

Computer System Design and Administration

Topic 2. Active directory secure service: LDAP (over SSL)



José Ángel Herrero Velasco

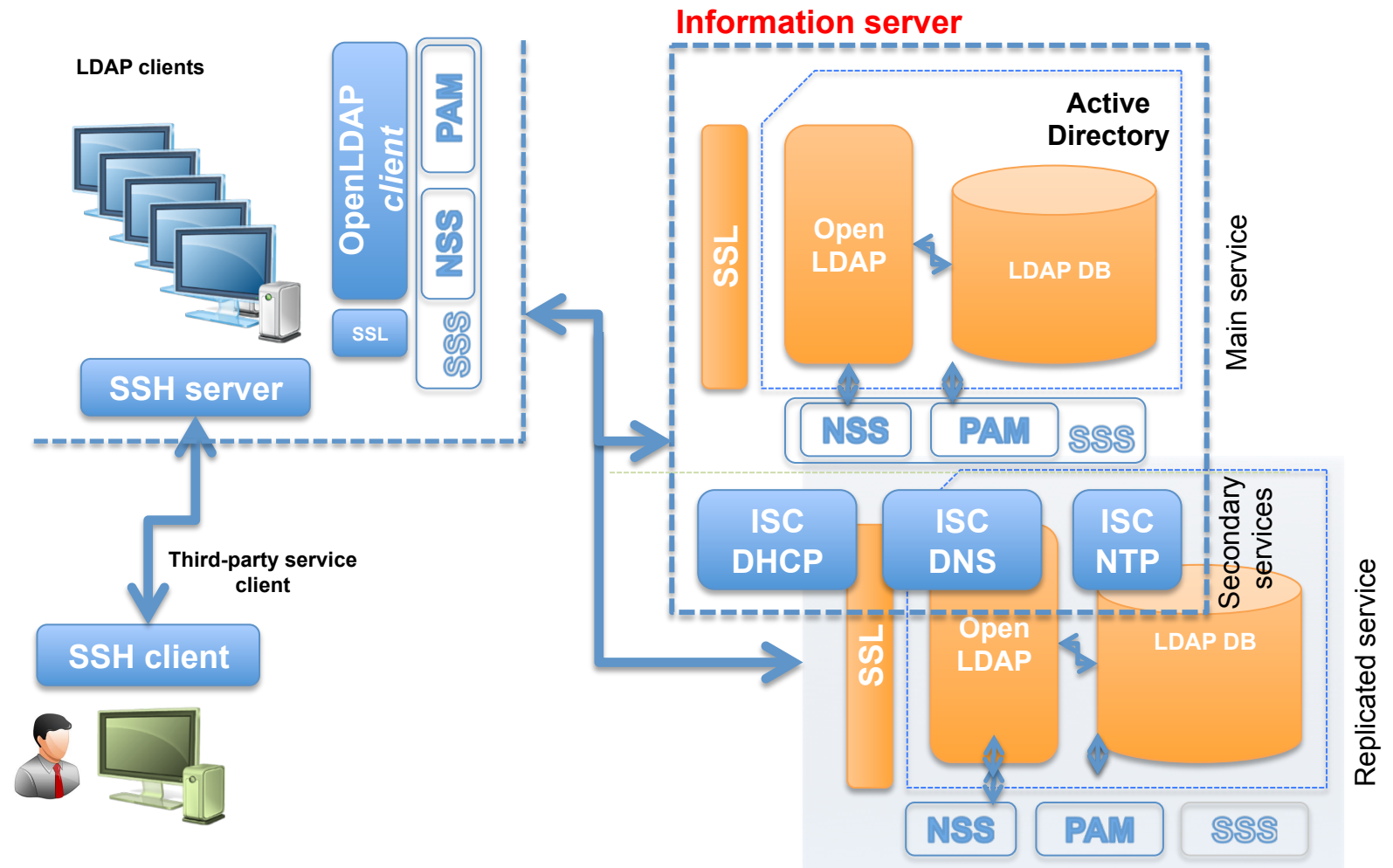
Department of Computer and
Electrical Engineering

This work is published under a License:

[Creative Commons BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Secure information service: Puzzle

“Single sign-on” model



Target: Building the “Single sign-on” core

- Implementation and development of a **secure** and **centralized** system for the management of ***account and computational information*** in an enterprise (corporative) environment, using **LDAP** protocol:

- **SSO components:**

1. **Centralized Active Directory** store:
 - **OpenLDAP.**
2. **Tools** for managing the information in the directory:
 - **LDAP-utils**, phpLDAPadmin...
3. A **mechanism** for authenticating user identities:
 - **OpenLDAP (itself)**, Kerberos.
4. Centralized **identity** and **authentication** aware versions of C-library routines:
 - **INTEGRATION: NSS/PAM** (SSSd).

- **TLS/SSL security:**

- TLS/SSL encrypted communications.

**Secure
Single sign-on
(VALIDATION)**



*Identification
+
authentication*

- Any valid user in the organization can log in any system with the same credentials.

How to manage the computational information of a corporative environment?

- A **directory service** is just a “**database**” used by an enterprise environment to manage *centrally* their huge amounts of computational data:

- It is (such services) **distinguished** by having:
 - Data object relatively **small**.
 - Information is **attributed-based**.
 - High levels of **read accesses**:
 - **Searching is a common operation.**
 - Low **volatility**:
 - **Storage information which suffers few changes.**
 - **Updates are limited to owners and *admins*.**

You can understand it as a **specialized database.**

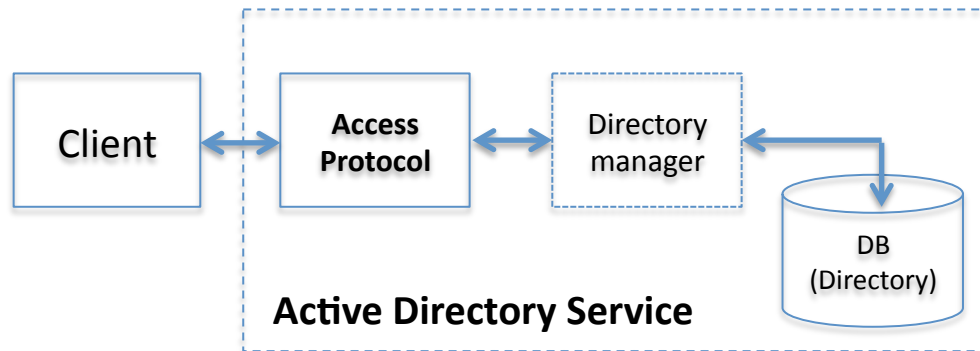
NO transactional, NO SQL support.

