Lightweight Directory Access Protocol

Computer Systems (CS2000)

Trimester 2 2020

Contents

X.500

- DAP (Directory Access Protocol)
- DSP (Directory System Protocol)
- DISP (Directory Information Shadowing Protocol)
- DOP (Directory Operational Bindings Management Protocol)
- Used the OSI networking stack

History of LDAP

- Originally started as a front end to X.500
- Provides much of X.500's functionality at a lower implementation cost
- Removed redundant and rarely used operations
- Uses TCP rather than OSI stack
- University of Michigan wrote first LDAP implementation
- Most early LDAP implementations were based on it
- U.Mich eventually realised didn't need X.500 and wrote lightweight server
- Meant it was easier to deploy, and more people started using it

What is LDAP?

- Standard protocol that can be used to access information over a network
- Directory Service
- Described in RFC1777 and RFC2251
- Stores attribute based data
- Data generally read more than written to
- Client-server model
- Based on entries collection of attributes
- Globally unique distinguished name (DN) – like domain name
- Entries arranged in hierarchical tree structure

Directory Services

- Directories are a specialised database
- Used in everyday lives - phone book, TV guides, etc
- Everyday directories fairly static
- Online directories are dynamic, flexible, secure and can be personalised
- Examples of online directories are finger, DNS, NIS and unix password file

Categories

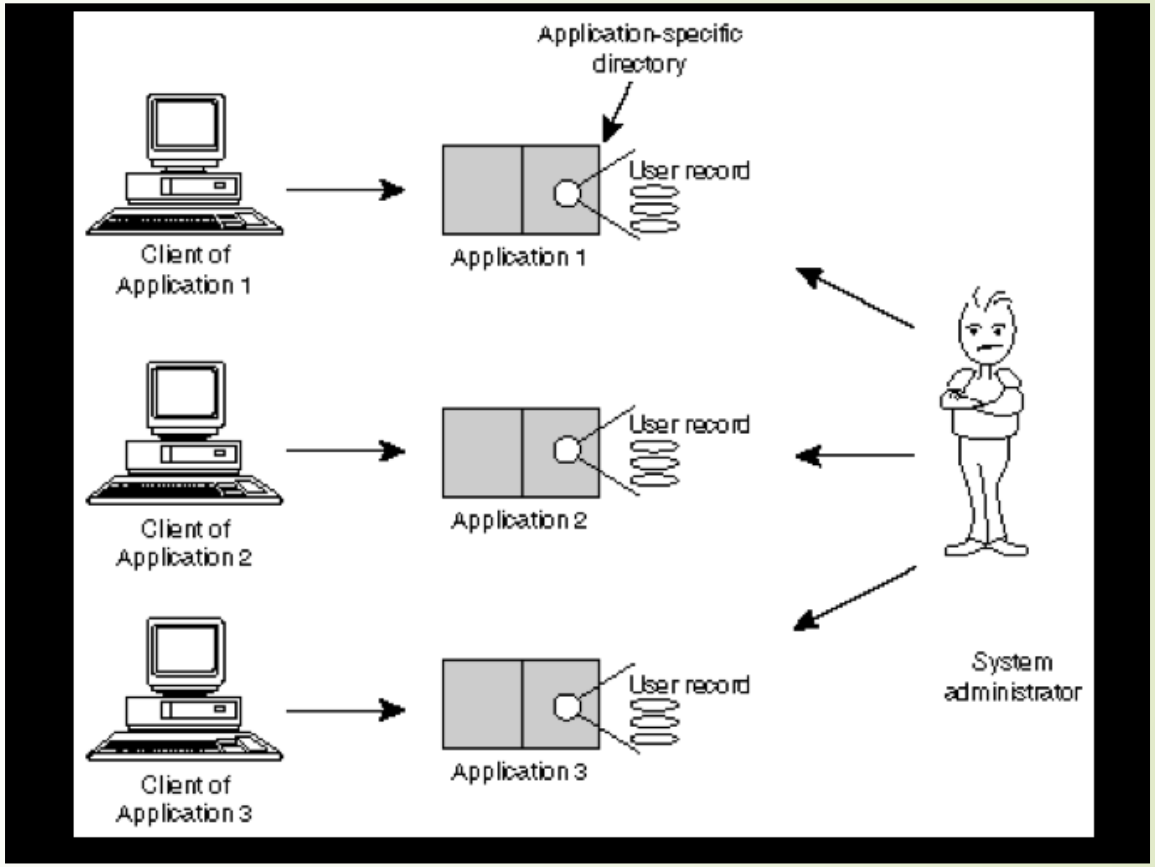
- NOS-based directories
  - *Directories such as Novell's NDS (now NetIQ), Microsoft's Active Directory, and Banyan's StreetTalk Directory are based on a network operating system*
  - *NOS-based directories such as these are developed specifically to serve the needs of a network operating system.*
  - *Novell is now owned by MicroFocus https://www.microfocus.com/novell/?utm_medium=301&utm_source=novell.com*
- Application-specific directories
  - *These directories come bundled with or embedded into an application.*
  - *E.g. IBM (Lotus) Notes name and address book, the Microsoft Exchange directory, and Novell's GroupWise directory.*

Categories (cont.)

- Purpose-specific directories
  - *These directories are not tied to an application, but are designed for a narrowly defined purpose and are not extensible.*
  - *E.g. the Internet's Domain Name System (DNS).*
- General-purpose, standards-based directories
  - *These directories are developed to serve the needs of a wide variety of applications.*
  - *E.G. LDAP & X500*

Why use LDAP?

