How we implemented an LDAP directory for Laboratories

A Case Study at Hong Kong Institute of Vocational Education (Tsing Yi), Department of ICT

Nick Urbanik `<nicku@nicku.org>`

Original work available with all LATEX and xfig source files at

`http://nicku.org/slug-talks/ldap-slug-talk/`

Sydney Linux Users Group (SLUG)

Building 2, Level 4, Room 410, UTS Broadway

24 June 2005, 8.20 pm

Ummm, Err, ummm… Errrrrr … what shall we do?

What do you already know about LDAP?

■ How many know that a directory is tree shaped?

What do you already know about LDAP?

- How many know that a directory is tree shaped?
- How many have worked with a directory before?

What do you already know about LDAP?

- How many know that a directory is tree shaped?
- How many have worked with a directory before?
- How many know about SNMP object IDs?

What do you already know about LDAP?

- How many know that a directory is tree shaped?
- How many have worked with a directory before?
- How many know about SNMP object IDs?
- How many know . . .

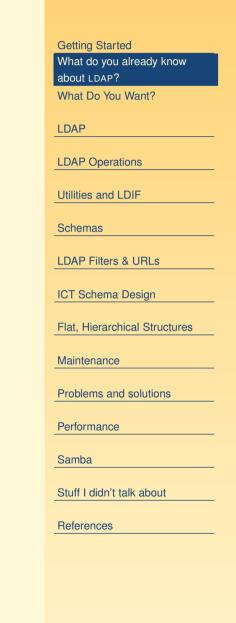# What do you already know about LDAP?

- How many know that a directory is tree shaped?
- How many have worked with a directory before?
- How many know about SNMP object IDS?
- How many know . . .
- What. . . is the air-speed velocity of an unladen swallow?

What do you already know about LDAP?

- How many know that a directory is tree shaped?
- How many have worked with a directory before?
- How many know about SNMP object IDs?
- How many know . . .
- What. . . is the air-speed velocity of an unladen swallow?
- The European swallow appears to do about $11\,\mathrm{ms}^{-1}$

What Do You Want?

■ I could talk for ten hours

What Do You Want?

- I could talk for ten hours
- (actually, I could go on for twenty after a few beers)
    - ◆ . . . but we have only an hour

What do you already know about LDAP?
What Do You Want?

LDAP

LDAP Operations

Utilities and LDIF

Schemas

LDAP Filters & URLs

ICT Schema Design

Flat, Hierarchical Structures

Maintenance

Problems and solutions

Performance

Samba

Stuff I didn't talk about

References

LDAP at HKIVE(TY) - p. 4/99

What Do You Want?

- I could talk for ten hours

- (actually, I could go on for twenty after a few beers)
  - ◆ . . . but we have only an hour

- What topics do you want us to cover here?
  - ◆ I think we need to understand the basics of LDAP itself
    - operations
    - some simple tools
  - ◆ . . . to make sense of other topics, especially programming
  - ◆ The basics of the way inheritance works in LDAP is important when understanding how to design a schema

Reasons for LDAP and problems with alternatives

We don't have time for this!

Account Information

- The computer uses numbers to refer to users and groups
- Humans prefer to use names (like nicku)
- When you create files in your shared network drive, the client must access them using the same numbers
- The user ID numbers and group ID numbers must be the same on all computers
- Otherwise won't be able to read own files!

Network Accounts

- $ `ls -ln file`

  `-rw-rw----    1 500      500                2057 Nov  1  2000 file`

- Now nicku with user ID number 500 and group ID 500 can read and write this file

- . . . But nicku with user ID number 2270 and group ID number 2270 cannot access the file at all:

  $ `id`

  `uid=2270(nicku) gid=2270(nicku) groups=2270(nicku),14171(staff)`

- The user ID numbers and group ID numbers on files on a network drive are fixed
- The user ID numbers should remain unchanged for all users who read/write the network drive.

Methods of achieving this

■ Have a *directory server* of some kind

■ The directory server associates a fixed user ID number
with each login ID

■ ... and a fixed group ID number for each group ID

■ On NT, these are called SIDs (security IDs)

Directory systems for authentication

- Proprietary:
  - Novell Directory Services (NDS)
  - Microsoft Active Directory (M? AD)
  - NT 4 domain
  - NIS + (Network Information System plus)
  - NIS

- Open protocols:
  - LDAP
  - Hessiod

Proprietary application directories

- Application-specific directories:
  - ◆ Lotus Notes
  - ◆ cc:Mail
  - ◆ Microsoft Exchange
  - ◆ Novell GroupWise

- These directories come bundled with, or, embedded into an application such as email.

- If add another such application, must manage one more directory ("$N + 1$ directory problem")

- If add another user, must add to all the directories.

Problem with proprietary directories

- Need put the same user into many different directories

- Need maintain N times the number of user accounts, where N is the number of directories.

- This is just too much work.

- The accounts get out of sync.

Why not buy Microsoft AD?

- Microsoft leverage their monopoly on the desktop to "embrace and extend" free software written by others

- Example:
  - Kerberos is a "Network Authentication Service", an IETF standard (see RFC 1510)
  - Kerberos is written by cooperating programmers round the world
  - Microsoft took Kerberos, and modified the protocol very slightly (they classified this change as a "trade secret")
  - So that MS destops can use MS Kerberos servers, but not non-MS Kerberos servers.

- Although MS claims to support standards, MS solutions are highly proprietary

- Designed to lock the user into an all-MS solution.

