Directories and LDAP (Lightweight Directory Access Protocol)

Francesco Bergadano
Università degli Studi di Torino, Computer Science Department
Corso Svizzera, 185 – 10149, Torino, Italy

INTRODUCTION

Directories and LDAP

References:

Brian Arkills, “LDAP Directories Explained”, Independent Technology Guides, 2005

G. Williamson, D. Yip, I. Sharoni, K. Spaulding, “Identity Management, a Primer”, MC press, 2009

What is a “directory”?

- A set of records
  - Each with a defined structure and containing defined attributes (e.g. name, birth date, address)
  - Organised in a hierarchy
  - Searchable in a fast, hierarchical, decentralized way
- Example:
  - A telephone directory
  - A list of departments in some organization, each with a list of employees

Information structure in a directory

- Each piece of information in a directory is usually called a record, an entry, or a directory object
- Each entry comprises a set of attributes, or properties, including a «type» and a «value»
- Some attributes may have more than one value

Directory vs Data Base

- Read optimized
- Information consistency may not be required, and there are no implied locking mechanisms designed to avoid simultaneous writes
- No stored procedure, no complex queries, no «joins»
- Appropriate for hierarchical information, to be distributed over many servers, and with the possibility of replicas
- Attributes may have more than one value

Example

A directory may be used to store

- information about the employees of some organization, with their personal data, addresses, and access rights to enterprise applications
- a hierarchical set of products or components, with images, technical characteristics, prices and applicable discounts
- a list of installed computers with their hw and sw characteristics

Use of a directory

- Manual
  - Using a command line application
  - Using a client-side graphic interface
  - Using a Web application
- Integrated in some software, for instance:
  - Email lookup in MUAs (Mail User Agents) – the user will click on a name or link and obtain the corresponding email address
  - Server-side authentication: the system looks up a user name on a directory, and finds correponding credentials to be verified

Directory standards

- X.500
  - ITU-T standard
  - Documentation is not free
  - Very complex
  - Includes X.509, for PKIs and certificates
- LDAP – Lightweight Directory Access Protocol
  - Open
  - Free
  - Documented via RFCs (www.ietf.org)
  - Simplified for practical applications
  - Relies on X.509 for certificate formats and PKIs

LDAP summary

- **Namespace**: a hierarchical organization of entry names and containers, with attribute values
- **Client operation** and protocol: how a client can communicate with an LDAP server
- **Schema**
- **Management**

Directories and LDAP

NAMESPACE

Namespace

- **Names**: naming of directory entries with values
- **Space:** organization & hierarchical structure of entries, ensuring that object names do not conflict

LDAP & DNS interaction

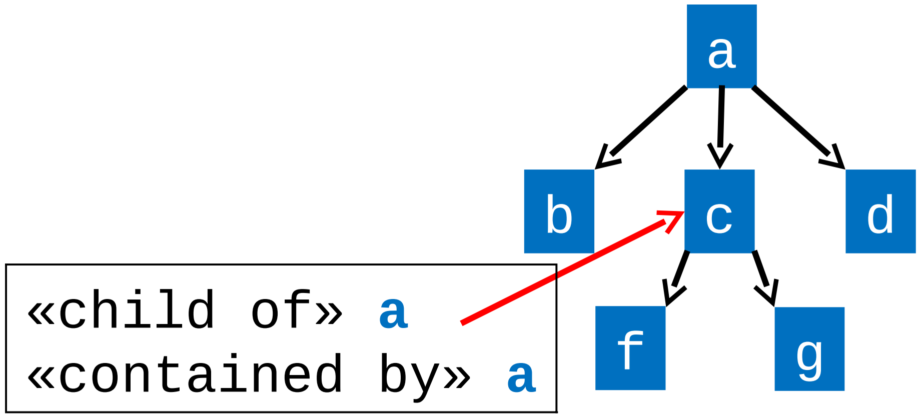
- LDAP clients use DNS to locate and LDAP server (possibly via a DNS «SRV» record), i.e. the IP address of the LDAP server for domain.com will be found in the associated SRV record
- DNS names may be used as entries in a directory tree

Some DNS record types:

- Address (A): IP address assigned to name
- MX: Mail eXchange for this domain
- SRV: LDAP server for this domain

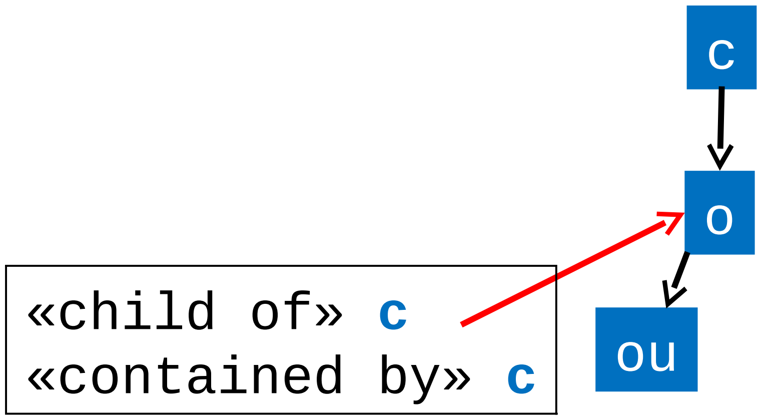
LDAP directories (1)

- Entries are organized in a hierarchy, so that
  1. information may be distributed over multiple servers
  2. access control can also be hierarchical
- Intermediate nodes in the hierarchy are called «container» entries (e.g., a, c are container entries in the directory below)
- Other nodes are called «leaf» nodes (f and g below)



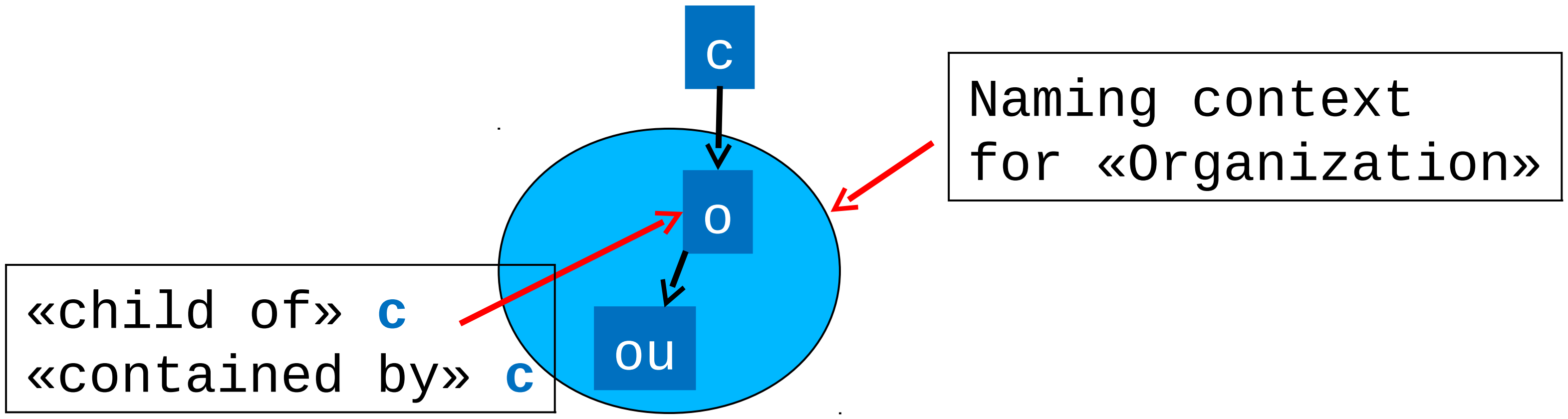
LDAP directories (2)

- A leaf entry may become a container, by creating children of that entry
- An entry is a list of attribute-value pairs
- There may be more than one value per attribute
- There is no attribute specifying whether an entry is a container and no attribute for listing the children
- Typical containers are country (c), organization (o), organizational unit (ou), as in X.500



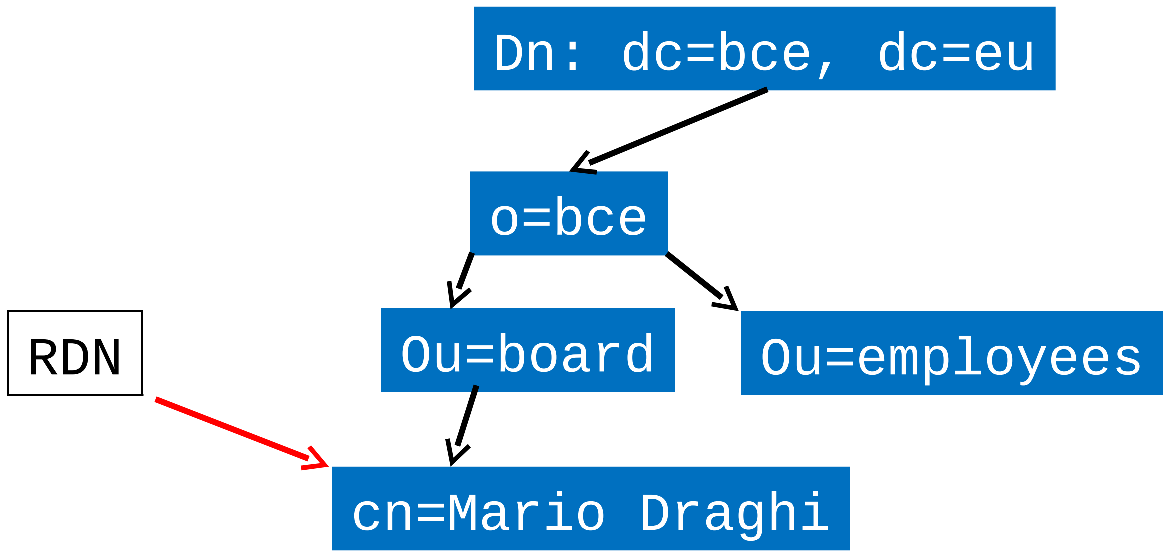
Directory structure and Objectclass

- Directory structure defines how attributes may be used
- Every entry has the «Objectclass» attribute, that is assoiated to defined structure rules (e.g. «Organizationalunit» must be a child of «Organization»)
- Directory subtrees are also called «Naming Contexts»