- It is **defined** as:
 - Hierarchical collection of **objects** and **attributes** arranged in a particular way:
 - **Sets what information is stored and how it should be organized.**
 - **Allows locating information easily and quickly.**

- It is **composed** by:
 - **Front-end**: Access protocol.
 - **Back-end**: Directory manager:
 - **(Specialized database).**

- It **implements** a:
 - **Server-client** service.

- In real life...:
 - Phone book, library catalog.



LDAP: Directory service

- **LDAP** → **L**ightweight **D**irectory **A**ccess **P**rotocol:
 - **Open, standard and cross-platform protocol** designed to provide a “*lightweight*” access to distributed directory information on TCP/IP networks:
 - Originally:
 - **Developed by the University of Michigan, in 1993.**
 - **Based on DAP protocol (Access protocol of X.500):**
 - » Designed for allowing **TCP/IP clients** access to **X.500 active directory service.**
 - » Initially, it replaced DAP protocol (Directory Access Protocol) in X.500 as *front-end* of the service.
 - Nowadays:
 - **Provides a full directory service → LDAP is anything but lightweight:**
 - » Linux implementation: **OpenLDAP.**
 - » Microsoft’s Active Directory.
 - **For many systems and applications:**
 - » Mail/Web servers.
 - **Key:**
 - “... Write once, read many times...”.
 - **Main features:**
 - **Read-write ratio:** reads optimized.
 - **Extensibility:** LDAP schemas.
 - **Distribution:** with LDAP data can be near where it is needed.
 - **Replication:** with LDAP data can be stored in multiple locations.

LDAP: Directory service

– Other features:

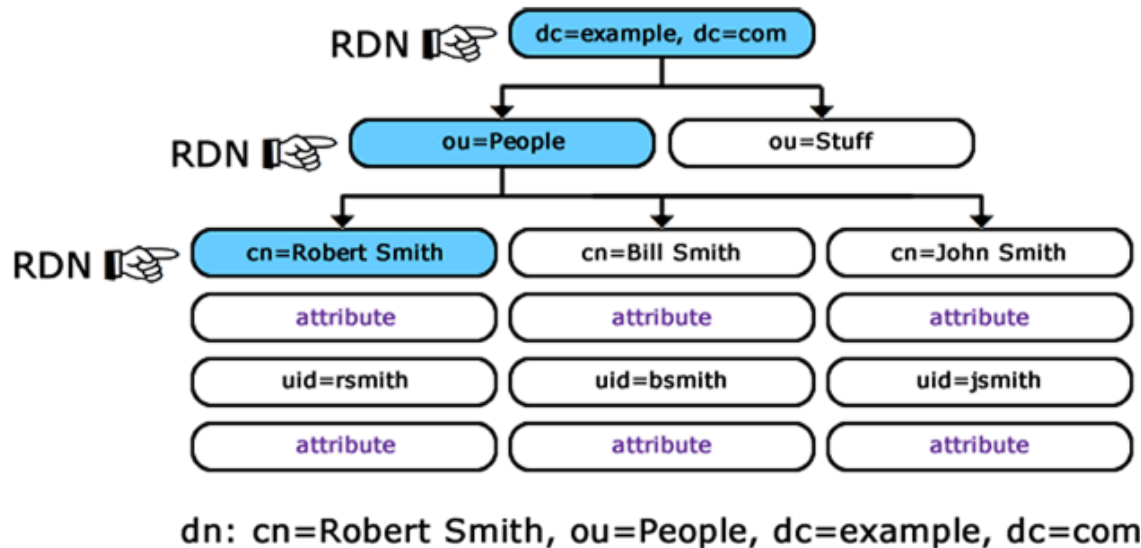
- Use **TCP/IP** protocols (**application layer**) instead of **OSI**.
- It is a stand-alone service:
 - **389/636 ports.**
- It supports secure communications (encrypted):
 - **SSL/TLS.**
- Nowadays, version 3 of the protocol (LDAPv3):
 - **RFC 2251 y RFC 2256 (doc. Base), RFC 2829 (auth), RFC 2830 (SSL/TLS)...**
- **Open standard:**
 - **Many implementations.**
 - **OpenLDAP:**
 - » Developed by GNU “opensource”: **GPL.**

– It's based on 4 models:

- **Information model:**
 - **Structure of information stored in an LDAP directory.**
 - **LDAP defines the content of messages exchanged between a LDAP client and server.**
- **Naming model:**
 - **How information is identified and organized.**
- **Functional model:**
 - **It describes what operations can be performed on the information stored in LDAP directory.**
- **Security model:**
 - **It describes how the information can be protected from unauthorized access.**

LDAP: Data model

- Hierarchical structure (tree): Directory Information Tree
 - Directory with a **tree structure (DIT)**.



Source: <https://docs.typo3.org>.

- The DIT (tree) can be *geographically* distributed on many servers:
 - Distribution (“main feature”).

LDAP: Data model

- Every *branch* (leaf) of the tree (DIT) composes a **LDAP entry**:
 - They represent **objects** from real life.
 - It is the minimal information unit for LDAP.

- Every *entry* →

- **Unique ID**

(*Distinguished Name, DN*):

- It establishes the **search path** to the data (sequence of **RDNs**):

- **dn: unique=3,dc=People,dc=ds,dc=example,dc=org.**

- **Attributes:**

- They include information of the entry (object):
 - **cn, ou, objetClass, etc.**

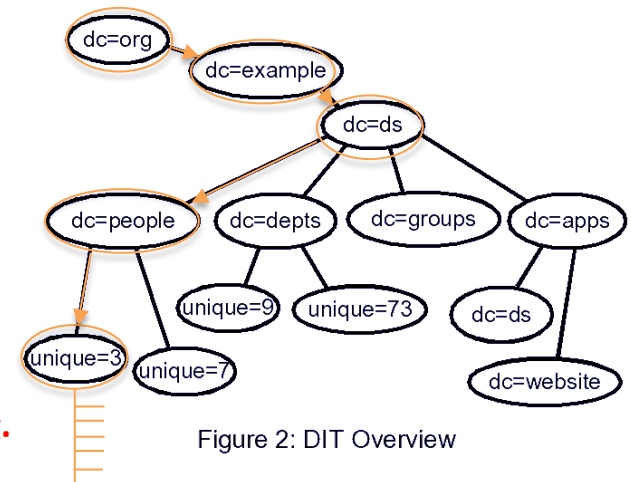


Figure 2: DIT Overview

LDAP: Data model

- Every *attribute* includes:

- Name (type).
- Value(s):
 - Multiples values.

