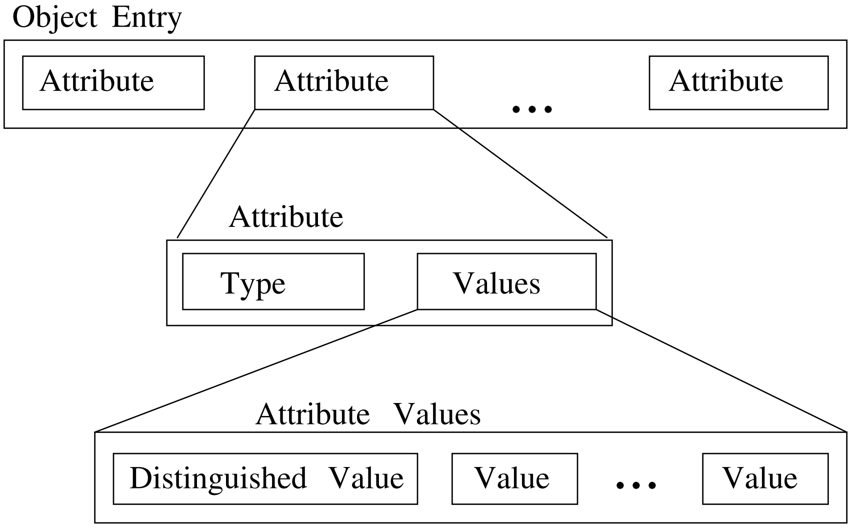


Figure 6. Structure of an Object Entry

For example, a person entry of object class "internetPerson" might have an entry consisting of the following attributes, only the first two being obligatory:

commonName= Juha Heinanen
surname= Heinanen
roomNumber= SL408
telephoneNumber= +358 31 162578 & +358 31 162111
rfc822Mailbox= jh@tut.fi
userid= jh
userPassword= secret

In order to manage a possibly large DIB, the object entries are arranged hierarchically as vertices of a tree, called the Directory Information Tree (DIT). Entries close to the root of the tree usually represent countries or organizations and the leaf entries persons or application processes.

Every entry in the DIT has a Distinguished Name (DN) which identifies it uniquely and unambiguously. The DN of an entry is a sequence of Relative Distinguished Names (RDNs) of entries on the path in the DIT from the root to the named entry. The RDN of an entry in turn consists of a set of designated values, so called *distinguished values*, of some of its attributes. Figure 7 shows an example of naming entries in the DIT.

In order for a DN to uniquely identify an object, the RDNs of all entries below each immediate superior entry have to be distinct. Usually a single distinguished value (eg. the Common Name of a person) suffices to make the RDNs distinct. If this is not the case, additional distinguished values (eg. Organizational Unit Name of a person) may be employed.

It is the responsibility of the *naming authority* of each non-leaf entry to ensure the uniqueness of RDNs by appropriately assigning distinguished attribute values. The relationship of the naming authorities is reflected by the hierarchical structure of the DIT. The responsibility is passed down the tree from superior to subordinate authorities, eg. from the National Naming Authority to Organizational Naming Authorities.

inform its client about the result or automatically redirect the operation on to another DSA.

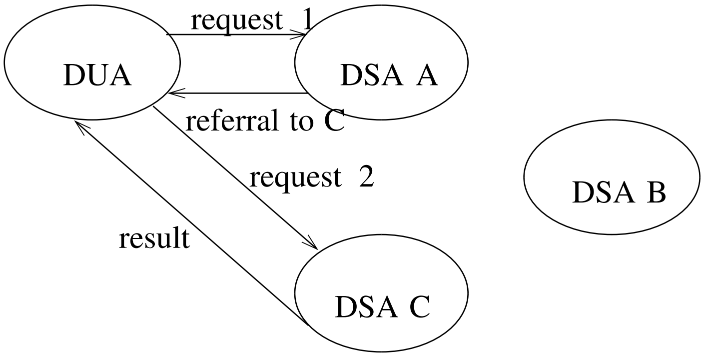


Figure 4. Referral Operation

- *Multicasting*. The DSA sends the same request in parallel or sequentially to multiple other DSAs in order to resolve it (Figure 5). The results are then combined and returned to the requestor. Multicasting is normally used only when the DSA for some reason doesn't have a complete knowledge of which DSAs hold which parts of the DIB.

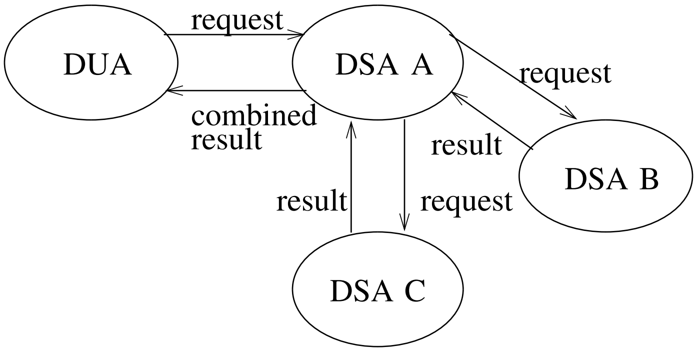


Figure 5. Multicasting Operation

The knowledge about other DSAs is normally provided by the strict hierarchical structure of the DIB as discussed in the next section. This hierarchy assigns each DSA (except the ones just below the top) one superior DSA and zero or more subordinate DSAs. In order to improve response time and to solve the missing top problem, a DSA can also maintain direct cross references to other DSAs.