- Centrally manage users, groups and other data
- Don't have to manage separate directories for each application
  - *Stops the "N + 1 directory problem"*



Why use LDAP?

- Distribute management of data to appropriate people
- Allow users to find data that they need
- Not locked into a particular server
- Ability to distribute servers to where they are needed
- Standards available for sharing data

What LDAP can do

- Find things - local office phone book, Internet directories such as VeriSign, BigFoot etc
- Manage things - users and groups.
- Lightweight database applications - sort bookmarks, small bits of data
- Security - store username and passwords, digital certificates, PKI etc

What LDAP can't do

- LDAP isn't a relational database
  - *No rollback for failed operations*
- LDAP isn't a filesystem
  - *No locking or seeking*
  - *Can't search blobs of data*
- LDAP isn't good for dynamic objects
  - *Optimised for read, not write*
  - *No locking to ensure transactions occur in a certain order - corruption could occur*
- No generic SQL-like reporting language
- Not useful without applications

LDAP vs Databases

Read-write ratio - LDAP is read optimised

Extensibility - LDAP schemas are more easily changed

Distribution - with LDAP data can be near where it is needed, and highly distributed

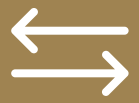
Replication - with LDAP data can be stored in multiple locations

Different performance - directories used for many different applications, and queries are usually simpler, but many more of them

Data sharing - LDAP is designed for sharing data, databases designed for one application

LDAP vs Databases

Database objects have complex relationships



Transaction model - *LDAP transactions are simple - usually changing one entry, databases can modify much more*

Size of information - *LDAP is better at storing small bits of information*

Type of information - *LDAP stores information in attributes*

LDAP vs Databases

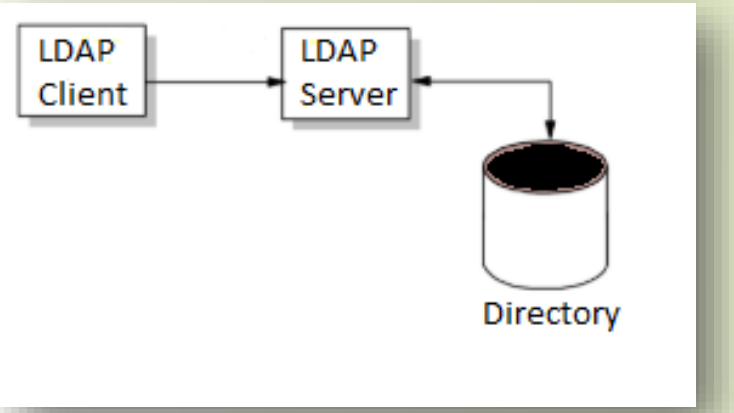
Naming model – *LDAP is hierarchical*

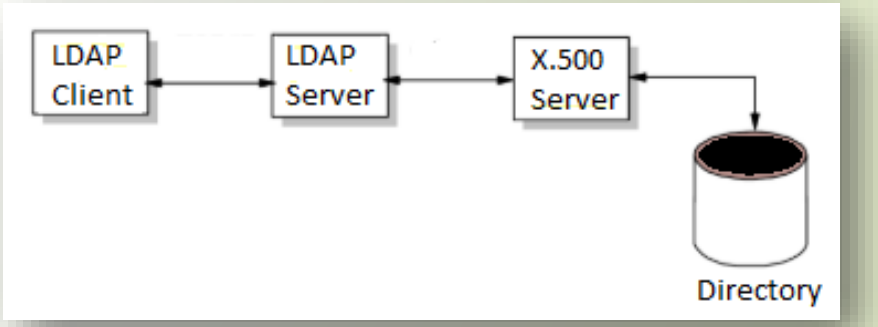
Schemas - *Database schemas are entirely user defined, directories have core schemas to help interoperability*

Standards are more important for directories – *LDAP clients can talk to any LDAP server, but database client can only talk to the database it was designed for.*

LDAP

- Two general types of directory servers:
  - *Stand alone LDAP servers*
    - LDAP is only access mechanism
    - Data stores tuned for LDAP
  - LDAP gateway servers
    - *Translate between LDAP and some other protocol*
    - *Original use of LDAP - gateway to X.500*
- Doesn't really matter where data is stored, as long as it is accessible via LDAP





Common LDAP Applications

- White pages
- Authentication and authorisation
- Personalisation
- Roaming profiles
- Public Key Infrastructure (PKI)
- Message delivery

LDAP Models

- Information Model
  - *Defines the types of data and basic units of info stored in directory*
- Naming Model
  - *Defines how to organise and refer to the data*
- Functional Model
  - *Defines operations in LDAP protocol*
    - Authentication (binding), Query (searches and reads), update (writes)
- Security Model
  - *Defines framework for protecting information*
    - Authentication methods
- LDAP models tutorial:

