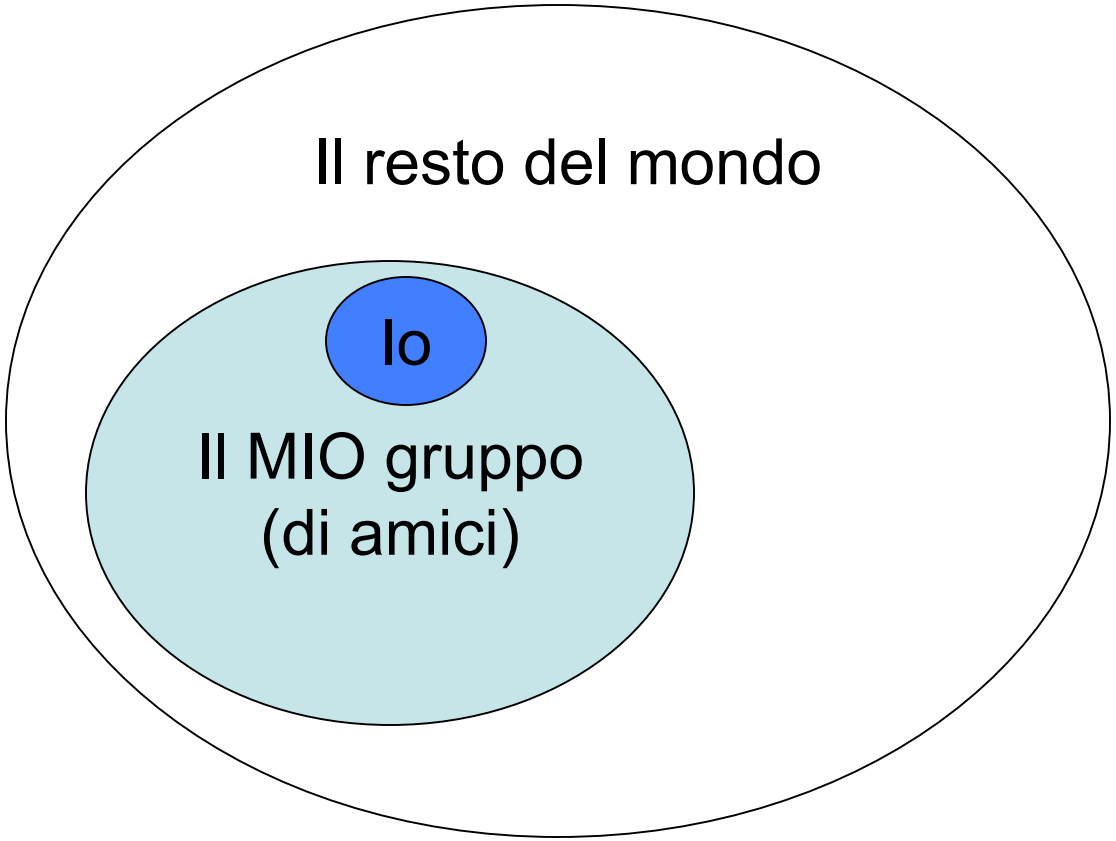
Modello sociale



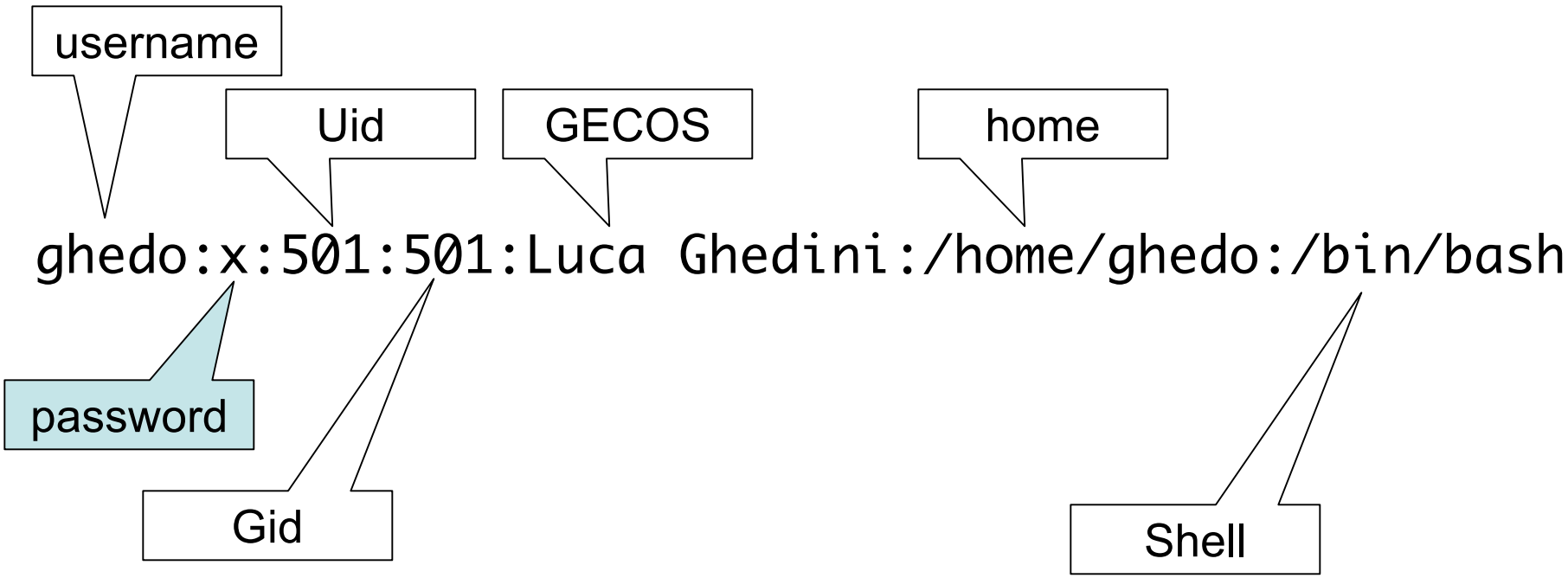
Utenti in locale

In unix tutto è file: Le protezioni si applicano ai file.
Ogni file ha 1 SOLO proprietario.

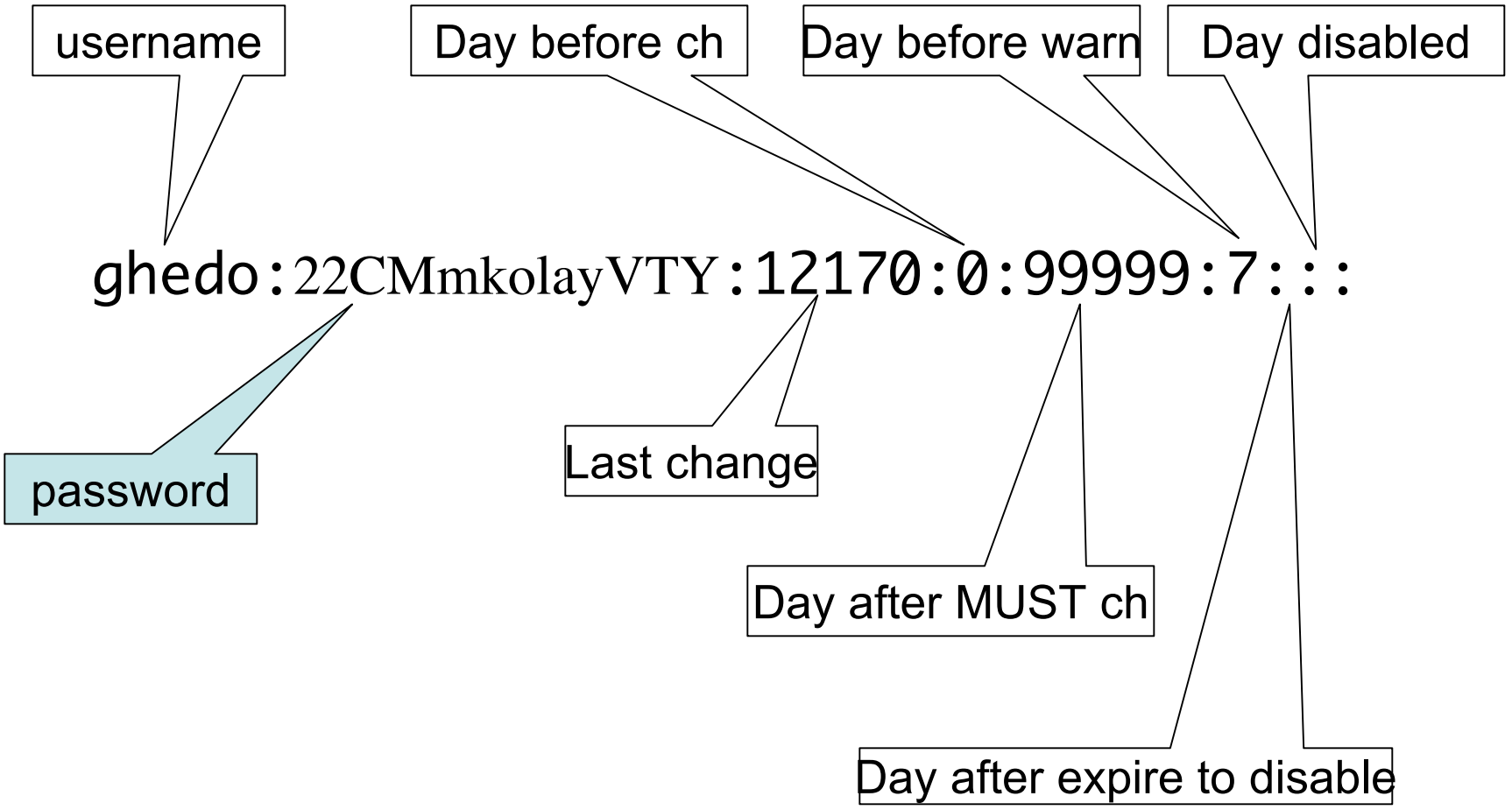
File di dati fondamentali:

- /etc/passwd (informazioni sugli utenti)
- /etc/shadow (password utenti)
- /etc/group (informazioni sui gruppi)

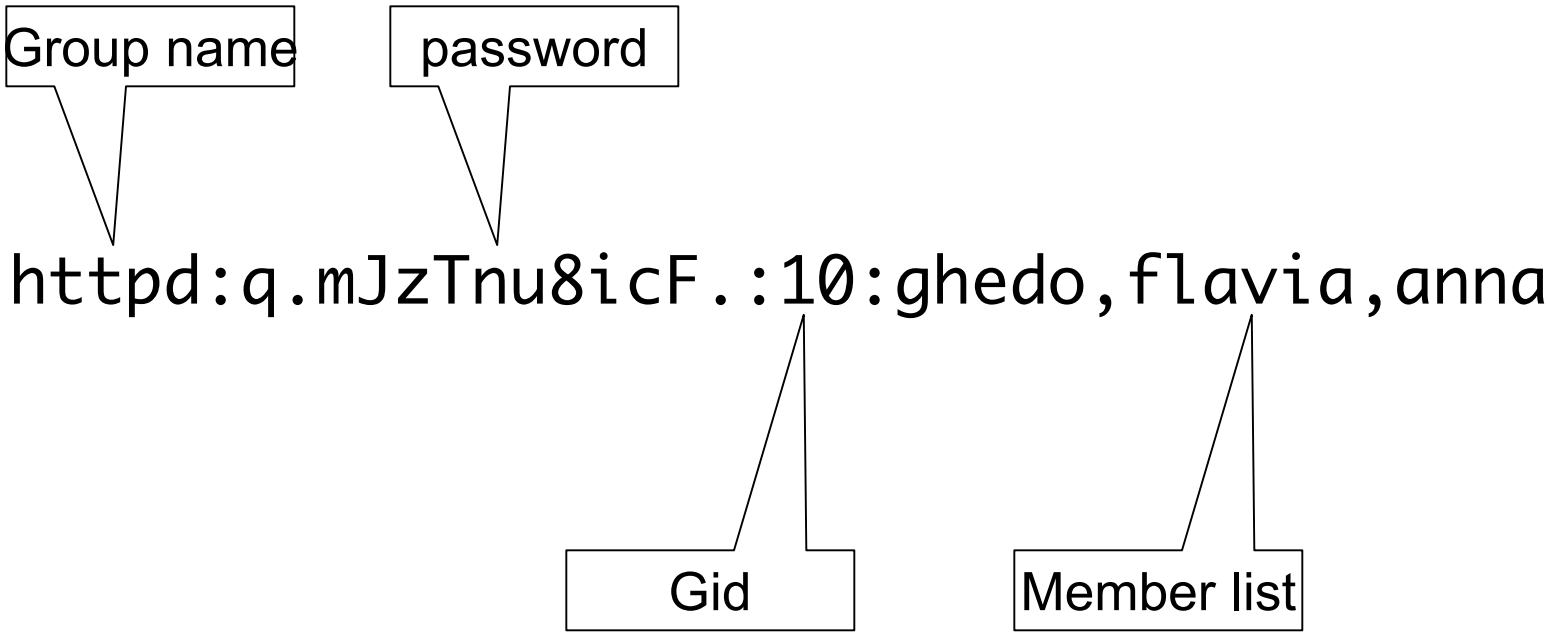
/etc/passwd


username
Uid
GECOS
home
ghedo:x:501:501:Luca Ghedini:/home/ghedo:/bin/bash
password
Gid
Shell

/etc/shadow

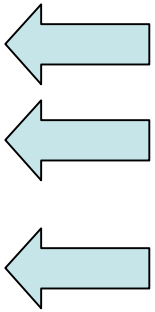

username
Day before ch
Day before warn
Day disabled
ghedo:22CMmkolayVTY:12170:0:99999:7:::
password
Last change
Day after MUST ch
Day after expire to disable

/etc/group

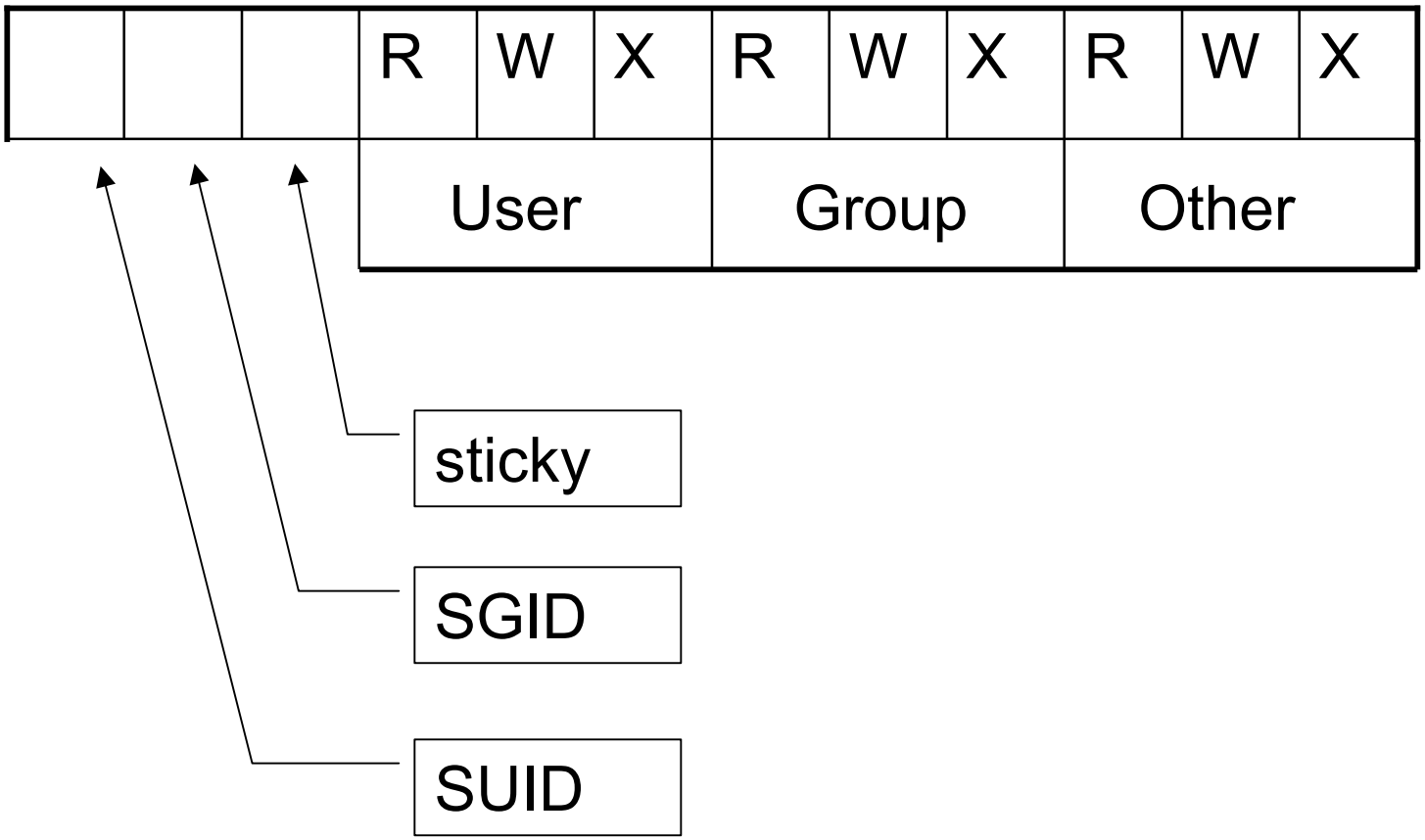

Group name
password
httpd:q.mJzTnu8icF.:10:ghedo,flavia,anna
Gid
Member list

Comandi utenti

useradd
passwd
chown
chfn
id
groupadd
groupdel
usermod



UGO!