- Could be an expensive and insecure mistake.

LDAP

■ Single sign on—the **𝕳𝖔𝖑𝖞 𝕲𝖗𝖆𝖎𝖑**.

■ King Arthur and his knights support this quest

■ The knights who say *Ni* all concur with a resounding *Ni!*

LDAP— Why?

- Non-proprietary, IETF standard
  - No vendor lock-in
  - Use standard software components
- Supports authorisation as well as authentication
  - E.g., access if "staff, or year 3, group W, CSA student"
- Very general purpose: use for email, system authentication, application authentication, . . .
- Reasonably secure
- Robust
- Extensible
- Good open source implementation available at
  `http://www.OpenLDAP.org/`

LDAP Terminology

- LDAP model is *hierarchical*, i.e., tree-structured
- Each object in a directory is an *entry*
- Each individual item in an entry is an *attribute*
- Each entry has a unique full name called its *distinguished name* or *dn*
- Each entry has a short name that is unique under its parent, called its *relative distinguished name*, or *rdn*.
- The organisation of names in the directory is called the *namespace*
- An important initial task is *namespace design*

What is LDAP?

- The LDAP *protocol*, a standard Internet protocol
- Four *models*:
  - ◆ *information model*— name directory data
    what you can put in
    directory
  - ◆ *naming model*—how
  - ◆ *functional model*—what
    you can do with data
  - ◆ *security model*—no
    unauthorised access
- LDAP Data Interchange Format (LDIF), a standard text format for representing directory data
- LDAP *server software*
- *command line utilities* (`ldapsearch`, `ldapmodify`,...)
- LDAP API

The LDAP Protocol

- LDAP is a *message-based* protocol
  - ◆ client sends one or more requests to server, one message per request
    - ■ Each message has its own *message ID*
  - ◆ server replies with one or more replies. Each reply has message ID matching that of request.
  - ◆ Can send several messages at once; results can be out of order, no problem

Simple Search Examples

1. Search operation →

2. Returned entry ←

3. Result code ←

LDAP client   LDAP server

- Here a client gets one single entry from the directory

1. Search operation →

2. First entry returned ←

3. Second entry returned ←

...

5. Nth entry returned ←

6. Result code ←

LDAP client   LDAP server

- A client gets multiple responses from the directory

Multiple Simultaneous Requests

1. Search operation, msgid=1
2. Search operation, msgid=2
3. Returned entry, msgid=1
4. Returned entry, msgid=2
5. Result code, msgid=2
6. Result code, msgid=1

LDAP client

LDAP server

- A client sends multiple requests to the directory

- Note that each request has its own `msgid`

- Responses may come out of order (see last two result codes); that's okay.
    - ◆ These details are hidden from programmer by the `SDK` (software development kit)
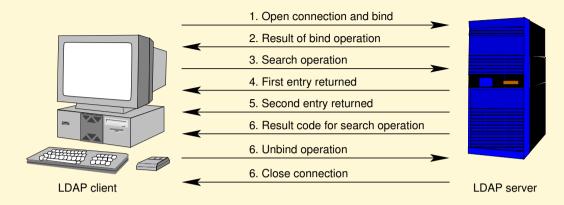
LDAP Protocol Operations

- *Interrogation operations*: `search, compare`
- *Update operations*: `add, delete, modify, modify DN` (rename)
- *Authentication and control operations*: `bind, unbind, abandon`
  bind operation allows a client to identify itself sending identity and authentication credentials
  unbind operation allows client to terminate session
  abandon operation allows a client to tell the server it does not need the results of an operation it had requested earlier

Typical LDAP Exchange

1. Open connection and bind →

2. Result of bind operation ←

3. Search operation →

4. First entry returned ←

5. Second entry returned ←

6. Result code for search operation ←

6. Unbind operation →

6. Close connection ←

LDAP client   LDAP server

- The *bind operation* provides a *distinguished name* (DN) and other credentials to authenticate against the directory
- The *unbind operation* is a request to disconnect

LDAP Encoding: BER

- The LDAP protocol uses the *Basic Encoding Rules*, BER to encode various data types in a platform independent way

- These are the same rules as used in SNMP

- Therefore it is not a simple text-based protocol, like HTTP or SMTP.

LDAP Search Operation

- Used to search for entries and retrieve them
  - This is the only way to read the directory

- Takes eight parameters, including:
  - DN of base object for search — see slide §26
  - search scope — see slide §26
  - search filter — see slide §55
  - list of attributes to return

Search Scope

dc=ict,dc=edu,dc=hk

ou=people

Search scope = *base*

dc=ict,dc=edu,dc=hk

ou=people

Search scope = *one*

dc=ict,dc=edu,dc=hk

ou=people

Search scope = *subtree*

- In each case, the search base is
  `ou=People,dc=ict,dc=edu,dc=hk`

The Compare Operation

- Not very useful
- I use it for determining if a user belongs to a particular group
- main difference from search:
  - If compare on an attribute that does not exist in a particular entry, returns code indicating this
  - If search for an attribute that does not exist in a particular entry, then get nothing returned.