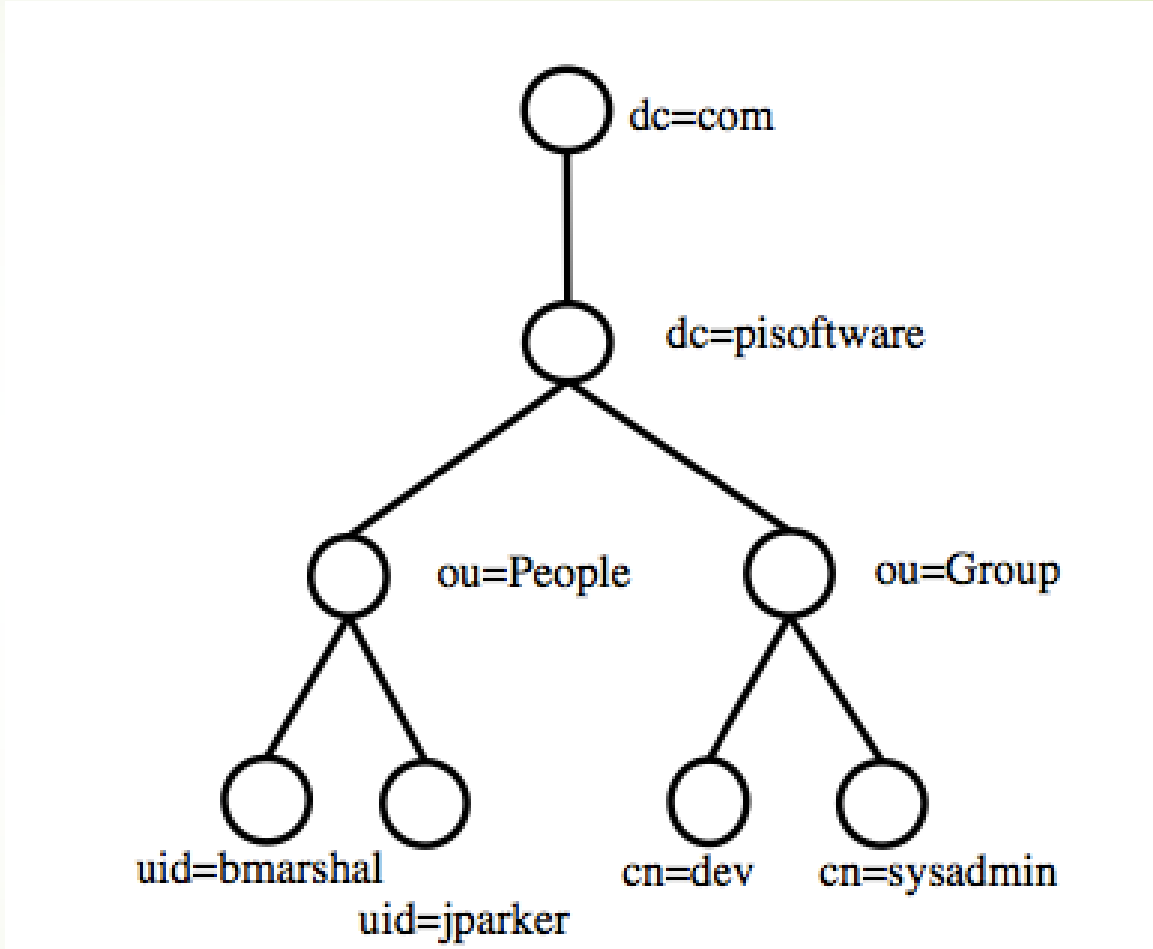
  http://etutorials.org/Server+Administration/ldap+system+administration/Part+I+LDAP+Basics/Chapter+1.+Now+where+did+I+put+that.+or+What+is+a+directory/1.3+LDAP+Models/

Terminology

- **DN**=Distinguish Name
- **RDN**=Relative Distinguished Name
- **DIT**=Directory Information Tree
- **LDIF**=LDAP Data Interchange Format
- **DSML**=Directory Services Markup Language
- **OID**=Object Identifier

Namespaces - Heirarchical

- Directory tree is similar to *nix file system
  - *Each entry in LDAP can both contain data and be a container*
  - *In *nix, an entry is either a file or a directory*
  - *LDAP distinguished names are read from bottom to top, unix file systems from top to bottom*



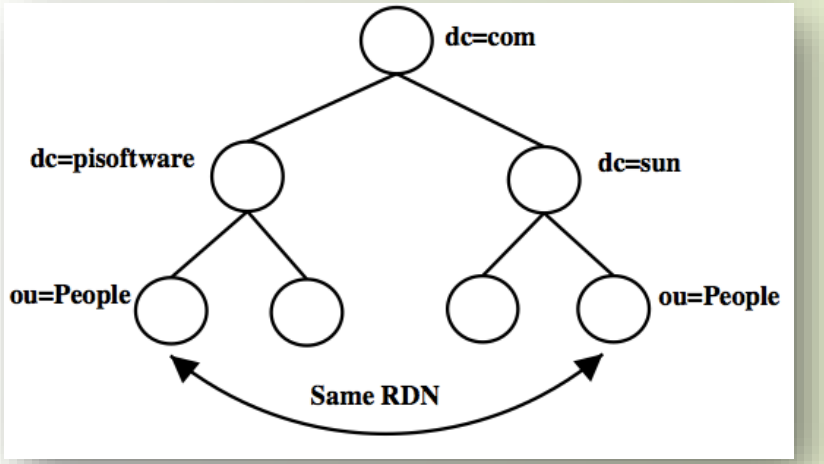
* dc may be C (country) or O (organisation)

Distinguished Names

- Defined in RFC2253 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names"
- Built up by starting at the bottom, and connecting each level together with commas
- Contain two parts - Left most part is called relative distinguished name
- Remainder is base distinguished name

Distinguished Names (cont.)