Bit File Dir

	`file`	`directory`
R	Il file è leggibile da [U|G|O]	Consente di listare la dir
W	Il file è scrivibile da [U|G|O]	Rimozioni / aggiunte di entry dalla dir
X	Il file è eseguibile da [U|G|O]	cd nella dir

SUID SGID STICKY

	file	directory
Sticky	Suggerise al SO di conservare copia del programma nella swap (obsoleto)	Solo il propritario può cancellare un file Sticky anche se la dir è scrivibile da tutti
SGID	Il gid dell'utente che esegue un file SGID è assimilato al proprietario del file	I file creati dentro la dir sono asseganti al gruppo della dir ($file_{GID}$:=dir_{GID})
SUID	Il uid dell'utente che esegue un file SUID è assimilato al proprietario del file	

Note per le dir

“Le directory sono file che conservano l’elenco dei file contenuti in esse”

- il diritto di lettura su una directory è necessario per poter elencare i file contenuti
- il diritto di scrittura su una directory è necessario e **sufficiente** per poter **creare** e **cancellare** file al suo interno (non importa avere diritti sul file)
- il diritto di “esecuzione” va interpretato come possibilità di entrare nella directory (è sufficiente questo, ad esempio, per poter lanciare un programma che sta dentro una directory, anche se non si ha il diritto di lettura, a patto di conoscerne il nome)

Real & effective

Ogni utente è caratterizzato da una coppia UID GID (presa dal file passwd)

Se l'utente esegue programmi con SUID/SGID impostati, dell'ambito di quella esecuzione assume i UID/GID del programma.

Questi valori di UID/GID sono detti **effective**

Il sistema utilizza gli effective per ogni controllo a run time

La collaborazione

In un sistema unix la collaborazione fra utenti diversi avviene tramite la appartenenza ad un gruppo comune.

Due approcci

1. Creare più utenti con lo stesso Gid
2. Creare utenti con Gid differente ma assegnarli ad un gruppo comune (con Gid differente da entrambi)

1 più semplice ma meno sicuro: se utenti diversi partecipano a progetti diversi?

2 più complesso ma più sicuro (1 gruppo per progetto)

In the large…

- Quando gli utenti diventano migliaia ?
- Quando gli utenti operano su più elaboratori?

L’approccio a file diventa inefficiente

database di validazione degli utenti
servizi centralizzati di autenticazione

nis
pam
ldap

NIS

oramai solo storico

basato su rpc

uno o più server su cui gira ypserv

sui client gira ypbind

Nei file passwd/shadow/group una linea con + che indica al sistema operativo che il file non contiene tutto, ma che occorre interrogare NIS

problemi: complicato, non trasparente alle applicazioni, non esportabile fuori da unix, NON FLESSIBILE (il meccanismo del +…)

PAM

Separazione fra le sorgenti di informazioni e le altre componenti del sistema.

file nsswitch.conf indica per ogni risorsa le fonti da consultare e con che ordine:

es:

```
passwd:  files nis ldap
group:   files ldap
hosts:   ldap [NOTFOUND=return] dns files
```

Nss

<entry> ::= <database> ":" [<source> [<criteria>]]*
<criteria> ::= "[" <criterion>+ "]"
<criterion> ::= <status> "=" <action>
<status> ::= "success" | "notfound" | "unavail" | "tryagain”
<action> ::= "return" | "continue"
<action> ::= "return" | "continue" | "forever" | <n>
<n> ::= 0...MAX_INT

Nascita di una interfaccia LOCALE standard (nss)

La fonte xxx è implemetata nella libreria nss_xxx.so: chiunque può scrivere la propria fonte

Un altro passo

PAM risolve i problemi di standardizzazione di interfaccie in locale. Non risolve il problema della complessità di una gestione per migliaia di utenti.

Primo approccio: scrivere una libreria nss per accedere ad un database.

Problemi: ogni sistemista scrive la sua: incompatibilità, difficile testabilità e manutenzione. NESSUNA interoperabilità

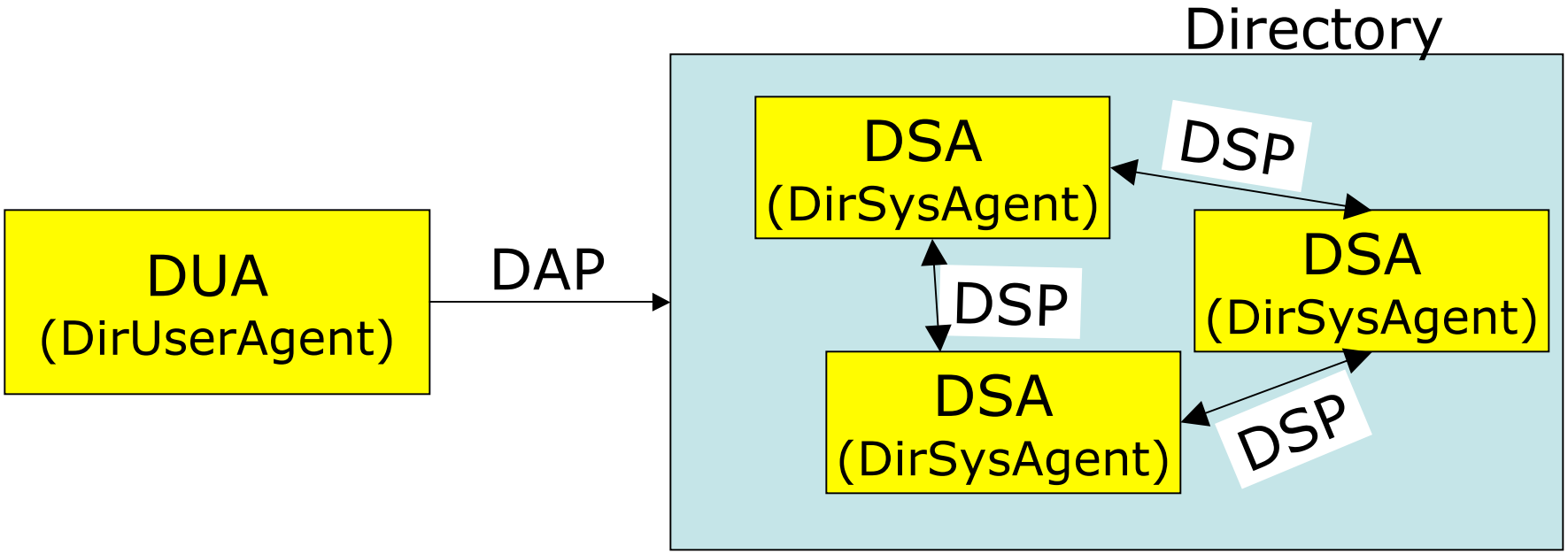
Secondo approccio: standardizzare il protocollo di accesso ad una base dati standardizzata

LDAP e directory services

Cosa è Directory

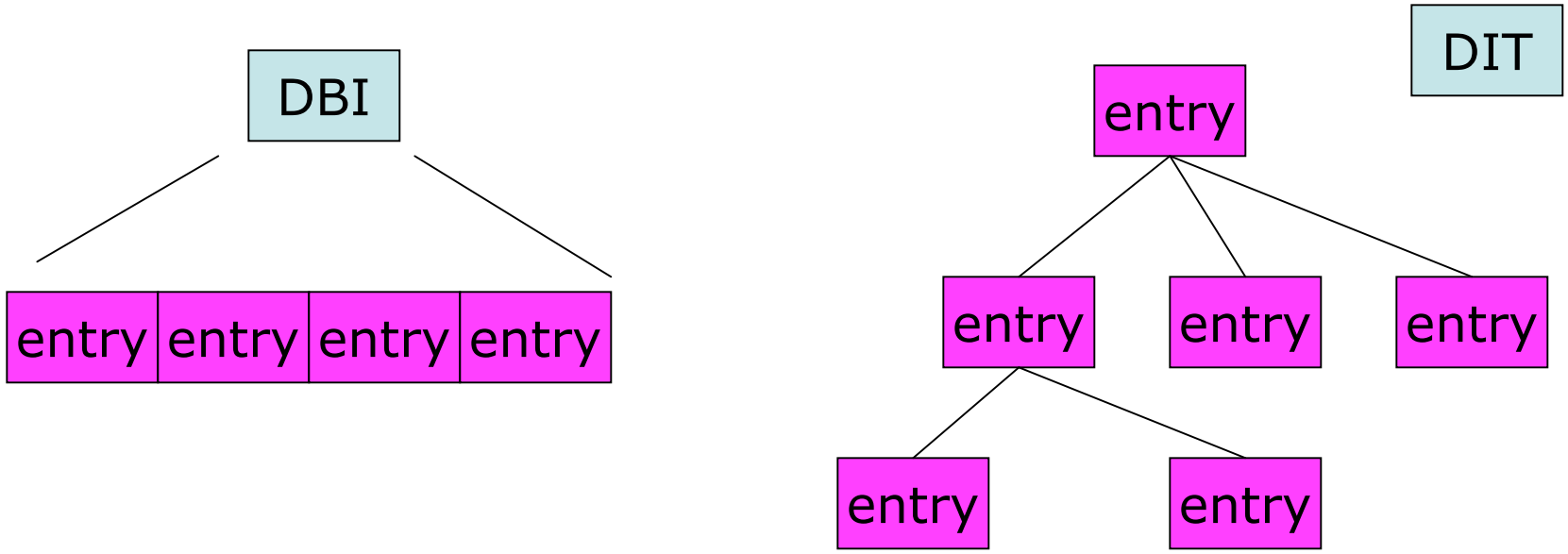
- Lightweight Directory Access Protocol
- mutuata da x500 (OSI -> ASN1, ROSE)
- Directory Services (RFC 1777)
- “molte letture poche scritture” (elenco telefono)
- Ne transizioni ne rollback
- Replica dei dati
- Architettura a master-slave
- Consistenza fra le copie lasca
- Possibilità di avere amministratori locali