3. Structure of Directory Information

As explained above, the DIB holds information of real world objects. Each such object is represented in the DIB as an *object entry*. An object entry is structured into a number of *attributes*, each with a type and one or more values (Figure 6). The types of attributes that must and may be present in an entry are specified by the *class* of the object.

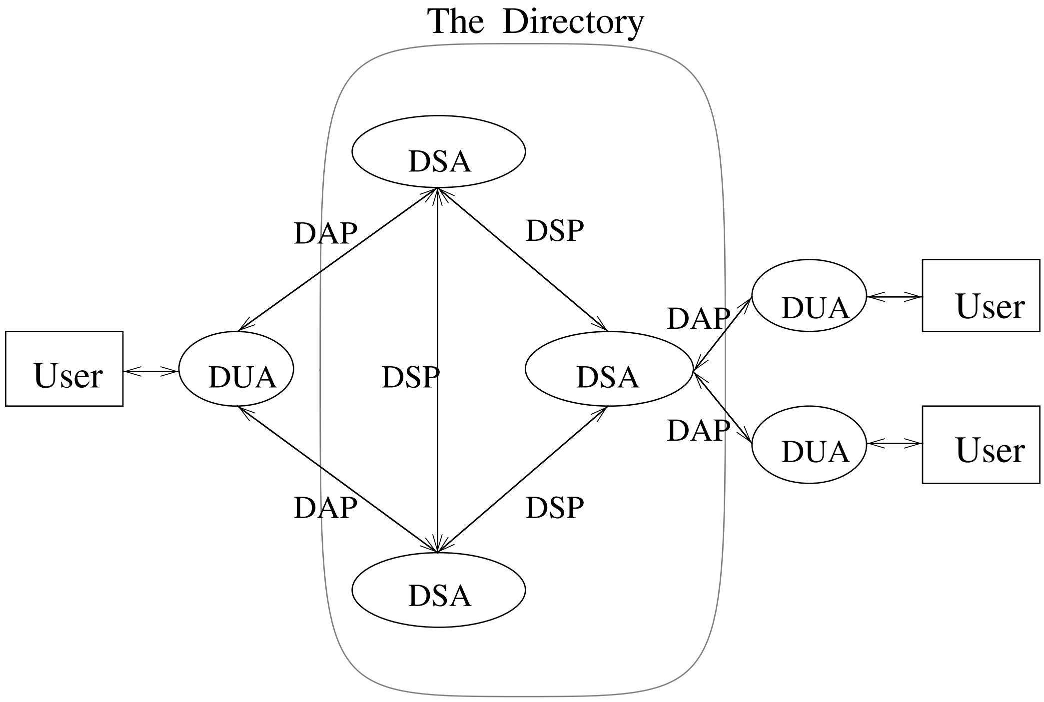


Figure 2. The Directory Provided by Multiple DSAs

A request from a DUA can involve several DSAs if the request references a part of the DIB not contained in the DSA that originally received the request. In such a case there exists three basic strategies on how the request will be processed:

- *Chaining*. The DSA passes the request to another DSA, which again can forward it to another DSA, etc. The responses return along the reverse path and finally the initial DSA returns the answer to the DUA (Figure 3). The DUA can ask the DSA not to do chaining and/or restrict the scope or time taken for the operation.

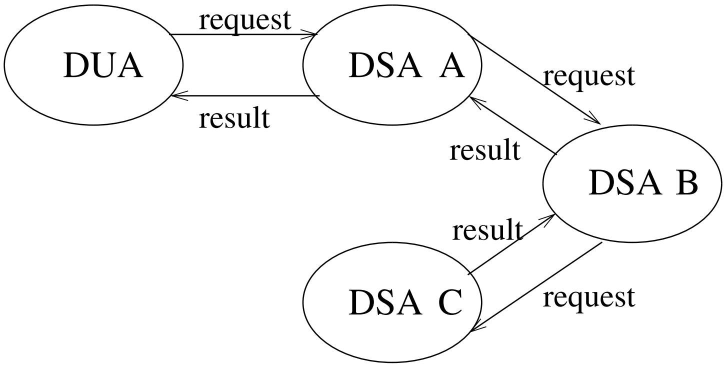


Figure 3. Chaining Operation

- *Referral*. The DSA determines which DSAs should be more able to process the request and returns the names and addresses of those DSAs back to the DUA or, in case of chaining, back to the DSA (Figure 4). The DUA or DSA may either

DU is given remote access using OSI provided means. The database is specialized and it doesn't have to be able to cope with all the problems usually associated with database distribution. For example,

- the database is structured in a hierarchical manner,
- it assumed that the frequency of queries is considerably higher than that of updates, and
- there is no need for instantaneous global commitment for updates.