Object Naming

- RDN = relative distinguished name, a unique ID within a container, it is one of the attributes of the entry, called a «naming» attribute
- Attribute types may also be named via an «OID» (object identifier), a numeric value such as 2.5.4.5

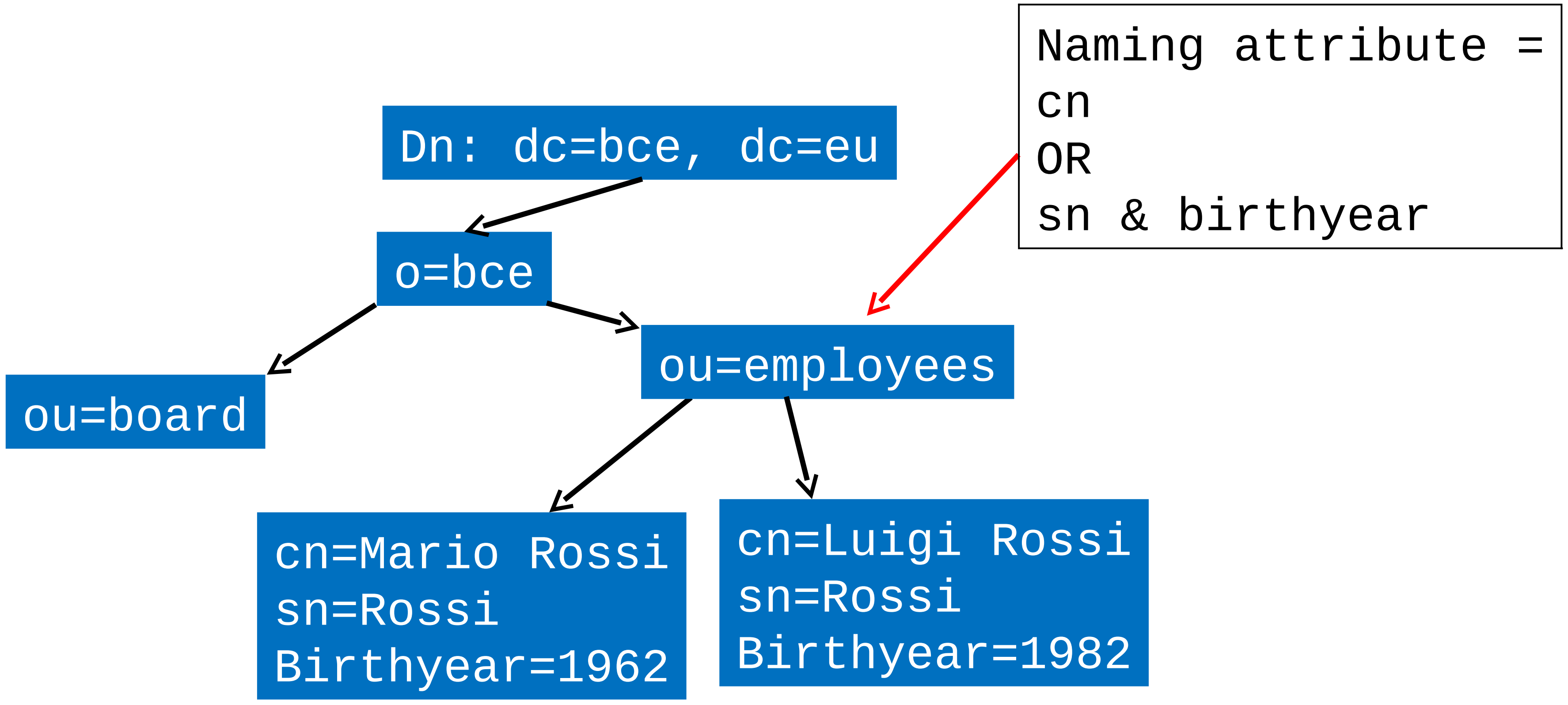


Typical attribute names

- Typical attributes names, sometimes used as naming attributes:
  - Common Name (cn)
  - Organization (o)
  - Organizational Unit (ou)
  - Country (c)
  - Domain Component (dc)
  - User Identifier (uid)
- Naming attributes should be public and static, otherwise search will not work

RDN with multiple attributes

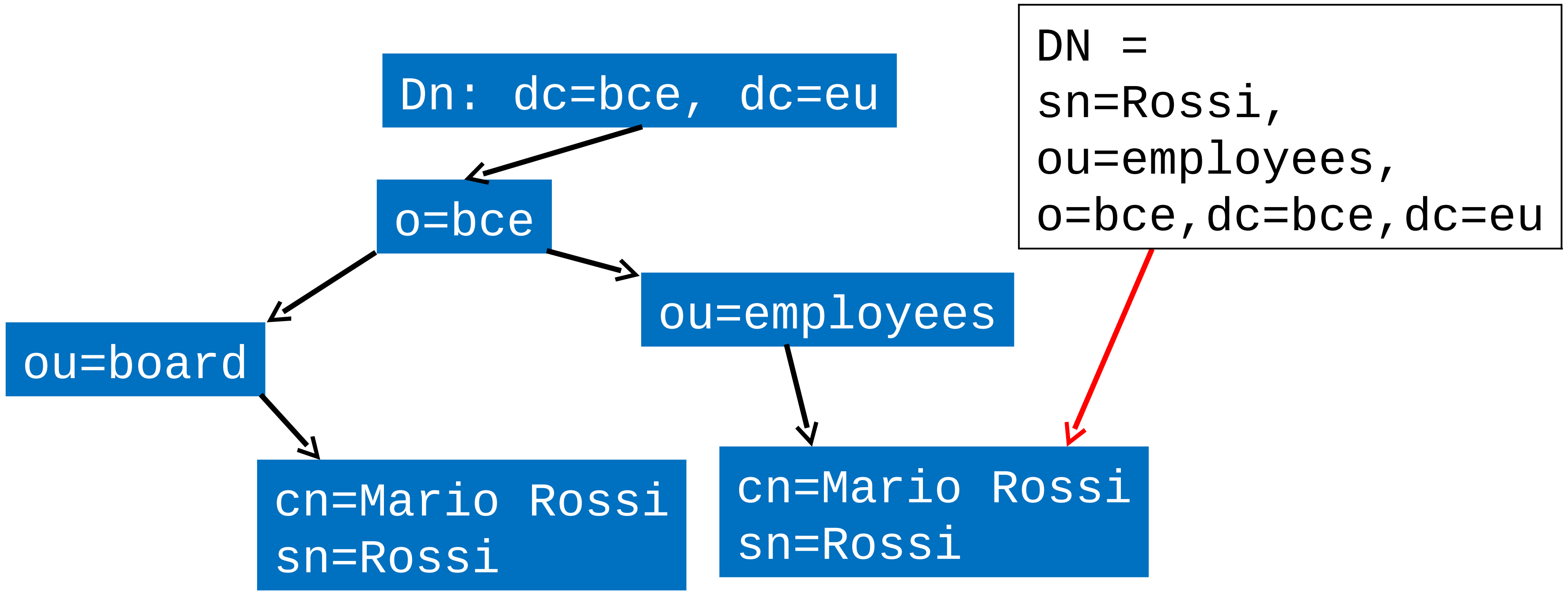
- RDN may consist of 2 or more attributes if values are not unique for just 1 attribute