- E.g.: uid=bmarshal,ou=People,dc=pisoftware,dc=com
- RDN is uid=bmarshal
- DN is ou=People,dc=pisoftware,dc=com
- In each base DN, each RDN is unique
  - *This ensures no two entries have the same DN*



LDAP Entry

- Entries are composed of attributes
- Attributes consist of types with one or more values
- Type describes what the information is
- Value is the actual information in text format
- Attributes have a syntax which specifies what type of data - see Schema later on

Schema

- Set of rules that describes what kind of data is stored
- Helps maintain consistency and quality of data
- Reduces duplication of data
- Ensures applications have consistent interface to the data
- Object class attribute determines schema rules the entry must follow

Schema (cont.)

- Schema contains the following:
  - *Required attributes*
  - *Allowed attributes*
  - *How to compare attributes*
  - *Limit what the attributes can store – e.g. restrict to integer etc.*
  - *Restrict what information is stored - e.g. stops duplication etc.*

Objectclass

- Used to group information
- Provides the following rules:
  - *Required attributes*
  - *Allowed attributes*
  - *Easy way to retrieve groups of information*
- Entries can have multiple object classes
- Required and allowed attributes are the union of the attributes of each of the classes

Objectclass Inheritance

- Object classes can be derived from others
- Extends attributes of other objectclass
- No multiple inheritance
- Can't override any of the rules
- Special class called top - all classes extend
  - *Only required attribute is objectclass*
  - *Ensures all entries have a objectclass*

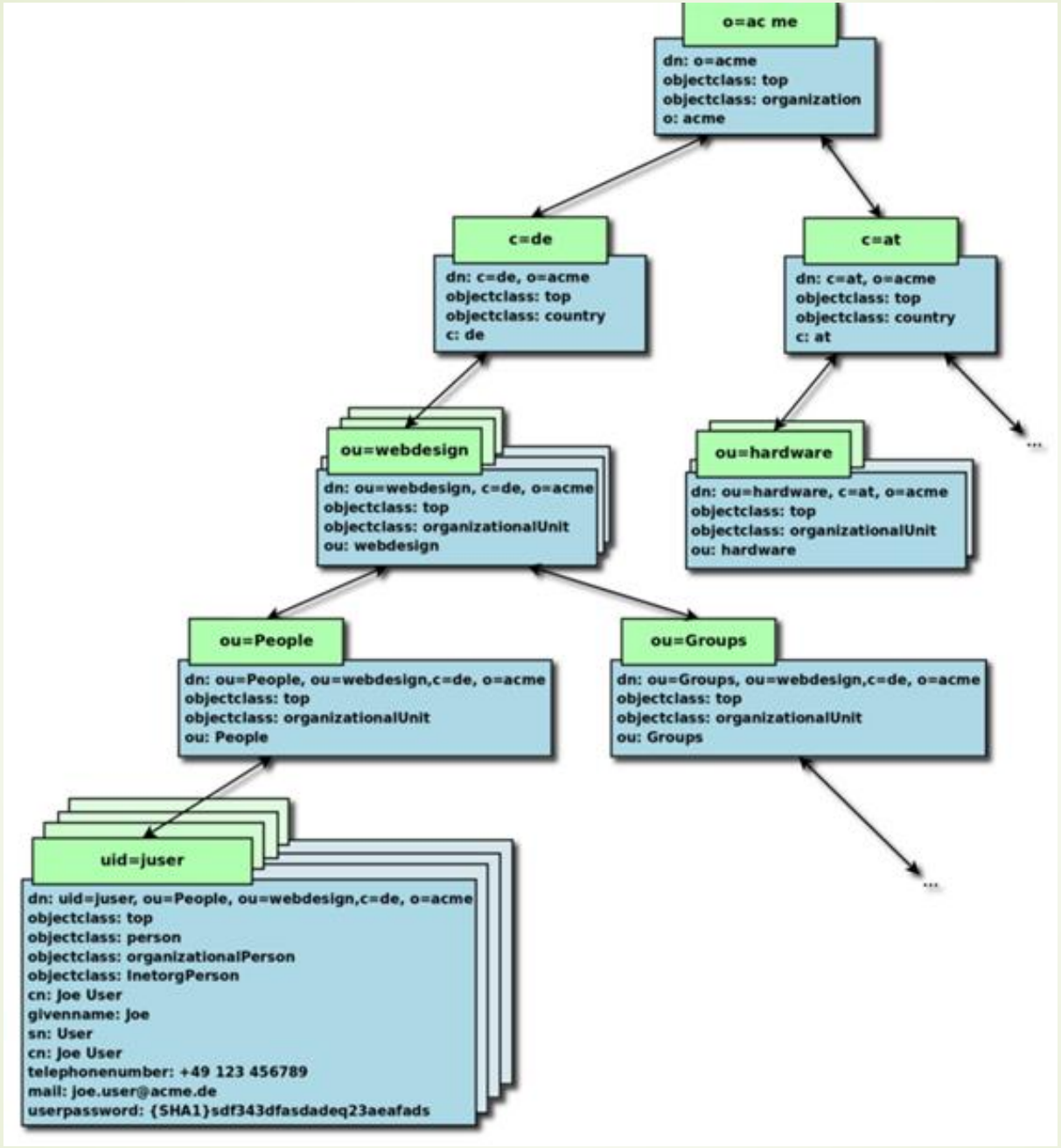
- https://ldap.com/object-classes/

Attributes

- Attributes have:
  - *Name - unique identifier, not case sensitive*
  - *Object identifier (OID) - sequence of integers separated by dots*
  - *Attribute syntax: Data attributes can store*
    - e.g. integer, string etc. How comparisons are made
  - *If multi valued or single valued*