- Attribute types:

- **Data attributes:**

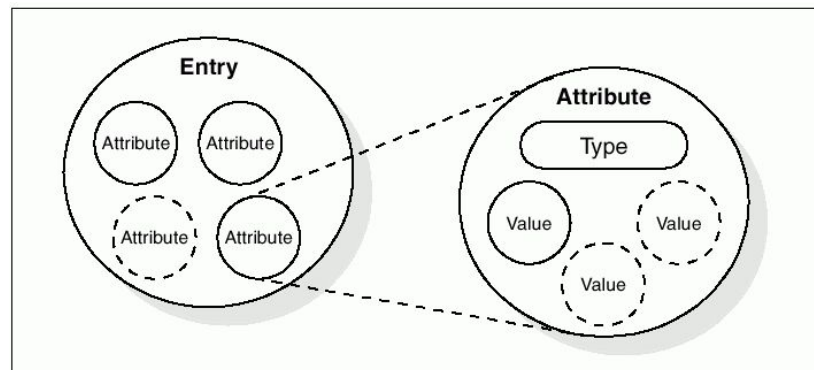
- They contain data from the *entries*:
 - **UID, CN (name), SN (surname), OU, etc.**

- **Operative attributes** (*slapcat*):

- ... Or *meta-attributes*.
- Server has only access to:
 - **Modification dates.**

- **LDIF:**

- LDAP Data Interchange Format.



Source: <http://www.redbooks.ibm.com>.

- **objectClass.**
- **dc** (domain component).
- **uid** (username).
- **cn** (common name).
- **st** (nombre del estado).
- **sn** (surname).
- **o** (organisation name).
- **ou** (organizational unit).
- ...

For example:

```
dn: cn=Jose A.,dc=ce,dc=unican,dc=es
objectClass: person
uid=jherrero
cn=Jose A.
uidNumber: 2001
...
```

LDAP: Data model

- Attributes are well-defined in the *schema files*:
 - Notation (**syntax**).
 - Meaning (**semantics**).
 - **Dependance relationships**, heritage...
- *The schemas* define the rules concerning what **objects** can be storage into a DIT:
 - **ObjectClass** attribute:
 - Specifies what attributes **an entry** can contain:
 - **List of attributes for every object.**
 - They establish where in a DIT a certain object can appear.
- *Schema-checking*:
 - Ensures that the relationships among attributes are correct according to the schemas before adding a new entry.

LDAP: Naming model

- It defines how entries are **identified** and **organized**:
 - *Tree-like structure* called the Directory Information Tree (**DIT**).
 - Entries are arranged within the DIT based on their *distinguished name* (**DN**) → RDNs.
 - They are used as *primary keys* of entries in the directory:
- dn: cn=Jose A., dc=ce, dc=unican, dc=es
- RDN
RDN
RDN
RDN
- The organization of the entries in the DIT are restricted by their corresponding **objectclass** definitions:
 - According to the *schemas*.
 - The DNs are an important key for LDAP client **requests**.

LDAP: Operational model

- **Methods:**

- LDAP provides to users methods to:

- Connect and disconnect to LDAP DB (TCP/IP model).
 - Search information.
 - Compare information.
 - Add new entries.
 - Modify entries.
 - Remove entries.

} LDAP
protocol

- **Operations (functions):**

- ... Which carry on requests to...:

- *Search, modify and remove* entries.

- Most relevant:

- **Abandon** (Abandonar): cancel a operation previously sent to the server.
 - **Add** (Agregar): Add a new entry to directory.
 - **Bind** (Enlazar): Create a new session on LDAP server (TCP/IP model).
 - **Compare** (Comparar): Compare entries in a directory by criteria.
 - **Delete** (Eliminar): Remove an entry from directory.
 - **Extended** (Extendido): Carry out extended operations.
 - **Rename** (Cambiar nombre): Rename an entry from directory.
 - **Search** (Buscar): Search an entry by criteria.
 - **Unbind** (Desenlazar): Close a session on LDAP server (TCP/IP model).

} OpenLDAP
tools

LDAP: Security model

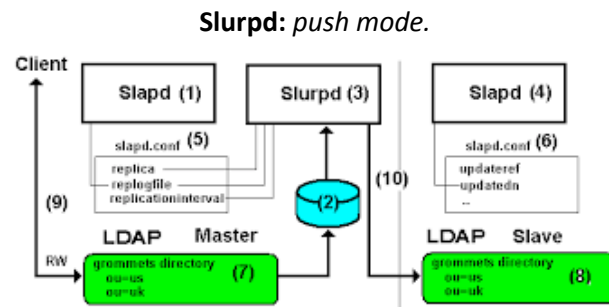
- **Access control:**
 - It defines the mechanisms to assure that:
 - **Access to LDAP information** is restricted → **control access list.**
 - LDAP client and server communications are safe.
- **Authentication:**
 - Assurance that the opposite party (machine or person) really is who he/she/it claims to be.
- **Integrity:**
 - Assurance that the information that arrives is really the same as what was sent:
 - Messages exchanged.
- **Confidentiality.**
 - Protection of information disclosure by means of data encryption to those who are not intended to receive it.

openLDAP: a LDAP protocol deployment

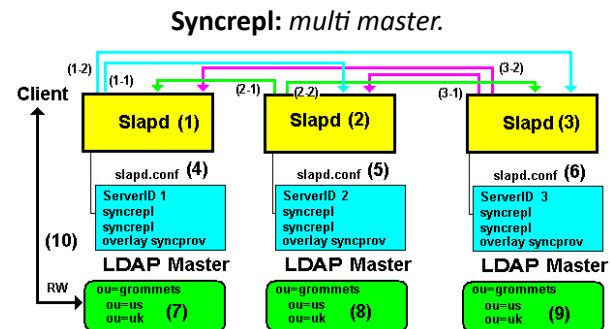
- **openLDAP** is a LDAPv3 protocol implementation for GNU:
 - Developed and maintained by “**The OpenLDAP project**”.
 - Opensource → OpenLDAP public license:
 - <http://www.openldap.org/software/release/license.htm>.
 - It supports:
 - **SSL/TLS security.**
 - **Replication.**
 - Authentication integration frameworks supports → SASL/GSSAPI.
 - Third-party authentication mechanisms → *kerberos 5*.
 - *Password algorithms* → *Crypt, MD5* and *SHA*.
 - *Backend systems* → *LDBM* y *DB2*.
 - Multi-Platform support → Linux, UNIX (AIX, Solaris, BSD...), MS Windows.
 - APIs to C, C++, PHP, Python...
 - <http://www.openldap.org>.
- Others:
 - **389 Directory server** (www.port389.org):
 - Superior documentation.
 - Open source too!

openLDAP: Daemons involved

- OpenLDAP runs as an OS **stand-alone service**.
- The suite includes:
 - **slapd**:
 - **Listens** to clients' requests to the LDAP DB.
 - **Performs** operations on the LDAP DB.
 - **Sends** results to clients.
 - Manages the LDAP DB **replication**.
 - **Libraries** implementing the LDAP protocol.
 - **Utilities, tools...**
- Replication service:
 - Adds **high ability** to the LDAP service.
 - Keeps the secondary (es) LDAP DB **fully updated**.
 - Up to 2.4 → "old style":
 - **Slurpd** daemon.
 - Only *push mode*:
 - **The master node pushed changes to the slaves.**
 - **Actually** → "new style":
 - **Syncrepl** replication.
 - Multi-master capabilities:
 - **Active (live) synchronization.**