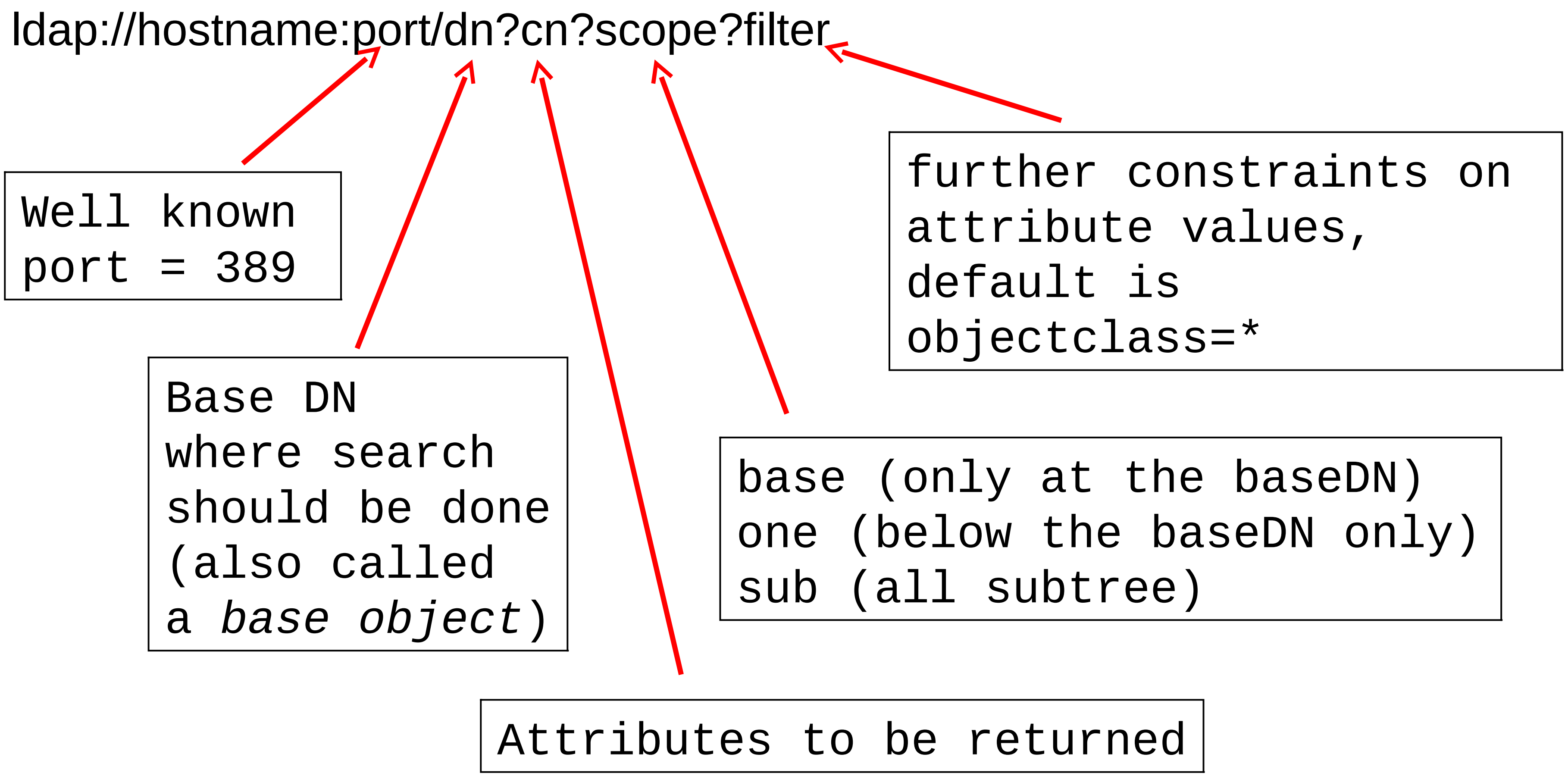
Naming attribute = cn OR sn & birthyear
Dn: dc=bce, dc=eu
o=bce
ou=board
ou=employees
cn=Mario Rossi sn=Rossi Birthyear=1962
cn=Luigi Rossi sn=Rossi Birthyear=1982

Distinguished Name (DN)

Fully qualified and complete, with position in the hierarchy, not fully stored in the entry, includes attribute values up the hierarchy



URL naming



ldap://hostname:port/dn?cn?scope?filter

Well known
port = 389

Base DN
where search
should be done
(also called
a *base object*)

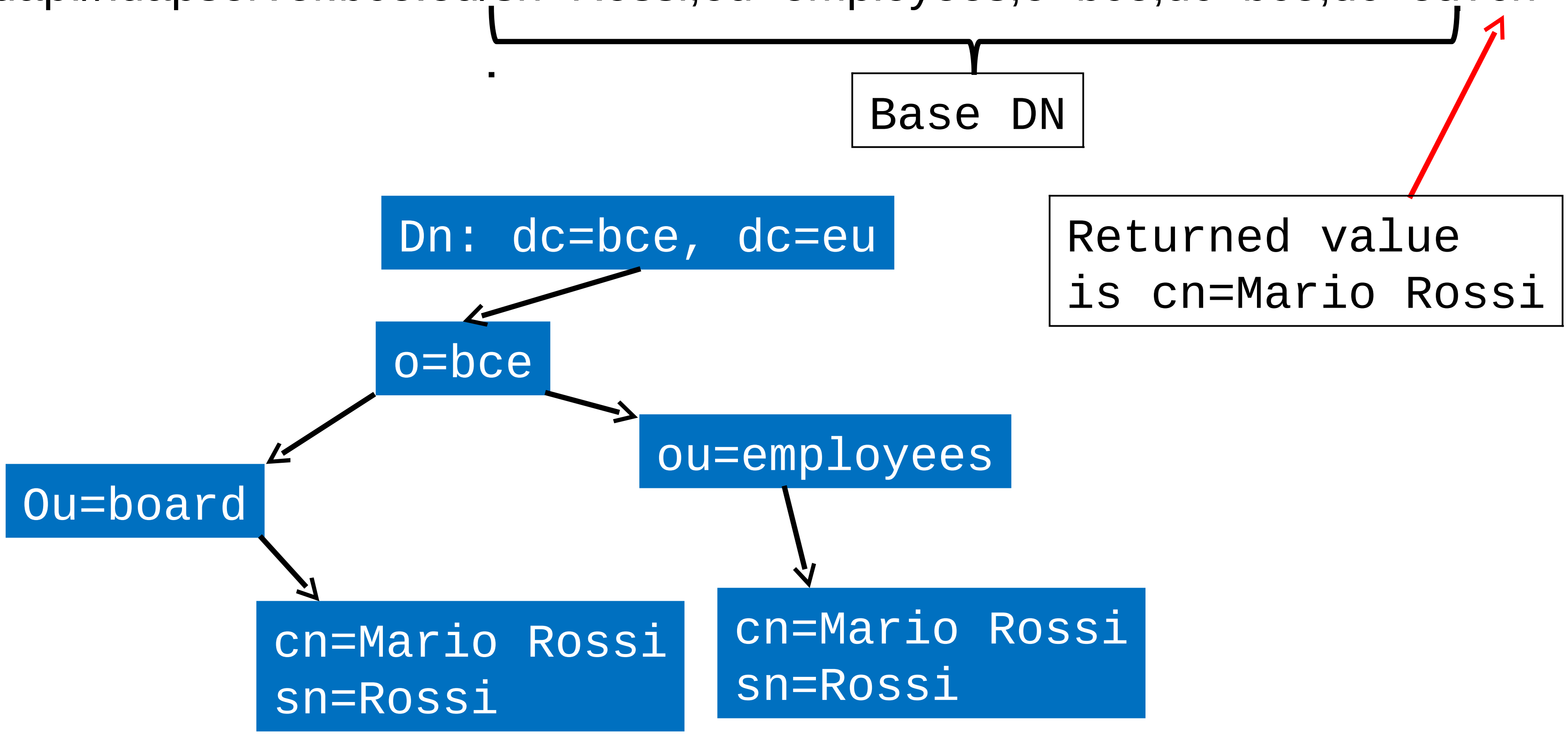
Attributes to be returned

base (only at the baseDN)
one (below the baseDN only)
sub (all subtree)

further constraints on
attribute values,
default is
objectclass=*

URL naming - example

ldap://ldapserver.bce.eu/sn=Rossi,ou=employees,o=bce,dc=bce,dc=eu?cn

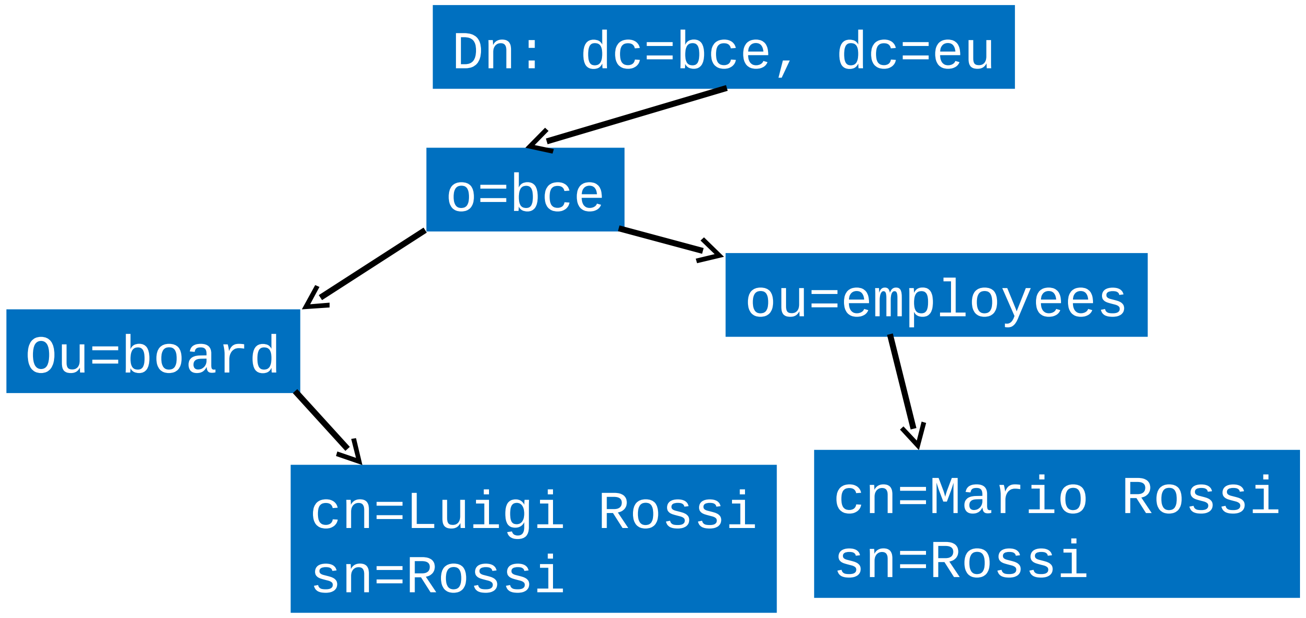


URL naming – more examples

ldap://ldapserver.bce.eu/dc=bce,dc=eu?cn?sub?(sn=Rossi)

ldap://ldapserver.bce.eu/dc=bce,dc=eu??one?(sn=Rossi)

ldap://ldapserver.bce.eu/dc=bce,dc=eu??sub?(cn=Mario%20Rossi)