Attribute Abbreviations

- See RFC2256
- **uid** =User id
- **cn**=Common Name
- **sn**=Surname
- **l**=Location
- **ou**=Organizational Unit
- **o**=Organization
- **dc**=Domain Component
- **st**=State
- **c**=Country



http://abhishekraw.blogspot.com/2013/07/lightweight-directory-access-protocol.html

Replication

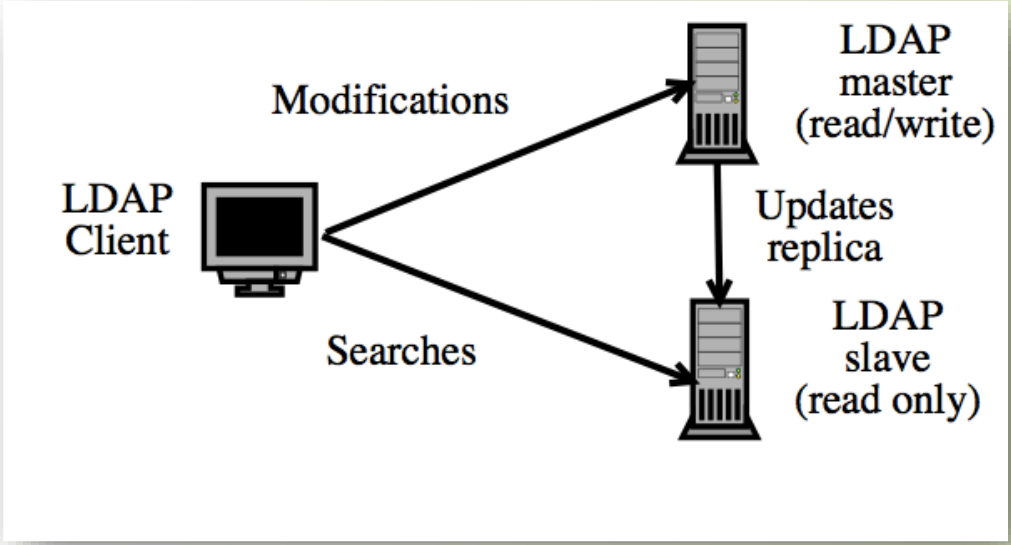
- Replication is method used to keep copy of directory data on multiple servers
- Increases:
  - *Reliability - if one copy of the directory is down*
  - *Availability - more likely to find an available server*
  - *Performance - can use a server closer to you*
  - *Speed - can take more queries as replicas are added*

Replication (cont.)

- Temporary inconsistencies are ok
- Having replicas close to clients is important - network going down is same as server going down
- Removes single point of failure