Add Operation

- Creates a new entry, given two parameters:
  - DN of new entry
  - list of attributes and their values to put in the new entry
- Will succeed if and only if:
  - parent of new entry exists
  - no entry of same name exists
  - new entry matches requirements of schemas
  - access control allows operation

Delete Operation

- Deletes an entry
- Takes DN of entry to delete
- Succeeds if:
  - entry exists
  - entry has no children
  - access control allows operation

Modify DN (Rename) Operation

- Used to rename or move an entry from one place in tree to another
- Has four parameters:
  - Old DN
  - New DN
  - New RDN for entry
  - optional flag indicating whether to delete the old RDN attribute from the entry
- Succeeds if:
  - entry exists
  - new name not already used
  - access control allows operation

Modify Operation

- Allows updating existing entry
- Can add, delete or replace attributes
- Can modify many attributes in one modify operation
- Succeeds if and only if:
  - entry exists
  - all attribute modifications must succeed
  - resulting entry obeys schemas
  - access control permits modification

Bind Operation

- authenticates client to the directory
- Three bind types:
  - ◆ *simple bind*, where send DN and password in clear text to server
    - Need to use TLS to encrypt communication in this case
  - ◆ SASL bind
    - SASL = Simple Authentication and Security Layer
    - A standard protocol independent way of negotiating and performing authentication
  - ◆ *anonymous bind*, where send DN but no passwords
- Client can bind, perform operations, bind again, and perform other operations

Command Line Utilities

- With Open<small>LDAP</small>, the main utilities (in RH Linux, in the package `openldap-clients`) are:

  ldapsearch Query directory

  ldapmodify Perform the modify operation on an entry — see §38

  ldapdelete Delete an entry

  ldapadd Add an entry

  ldapmodrdn Rename an entry

  ldapcompare Compare operation

  ldappasswd Change <small>LDAP</small> password using LDAPv3 Password Modify (RFC 3062) extended operation

- Each one has a detailed `man` page

Common Parameters

- All commands use the SASL (Simple Authentication and Security Layer) protocol by default
  - ◆ But won't work in HKIVE Tsing Yi:
  - ◆ . . . we use simple authentication here (we send plain text passwords over link encrypted with Transport Layer Security i.e., TLS or SSL)
- "-x" use simple authentication instead of SASL
- specify hostname of server with -h, e.g., `-h ldap.vtc.edu.hk`
- Specify a DN to bind with using -D (see §38)
- Specify a password on command line with -w ⟨*password*⟩ or interactively prompt using -W
  - ◆ See §38, §64 for examples

`ldapsearch`

- Specify *base* of search with `-b` ⟨*DN of search base*⟩
  - ◆ Default can be specified as a line in `/etc/openldap/ldap.conf`, e.g.,

    `BASE dc=tyict,dc=vtc,dc=edu,dc=hk`

    `HOST ldap.tyict.vtc.edu.hk`

- Specify *scope* of search with `-s [base|one|sub]`
  - ◆ Default scope is subtree scope

- See §61 for more examples.

LDAP Data Interchange Format LDIF

- A standard defined in RFC 2849

- Used to import, export directory data in a standard way
  - ◆ A bit like how all spreadsheets understand tab-delimited text files

- Can also specify update operations to directory entries.

Example LDIF

```
dn: uid=nicku,ou=People,dc=ict,dc=vtc,dc=edu,dc=hk
uid: nicku
cn: Nick Urbanik
givenName: Nick
sn: Urbanik
mail: nicku@sysadmin.no-ip.com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
loginShell: /bin/sh
uidNumber: 1000
gidNumber: 1000
homeDirectory: /opt/nicku
mail: nicku@nickpc.tyict.vtc.edu.hk
description: Interested in free software
```

Update Operation in LDIF

```
$ cat /tmp/update-nick.ldif
dn: uid=nicku,ou=People,dc=ict,dc=vtc,dc=edu,dc=hk
changetype: modify
replace: mail
mail: nicku@nicku.org
—
add: title
title: No longer a lecturer in Hong Kong
—
add: jpegPhoto
jpegPhoto:< file:///tmp/penguin.jpg
—
delete: description
—

$ ldapmodify -x \
-D 'uid=nicku,ou=People,dc=ict,dc=vtc,dc=edu,dc=hk' \
-W -f /tmp/update-nick.ldif
Enter LDAP password:
modifying entry "uid=nicku,ou=People,dc=ict,dc=vtc,dc=edu,dc=hk
```

LDAP Schemas

- The directory has a set of rules that determine the allowed objectclasses and attributes
- Called the *schemas*
- Can be defined in
  - ASN.1, or
  - University of Michigan style, or
  - LDAPv3 style
- Each object, and its syntax, are both defined using OIDs, as in SNMP.

Side track on Object IDs

- LDAP uses a tree structure of *Object IDs* (*OIDs*), the same as SNMP, to identify objects and attributes

- Better not to invent your own to avoid clashing with those used in other schemas

- Apply to Internet Assigned Numbers for your own enterprise number
    - from Application Forms choose
      Private Enterprise Numbers (SNMP)

- See ours (11400) at IANA `http://www.iana.org/ assignments/enterprise-numbers`, grep for `nicku`.

Tree of object IDs

Attributes — Defined in Schema

- For each attribute, schema defines:
  - Name
  - Description
  - Permitted compare operations
  - Syntax (i.e., data type).