Mondo Dir

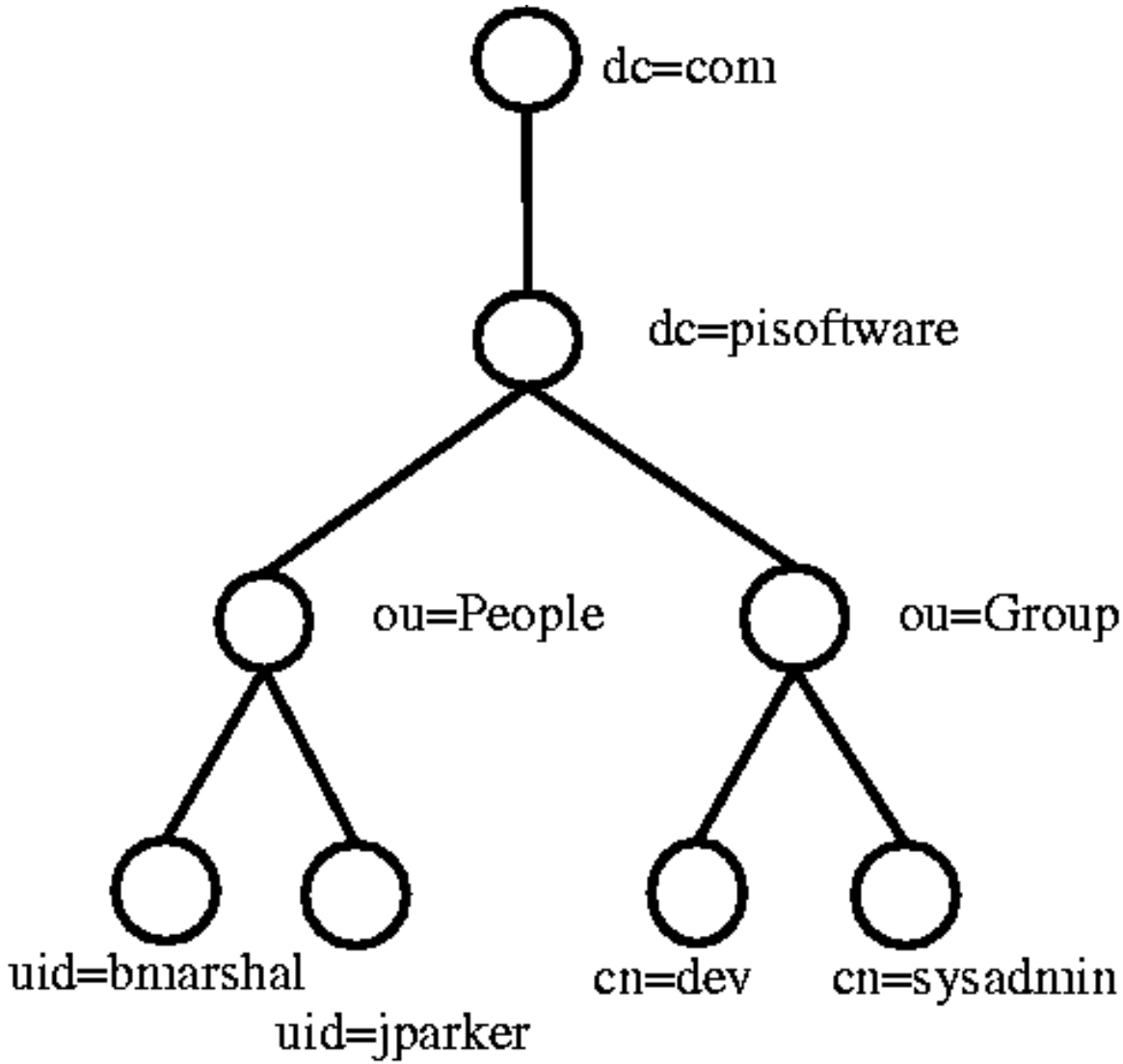

Directory
DUA
(DirUserAgent)
DAP
DSA
(DirSysAgent)
DSP
DSA
(DirSysAgent)
DSP
DSA
(DirSysAgent)
DSP

DBI & DIT

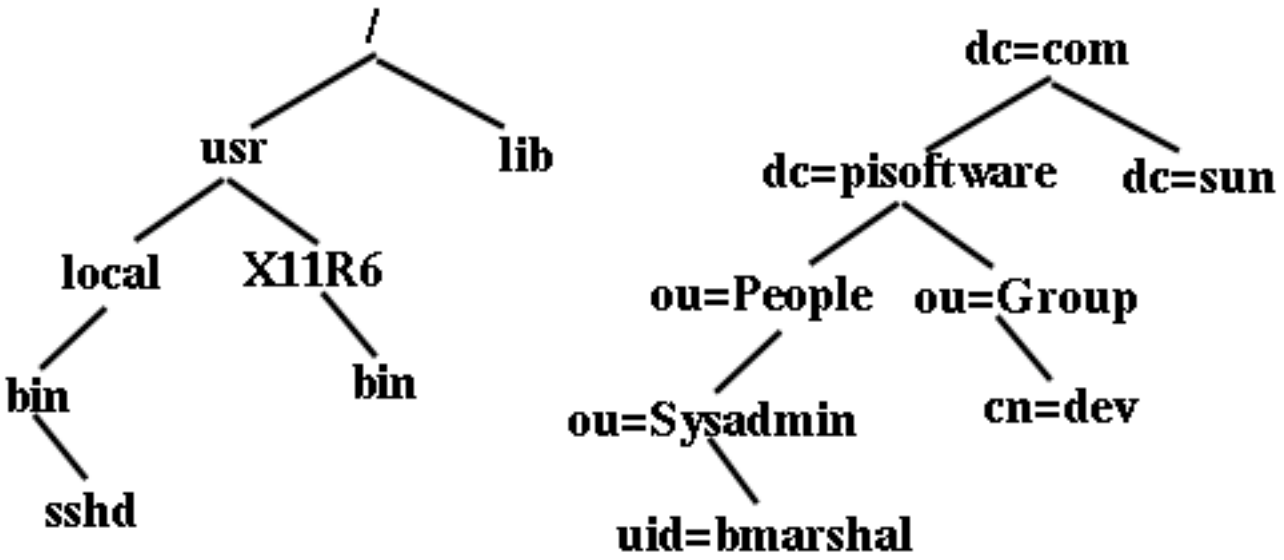
- La base di dati della Dir è detta DBI (Directory Information Base)
- DBI è una serie di **entries** (o object)
- Entry consistono di sequenze di **attributes**
- Gli attributes consistono di **types** con (molteplici) **values**
- Ogni entry ha un DN (Distinguis Name)
- Le entry sono gerarchicamente legate. La gerarchie è chiamata DIT (Dir Information Tree). Una DBI è rappresentata in un DIT



Dir NameSpace



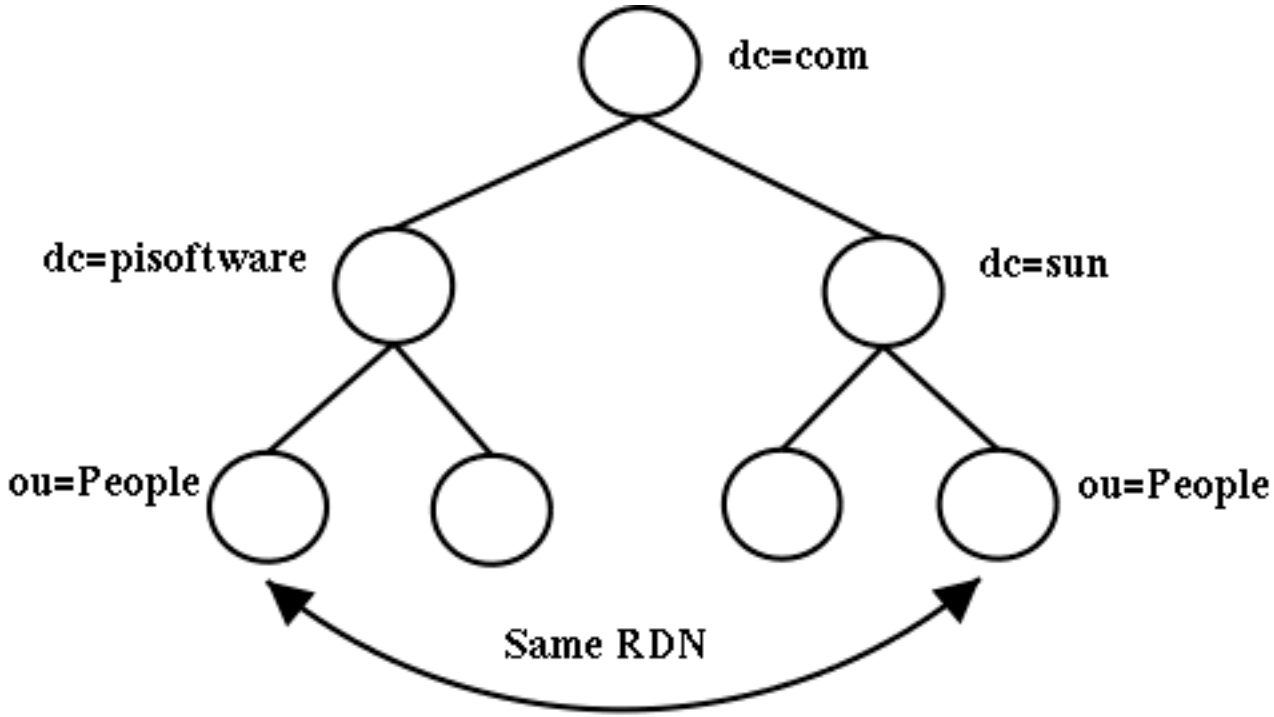
confronto con un file system



- Nel FS esplorazione dalla radice nella DIR dalle foglie
- Non c’è una root nella DIR