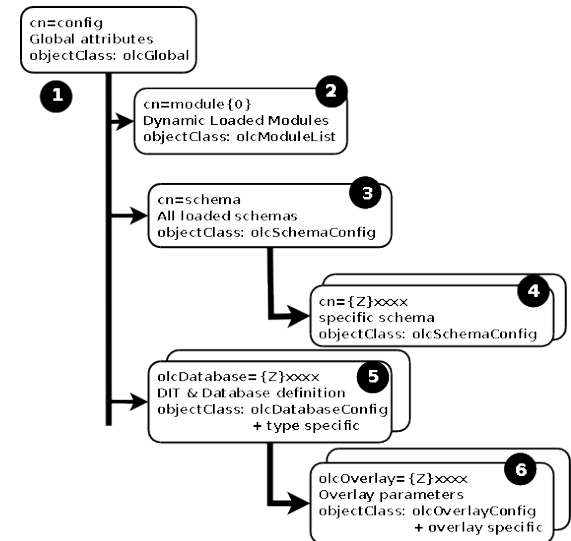
Source: www.zytrax.com.



Source: www.zytrax.com.

openLDAP: OLC configuration (cn=config)

- Up to 2.3 → “old style”:
 - “Main single file” of base configuration:
 - /etc/ldap/slapd.conf.
 - Other files:
 - schema files, module files, includes...
- From 2.3 to 2.4:
 - A new service configuration mechanism appears:
 - Henceforth, there **will not be** a single main configuration file.
 - Both configuration methods can be used:
 - You can even use a *conversion* method: slapd.conf → slapd.d/.
- Actually → “new style” OLC: “On Line Configuration”:
 - It is not necessary restart service.
 - Service configuration is stored in a DIT:
 - **cn=config**.
 - Located in a system directory.
 - /etc/ldap/slapd.d (Initialization LDIF files).
 - Any change must be done through **LDIF files**, using:
 - **LDAP client tools:**
 - **ldapmodify, ldapadd, ldapsearch...**



openLDAP: LDIF files

- They are used to exchange information with LDAP directory:
 - OLC configuration and Corporate DITs.

- **LDIF: LDAP Data Interchange Format:**

- They allow **importing and exporting** data to/from a LDAP directory using a text file:
 - ... And LDAP operations:
 - **OpenLDAP “tools”.**
- They allow **adding and removing** information to/from a LDAP directory:
 - Example:

```
dn: uid=ruizsr,ou=People,dc=localdomain
sn: Ricardo Ruiz
uid: ruizsr
cn: Ricardo Ruiz
givenName: ruizsr
uidNumber: 9034
gidNumber: 90
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: krb5Principal
objectClass: shadowAccount
homeDirectory: /afs/atc.unican.es/u/r/ruizsr
userPassword: {KERBEROS}ruizsr@atc.unican.es
shadowLastChange: 13684
shadowMin: 1
shadowMax: 3650
shadowWarning: 10
shadowInactive: 10
shadowExpire: -1
shadowFlag: 0
gecos: ruizsr@unican.es,26772,F. CIENCIAS
loginShell: /bin/bash
krb5PrincipalName: ruizsr@ATC.UNICAN.ES
```

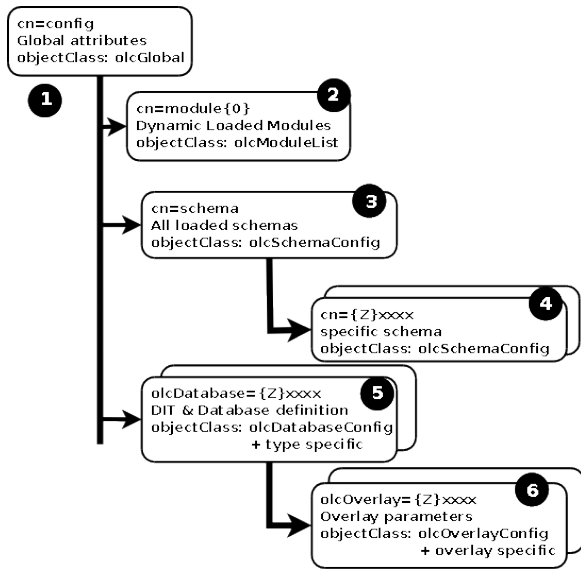
```
$ ldapadd -x -D "cn=admin,dc=localdomain" -W -f example.ldif
```

openLDAP: Where are the data???

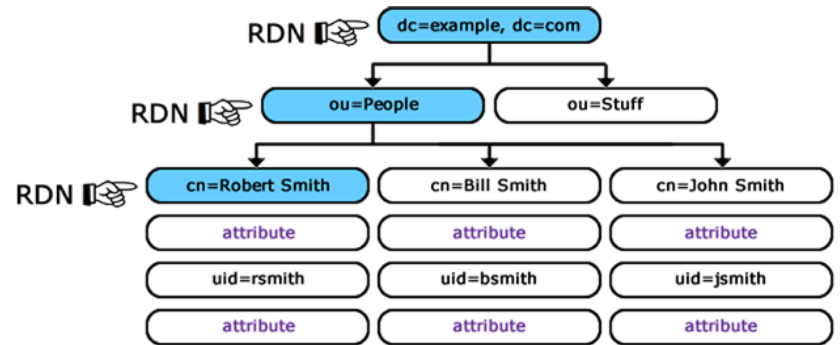
- OpenLDAP keeps 2 DITs (at least):

– OLC DIT and Backend DITs:

DN → cn=config



DN → dc=example,dc=com



dn: cn=Robert Smith, ou=People, dc=example, dc=com

The BACKEND

– /etc/ldap/slapd.d:

- LDIF files hierarchy.