- LDAP server ensures that all added data matches the schema

- Each attribute belongs to one or more **objectClass**es
- objectClasses are defined in schemas
- Defines what attributes *must*, or *may* be present in an entry
- objectClass definition includes:
  - ◆ Name of objectClass
  - ◆ What subclass this is derived from
  - ◆ The type of objectClass: *structural*, *auxiliary* or *abstract*
  - ◆ Description
  - ◆ List of *required* attributes
  - ◆ List of *allowed* attributes

Object Class and Attributes

- The entry can use all the attributes allowed in all the objectClasses.
  - ◆ See in slide §50 how LDAP attributes differ from attributes in, say, a Java class

LDAP Object Class Inheritance

- **LDAP** implements a limited form of object oriented inheritance

- One entry may contain many objectClasses
  - ◆ We say, "an entry belongs to many classes"

- Cannot override any schema rules defined in superior class

- Example: top ◁— person ◁— organizationalPerson ◁— inetOrgPerson
  - ◆ In `/etc/openldap/schema`, `core.schema` defines person, organizationalPerson; `inetorgperson.schema` defines inetOrgPerson

- A class derived from another class includes the attributes of its superior class(es)

LDAP Object Class Type

- objectClass has a type: *structural*, *auxiliary*, or *abstract*

- Default is *structural*

- **Structural** is for the fundamental, basic aspects of the object, e.g., **person**, **posixGroup**, **device**.

- **Auxiliary** classes place no restrictions on where an entry is stored, and are used to add more attributes to structural classes.

- **Abstract** classes are not usually created by users; the class **top** and **alias** are abstract.

Structural Classes

■ Rule of `LDAP` standards: if an entry belongs to more than one *structural* class, they must be related by inheritance
  ◆ Open`LDAP` 2.0.x does not implement this restriction, but Open`LDAP` 2.1.x and later versions (including 2.2.x) do.

■ To get around this, can either:
  ◆ Implement a new objectClass that is of type auxiliary that allows the attributes you require—see
    `http://www.openldap.org/faq/data/cache/883.html`
  ◆ Implement a new objectClass that inherits from both unrelated structural classes and use that—See
    `http://www.openldap.org/faq/data/cache/807.html`
    .

Entries: Selecting Object Class Types

- Entries contain one or more *objectClass*es
- Choose the attributes you need
- Select the objectClasses that provide these attributes
- Add the objectClass to your entry.

Rules for LDAP Entries

- Each entry must be a member of the objectClass `top`
- Each entry must be a member of the objectClass that provides the attributes
- Exactly one objectClass should be structural, the rest auxiliary (or abstract)
  - ◆ An entry may belong to more than one structural class if all structural classes are related by inheritance

Namespace of attributes

- There is only one namespace for attributes
- The definition of the attribute `cn` (common name) is the same for all objectClasses that support the `cn` attribute.

Example objectTypes

- Here is the definition for person from core.schema:

```
objectclass ( 2.5.6.6 NAME 'person'
   SUP top STRUCTURAL
   MUST ( sn $ cn )
   MAY ( userPassword $ telephoneNumber $
        seeAlso $ description ) )
```

- This says a person entry *must* contain:
  - ◆ a surname (`sn`) and
  - ◆ common name (`cn`),

- and *may* contain a userPassword, a telephoneNumber, a description, and a reference to another `LDAP` entry.

Want to support network login

- Does the objectClass person provide what is needed for network login?

- For network accounts, need replace (at minimum):

    - `/etc/passwd`   - `/etc/group`
    - `/etc/shadow`

- So in addition to attributes of person, need:

    - User ID name (log in name)
    - User ID number
    - Primary group ID number

    - Gecos information (fifth field of `/etc/passwd`)
    - Home directory
    - Login shell

- Also the password aging information from `/etc/shadow`

Supporting network login

■ **Use the existing objectClass** `posixAccount`:

```
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 ) )
```

■ **Provides fields from** `/etc/passwd`

Authorisation as well as authentication

- Suppose you have an online web-based quiz, want only staff, or year 3, group W, CSA student to be allowed to log in.

- For this to work:

- Each person entry has attributes including:
  - ◆ `Course`, e.g., 41300
  - ◆ `classCode`, e.g., W
  - ◆ `Year`, e.g., 3
  - ◆ `acType`, e.g., STU or STF

LDAP filters

- **LDAP** provides a standard method for selecting authenticated users who match authorisation criteria

- The filter to select staff or students in year 3, CSA, group W is:
  ```
  (|(acType=STF)
  (&(year=3)(course=41300)(classcode=W)))
  ```
  (This line is wrapped to fit on the slide, but normally given on one line)

- All filters are enclosed in parentheses

- Filters can be combined with OR '`|`', AND '`&`'

Find this in `/usr/share/doc/openldap-devel-2.2.23/rfc/rfc2254.txt`

```
filter      = "(" filtercomp ")"
filtercomp = and / or / not / item
and         = "&" filterlist
or          = "|" filterlist
not         = "!" filter
filterlist = 1*filter
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     = value
any         = "*" *(value "*")
final       = value
attr        = AttributeDescription from
              Section 4.1.5 of [1]
value       = AttributeValue from
              Section 4.1.6 of [1]
```

- **[1] is RFC 2251.**

- **Grammar is defined in RFC 822**

Examples of Filters from RFC 2254

Return all entries in the scope of the search with attribute `cn` having the value "Babs Jensen":

`(cn=Babs Jensen)`

Return all entries in the scope of the search which do `not` have the attribute `cn` with the value "Tim Howes":

`(!(cn=Tim Howes))`

Return all entries in the scope of the search which have the attibute

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

Return all entries having an attribute `o` (i.e., organisation) which contains the strings `univ`, `of`, `mich` with zero or more of any characters between, and with any number of any characters at the end.