Distinguished Name (DN RFC 1485)

Elenco degli attributi separati da virgole a partire dal fondo.
DN=BDN (Base DN) + RDN (Relative DN)



DN: 1 esempio

- DN is uid=bmarshal,ou=People,dc=pisoftware,dc=com
- RDN is uid=bmarshal
- BDN is ou=People,dc=pisoftware,dc=com

- ogni RDN distingue le entries fra i pari
- In ogni BND ogni RDN è unico => **ogni entries ha un DN unico**
- un RDN può esser composto dalla concatenazione di più attributi (per renderlo unico)
- In ogni entries è specificato quali attributi ne costituiscono il RDN

Alias e Collective attribute

- Una entries può avere sopranomi per le entries (alias)
- Ogni alias deve essere NON ambiguo (ma non unico): più nomi per lo stesso oggetto.
- Un alias può riferire ad un altro alias! (loop…)

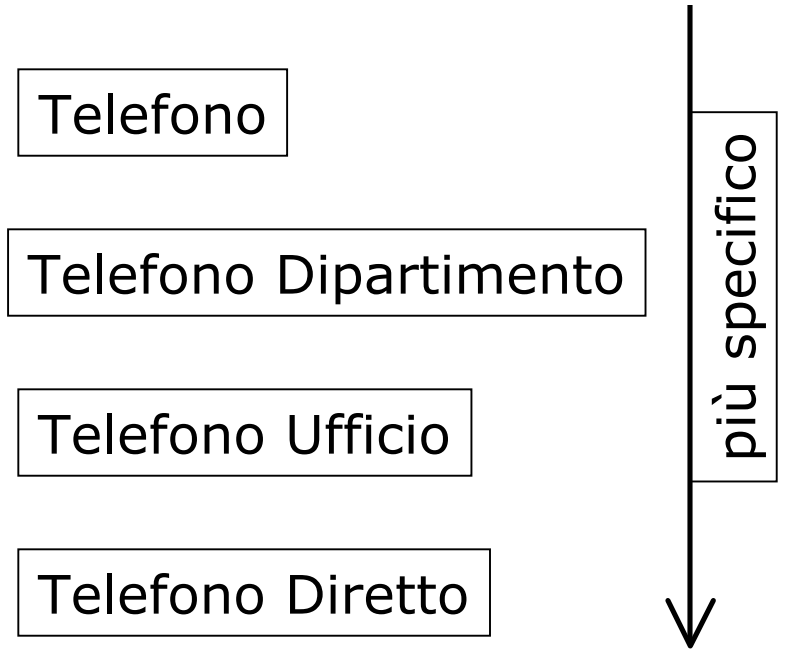
Come fare quando lo stesso attributo assume lo stesso valore in moltissime entries? (scomodo da inserire)

subentries

- Entries speciali che contengono gli attributi comuni uniti ad un riferimento nel DIT per indicare a quali entries vanno aggiunti.
- Durante una lettura vengono riportati anche loro come se fossero parte della entries.
- Durante una scrittura **sono assenti**!
- Solo in manager della DIR le può alterare