– /var/lib/ldap

- Usually, HBD backend
 - **Oracle Berkeley DB.**

openLDAP: Commands & tools

- **Server tools:**
 - **Most significant commands:**
 - **slapadd:** Adds entries from an *LDIF* file.
 - **slapcat:** Gets information (entries) from LDAP directory (*LDIF format*).
 - **slapindex:** Re-indexes the LDAP directory.
 - **slappasswd:** Creates a new password for LDAP (console).
 - **Considerations:**
 - These commands access the ldap folder: `/var/lib/ldap:`
 - **You can not run them from other (remote) hosts.**
 - It's important that the ldap service is stopped.
- **Client tools:**
 - **Most significant commands:**
 - **ldapadd:** Adds entries from an *LDIF* file.
 - **ldapmodify:** Modifies entries from an *LDIF* file.
 - **ldapdelete:** Deletes entries from LDAP directory.
 - **ldapsearch:** Searches information according to filters.
 - **ldappasswd:** Changes the password attribute from a DIT entry.
 - **Considerations:**
 - That tools are installed from a third-party package.
 - To use them, **the ldap service must be in operation:**
 - They access the ldap directory through the ldap service.
 - You can run them from other (remote) hosts.

openLDAP: Server & client side installation

- From debian **repositories**:
- **OpenLDAP**:
 - Installation of libraries and tools (clients/server).

```
servidor {  
  $ apt-get install slapd ldap-utils  
  $ dpkg-reconfigure slapd (opcional)
```

```
cliente {  
  $ apt-get install ldap-utils libpam-ldap  
  libnss-ldap nscd
```

openLDAP: Service configuration

- From 2.4.23, the LDAP **service** configuration is changed → “*new style*”:
 - **OLC configuration: DIT** → `cn=config`
\$ `/etc/ldap/slapd.d`
- It contains the same *elements* and *features* as “*old style*”.
- But now...:
 - **Do not need to restart the LDAP service:**
 - “**On the fly**”.
 - Through LDIF files + client tools.

openLDAP: Service configuration

- “new style” basic procedures:

- Search of configurations:

```
$ ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config"
```

- Modification (added) of configuration:

```
$ cat <file.ldif>
```

```
dn: olcDatabase={1}hdb,cn=config
```

```
changetype: modify
```

```
add: olcDbIndex
```

```
olcDbIndex: cn pres,sub,eq
```

```
$ ldapmodify -Y EXTERNAL -H ldapi:/// -f <file.ldif>
```

openLDAP: Daemon configuration

- **Server** daemon configuration:

- This file sets the running parameters of the LDAP daemon:

```
$ vi /etc/default/slapd
```

- LDAP user and group:

```
SLAPD_USER="openldap", SLAPD_GROUP="openldap"
```

- Protocol version (type), server hostname and TCP ports:

- `ldap://.../` → service instance for LDAP over TCP (389 port).

- No security.

- `ldaps://.../` → service instance for LDAP over TCP (636 port).

- SSL/TLS security.

- `ldapi://.../` → service instance for LDAP over IPC (*Unix-domain socket*) for service **maintenance tasks**.

- Local scope:

```
SLAPD_SERVICES="ldap://server-01.localdomain:389/ ldaps:/// ldapi://"
```

- Additional parameters:

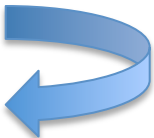
- *Debug modes...*:

```
SLAPD_OPTIONS="-g ..."
```

openLDAP: Client side configuration

- **Client** side configuration (elements most relevant):
 - Main file of **LDAP client side**:
 - `$ vi /etc/ldap/ldap.conf`
 - **Name Service Switch (NSS)** LDAP configuration files:
 - Needed for **identification** of OS entities (users, groups...) managed by LDAP directory.
 - `$ vi /etc/libnss-ldap.conf**`
 - **Pluggable Authentication Modules (PAM)** configuration files:
 - Needed for **authentication** of PAM clients/apps managed by LDAP directory.
 - `sshd`
 - `$ vi /etc/pam_ldap.conf**`
 - `$ vi /etc/pam.d/sshd`
- ** Both files maintain an identical configuration.
- **Name Service Switch (NSS)** main configuration file:
 - It sets the identification methods and in what order they will be used
 - Identification of users, machines, services, apps...
 - `$ vi /etc/nsswitch.conf`

openLDAP: Secure communications

- OpenLDAP supports secure communications in client-server transactions:
 - Using SSL/TLS layer.
 - **Protocols:**
 - **SSL: Secure Sockets Layer:**
 - Is part of the Transport and Session Layer (OSI).
 - **TLS: Transport Layer Security:**
 - SSLv3 is the predecessor of TLS.
- 
- TLS: SSLv3 Update
- **SSL/TLS** are cryptographic protocols that provide communication security over a computer network (TCP/IP):
 - **Symmetric and Asymmetric crypt:**
 - Size keys up to 256 bits (symmetric) and 4096 bits (asymmetric).
 - Originally developed (SSL) by Netscape (Mastercard, Bank of America, MCI y Silicon Graphic) in the 1990s.
 - Clients-server communications.
 - **TLS** aims primarily to provide **authentication, privacy and data integrity** between two communicating computer applications:
 - Client and server communication has the following properties:
 - *Privacy* → to encrypt the data transmitted (**symmetric crypt**).
 - *Authenticity* → to authenticate the ends (**asymmetric crypt**).
 - *Integrity* → message integrity check (**message authentication code**).
 - ... To prevent *eavesdropping* and *tampering*.
 - Most famous and used implementations:
 - **SSLey, OpenSSL, GnuTLS.**
 - Protocol versions:
 - SSLv2, SSLv3.
 - TLS 1.0, TLS 1.1, TLS 1.2, TLS 1.3 (*).
 - Some **services/protocols** that use SSL/TLS:
 - https, ssh, ldaps, smtps/pop3s/imaps.

TLS: Transport Layer Security

[Internet Engineering Task Force]



- Developed from SSLv3:
 - By IETF (1999). RFCs updated: RFC 5246 and RFC 6176.
- Based on **PKI (asymmetric crypt)** and **symmetric crypt**:
 - **Private/public** keys & **session** keys.
 - **Digital certificates** X.509 (defined by UIT-T).
 - **CAs** (Certificate Authorities).
- Server (**secure** service):
 - **Service certificate** → [(public key)certificate]CA_{pk}
- A digital certificate signed by a CA provides 2 important features:
 - When a CA issues a signed certificate, it certifies the **identity** of the organization which is providing the secure service.
 - Client apps are able to recognize the service certificate **automatically** without asking users.
- There are *self-signed certificates* too:
 - **Unsafe!!**
 - Only local use.

CA: Trusted entity that *issues* and *revokes* **digital certificates** which are used by an organization to validate its identity and to ensure its communications.