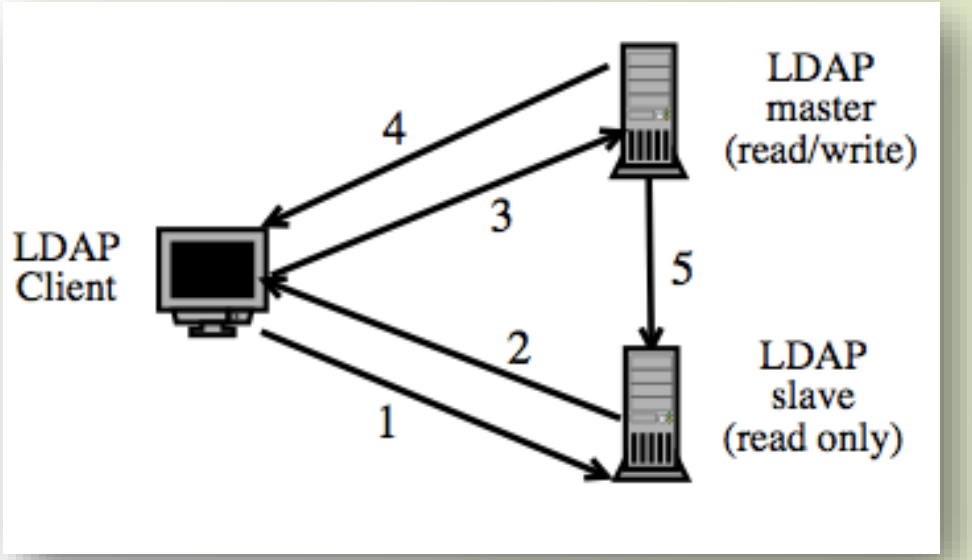
Replication options – Mods to Master

1. Client sends modification to the master
2. Master updates replica with change



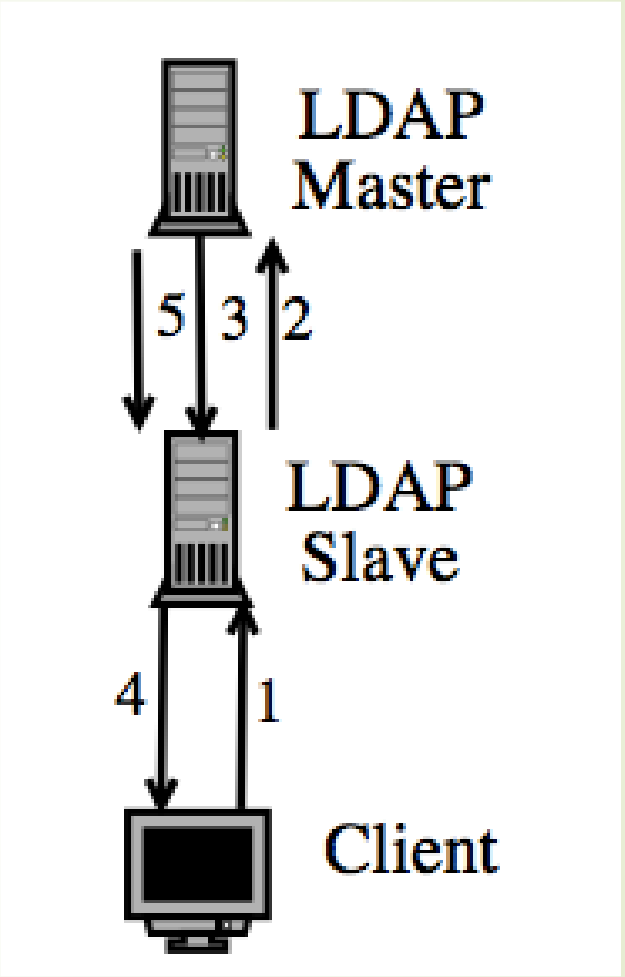
Replication Options - Referrals

1. Client sends modification to replica
2. Replica returns referral of master to client
3. Client resubmits modification to master
4. Master returns results to client
5. Master updates replica with change



Replications Options - Chaining

1. Client sends modification to replica
2. Replica forwards request to master
3. Master returns result to replica
4. Replica forwards result to client
5. Master updates replica



Replication – Single Master

- Single master
  - *Single server with read-write copy of data*
  - *Use read-only replicas to handle load of reading and searching*
  - *Uses referrals or chaining to deal with mods being sent to replicas*
  - *Obvious single point of failure*
  - *Simplest to implement*

Replication – Floating Master

- Floating master
  - *Only one server with writeable copy of data at any time*
  - *If master becomes unavailable, new master is chosen from remaining servers*
  - *Actual algorithm used depends on server – usually voting*

Replication - Cascading

- Cascading replication
  - *Master replicates to hub which replays updates to further servers*
  - *Helps balance very heavy loads - master can be used for all writes, and pass off all replication to hub*
  - *Can replicate out to local hubs in geographically dispersed topologies - reduces cost*
  - *Increase performance - put all reads on "bottom" servers*

Replication - Fractional

- Fractional replication
  - *Replicates subsets of attributes*
  - *Useful for synchronising from intranet to extranet and removing data for security reasons*
  - *Can help reduce replication costs by removing unneeded data*

Replication – Multi-master

- Multi-master
  - *More than one server with writable copy of data*
  - *Servers are responsible for ensuring data propagates*
  - *Need a conflict resolution policy - most current servers use a last writer wins policy*
  - *Can have automatic failover when one is unavailable*
  - *Most complex to implement*

LDIF

- LDAP Data Interchange Format
  - *Represents LDAP entries in text*
  - *Human readable format*
  - *Allows easy modification of data*
  - *Useful for doing bulk changes*
    - dump db, run a script over, import back
  - *Can use templates for additions*
  - *Good for backups and transferring data to another system*

LDIF (cont.)

- If attribute contains non-ascii chars, begins with a space, colon or less than, the value is stored in base64 and type separator is ::
- Can mix plain text and encoded values
- Can specify URLs, type separator is :<
- Entries in LDIF are separated with a blank line
- Continued lines start with spaces
- Utilities to convert from database to ldif and back
  - *slapcat: ldbm database to ldif*
  - *slapadd: ldif to ldbm database*
- Few non-LDAP related servers know how to deal with it

LDIF Example

dn: uid=bmarshal,ou=People, dc=pisoftware,dc=com

uid: bmarshal

cn: Brad Marshall

objectclass: account

objectclass: posixAccount

objectclass: top

loginshell: /bin/bash

uidnumber: 500

gidnumber: 120

homedirectory: /mnt/home/bmarshal

gecos: Brad Marshall,,,,

userpassword: {crypt}lDIOnUp17x2jc

DSML

- XML-based file format
  - *Can store both schema and data*
  - *Can use as generic import/export format for directory data*
  - *Many existing servers can deal with XML*
  - *More complex than LDIF*
  - *Not all LDAP servers support it*

DSML Example

```
<dsml:directory-entries>
  <dsml:directory-entry dn="uid=mball, ou=consultant,
o=sark.com,c=us">
    <dsml:objectclass>
      <dsml:oc-value>top</dsml:oc-value>
      <dsml:oc-value>person</dsml:oc-value>
      <dsml:oc-value>organizationalPerson</dsml:oc-value>
      <dsml:oc-value>inetOrgPerson</dsml:oc-value>
    </dsml:objectclass>
    <dsml:attr
name="sn"><dsml:value>Ball</dsml:value></dsml:attr>
    <dsml:attr
name="uid"><dsml:value>mball</dsml:value></dsml:attr>
    <dsml:attr
name="mail"><dsml:value>mball@sark.com</dsml:value></dsml:attr>
    <dsml:attr
name="givenname"><dsml:value>Michael</dsml:value></dsml:attr>
    <dsml:attr name="cn"><dsml:value>Michael
Ball</dsml:value></dsml:attr>
  </dsml:entry>
</dsml:directory-entries>
```

https://www.infoworld.com/article/2076231/dsml-gives-you-the-power-to-access-your-ldap-information-as-xml.html

Search Filters

- Consists of:
  - *Base and scope - what part to search*
  - *Filter - what criteria attributes must fulfil to be returned*
  - *Attributes - what attributes to return*
- Prefix notation
- Standards
  - *RFC 1960: LDAP String Representation of Search Filters*
  - *RFC 2254: LDAPv3 Search Filters*