Attributi in gerarchia

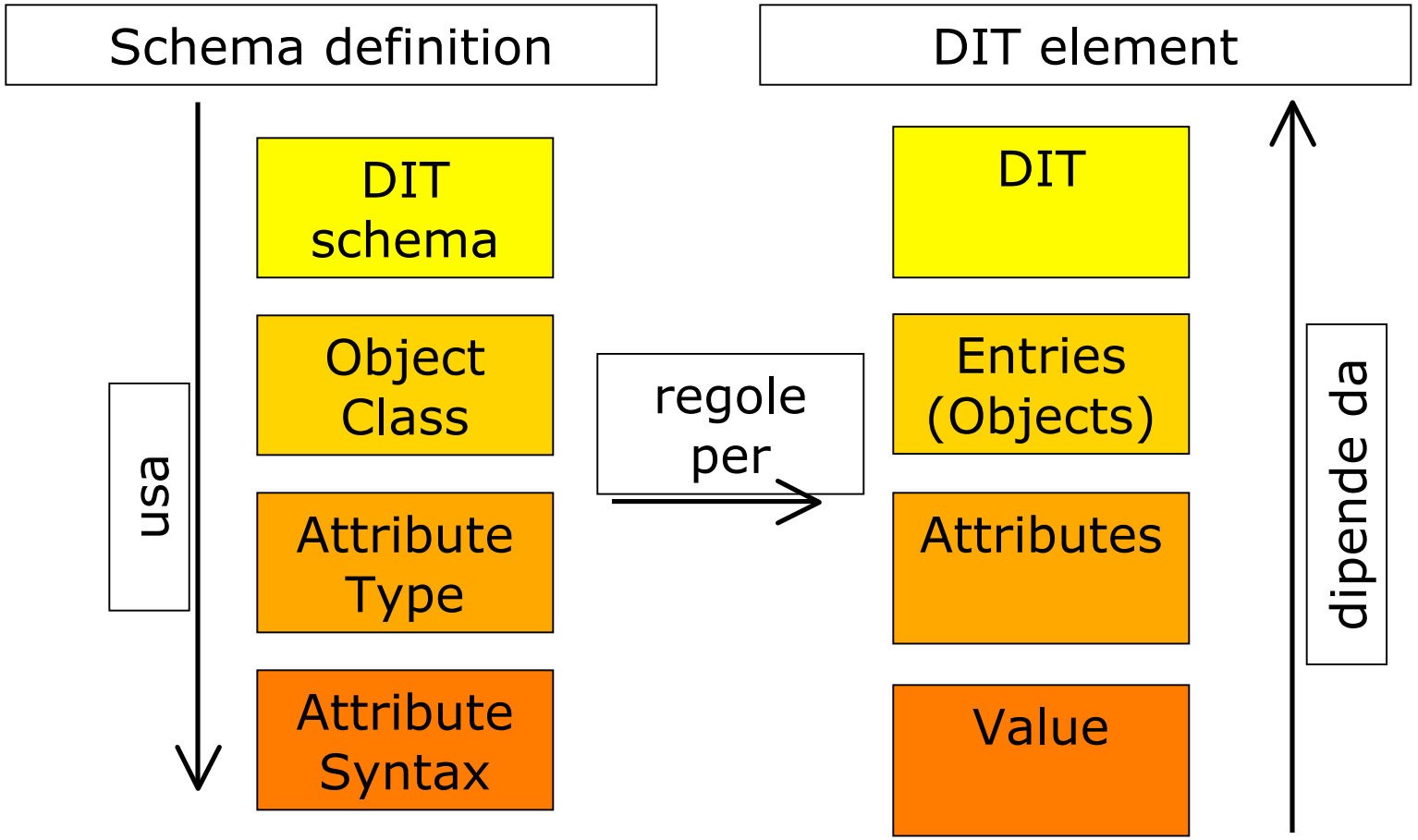
Sono possibili gerarchie di attributi



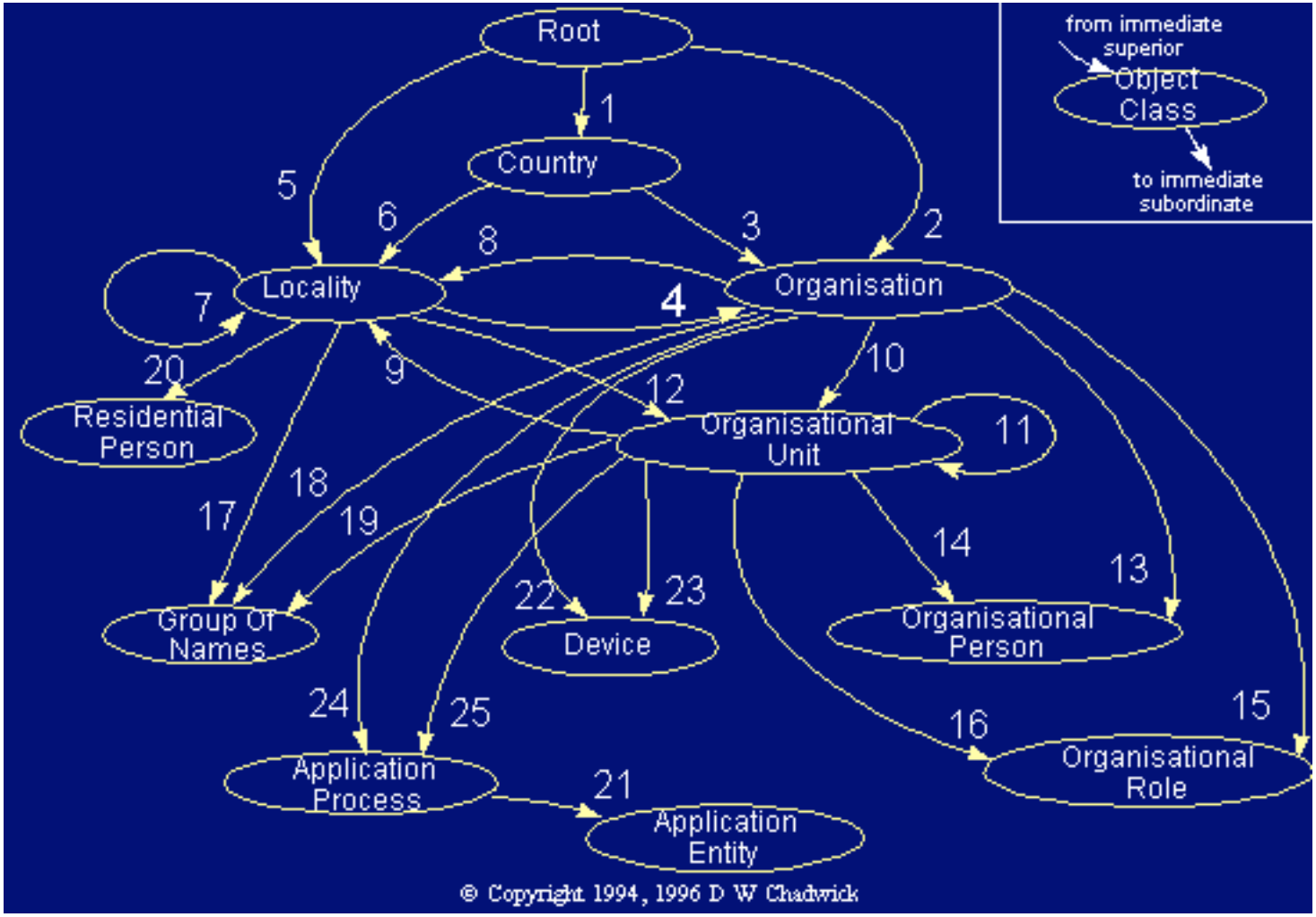
In risposta ad una interrogazione vengono restituiti tutti gli attributi più specifici (oltre a quello richiesto)

Schema

Il modo per descrivere le parti del sistema è detto **schema** e la sintassi usata è ASN1



Esempio di organizzazione di DIT



Esempi di Attribute

```
attributetype ( 1.3.6.1.1.1.1.0 NAME 'uidNumber'
        DESC 'An integer uniquely identifying a user'
        EQUALITY integerMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )


attributetype ( 1.3.6.1.1.1.1.4 NAME 'loginShell'
        DESC 'The path to the login shell'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

Esempi di Object Class

```
objectclass(
        1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
        DESC 'Abstraction of an account with POSIX attributes'
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description )
 )

objectClass (
        1.3.6.1.4.1.4203.1.4.5 NAME 'OpenLDAPperson'
        DESC 'OpenLDAP Person'
        SUP ( pilotPerson $ inetOrgPerson )
        MUST ( uid $ cn )
        MAY ( givenName $ labeledURI $ o )
)
```

Schema

- Insieme di regole che descrivo i dati immagazzinati
- L'uso di schemi rende chiara la struttura della Dir
- Per ogni entries (object) lo schema indica le regola da seguire per memorizzare dati indicando:
  - Attributi indispensabili/facoltativi
  - Come comparare gli attributi
  - Il tipo di dato memorizzabile in un attributo

Alcuni attributi standard

uid	User id
cn	Common name
sn	Surname
l	Location
c	Country

ou	Organization Unit
o	organization
dc	Domain Component
st	State

Filter (RFC 1558)

Selezione sugli attributi

<filter> ::= '(' <filtercomp> ')'
<filtercomp> ::= <and> | <or> | <not> | <item>
<and> ::= '&' <filterlist>
<or> ::= '|' <filterlist>
<not> ::= '!' <filter>
<filterlist> ::= <filter> | <filter> <filterlist>
<item> ::= <simple> | <present> | <substring>
<simple> ::= <attr> <filtertype> <value>
<filtertype> ::= <equal> | <approx> | <greater> | <less>
<equal> ::= '='
<approx> ::= '~='
<greater> ::= '>='
<less> ::= '<='
<present> ::= <attr> '=*'
<substring> ::= <attr> '=' <initial> <any> <final>
<initial> ::= NULL | <value>
<any> ::= '*' <starval>
<starval> ::= NULL | <value> '*' <starval>
<final> ::= NULL | <value>

Esempi di Filtri

```
(cn=Babs Jensen)
```

```
(!(cn=Tim Howes))
```

```
(&
   (objectClass=Person)
   (|
      (sn=Jensen)(cn=Babs J*)
    )
)
```

```
(o=univ*of*mich*)
```

```
(objectclass=posixAccount)
```

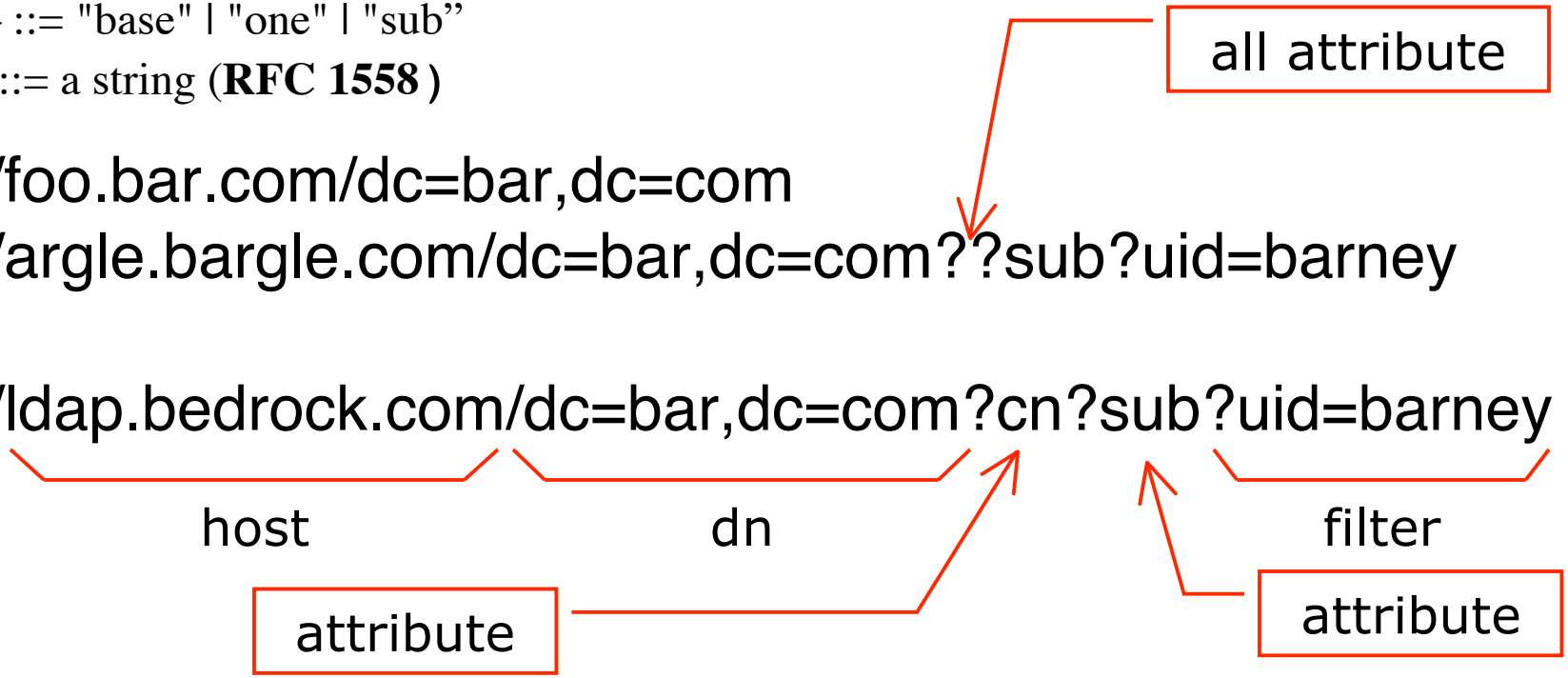
```
(&(|(uid=jack)(uid=jill))(objectclass=posixAccount))
```

LDAP URL (RFC 1959)

<ldapurl> ::= "ldap://" [<hostport>] "/" <dn> ["?" <attributes>["?" <scope> "?" <filter>]]
<hostport> ::= <hostname> [":" <portnumber>]
<dn> ::= a string (**RFC 1485**)
<attributes> ::= NULL | <attributelist>
<attributelist> ::= <attributetype>| <attributetype> ["," <attributelist>]
<attributetype> ::= a string (**RFC 1777**)
<scope> ::= "base" | "one" | "sub"
<filter> ::= a string (**RFC 1558**)

ldap://foo.bar.com/dc=bar,dc=com
ldap://argle.bargle.com/dc=bar,dc=com??sub?uid=barney

ldap://ldap.bedrock.com/dc=bar,dc=com?cn?sub?uid=barney



LDIF (RFC 1960)

Formato di scambio ASCII (LDAP Interchange Format):

- Entries rappresentate in ascii
- Human readable
- Permette facilmente di trasferire/modificare dati