Escaping characters in DNs

,	\,
space	%20
\	\\
+	\+
”	\”
<	\<
>	\>
#	\#
;	\;

Directory-enabled services

Email server:
Sendmail -> ldap server
Microsoft Exchange -> active directory server
When email is received, server looks up email address to find if user is local and where the corresponding mailbox is located

Email client:
To look up destination email address based on user-given search parameters (e.g. last name)

Software distribution to clients via active directory

Directories and LDAP

CLIENT OPERATION

LDAP clients

- Browser accepting LDAP urls
- Web-based (the user accesses an HTTP to LDAP gateway)
- Custom client

LDAP search

Client sends to LDAP server a search request with

Mandatory parameters:

- Base DN (also called a base object)
- Scope (base, one, subtree)
- Search filter (attributeType comparisonOperator attributeValue)

Optional parameters:

- Attributes to be returned (all if not present)
- derefAliases
- sizelimit, timelimit
- typesonly (only attribute types are returned, not values)

Search filters

May be combined with &, |, !
Example: (& (! cn=Mario%20Rossi)(sn=Rossi))

Comparison operators:
=,<=,>=, ~= (about equal, implementation-dependent)

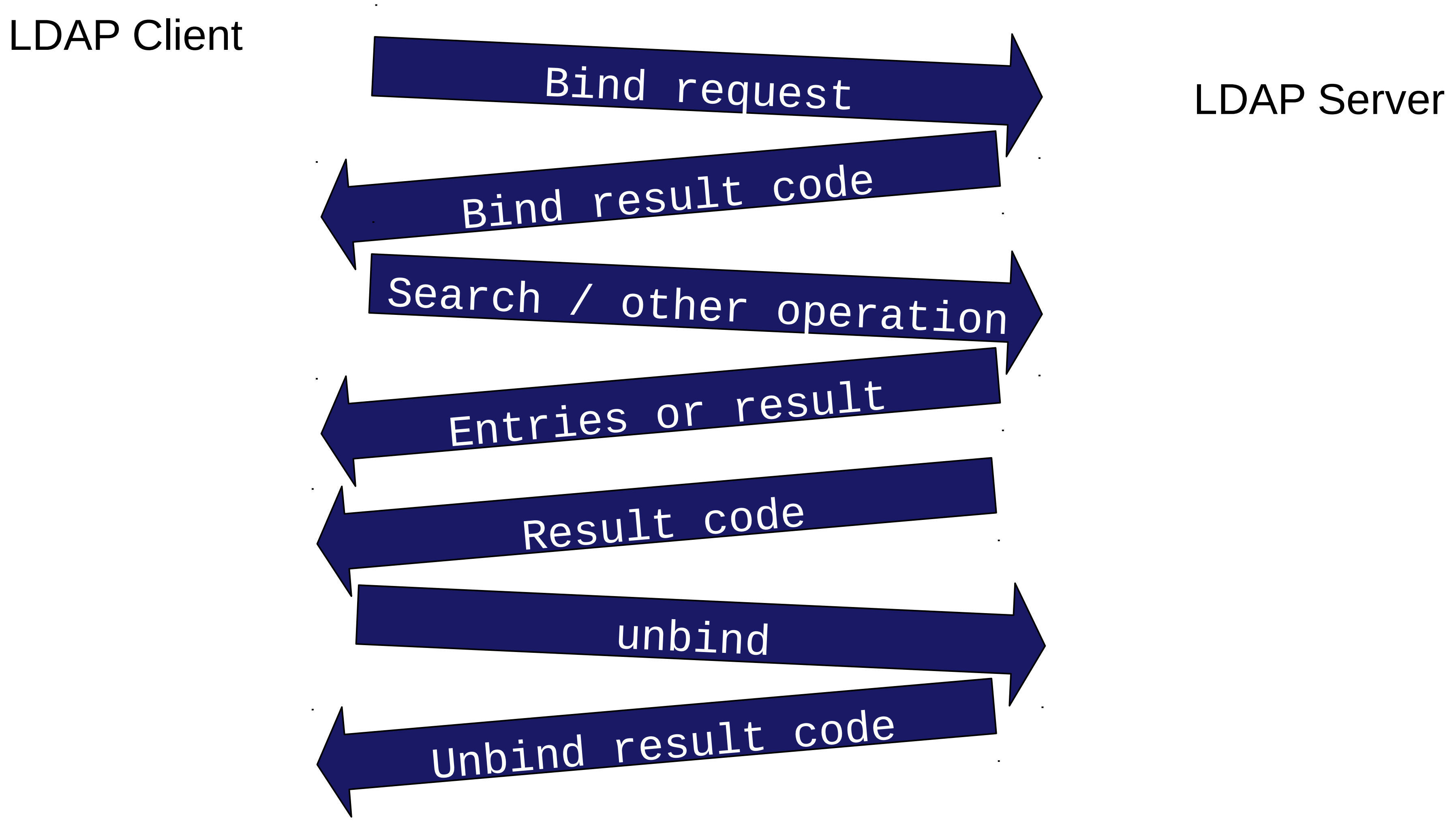
Attribute values may include *

GUIs in clients may simplify search syntax/interface

Optional LDAP search parameters

- derefAliases:
  - neverDerefAliases (alias is returned as is)
  - derefInSearching (only in subtree)
  - derefFindingBaseObs (only in base DN)
  - derefAlways (always deref)
- Sizelimit (max number of entries returned, 0 if no limit). Limit may be set on server, so client may only limit further.
- Timelimit (max time in seconds to complete the search)

LDAP protocol



LDAP protocol

Protocol stack
- TCP with keepalive
- UDP (CLDAP = connectionless LDAP)

Messages
- Operations
- Controls
- Client options

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

Credentials (username, password) could be within the namespace, in this case passwd change can be done via modify

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

Scope is base, one or subtree
Optional_parameters include

- attributes to be returned
- derefAliases
- timelimit, sizelimit
- typesonly

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

returns true or false

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

Adds an entry, if attribute value pairs include objectclass, the schema will be checked.
Attribute-value pairs are separated by a semicolon (;).

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

Completely deletes an entry

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

Deletes attribute-value pairs and/or adds values to existing attributes.
Attribute-value pairs are separated by a semicolon (;).

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

Flag tells if old RDN is kept as an attribute

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

TCP keepalive possible

LDAP operations

bind DN credentials
search baseDN scope filter optional_parameters
compare DN attributeType attributeValue
add DNofNewEntry attribute_value_pairs
delete DNofEntryToBeDeleted
modify DNtoBeModified attribute_value_pairs
rename DN newRDN flag [newContainerDN]
unbind
abandon MessageID

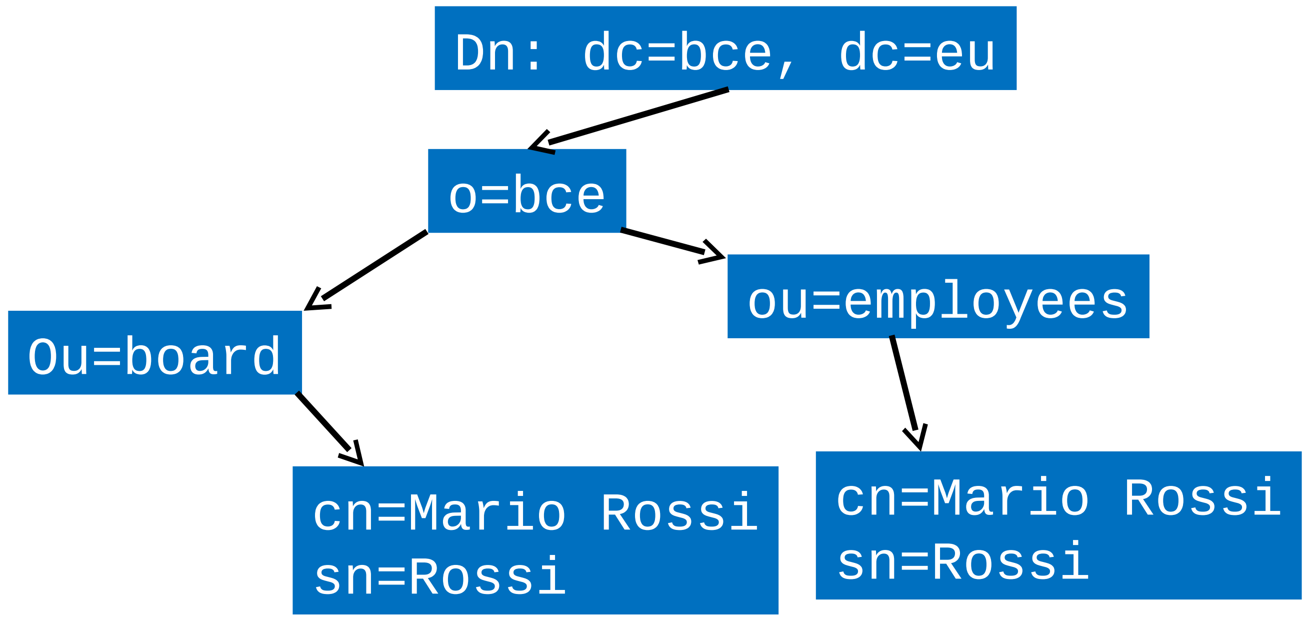
MessageID is an ID of a previous message that is blocked or taking too long to respond.