Search Filter Operators

- Basic filter types
  - *Presence* – (objectClass=*)
  - *Equality* – (objectClass=person)
  - *Substring* - (name=*curtincollege*)
  - *Greater-than or equal* – (msDS-LockoutDuration>=3)
  - *Less-than or equal* – (msDS-LockoutDuration<=3)
  - *Approximate* – (givenName~=Bobb)
  - *Extensible*
- Joining filters
  - *& and* - (&(attribute1=X)(attribute2=Y))
  - *| or* – (|(attribute1=X)(attribute2=Y))
  - *! Not* – (! (attribute1=X)(attribute2=Y))

https://ldap.com/ldap-filters/

Search Filter Details

- Presence
  - *Simple checks for presence of attribute*
  - *Can return everything by using (objectClass=*)*
- Equality
  - *Checks for entries containing given attribute and value*
  - *Most commonly used search*
  - *Fastest and easiest to index*
  - *Case sensitivity depends on attribute syntax*

Search Filter Details (cont.)

- Substring matching
  - *Returns entries with values matching given substrings*
  - *Useful for finding information, eg in white pages etc*
  - *Not wildcard or regular expression matching*
  - *Restricted to matching start, middle or end of strings*
  - *Slower than presence or equality*
  - *Middle substring matches slower than start or end*

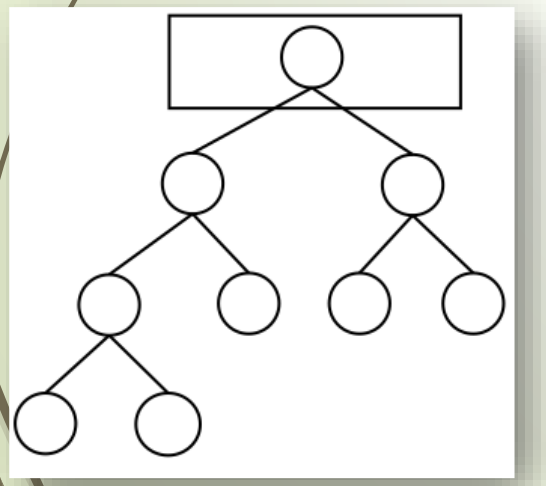
Search Filter Details (cont.)

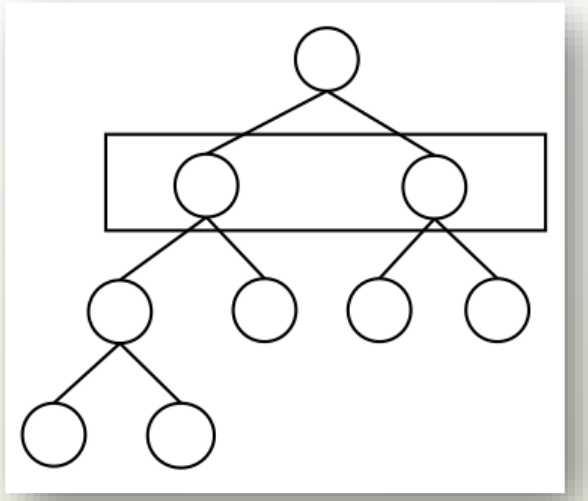
- Ordered matching (greater-than, less-than)
  - *Order determined by attributes syntax and matching rules*
- Approximate filter
  - *Algorithm used determined by server*
  - *Common to have soundex*[1]
- Extensible
  - *Allows applying an explicit matching rule to a search filter*
  - *Not very common, but powerful*

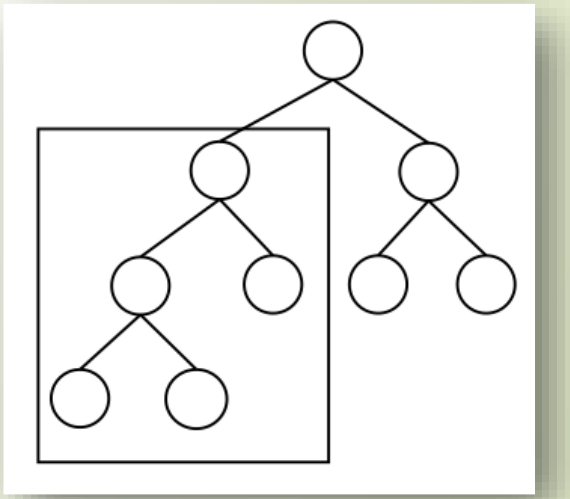
[1] Soundex is a phonetic algorithm for indexing names by sound

Search Scope

- 3 types of scope:
  - *Base = limits to just the base object*
  - *Onelevel = limits to just the immediate children*
  - *Sub = search the entire subtree from base down*







LDAP Protocol

- Uses client server model
- Message oriented protocol - client sends messages to server and gets replies
- Can issue multiple requests at once - each response has message id to identify
- 9 basic protocol operations - interrogation, update and authentication
- LDAPv3 provides extended operations and controls
- Uses simplified version of Basic Encoding Rules (BER) – not plain text

LDAP Protocol Operations

- Interrogation operations
  - *search*
  - *compare*
- Update operations
  - *add*
  - *delete*
  - *modify*
  - *rename*
- Authentication and control operations
  - *bind*
  - *unbind*
  - *abandon*