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

More Filter Examples

- Note that a filter such as `(age>21)` is not allowed.
- Use `(!(age<=21))` instead.
- Similarly, instead of `(age<21)`, use `(!(age>=21))`.
- search for all students in group X, year 3, CSA course, who enrolled this year:

  ```
  (&(year=3)(course=41300)(classcode=W)
  (registrationDate=*-03))
  ```

  Note that there is a substring match on `registrationDate` here. A substring match is like a wildcard in filename matching.

Escaping Characters in a Filter

Character	Escape Sequence
`*` (asterisk)	`\2A`
`(` (left parenthesis)	`\28`
`)` (right parenthesis)	`\29`
`\` (backslash)	`\5C`
`NUL` (the null byte)	`\00`

Using the command line tool ldapsearch

- `$ ldapsearch -x -h ldap.vtc.edu.hk \`
  `-b "dc=vtc.edu.hk" \`
  `"(&(department=ICT)(site=TY)`
  `(|(acType=STF)`
  `(&(year=3)(course=41300)(classcode=W))))" cn`

- The result is a list of all the DNs that match the filter, with the students' and staff names.

- Can filter out the DNs and blank lines by piping the command though `grep '^cn:' | sort`

Output of this `ldapsearch` without staff

```
cn: CHAN Kwok Kam
cn: CHEUK Suk Lai
cn: CHUNG Ming Kit
cn: LAI Man Chiu
cn: LAM Lai Hang
cn: LAU Siu Ying
cn: LAW Yuk Woon
cn: LI Kim Wah
cn: LI Siu Kai
cn: LI Yuet Cheung
cn: MA Hei Man
cn: MO Hoi Yu
cn: POON Chun Chung
cn: TAM Kin Fai
cn: TSO Yee Yee
cn: WONG Chi Man
cn: WONG Hoi Shan
cn: WONG Siu Fai
cn: WOO Kin Fan
```

Get All the Results

```
$ ldapsearch -x -h ldap.vtc.edu.hk -b 'dc=vtc.edu.hk' \
"(&(department=ICT)(site=TY)(|(actype=STF)(&(year=3)
(classcode=W)(course=41300))))" cn \
| grep '^cn: ' | sed 's/^cn: //;s/^\(.\{15\}\).*/\1/' | sort | column
```

Andy LAI	C M Ho	LEE HUNG KIN	SIU CHONG PUI
CHAN CHIN PANG	Curtis H.K. Tsa	LEE KOON HUNG K	SIU WAI CHEUNG
CHAN Kwok Kam	Esther YUEN	LEUNG KAM SHEK	Stella Chu
CHAN KWOK KEUNG	Eva Chung	LI Kim Wah	TAM CHI HO
CHAN SHIU CHUAN	FONG CHI KIT	LI Siu Kai	TAM Kin Fai
CHAN TAI HING	Henry Leung	LI Yuet Cheung	TSANG KWOK TUNG
CHAN TAI MING R	HO CHUN WAH	MA Hei Man	TSO Yee Yee
Charles Wu	HO KIM MAN ALBE	MA SUI WAH	WONG Chi Man
CHEUK Suk Lai	Josephine Wan	MICHAEL LEUNG	WONG Hoi Shan
CHEUNG KAM HOI	Karl Leung	MO Hoi Yu	WONG Siu Fai
CHEUNG SAI MING	Ken LI	MONTAGUE NIGEL	WONG WAI YIP FR
CHIK FUNG YING	Kit K. KO	NG HOI KOW	Wong Y.L. Lawre
CHIU SUET FAN J	LAI HING BIU	NG SZE CHIU EDD	WOO HUNG CHEUNG
Chou Siu Chuen	LAI Man Chiu	Nick Urbanik	WOO Kin Fan
CHUNG Ming Kit	LAM Lai Hang	PATRICK K.S. TO	YIM KWOK HO
CHU SHING TSU J	LAU KWOK ON	POON Chun Chung	Y.K. Leung
Clarence Lau	LAU Siu Ying	Rick Liu	
Clarence Lo	LAW Yuk Woon	SCOTT ALBERT HE	

Samba

ldapsearch

- Needs the `-x` option to work here
- Check ssl works with the `-ZZ` option
- Can "bind" as a user to get all the info you are allowed to see after binding:

```
$ ldapsearch -x -W -D \
"uid=nicku,ou=People,dc=tyict,dc=vtc,dc=edu,dc=hk" \
'(uid=nicku)'
```

- Can then see own passwords

LDAP URLs: RFC 2255

- Have the form:
- ldap://⟨*host*⟩:⟨*port*⟩/⟨*base*⟩?⟨*attr*⟩?⟨*scope*⟩?⟨*filter*⟩

```
ldapurl    = ldap://" [hostport] ["/"
                [dn ["?" [attributes] ["?" [scope]
                ["?" [filter] ["?" extensions]]]]]]
```

- The ⟨*base*⟩ or dn is the distinguished name of the starting entry for your search.
- ⟨*scope*⟩ is one of base, one or sub
- Examples:

```
ldap://ictlab/ou=People,dc=tyict,dc=vtc,dc=edu,dc=hk?uid?one?(uid=nicku)
```

`mod_auth_ldap` with Apache

- **`mod_auth_ldap` is part of the `httpd` RPM package on Fedora Core versions 1 to 4.**

- **Here we allow staff or students from group W, year 3 CSA to access the web pages under `http://hostname/group-w/` if the user provides a correct password:**

```
<Location "/group-w">
        AuthType Basic
        AuthName "\LDAP authentication to class W only"
        AuthLDAPURL ldap://ldap.tyict.vtc.edu.hk/
ou=People,dc=tyict,dc=vtc,
dc=edu,dc=hk?uid?one?(|(acType=STF)(&(course=41300)
(classCode=W)(year=3)))
        require valid-user
</Location>
```

- **See `http://httpd.apache.org/docs-2.0/mod/mod_auth_ldap.html`, and also `http://httpd.apache.org/docs-2.0/mod/mod_ldap.html` for manual.**

Authenticating web pages—continued

- Unfortunately, `mod_auth_ldap` disappeared from Red Hat 8.0 and 9, to reappear in Fedora Core but not RHEL 3, where another module was provided that did not work the same.