LDAP operations examples

Exercise: do examples with command line OpenLDAP, using all the previous operations.

Example: rename entry ou=employees,cn=Mario Rossi
into ou=employees,cn=Mario Bianchi,sn=Bianchi *

* with flag Delete-Old-RDN
= TRUE, and no
new parentDN



LDAP controls

Extra parameter in an LDAP operation

Advertised in the root under the SupportedControl attribute

Effects:

- how the result is returned
- access entries that would otherwise be ignored
- added functionality

Designted with an OID (Object Identifier,

a string of numbers such as 3.4.7.2)

LDAP controls - examples

- **Paged search control**: returns one page at a time, thus allowing more results than sizeLimit
- **Server-side sort control**: server will sort based on given attribute

Client options

These last for a whole client-server session, hence non related to a single search operation like controls.

Examples:

LDAP_OPT_DEREF

LDAP_OPT_SIZELIMIT

LDAP_OPT_TIMELIMIT

LDAP_OPT_REFERRALS

LDAP_OPT_SSL

LDAP_OPT_REFERRAL_HOP_LIMIT

Client options

These last for a whole client-server session, hence non related to a single serch operation like controls.

Examples:

LDAP_OPT_DEREF

LDAP_OPT_SIZELIMIT

LDAP_OPT_TIMELIMIT

LDAP_OPT_REFERRALS

LDAP_OPT_SSL

LDAP_OPT_REFERRAL_HOP_LIMIT

LDAP_DEREF_NEVER
LDAP_DEREF_ALWAYS
LDAP_DEREF_SEARCHING (subtree only)
LDAP_DEREF_FINDING (baseDN only)

Client options

These last for a whole client-server session, hence non related to a single serch operation like controls.

Examples:

LDAP_OPT_DEREF

LDAP_OPT_SIZELIMIT

LDAP_OPT_TIMELIMIT

LDAP_OPT_REFERRALS

LDAP_OPT_SSL

LDAP_OPT_REFERRAL_HOP_LIMIT

LDAP_OPT_ON,
LDAP_OPT_SUBORDINATE_REFERRALS
LDAP_OPT_EXTERNAL_REFERRALS

C language APIs (RFC 1823)

- **Syncronous/Asyncronous, e.g.**
    - **ldap_add_s** (blocks till result is returned)
    - **ldap_add**

- **Results and errors found in the LDAPMessage data structure, e.g.**
    - **ldap_first_entry** (returns entry)
    - **ldap_next_entry**
    - **ldap_first_attribute**
    - **ldap_next_attribute**

C language APIs - examples

ldap_open	open a connection to an LDAP server
ldap_add_s	synchronously add an entry
ldap_bind	asynchronously bind to the directory
ldap_bind_s	synchronously bind to the directory
ldap_unbind	synchronously unbind from the LDAP server, close connection
ldap_compare	asynchronous compare to a directory entry
ldap_compare_s	synchronous compare to a directory entry
ldap_delete	asynchronously delete an entry
ldap_delete_s	synchronously delete an entry

Directories and LDAP

SCHEMA

Schema

- A set of **rules** and entry **type definitions**
- The schema is published under the **subschemaSubentry** attribute of any directory entry. The corresponding value is the DN of the entry publishing the schema.

  $\longrightarrow$ thus, the schema is easy to read by clients and can also be more easily modified and maintained.
- Schema checking is done on any add, modify, modifyDN operation
- Includes
  - **Structure rules, Content rules, Name Form** for Object Class
  - **Syntax** and **matching rules** for attributes

Schema components

- **Object Class**: entry types allowed
- **Structure rules**: how the Object Class relates to other objects in the namespace
- **Name form & syntax**: attribute names and value types
- **Matching rules**: how to compare data values

Default schema: defined in RFC 2252

Can be extended in implementations

Distributed schemas

Widespread LDAP implementations are distributed with a set of schema specifications ready to use.

For example, in OpenLDAP, schema files are installed in /usr/local/etc/openldap/schema, and may be included in the slapd.con configuration file.

For example include the files:

```
/usr/local/etc/openldap/schema/core.schema
/usr/local/etc/openldap/schema/cosine.schema
/usr/local/etc/openldap/schema/inetorgperson.schema
```

Objectclass

- Every directory entry has an objectclass attribute, and the values correspond to objectclass definitions in the schema
- The objectclass defines required (**Must**) and optional (**May**) attributes
- Directory implementations allow to query for objects with a selected objectclass value, in the whole directory tree, e.g., objectclass=person

Objectclass types

- **Abstract** (templates):
  - They cannot inherit from other objectclasses
  - No entries with data belong to them, only used to define structural classes
- **Structural**: uses inheritance via subclass/superclass
- **Auxiliary**: existing class + new attributes (no real inheritance)

Objectclasses inherit from other objectclasses, by including the corresponding attributes, and adding more attributes

Elements of an objectclass

- **OID** – unique numeric object identifier
- **Name** – the name of the objectclass
- **Superclass** – the superclass it inherits from
- **Category** – abstract, auxiliary or structural
- **Must** attributes
- **May** attributes
- **Naming attributes** – those used for RDNs

Elements of an objectclass

The schema may be stored

- In a file, e.g.

  /usr/local/etc/openldap/schema/core.schema), or
- In the directory itself, e.g.

  cn=person, ou=schema, must=attr1, may=attr2, may=attr3

Creating entries

When we create a new entry in a directory we must associate to it one or more objectclasses, and this can be done in one or more of the following ways:

- Use **existing objectclasses** in the schema
- Create and use new objectclasses that are **subclasses** of any existing objectclass
- Define and use new **auxiliary classes**

Syntax

- Data format used by attribute types and matching rules
- Defined via ASN.1
- Examples: image, string:

```
attributetype ( 0.9.2342.19200300.100.1.60
   NAME 'jpegPhoto'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.28 )

attributetype ( 2.16.840.1.113730.3.1.1
   NAME 'carLicense'
   EQUALITY caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

Attributes

- Defined by OID, name, superior class (hence also in a hierarchy, like objectclass), equality, order, substring matching rules, syntax, number of values, modifiable
- Can have multiple names (1° one is the canonical name and is normally returned by searches)
- Subtypes possible, e.g.

```
 attributetype (2.5.4.3
    NAME ‘cn‘
    SUP ‘name’)
 attributetype (2.5.4.41
    NAME ‘name'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