The Directory can also be viewed as an abstract object, which provides a number of services for the DU. The DU doesn't access the Directory directly but by the use of a Directory User Agent (DUA) (Figure 1). The Directory provides one or more *access points* at which such accesses can take place. The DUA itself is a normal application process which may support different user interface styles. The examples in this paper use a line oriented DUA called Dish (DIrectory SHell), which is part of the QUIPU distribution.

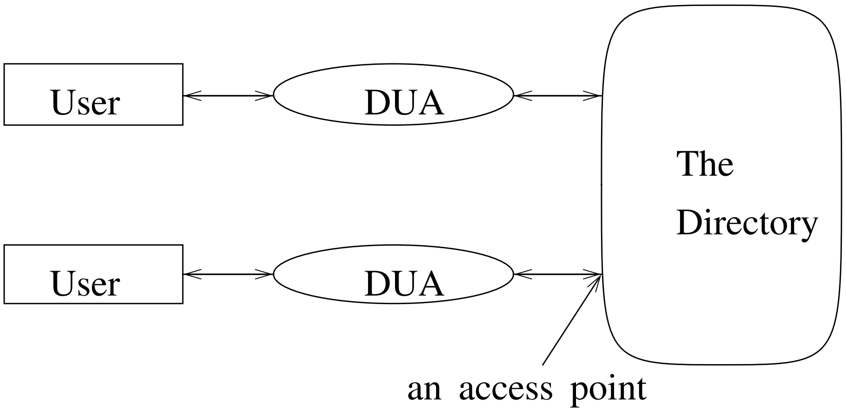


Figure 1. Access to the Directory

A DUA's access to the Directory is implemented over an OSI Association through the Directory Access Protocol (DAP). The DAP in turn uses the Remote Operations Service (ROS), which allows specification of the services cleanly as a set of remote procedure calls. The services allow retrieving and modifying of Directory information and will be described in more detail in Section 4.

Internally the Directory is realized as a collection cooperating Directory Service Agents (DSAs) which provide the Directory Service in a distributed manner (Figure 2). Each DSA is an application process that represents some portion of the DIB. Its role is to provide access to the DIB for DUAs and other DSAs. In doing so, a DSA may either use the information stored in its local database or interact with other DSAs to carry out the request. Alternatively, a DSA may redirect the requestor to another DSA.

As mentioned above, DUAs connect to the Directory through the Directory Access Protocol (DAP). The interactions between a pair of DSAs are supported by a slightly different application layer protocol called the Directory Service Protocol (DSP). The detailed specifications of these protocols can be found in [ISO88e].

Introduction to the OSI Directory

Juha Heinänen

Software Systems Laboratory
Tampere University of Technology
P.O. Box 527, SF-33101 Tampere, Finland
jh@tut.fi

Abstract

The OSI Directory will have an important role in supporting various OSI applications such as electronic mail and file transfer as well as providing ordinary white page/yellow page services. This paper gives an introduction to the OSI Directory as defined in the Draft International Standard. Examples from the QUIPU implementation of the Directory are used to illustrate the concepts.

1. Introduction

The work on the OSI Directory (in short, the Directory) has originally started from the telecommunications users' needs to find out how a person can be reached via telephone, telefax, electronic mail, etc. The OSI directory service as defined today has a more general applicability. It can hold information not only about individuals but also about OSI applications (such as the presentation addresses of application entities). Other uses of the Directory include support for MHS routing and distribution lists, yellow page services, and wide area authentication.

This paper gives an overview of the OSI directory system drawing examples from the QUIPU [Kil88] implementation that is part of the ISODE [Ros89] package. The text is based on the Draft International Standard [ISO88a-h] (the IS version, although approved, is not available yet) and has been influenced by the earlier tutorials [Hui88], [Lun88], and [Kil89].

We start by viewing the Directory as a distributed database. Next we describe how the directory is organized in the form of a Directory Information Tree (DIT) and how information about objects is stored in its entries. Then we discuss the services the directory provides to its users. Finally, we cover some topics that are left as future work for the next revision of the directory standard.

2. The Directory as a Distributed Database

In OSI terms the Directory is a collection of open systems that cooperatively hold information about a set of real world objects. An object can be anything that can be named in some world, but usually the world is restricted to that of telecommunications and information processing areas. Examples of typical directory objects include countries, organizations, persons, computer systems, and application processes such as the FTAM process.

The set of information managed by the Directory is called the Directory Information Base (DIB). The Directory User (DU) is an entity or person which accesses the Directory. In more concrete terms the Directory can be thought of a distributed database to which the