TLS/SSL: The protocol

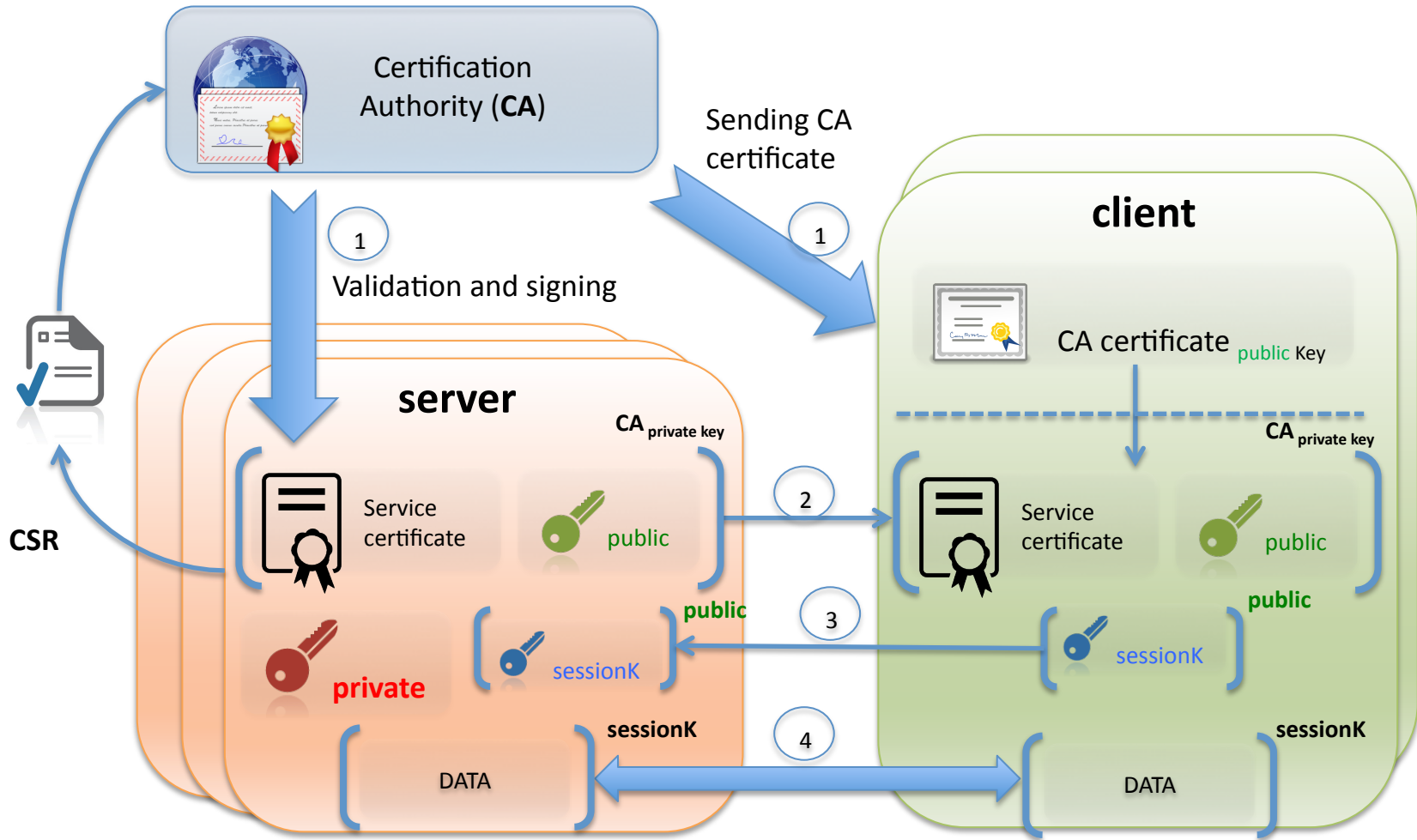
- 3 stages:
 - **HANDSHAKE:**
 - Both client and server **negotiate** the crypt algorithms to authenticate themselves and encrypt the information.
 - There are actually several options:
 - **Public-key cryptography: RSA, Diffie-Hellman, DSA (Digital Signature Algorithm) or Forteza.**
 - **Symmetric cryptography: RC2, RC4, IDEA (International Data Encryption Algorithm), DES (Data Encryption Standard), Triple DES or AES (Advanced Encryption Standard).**
 - **Hash functions: MD5, SSHA.**
 - **VALIDATION AND KEY EXCHANGE:**
 - **Step 1:** the ends are validated by digital certificate.
 - **Step 2:** they exchange keys to encrypt each other, according to the previous stage (**HANDSHAKE**).
 - **SECURE COMMUNICATION:**
 - The ends can begin the encrypted data transmission.
- The standards:
 - The first one: TLS (TLS 1.0) → RFC 2246.
 - At present (2014 October), **TLS 1.3** has been defined as a draft.

TLS/SSL: The protocol

- Client and server **message exchange** “in detail”:
 - **Step 1 [Hello]**, the ends agree on the algorithms to be used for keeping *confidentiality and authenticating*.
 - **Step 2 [server validation]**, server sends information about itself:
 - (service_{public key} + service certificate) ^{RSA} by CA_{private key}
 - **Step 3 (Optional) [client validation]**, server requests to client a X.509 certificate:
 - So they are both validated.
 - **Step 4 [session key production]**, which will be used to encrypt data:
 - It is often the client that produces this key.
 - **Step 5 [session key exchange]**, client sends this key to server:
 - (session key) ^{RSA} by server_{public key}
 - **Step 6 [Finish]**, It shows that client/server can start a new secure communication.

Optional

TLS/SSL: Mode of operation



openssl: SSL/TLS deployment

- Collaborative project from “**SSLeay**” (Eric Andrew Young[1] and Tim J. Hudson):
 - “European” branch of SSLeay.
 - “... 2014 two thirds of all web servers use OpenSSL”.
- Protocol implementations:
 - **Secure Sockets Layer (SSL v2/v3)**.
 - **Transport Layer Security (TLS v1.2)**.
- Some outstanding features:
 - Set of **encrypting libraries** written in C:
 - Provide *cryptographic functions* to software programmers.
 - They allow using *digital certificates*.
 - **Open source**.
 - **Multi-platform**:
 - Unix (Solaris, MAC OS...), Linux, Microsoft Windows...

gnuTLS: SSL/TLS deployment

- It's a **GNU project** to develop an implementation of SSL/TLS protocols.
- Sets of libraries and tools to make possible **secure communications** among clients and servers:
 - (API) Developed in C.
 - GNU *OpenSource* → GPL (LGPLv2.1+).
- Protocols:
 - SSL **v3.0**.
 - TLS 1.0, TLS 1.1 and **TLS 1.2**.
 - DTLS 1.0 and 1.2 (UDP).
- Provide an APIs to make *digital certificates*:
 - X.509, PKCS, OpenPGP...

SSL/TLS: More deployments...

- Other implementations of **SSL/TLS protocols**:
 - LibreSSL.
 - BoringSSL.
 - SharkSSL.
 - PolarSSL.
 - SecureBlackbox.
 - Network Secure Services.

- Are they actually a secure option?:
 - SSLv3: insecure!!!:
 - POODLE (<https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/>).
 - TLSv1.1 & TLSv1.2: safer!!!:
 - They solve many bugs of SSLv3 protocol.
 - TLSv1.3 (if approved):
 - For the moment (December 28, 2015), TLSv1.3 is not used very much (Developing...).
 - <https://tools.ietf.org/html/draft-ietf-tls-tls13-11>.

- But, what should we do if we want to deploy a fully safe service?:
 - We must always use SSL/TLS implementation updated.
 - DO NOT use SSLv3. It is no longer safe:
 - “False security”.