Attributes

- Defined by OID, name, superior class (hence also in a hierarchy, like objectclass), equality, order, substring matching rules, syntax, number of values, modifiable
- Can have multiple names (1° one is the canonical name and is normally returned by searches)
- Subtypes possible, e.g.

```
attributetype (2.5.4.3
   NAME ‘cn‘
   SUP ‘name’)
attributetype (2.5.4.41
   NAME ‘name'
   EQUALITY caseIgnoreMatch
   SUBSTR caseIgnoreSubstringsMatch
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

Hence, searching for name=* will also return all values of cn

Attribute description

- Part of the schema
- Includes attributetype and attributeoptions

e.g.:
Binary (return as binary, not converted to a string, useful for certificates)

Namespace vs Schema

Namespace

Data:

An instantiated tree of entries (attributes and values), each belonging to an objectclass

Schema

Type:

A tree of objectclasses, each defined with mandatory and optional attributes, and matching rules

Directories and LDAP

MANAGEMENT

Management

- Replication
- Referrals
- Aliases
- Distribution models
- Directory integration
- Data exchange formats

Replication

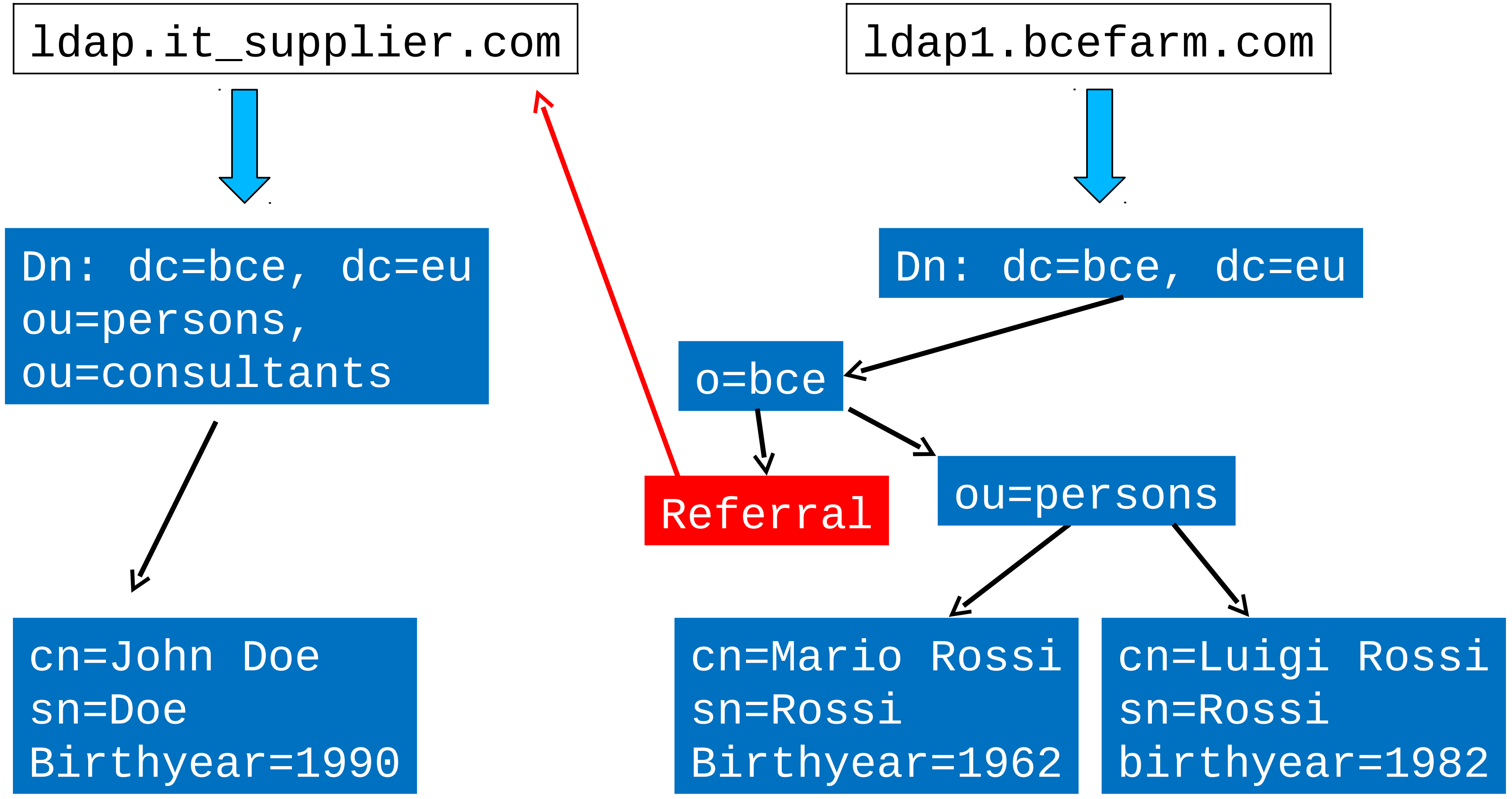
- Useful for availability & backup