- I ended up modifying Apache::AuthNetLDAP (available with my changes from `CPAN`)

- I used that on our servers in the department.
  - ◆ . . . a more portable method of authentication, *provided* we are using `mod_perl`

- I haven't tried it with the final `mod_perl` version 2 on FC4 yet.

ICT Schema Design

Authorisation of Students and Staff

- We need a new schema to support the required attributes
- We create three new objectClasses and associated attributes:
- The first is common to students and staff:

```
objectclass ( 1.3.6.1.4.1.11400.2.2.1 NAME 'institute'
   SUP top AUXILIARY
   DESC 'Any person in the institute, staff or student'
   MAY ( acOwner $ acType $ answer1 $ answer2 $
        answer3 $ batchUpdateFlag $ department $
        site $ instituteEmail ) )
```

- See slides 40–41 for more about the funny numbers.

Other objectTypes for IVE

- **Then on top of this, we have attributes for students:**

```
objectclass ( 1.3.6.1.4.1.11400.2.2.2 NAME 'student'
        SUP top AUXILIARY
        DESC 'A student in the institute'
   MAY ( academicYear $ award $ classCode $ course $
        courseDuration $ FinalYear $
        registrationDate $year $
        fullPartTime ) )
```

- **. . . and staff:**

```
objectclass ( 1.3.6.1.4.1.11400.2.2.3 NAME 'staff'
        SUP top AUXILIARY
        DESC 'A staff member of the insitute.'
   MAY ( titleDes $ employerID ) )
```

The whole schema for IVE

- ## The whole schema can be seen here:
  `http://ictlab.tyict.vtc.edu.hk/oids/institute.schema`
- ## If planets are alligned, then this local link will work.

Case Study: ICT laboratories

- Old system:
- An ancient DEC Alpha running NIS
- Hardware insufficient for demand
- *Very* expensive maintenance, stopped paying
- Technician reported a hardware failure close to first day of term
- New system:
- We were planning to introduce LDAP authentication gradually
- Failure required planning move faster
- Needed to maintain old legacy accounts, plus introduce new accounts

ICT case study

- We chose Open`LDAP` on Linux
- Running on an Acer Altos dual `CPU` P-III
- Migrated from the `NIS` using the migration scripts provided with Open`LDAP`
- Migrated from the `VTC` `LDAP` accounts using a Perl program, written (quickly!) for the purpose,
- Uses the `Net::LDAP` Perl modules

- After migrating the legacy accounts, and creating new accounts for staff, full and part time students, had more than 5000 accounts
- The `LDAP` server was using a high `CPU` load
- Was able to solve this using caching:
- Use `nscd` (name service caching daemon) on client
- Use memory in server to increase local cache size drastically.
- `CPU` load reduced to a very acceptable level.

- The ICT LDAP server namespace design:

dc=tyict,dc=vtc,dc=edu,dc=hk

ou=people   ou=group   ou=devices

uid=albertho · · · uid=nicku   cn=staff · · · cn=students   cn=ictlab · · · cn=printer7

- We chose a fairly flat directory structure
- Recommended by reference, pages 239, 249.
- Reason: flexibility:
- allows for change without major reorganisation of data.

Hierarchical Directory Structure

■ This directory structure is hierarchical:

dc=tyict,dc=vtc,dc=edu,dc=hk

- **ou=TY**
  - **ou=ICT**
    - **cn=people**
      - uid=albertho
      - uid=nicku
    - **cn=group**
      - cn=staff
      - cn=students
    - **cn=devices**
      - cn=ictlab
      - cn=printer7
  - ...
  - **ou=ENG**
- **ou=MH**
  - **ou=ICT**
  - ...
  - **ou=ENG**
- **ou=TM**
  - **ou=ICT**
    - **cn=people**
    - **cn=group**
    - **cn=devices**
  - ...
  - **ou=ENG**

New VTC LDAP Namespace

- This new VTC LDAP namespace was introduced in April 2003:

Hierarchical Directory Structure

- This is an alternative data arrangement
- Divide into different campuses
- Advantage: can easily delegate management to local campus
- But: suppose ENG changes to EE?
- Suppose staff move from one department to another?
- Suppose equipment is transferred?
- Not only need change the attributes in the entry, but also move the entry.
- Overall, a flatter structure is easier to manage.