OpenSSL/GnuTLS: Installation and creation of certificates

- From debian **repositories**.

- **OpenSSL:**

- Installation of **openssl** libraries and tools:

```
$ apt-get update.  
$ apt-get install libssl1.0.0 libssl-dev openssl ssl-cert
```

- Creation of *self-signed* certificate (*):

```
$ mkdir /etc/ldap/ssl  
$ cd /etc/ldap/ssl  
$ openssl req --newkey rsa:1024 --x509 -nodes --out CA_server-01.localdomain.cert  
--keyout CA_server-01.localdomain.cert  
--days 365
```

certificate



- **GnuTLS:**

- Installation of **GnuTLS** libraries and tools:

```
$ apt-get update  
$ apt-get install gnutls-bin ssl-cert
```

- Creation of *self-signed* certificate (*):

```
$ mkdir /etc/ldap/ssl  
$ cd /etc/ldap/ssl  
$ certtool --generate-privkey --outfile CA_server-01.localdomain.key  
$ certtool --generate-self-signed --load-privkey CA_server-01.localdomain.key  
--template CA_server-01.localdomain.info  
--outfile CA_server-01.localdomain.cert
```

(*) It can be useful for testing a service under construction → CA certificate in DGSi.

Checking

- **SSL/TLS certificates:**

```
$ openssl s_client -connect <nombre servidor>:636  
-showcerts
```

```
$ gnutls-cli-debug -p 636 <nombre servidor>
```

- **LDAP server** running and access to its active directory:

```
$ netstat -aptnu
```

```
$ nmap <nombre servidor>
```

```
$ slapcat
```

- **LDAP service** running:

```
$ ldapsearch -x -H...
```

```
$ getent shadow
```

```
$ id <username_ldap>
```

- **The whole LDAP service**, through a “third-party” service:

```
$ ssh -l <username_ldap> <nombre servidor>
```

openLDAP: “Fail over” strategies

- “Old style” **REPLICATION** method:

- slurpd:

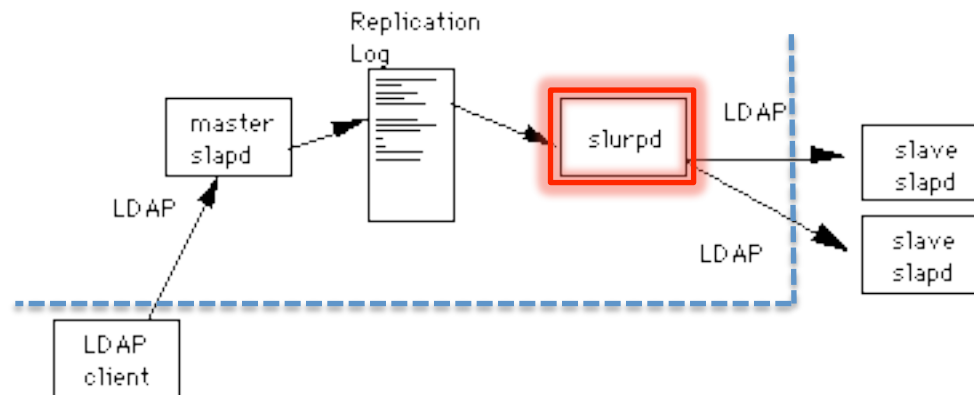
- Through an additional **daemon**, LDAP (openLDAP) will be able to deploy a “failover” schema itself:

- If the main daemon (**slapd**) goes down, the service keeps going through a secondary **slapd** instance running on a secondary server:

- » The switching is automatic (for client side).

- **slurpd** maintains the LDAP directory **REPLICATED** in a secondary directory:

- Running on different servers.



openLDAP: “Fail over” strategies

- “Old style” **REPLICATION** method:
 - **slurpd** was the first type of replication.
 - **slurpd** was a **standalone daemon** plagued with problems (briefly):
 - slurpd never rerouted requests.
 - It was not reliable.
 - It was extremely sensitive to the ordering of records in the *repllog*.
 - It could easily go out of sync, at which point manual intervention was required.
 - It wasn't very tolerant of unavailable servers.
 - It only worked in **push mode**.
 - It required stopping and restarting the master to add new slaves.
 - It only supported single master replication.
 - **slurpd** is no longer part of OpenLDAP:
 - From version 2.4.

openLDAP: “Fail over” strategies

- “New style” **REPLICATION** method:

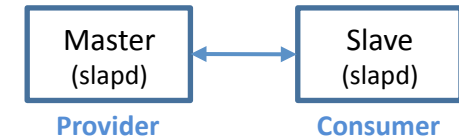
→ From version 2.4, openLDAP supports a few more replication modes.

- **SyncRepl**: lightweight *replication engine* for OpenLDAP

- **Syncrepl** has none of the “old style” weaknesses as regards replication.

- **Replication schema:**

- **Provider-consumer.**
- **Both of them can process client request:**
 - » Consumer only “reads”, does not “write/update”.



- ... And it adds:

- **MirrorMode (Active-Active Hot-standby).**
- **N-Way Multimaster Replication.**
- **And...:**
 - » More sophisticated Syncrepl configurations.
 - » Delta-syncrepl.
 - » Replicating `slapd` configuration (`syncrepl` and `cn=config`).

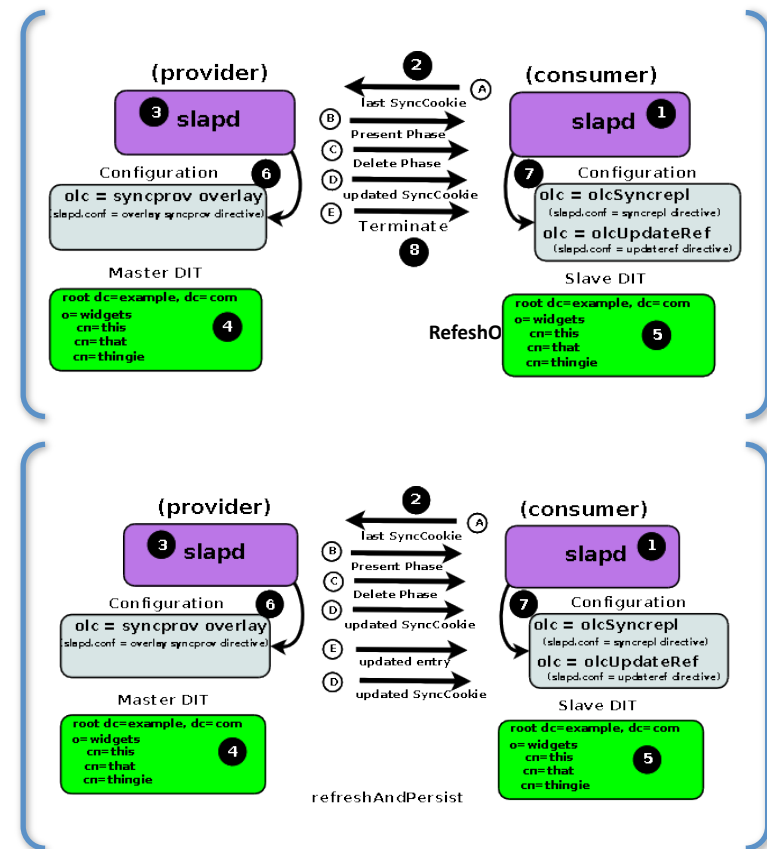
- **Optimization:**

- **Delta-syncrepl replication.**
- **Syncrepl Proxy mode.**
- **MirrorMode replication.**
- **N-Way Multi-Master replication.**

openLDAP: “Fail over” strategies

• Conventional Syncrepl replication: Basic LDAP Sync Replication:

- **Syncrepl engine** is executed as `slapd` threads.
- Replication operates at the **DIT level**, not the LDAP directory level:
 - Different DITs to different servers:
 - Even DIT fragments.
 - **Minimum unit** of synchronization:
 - The entry.
- **Incremental**:
 - Only changes after last sync.
- **Default replication schema**:
 - **Provider-consumer**.
 - **Consumer** always initiates the update process.
- **Operation modes**:
 - **RefreshOnly**:
 - **Consumer pull**:
 - » Burst mode.
 - » Replication cycle time.
 - **RefreshAndPersist**:
 - **Provider push**:
 - » Sync process remains active.
- Syncrepl tracks status of the replication content by maintaining and exchanging synchronization cookies.



Source: www.zytrax.com.

openLDAP: “Fail over” strategies

→ Optimization:

- **Delta-syncrepl** replication:

- Disadvantages of LDAP Sync replication:

- **LDAP Sync replication** is an *object-based* replication:
 - **When any attribute value is changed → the complete object (entry) is replicated.**
 - Both the changed and unchanged attribute values are processed.
 - Excess traffic generated for small changes.

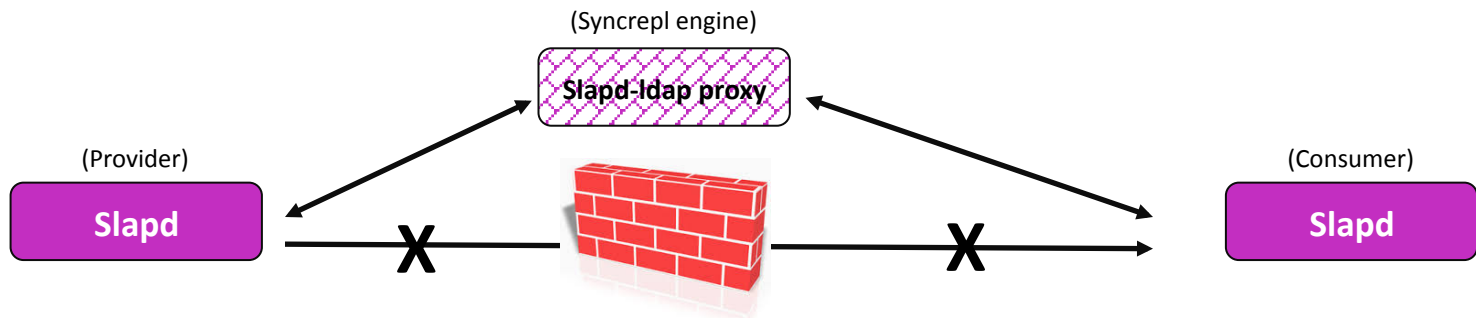
- **Delta-syncrepl:**

- Maintains a **changelog** on the provider.
 - Consumer checks the changelog for the **operations** it needs to perform on consumer directory.

openLDAP: “Fail over” strategies

- **Syncrepl Proxy Mode:**

- When *refreshAndPersist* is initiated from the consumer.
- **Firewalls** may need provider initiated **push-mode** replication.
- **Slapd-ldap proxy** is set up near (or collocated with) the provider that points to the consumer.
- Syncrepl engine runs on the proxy and points to provider.



openLDAP: “Fail over” strategies

- **MirrorMode** replication:

- It is an **Active-Active Hot-Standby** solution:

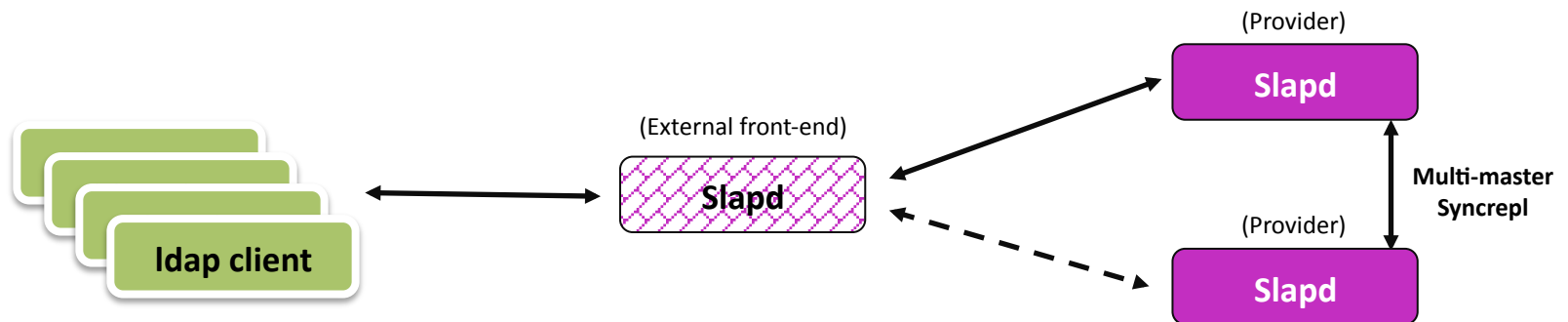
- **External *slapd* front-end is needed.**
- It is Not a Multi-Master solution.

- Syncrepl also allows the provider nodes to re-synchronize after any downtime.

- Delta-Syncrepl can be used.

- **2 providers** are set up to replicate from each other:

- An external frontend is employed to direct all writes to **only** one of the two servers.
- The second provider will only be used for writes if the first provider crashes.



openLDAP: "Fail over" strategies

N-Way Multi-Master:

- Uses **Syncrepl** to replicate data to multiple providers ("Masters").
 - Up to 4096 to be exact!
- Avoids a single point of failure.
- Supports complex topologies:
 - Providers can be located in several physical sites.
- Good for failover/High Availability || **NOTHING to do** load balancing.
- Requires synchronized time source - **ntp**.
- Providers must propagate writes to all the other servers:
 - Network traffic and write load spreads across all of the servers the same as for single-master.

Source: www.zytrax.com.

For more details:
<http://www.openldap.org/doc/admin24/replication.html>.