- ‘Partitions’ (i.e., naming contexts, or subtrees) may be replicated on other servers
- A replicated partition is called a ‘replica’

$$\begin{tikzcd}
 & \text{Single master (replicas are read-only)} \\
\text{Replication} \arrow[ur] \arrow[dr] & \\
 & \text{Multi master (all replicas are read/write)}
\end{tikzcd}$$

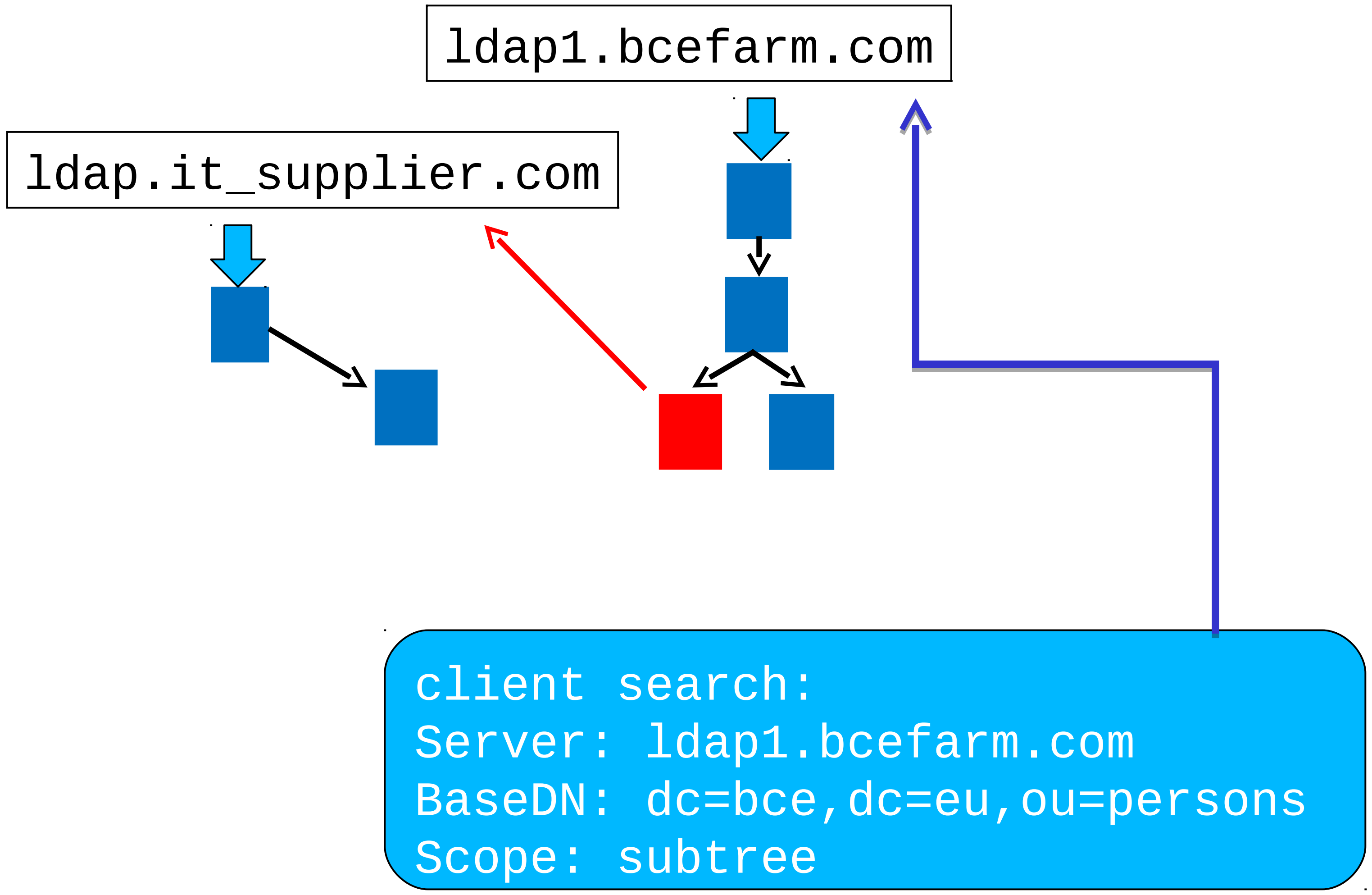
Referrals

- ‘Refer’, or redirect, the client to another server, port, baseDN
- Client will then ‘chase’ the referral for search (or modify, delete, etc.)
- Types of referrals:
  - ‘Subordinate/superior’ referrals (up/down the directory tree)
  - ‘External’ referrals (other DNS and server)
  - ‘Default’ referral if search is outside the namespace

Referrals - example

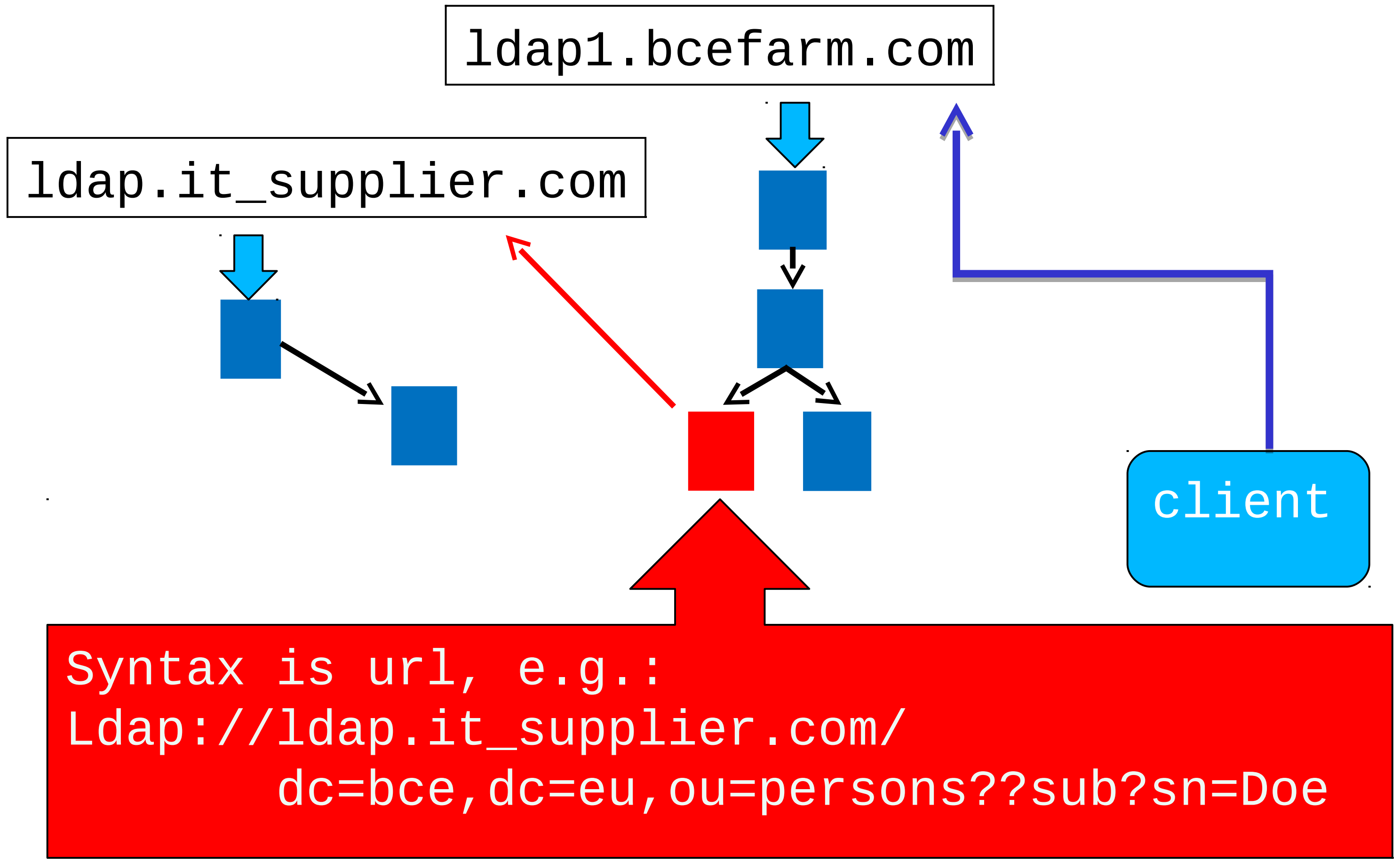

ldap.it_supplier.com
ldap1.bcefarm.com
Dn: dc=bce, dc=eu ou=persons, ou=consultants
cn=John Doe sn=Doe Birthyear=1990
Dn: dc=bce, dc=eu
o=bce
Referral
ou=persons
cn=Mario Rossi sn=Rossi Birthyear=1962
cn=Luigi Rossi sn=Rossi birthyear=1982

Referrals – example with client search



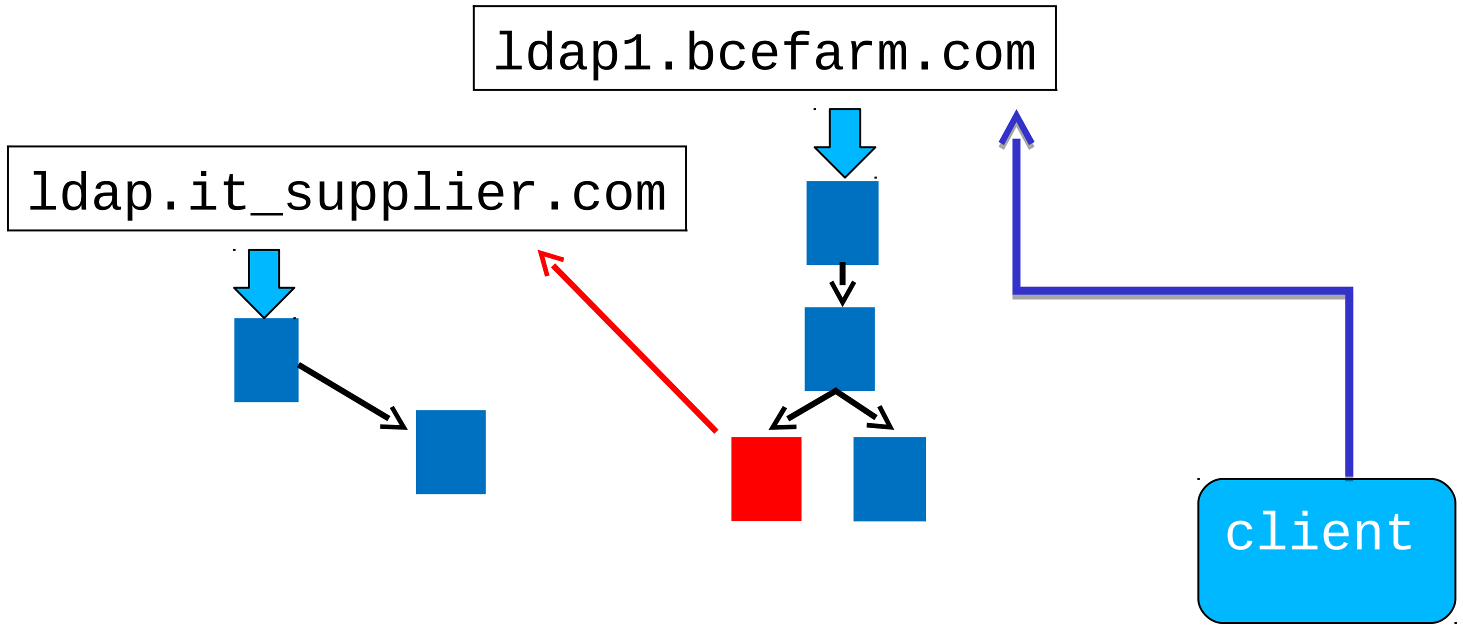
client search:
Server: ldap1.bcefarm.com
BaseDN: dc=bce,dc=eu,ou=persons
Scope: subtree

Referrals – syntax



