
What if we know the names of some object/resource, but want to find out more about them:
- What is the telephone no. of X?
- What department does Y work in?
- What is the CPU speed of the machine with IP a.b.c.d?

What if we want to find some resource/object, but don’t know the name, only know some description of it:
- Find a laser printer with printing speed higher than 50 ppm in your office building
- Find the model no. and year of manufacture of all cars that have V8 engines

Directory Service
- Sort of yellow pages for resources
- Stores collections of objects and their attributes
- Retrieves the attributes of an object given the object name
- Retrieves the set of names that satisfy a given description (set of attribute values)

X.500 Directory Service
- A hierarchically-structured directory service designed for world-wide use
- X.500 is standardised by ITU (international telecommunication union) and ISO
- Accommodates resource descriptions in a standard form and their retrieval for any resource (online or offline)
- Never fully deployed, but the standard forms the basis for LDAP, the Lightweight Directory Access Protocol, which is widely used – IETF RFC 2251.
- A secure access to directory through authentication is also supported.
Consists of a set of objects/entries
- Each object is a list of attribute-value pairs
  - Each attribute can have a single or multiple values
- Objects are organized in a hierarchical tree structure (Directory Information Tree or DIT)
- Name of an object (called its Distinguished Name) is based on its position in the tree
  - Each DN is a sequence of RDNs (Relative Distinguished Name)
  - Each RDN identifies a branch in the tree leading from the root to the object

Example

DN = Host_Name= star,CN=Main Server, OU=Math. & Com. Sc., O=Vrije Universiteit, C=NL
Object class = Computer
Host_Name = star
Arch = Intel Xeon
Core = 2
Speed = 2.3
L1-Cache = 2
L2-Cache = 4
Memory = 4096
Disk = 20G
IP address = 10.3.21.4, 144.36.43.3
Location = Building 4

Example (contd.)

Basic Operation

- Directory Information Base (DIB) on directory server
  - Contains all information
- Directory System Agent (DSA) on directory server
  - Handles client requests for adding, deleting, modifying, and searching objects in the directory
- Directory User Agents (DUA) on clients
  - Accesses information in directory. Supports operations for adding, deleting, modifying and searching objects in the directory
What if an object is not found?
- DSA can forward request to other DSAs that has it (Chaining)
- DSA can give DUA the address of the other DSA that has it, and the DUA can access it directly (Referencing)

Classes and Attributes
- Each object belongs to one or more class
- Each class has a set of attributes that defines attributes that an object of that class can have
  - Each class must have some standards-defined mandatory attributes
    - Ex. Object class, DN, etc.
  - Can have some standards defined optional attributes
  - In addition, it can have any other application defined attributes (mandatory or optional)
    - Example CPU-Arch, L1-Cache, L2-Cache, Core etc.
    - Optional attribute means an object of that class may or may not have that attribute
- Subclassing used to inherit attributes from other classes

Types of classes
- Structural
  - Objects of this classes can be instantiated
- Abstract
  - Cannot have objects of this class
  - Helps in inheriting attributes
- Auxiliary
  - Cannot have objects of this class
  - Helps in adding attributes to other classes

Example - top
- superclass of all other classes (X.500 defined)
  - All other classes are subclass of it

```
top OBJECT-CLASS ::= {
  KIND abstract
  MUST CONTAIN { objectClass }
  OID 2.5.6.0}
```
Example - *person*

- Defines a "person" (X.500 defined)

```
person OBJECT-CLASS ::= { SUBCLASS OF { top }
  KIND abstract
  MUST CONTAIN { commonName | surname }
  MAY CONTAIN { description | telephoneNumber | userPassword | seeAlso }
  OID 2.5.6.6 }
```

Example - *Employee*

- Defines an employee (application defined)

```
employee OBJECT-CLASS ::= { SUBCLASS OF { person }
  KIND structural
  MUST CONTAIN { employeeCode | division | position | address }
  MAY CONTAIN { email | mobileNumber | webpage }
  OID 1.3.6.1.4.1.10.2 }
```

Each attribute has a similar definition defining
- Syntax (string (case ignore or not), OID, DN, Boolean,…)
- Matching rules (exact match, substring match)
- An unique OID identifying it…

An employee object

- DN = CN=Arobinda Gupta,OU=CSE,O=IIT Kgp,C=IN
- objectClass = employee
- objectClass = person
- objectClass = top
- commonName = Arobinda
- surname = Gupta
- address = XYZ, IIT Campus, Kharagpur - 721302
- employeeCode = 10203
- division = CSE
- position = Associate Professor
- telephoneNumber = 03222283476
- email = agupta@cse.iitkgp.ernet.in
Example

```
class student
  subclass of person
  mandatory attributes
    Roll no <from person>
    Department
  optional attributes
    Minor department
    Has-computer
```

A student object

```
DN = CN=Ashish Tewari,OU=CS,O=IIT Kgp,C=IN
Object class = student
First name = Ashish
Last name = Tewari
Address = Patel Hall, IIT Kgp – 721302
Roll No = 00CS1004
Department = CS
Age = 21
Has-computer = yes
```

Directory Schema

- Schema defines what all can be there in the directory and their structure
- Defines what classes can be there in the directory
- Defines which class can contain what mandatory and optional attributes, and their syntax
- Defines inheritance hierarchy (who is a subclass of who)
- Defines DIT structure
  - Which object (i.e. what object class) can be a child of which objects in the DIT

LDAP

- Lightweight Directory Access Protocol
- Protocol to access a X.500 like directory service
- Defines methods to connect to and authenticate to server, search, add, modify, rename objects
- Methods are generic, i.e., independent of the objects and their attributes
- Directory accessed is commonly called LDAP Server (some differences from X.500)
Basic Operation

- LDAP client connects to LDAP server
  - IP address and port provided
- LDAP client binds to LDAP server
  - Client is authenticated, Authentication methods vary
- LDAP client performs one or more operations on directory data
  - Read/modify
- LDAP client closes connection

LDAP Operation: Add

- Provides DN of object to add and its attribute-value pairs
  - Must have at least one value for each mandatory attribute
  - May have values for optional attributes
- Can only add to leaf of DIT
  - Position to add in depends on DN provided
- Object attributes must conform to directory schema

LDAP Operation: Search

- Provides DN of starting object to search from
  - Can be root if entire tree is to be searched
- Provides scope of search
  - Base, one-level, or subtree
- Provides filter to match
  - Only results that match the filter are returned
    - Ex. (object class=students) AND (Department=CS)
    - Matching rules depends on syntax of attribute matched
- What attributes to return for matched objects
  - All or list provided
- Other inputs
  - Synchronous or asynchronous search
  - Paged search or not, no. of results to return in one page
  - Resource bounds (size limit, time limit)
  - ...

**Other LDAP Operations**

- **Modify**
  - Provide DN of object to modify
  - Provide list of attribute-value pairs to add/delete/change
- **Delete**
  - Provide DN of object to delete
  - Must be a leaf in DIT
- Other operations like rename etc.

- **Any update must make the new object conform to directory schema**
- **Appropriate success or error codes returned**
- **C/C++/Java and other libraries are there for using LDAP calls from programs**
- **LDAP also defines the exact format of messages on the wire**

**Large Directories**

- A single LDAP server may not store a complete directory
  - May be too large
  - May not be needed
    - Employee information in US and Japan should be in different servers in US and Japan
- A search/update may span across multiple LDAP servers