Directory Design Guidelines

- **Design as flat as possible given constraints:**
- **Replication**
- **Access Control**
- **Limitations of directory software**
- **Requirements of applications that use the directory**

Designing a Schema

- After selecting the schema attributes needed for your application, you may find that not all are available with the server

- Search web for more schemas

- If none provide all you need,

- Select a suitable structural base class

- Create an auxiliary class to be used with the base class

- Define the objectClass and its attributes

Designing a Schema: Example

- For our ICT LDAP server, we use enough attributes to be able to log in

- But we also want to select users on the basis of course, year, class

- Want to add these attributes to the existing objectClasses

- Create three object classes:

- Institute

- Student

- Staff

Maintenance

Building the original directory

- I built the original directory from the old failing NIS data, using some modified `padl` import scripts
- Then quickly wrote a nasty Perl script that reads the LDAP data from the VTC directory server, and builds POSIX accounts from that data
- The nasty Perl script stuck around, and we used it ever since.
- I extended it to read the student enrollment data directly.
  - ◆ ... this was only available in "unparseable" PDF files with about 7–10 students per A3 page!
- Henry now uses the Perl programs written by Gerald Carter that come with `samba`.

Problems and solutions

How we started up

- The original machine was an Asus Altos P-III with 256 MB RAM

- Running Red Hat 7.1, openldap 2.0.x.

- Was providing:
  - ◆ Home directories by NFS
  - ◆ Web service to the Internet
  - ◆ telnet :-(and SSH login to students to do their programming assignments on
  - ◆ . . . and now LDAP accounts for all our students (there were 5000 accounts).

Performance

Problems

- `CPU` load would get very high when assignments were left with tight, infinite loops (a load average of 10 or so)

- `CPU` load would get very high when classes logged in (a load average of about 4–6.

- Occasionally the `CPU` would go up to 10 and stay there solidly for a while, and the load would be all from `slapd`.

Solutions

- Openldap came with a tiny default value of memory and disk caching. Needed to increase this to a much higher level. In `/etc/openldap/slapd.conf`:
  ```
  cachesize 100000
  dbcachesize 25600000
  timelimit 60
  ```

- index for all the common searches your system will do
  - ◆ Enable logging of all search filters
  - ◆ index almost every attribute you see being searched for.

- Enable caching on the clients
  - ◆ Turn on the Name Service Caching Daemon (`nscd`)

- Add `RAM` to the directory server (We added to a total of 1GB)

- We didn't do this, but obviously, use replication to two or more `LDAP` servers, one master, others slaves, and round-robin `DNS` to select directory server.

The FAM storm problem

- An amazing problem occurred when older Red Hat (about 7.2) client machines were booted: the `fam` daemon (file alteration monitor) on the client would be involved in causing a storm of `LDAP` requests that would drive the `CPU` usage of the server to stratospheric limits.

- Used cricket `http://cricket.sourceforge.net/` to monitor `CPU` and network usage on server
  - ◆ See my notes on cricket, snmp, snmp version 3 and *all* my *free* network management notes `http://nicku.org/snm/`

- Wrote a perl program to watch the `LDAP` logs and send me an email if any problems.

- Upgraded clients to a later version of Red Hat, or turned off the `fam` service.

Problem with automounter

- We used the automounter to mount home directories when people log in
- The automounter uses `LDAP` version 2
- OpenLDAP 2.x.y, where $x > 0$ defaults to version 3 *only*
- Need enable `LDAP` version 2 in `/etc/openldap/slapd.conf` with
  `allow bind_v2 bind_anon_dn`

Problem with shared Gconf data

- When people log in twice both using Gnome, then things go horribly weird

- From memory: panel does not work properly, clicking on some things don't work.

- The problem appears to be that the same Gconf data is shared out over NFS, and there is a file lock to ensure exclusive access.

- I haven't found a work around except KDE or something that does not use GConf.

- I'd be very grateful for ideas here.

- It is now running nicely on a single P4 system that my friend Henry built, running a RHEL 3 clone (actually, the Department paid for a RHEL license, but never perhaps got around to using it)

- Just before I left, I tendered for an Adaptec hardware cluster system suitable for running the Red Hat cluster manager

- My friend Henry has been too busy to get it up and running. Besides, the old P4 system works well.

- Students do not log into the server very often any more. Better to ban this completely right from the start!

- If I started from scratch again, I would use a shorter DN suffix: `dc=tyict,dc=vtc,dc=edu,dc-hk` simply adds unnecessary bulk to the directory storage on disk. I would have used `o=ICT`.

- At home I use `dc=nicku,dc=org` which is not too much to type.

Samba gotchas

- **Refer to the latest version of**

  Samba-3 by Example: Practical Exercises in Successful Samba Deployment

  `http://samba.mirror.aarnet.edu.au/samba/`
  `docs/man/Samba-Guide/`

- **Carefully follow the steps in the section**

  Samba Domain with Samba Domain Member Server Using LDAP

  `http:`
  `//samba.mirror.aarnet.edu.au/samba/docs/man/`
  `Samba-Guide/unixclients.html#sdcsdmldap.`

- **You need to set up the smbldap tools so that they do not get overwritten as your samba setup is updated.**

- **The computers needed to be put in the same place (in the directory tree) as user accounts (this was true for recent versions of samba)**

- **You don't need `winbind` if you are running a samba `PDC`, only if you want to authenticate Linux machines to a Windows server.**

The Administrator account

- The biggest concern to me has been putting a root account in the directory

- My conscience screams at me!

- Latest samba supports non-root administrators for joining machines to the domain.

- I haven't tried that yet.

- Other concern: `smbldap` tools need to read administrator password

- So does samba

- samba reads it from `/etc/samba/secrets.tdb`

- Nice to write a program to read it from there for `smbldap` tools, so only need maintain it in one place.

- My crude attempt used `tdbdump`, part of samba.