```
Syntax is url, e.g.:
Ldap://ldap.it_supplier.com/
       dc=bce,dc=eu,ou=persons??sub?sn=Doe
```

Referrals – chaining



- Server may implement ‘chaining’, i.e. it will chase referrals on behalf of the client. Otherwise the client will obtain the referral syntax from the server and then autonomously chase the referral.
- When chasing a referral, the client may be asked for new credentials

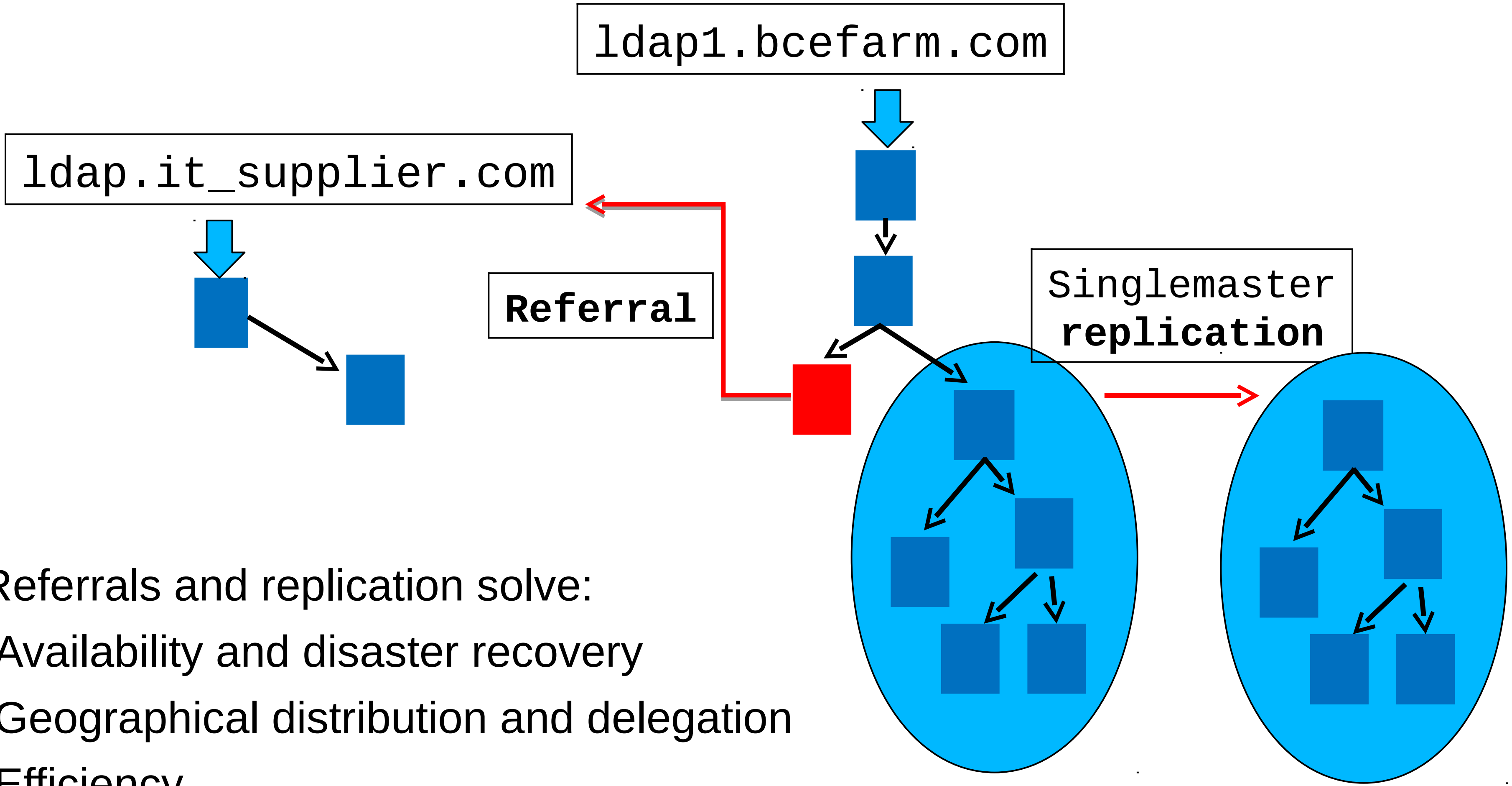
Aliases

- Similar to referrals, but
  - Alias points to another entry in the same directory, whilst referrals may point to any URL
  - Alias points to single entry, referrals may also point to subtrees
  - Alias is always resolved by the server, referrals may be chased by clients (unless chaining is in use)
- Example: same person in two subtrees (e.g. two University Departments, where some professor is teaching, but main affiliation and data are only in one Department).

Aliases - continued

- Client specifies (in the ldap search parameters) if the Server should ‘dereference’ aliases
- Aliases are resolved by the server only on search. Modify and delete are done only on the alias entry, not on the target
- Target data can be kept private if alias has a different RDN

Distribution models



Referrals and replication solve:

- Availability and disaster recovery
- Geographical distribution and delegation
- Efficiency

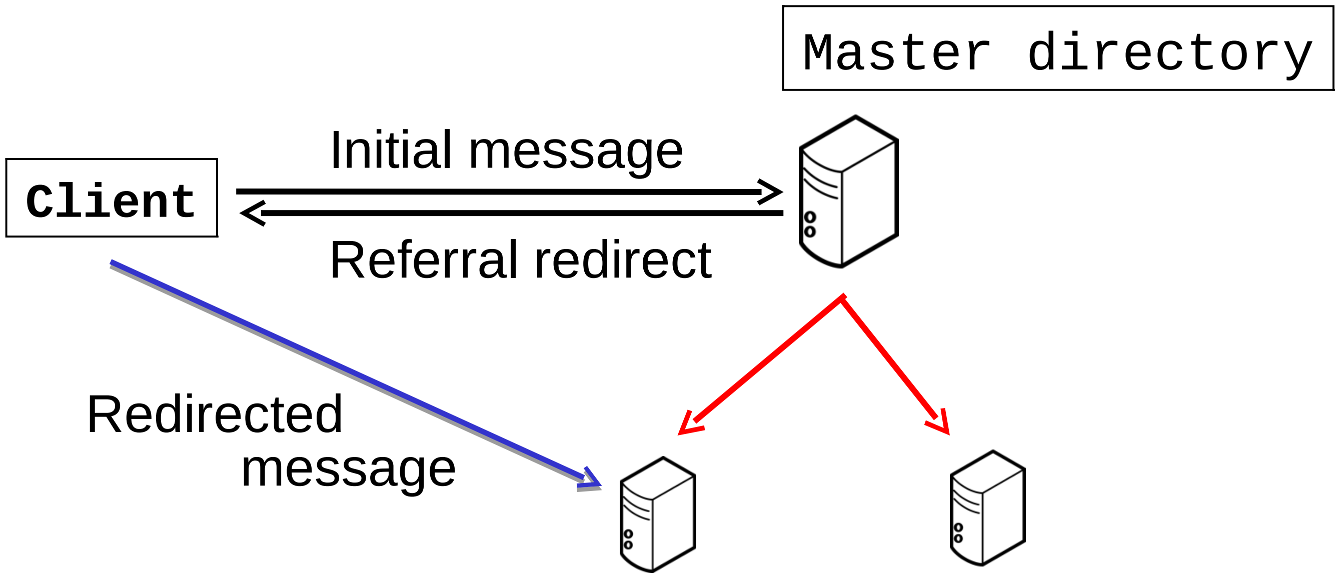
Directory Integration

- Not all directory products fully implement LDAP
- Many directories ad data sources may be present and need to be integrated, via
  - Analysis
    - Data sources & owners
    - Consumers of data
    - Subscribers (other directories and services)
  - Implementation
    - Metadirectory products
    - Custom integration via scripts and software tools

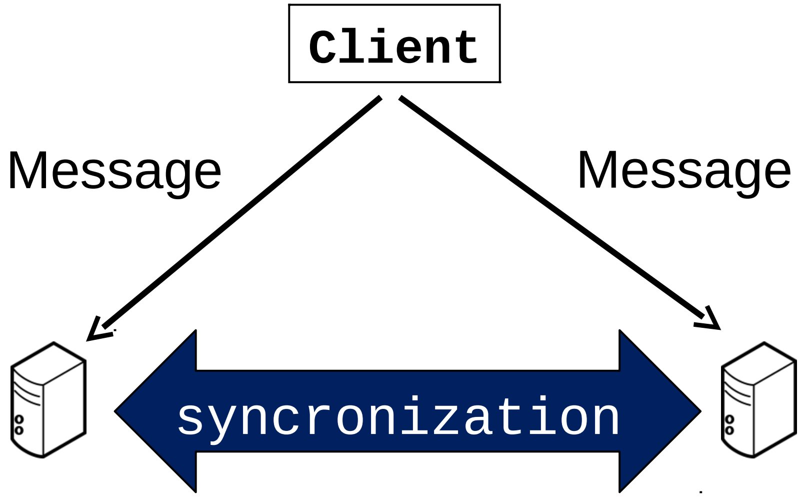
Metadirectories

- Master directory
- Syncronization
- Loose interconnection (just a common user interface)
- Harvester
- Connector (Harvester with extra funcionality, e.g. attribute name mapping)

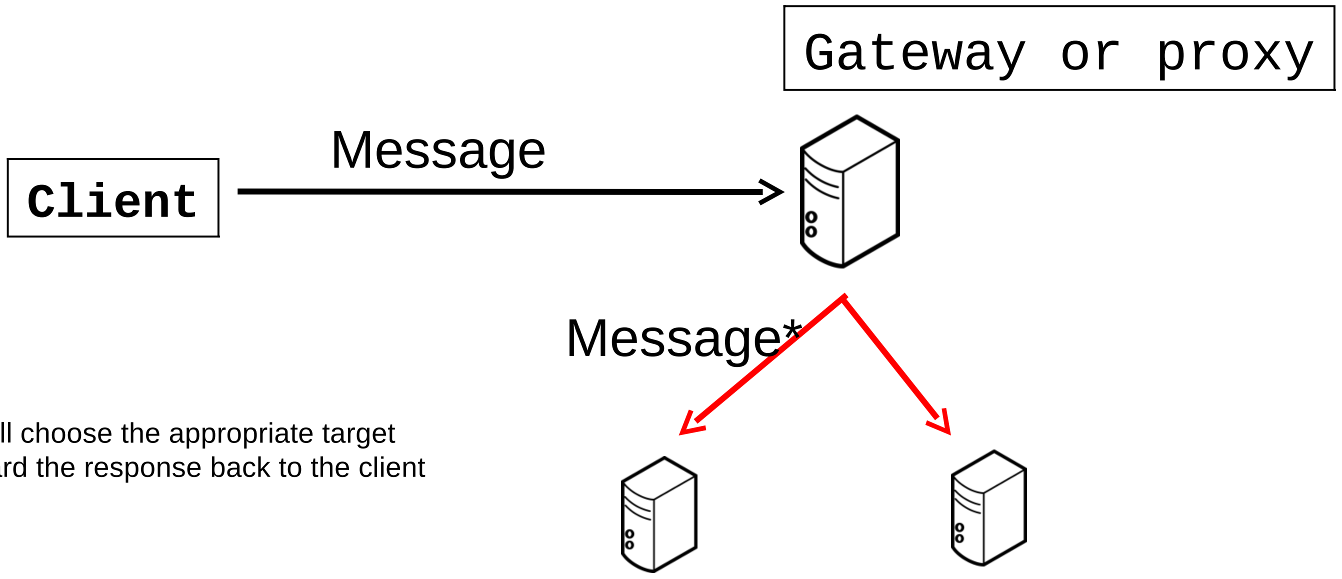
Directory integration via master directories



Directory integration via syncronization / multimaster replicas



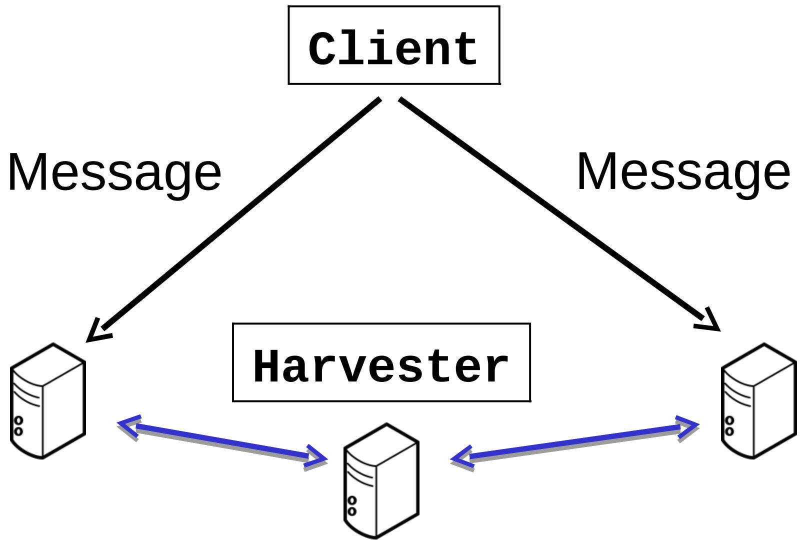
Directory integration via gateways (loose interconnection)



* The gateway will choose the appropriate target server, and forward the response back to the client

Directory integration via harvesters

The harvester will continuously query each dir for relevant entries, and write them to the other directory.



Data Interchange Formats

- LDIF – LDAP data interchange format
  - Text based (may be edited and modified by scripts)
  - Supports both LDAP data and LDAP operations (e.g. delete, add)
  - Syntax examples (see also lab examples):
    - #comments possible
    - changetype: add/modrdn/delete/modify
    - attribute::xxx (where :: force base64 coding)
- DSML – directory services markup language
  - XML-based, LDAP data and operations, over HTTP or via file
  - Supported by IBM, Oracle, Microsoft