```
dn: uid=lab2n36$,ou=People,o=labx,dc=ing,dc=unibo,dc=it
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
userPassword: e2NSeX28fXg=
loginShell: /bin/false
uidNumber: 62036
gidNumber: 0
homeDirectory: /dev/null
gecos: Macchina windows
uid: lab2n36$
cn: LAB2N36$
```

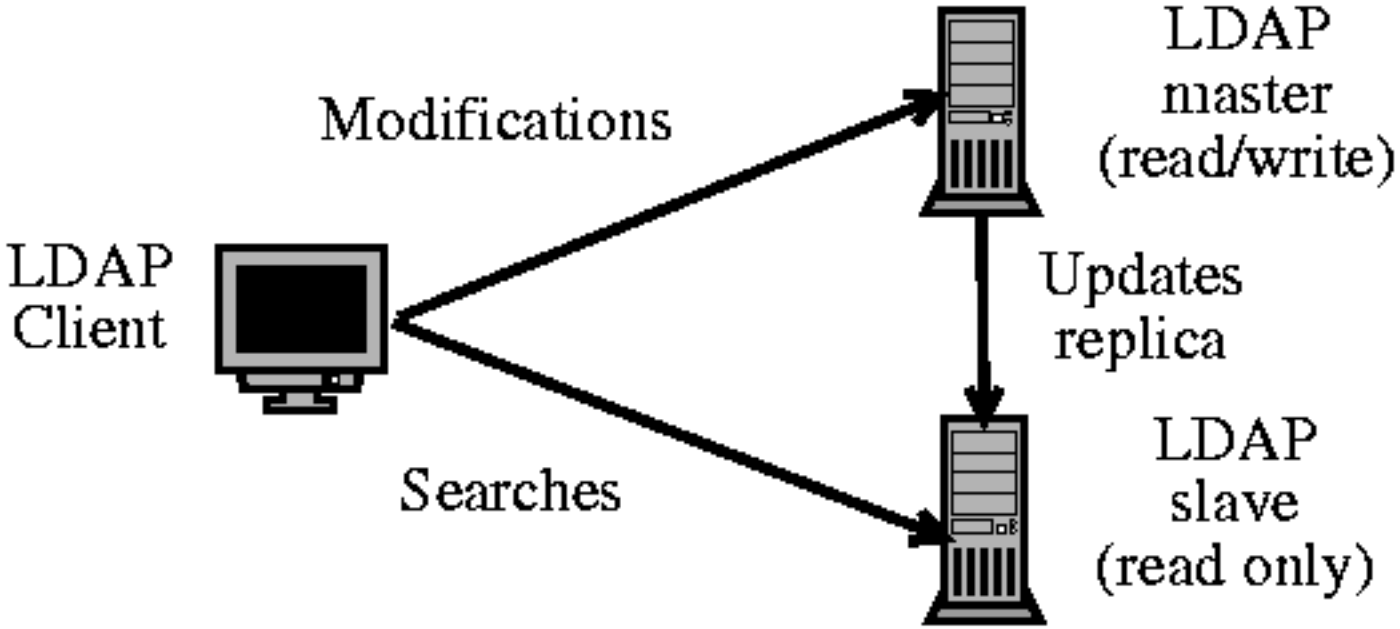
Si Replica!

Avere repliche: fault tollerance, performance.

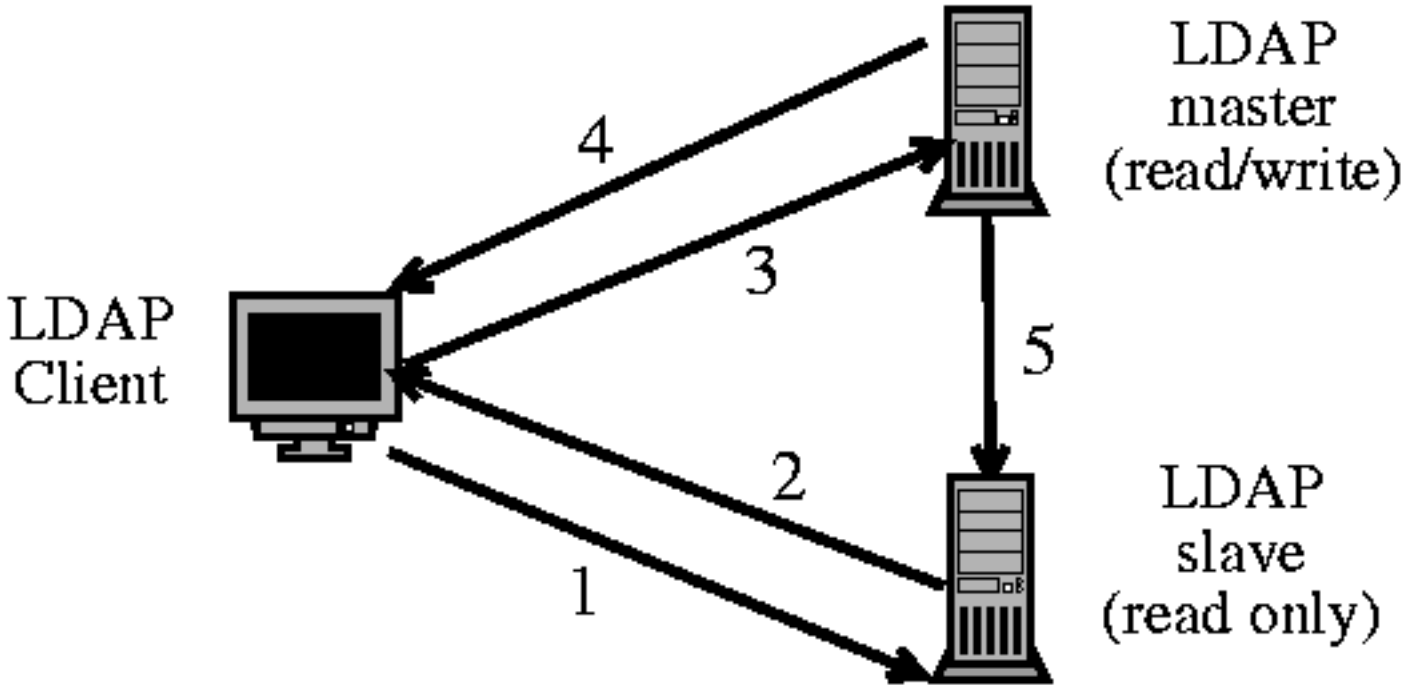
LDAP ha una architettura master-replica:
le modifica vengono propagate dal master alle repliche.
Lo standard NON dice con che tempistica ne in che modo.
Lo standard attuale in sviluppo (v3) prevede una architettura multimaster (protocolli ad hoc per la replica)

Nel caso mono master più modalità per gestire la scrittura delle etries (on master, by referrals, by chain)
Una replica può fungere da fonte di dati per aggiornarne un'altra

On Master

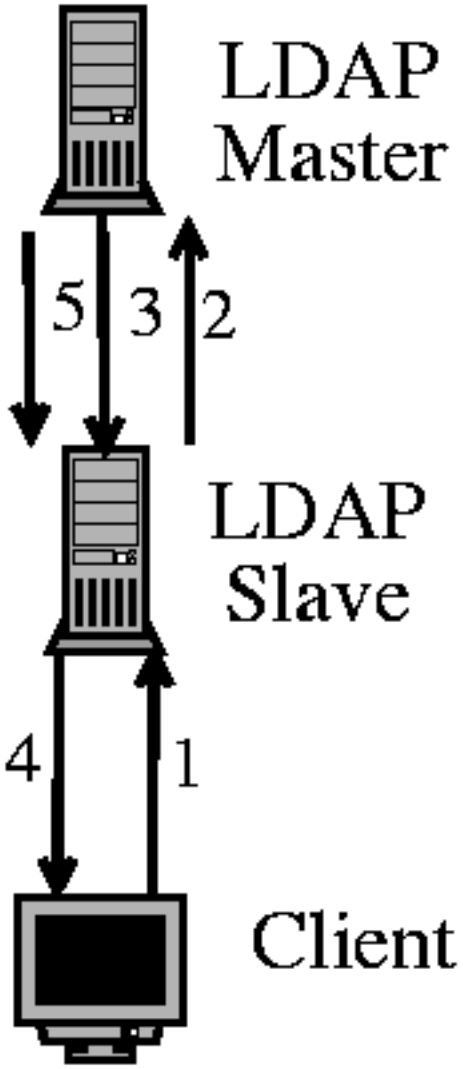

LDAP
master
(read/write)
Modifications
LDAP
Client
Updates
replica
Searches
LDAP
slave
(read only)