Didn't include

- replication
- distributed directories
- access control lists (for examples, see slapd.conf on ictlab, slapd.conf on nicku, and the
  program to make ACLs for student `LDAP` workshops).
- How the automounter is set up to run from `LDAP`
- simplicity of client setup using `authconfig` (or kickstart) with Red Hat/Fedora
- Setting up local user accounts and network user accounts
- Use of GQ or directory_administrator, `LDAP` Account Manager
- high availability
- the fabulous new Fedora Directory Server
- Life, the Universe and Everything.

References

- *LDAP System Administration*, Gerald Carter, ISBN 1-565-92491-6, O'Reilly, March 2003

- *Understanding and Deploying LDAP Directory Services* (2nd Edition), Tim Howes, Timothy A. Howes, Mark C. Smith, Gordon S. Good, ISBN: 0672323168, Publisher: Addison Wesley Professional, May 2, 2003

- *Understanding and Deploying LDAP Directory Services*, Timothy Howes, Mark Smith and Gordon Good, Macmillan, 1999. Tsing Yi library: TK 5105.595.H69 1999

- *LDAP Programming, Management and Integration*, Clayton Donley, ISBN: 1-930110-40-5, Manning Publications, 2003

- *LDAP Directories Explained: An Introduction and Analysis*, Brian Arkills, ISBN 0-201-78792-X, Addison-Wesley, 2003.

- *Understanding LDAP Redbook* (registration required): `http://www.redbooks.ibm.com/abstracts/sg244986.html`

- *LDAP Implementation Cookbook Redbook* (registration required): `http://www.redbooks.ibm.com/abstracts/sg245110.html`

- *Implementing LDAP*, Mark Wilcox, Wrox Press, 2000

- The many RFCs are helpful.

The RFCs

You could get a list of (most) of the relevant RFCs with something like this:

```
$ wget -O - ftp://ftp.isi.edu/in-notes/rfc-index.txt 2>/dev/null \
| perl -n00 -e "print if /ldap|lightweight/i and not /obsoleted\s*by/i"
1823 The LDAP Application Program Interface. T. Howes, M. Smith.
     August 1995. (Format: TXT=41081 bytes) (Status: INFORMATIONAL)

2164 Use of an X.500/LDAP directory to support MIXER address mapping.
     S. Kille. January 1998. (Format: TXT=16701 bytes) (Obsoletes RFC1838)
     (Status: PROPOSED STANDARD)

2247 Using Domains in LDAP/X.500 Distinguished Names. S. Kille, M.
     Wahl, A. Grimstad, R. Huber, S. Sataluri. January 1998. (Format:
     TXT=12411 bytes) (Status: PROPOSED STANDARD)

2251 Lightweight Directory Access Protocol (v3). M. Wahl, T. Howes, S.
     Kille. December 1997. (Format: TXT=114488 bytes) (Updated by RFC3377,
     RFC3771) (Status: PROPOSED STANDARD)

2252 Lightweight Directory Access Protocol (v3): Attribute Syntax
     Definitions. M. Wahl, A. Coulbeck, T. Howes, S. Kille. December 1997.
     (Format: TXT=60204 bytes) (Updated by RFC3377) (Status: PROPOSED
     STANDARD)

2253 Lightweight Directory Access Protocol (v3): UTF-8 String
     Representation of Distinguished Names. M. Wahl, S. Kille, T. Howes.
     December 1997. (Format: TXT=18226 bytes) (Obsoletes RFC1779) (Updated
     by RFC3377) (Status: PROPOSED STANDARD)

2254 The String Representation of LDAP Search Filters. T. Howes.
     December 1997. (Format: TXT=13511 bytes) (Obsoletes RFC1960) (Updated
     by RFC3377) (Status: PROPOSED STANDARD)
```

RFC numbers

And while we're crazy, let's see the RFC numbers:

```
$ cat ldap-rfc-list.txt | perl -n00 -e \
'{($rfc)=split;push @R, $rfc}END{print join " ", @R, "\n"'}
1823 2164 2247 2251 2252 2253 2254 2255 2256 2307 2587 2589
2649 2657 2696 2713 2714 2739 2798 2820 2829 2830 2849 2891
2926 2927 3045 3062 3088 3112 3296 3352 3377 3383 3384 3494
3663 3671 3672 3673 3674 3687 3698 3703 3712 3727 3771 3828
3829 3866 3876 3909 3928 4104
$ ls /usr/share/doc/openldap-devel-2.2.23/rfc
INDEX        rfc2293.txt   rfc2798.txt   rfc3296.txt   rfc3703.txt
rfc1274.txt  rfc2294.txt   rfc2829.txt   rfc3377.txt   rfc3712.txt
rfc2079.txt  rfc2307.txt   rfc2830.txt   rfc3383.txt   rfc3727.txt
rfc2247.txt  rfc2377.txt   rfc2849.txt   rfc3663.txt   rfc3771.txt
rfc2251.txt  rfc2587.txt   rfc2891.txt   rfc3671.txt   rfc3829.txt
rfc2252.txt  rfc2589.txt   rfc2926.txt   rfc3672.txt   rfc3866.txt
rfc2253.txt  rfc2649.txt   rfc3045.txt   rfc3673.txt   rfc3876.txt
rfc2254.txt  rfc2696.txt   rfc3062.txt   rfc3674.txt   rfc3909.txt
rfc2255.txt  rfc2713.txt   rfc3088.txt   rfc3687.txt   rfc3928.txt
rfc2256.txt  rfc2714.txt   rfc3112.txt   rfc3698.txt
```