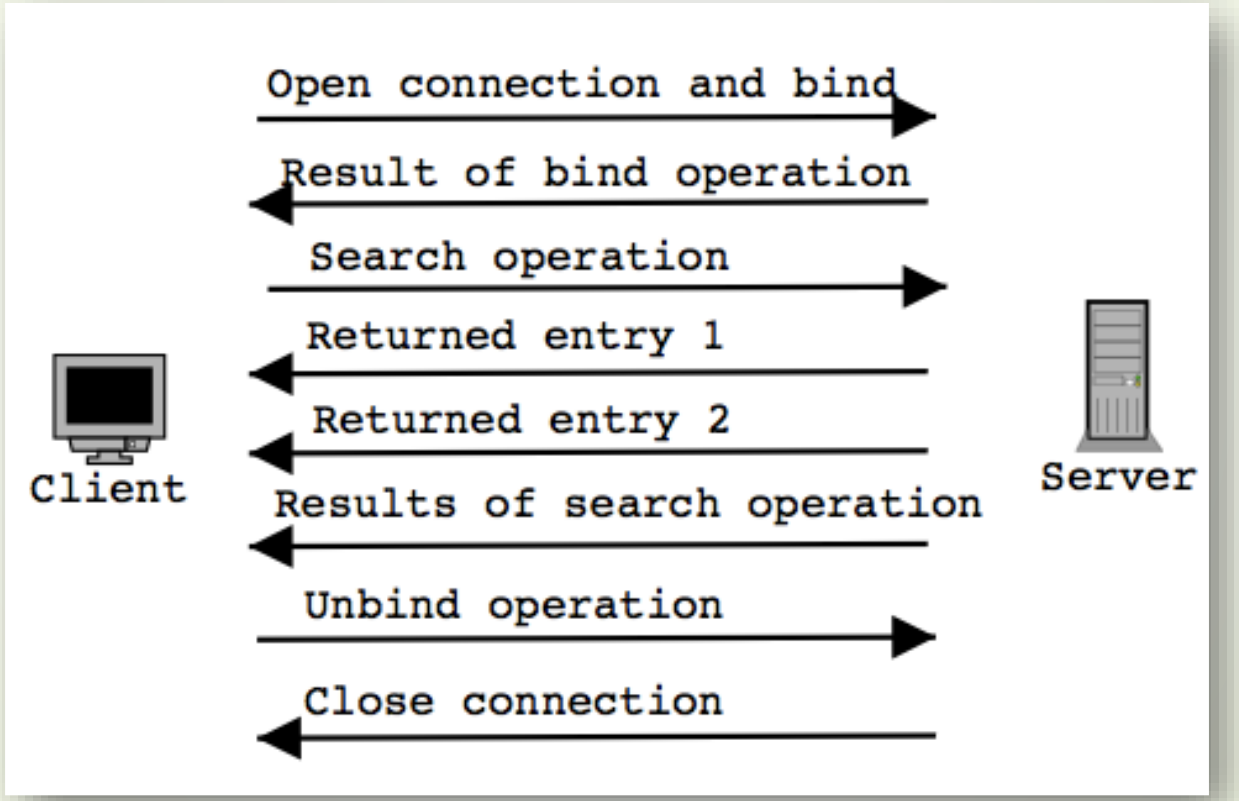
Bind Operations

- How client authenticates to directory
- Opens TCP connection, gives distinguished name and credentials (Port 389 by default)
- Server checks credentials are correct, returns result to client
- If credentials fail, returns an anonymous bind
- Authentication lasts while connection is open, or until client re-authenticates
- Credentials can be password, digital certificate or other

Authentication Levels

- There are 3 basic levels of authentication:
  - *Anonymous - no username and password*
    - Enabled by default, disable using “disallow bind_anon” in slapd.conf
  - *Unauthenticated - only username (sometimes called reference bind)*
    - Disabled by default, enable using “allow bind_anon_cred” in slapd.conf
  - *Authenticated - username and correct password*
    - Enabled by default, disable using “disallow bind_anon_simple” in slapd.conf

Typical LDAP Conversation



Conclusions

- LDAP is simple, yet versatile
  - *LDAP supports a wide variety of directory-enabled applications that have widely varying needs.*
- LDAP is ubiquitous
  - *A good implementation of LDAP was developed and distributed freely on the Internet by researchers at the University of Michigan.*
  - *LDAP implementations are available for every major and most minor computing platforms that are in use.*

Conclusions (cont.)

- LDAP directories are inexpensive and easy to understand
  - *Organisations that choose LDAP directories find them to be relatively inexpensive to deploy and maintain.*
  - *LDAP directory systems are simple to understand and deploy.*
- LDAP directory services simply work better
  - *The high reliability, performance, and scalability of LDAP directory products, combined with their general-purpose design, allows them to meet the most important directory services needs.*
- LDAP has a lot of "mindshare"

RFCs

- RFC 2251: LDAPv3 Protocol
  - *Describes the LDAP protocol itself*
- RFC 2252: LDAPv3 Attribute Syntax Definitions
  - *Defines attribute syntaxes used by LDAP and its applications, including how these values are represented as simple strings*
- RFC 2253: LDAPv3 UTF-8 String Representation of Distinguished Names
  - *Describes how distinguished names are represented as simple strings using the UTF-8 character set*
- RFC 2254:The String Representation of LDAP Search Filters
  - *Defines a scheme for representing LDAP search filters (which may be complex Boolean expressions involving multiple terms) as simple strings for use in LDAP URLs and APIs*
- RFC 2255: The LDAP URL Format
  - *Defines a uniform resource locator (URL) scheme to represent LDAP search requests*
- RFC 2256: A Summary of the X.500(96) User Schema for Use with LDAPv3
  - *Provides a list of useful directory schemas (attributes and object classes) that are pulled from the X.500 specifications*