By Referrals



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

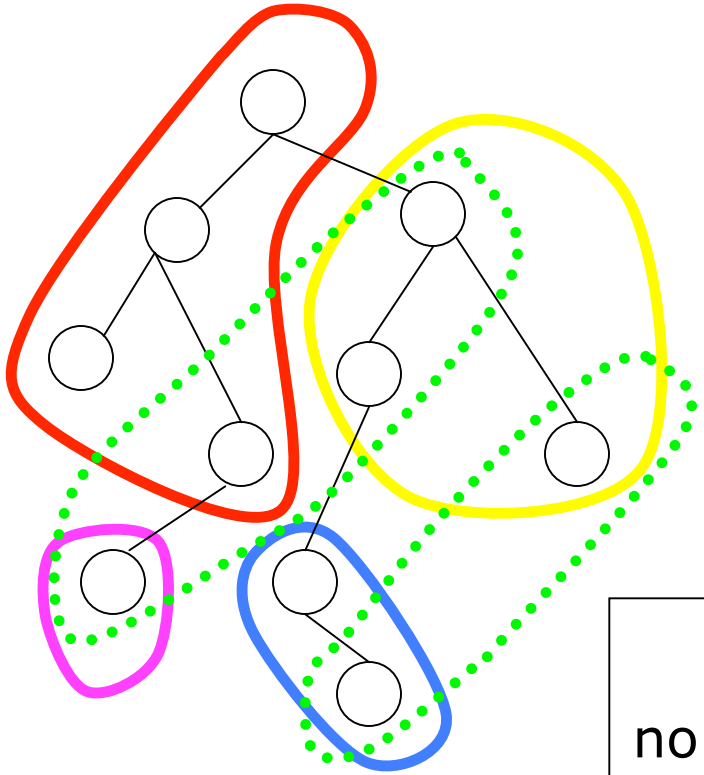
By Chain

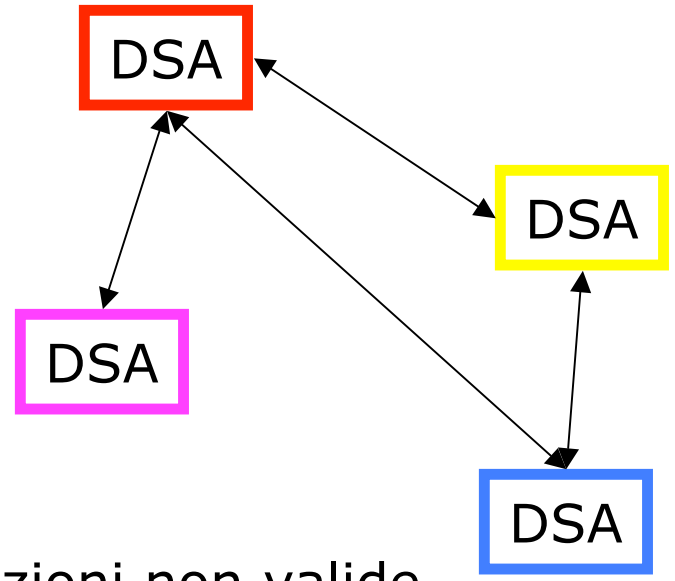


1. Client sends modification to replica
2. Replica forwards request to master
3. Master returns result to replica
4. Replica forwards result to client
5. Master updates replica

Delega

IL DIT è partizionabile in sottoalberi ciascuno dei quali può essere ospitato da un differente DSA





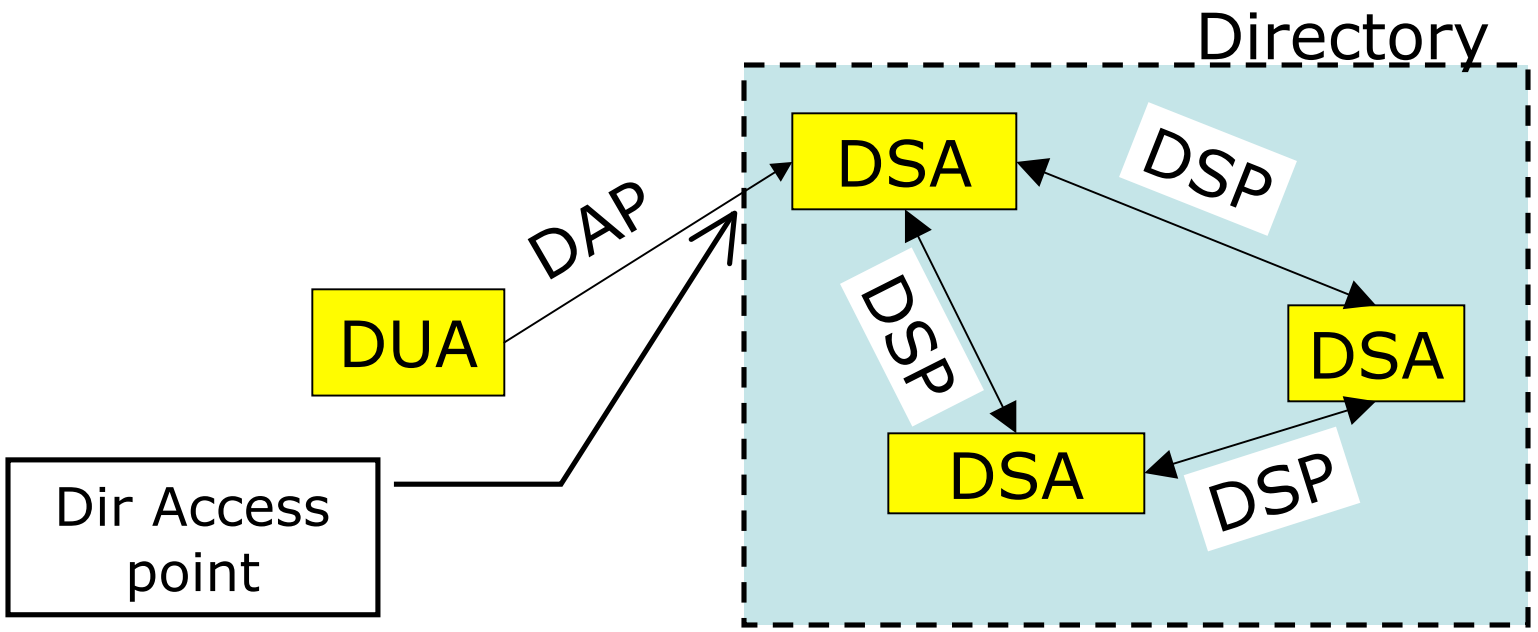
Partizioni non valide

Più sottoalberi per 1 DSA?

no

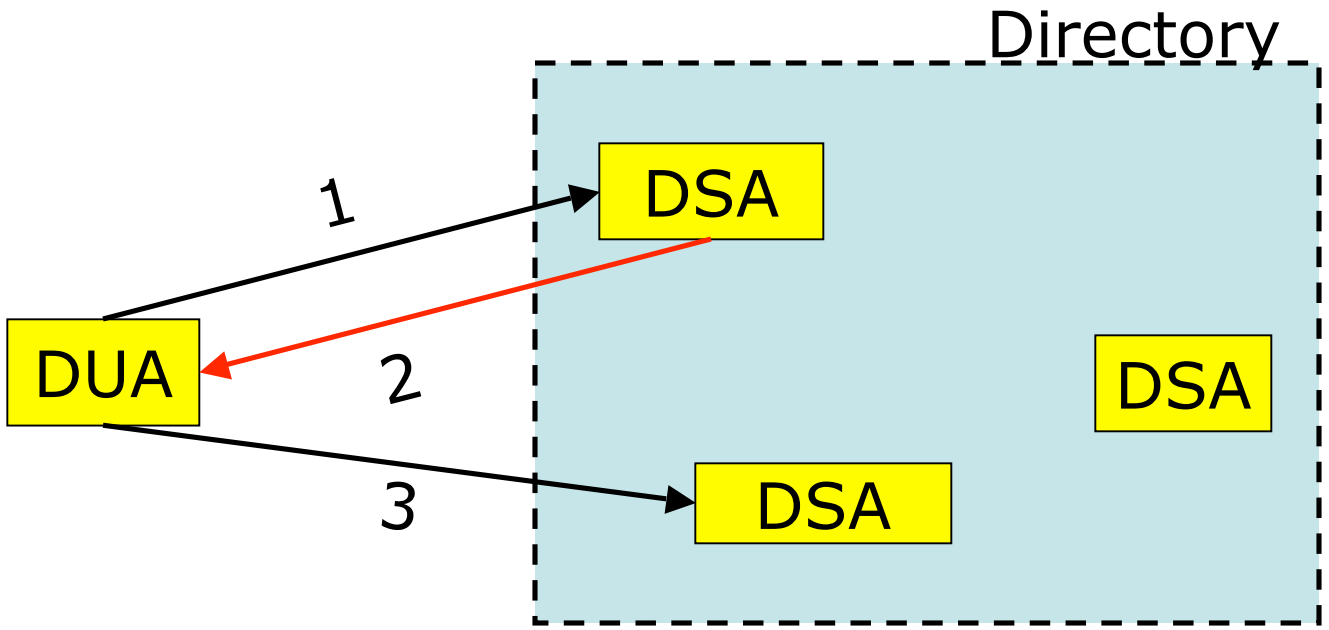
si

Dir con Catena



DUA fa richieste solo attraverso un DSA che funge da Access Point. Sarà sua cura coordinarsi con gli altri DSA per soddisfare la richiesta del DUA

Dir senza catena



- Ogni DSA può comportarsi come desidera sia quando ricevere una richiesta che quando riceve un **REFERRALS**
- Un ref contiene tutte le info possibili per aiutare chi lo riceve (per es elenco di tutte le DSA in grado di aiutarlo)