Using LDAP from Java+Tomcat and Python+Zope

Holger Blasum, Andreas Meisl

6th April 2003

(Talk at the GUUG-Frühjahrsfachgespräch Bochum, 28 March 2003.)

Abstract

We report on the implementation and installation and implementation of clients to Novell eDirectory via plain text and TLS/SSL and some performance measurements. Note: You can find source code and supplementary materials at http://www.blasum.net/holger/wri/comp/net/7appl/ldap/bochum2003/.

1 Use case

The Dept of Veterinary Science at Munich University is using Novell eDirectory as directory server for student user data[1]. Once per term, students can subscribe to elective courses; since July 2002 this is done via a web interface[2].

2 Directory services and LDAP

Examples for directories are telephone registers or the DNS system; directories adhere to the hierarchical data model. In 1988, CCITT(ITU) / ISO finalized the standard X.500 on directory services [Kla01, Chad96]. LDAP[3] is a trimmed down variety of the directory accesss protocol (DAP) for X.500 directory services. Originally designed for the replication of directory data, LDAP is now provided as an interface by popular directory services, e.g OpenLDAP[4], Novell eDirectory[5] , or Microsoft Active Directory[6].

In these applications, LDAP is typically used via TCP (although there is also a specification for connectionless UDP[7]).

[1]This setup is quite popular in Bavarian universities, see AK Netz PC, http://www-lan.uni-regensburg.de/ak/netz/.

[2]http://www.vetmed.uni-muenchen.de/studium/wahlpflicht/kurse/index.html

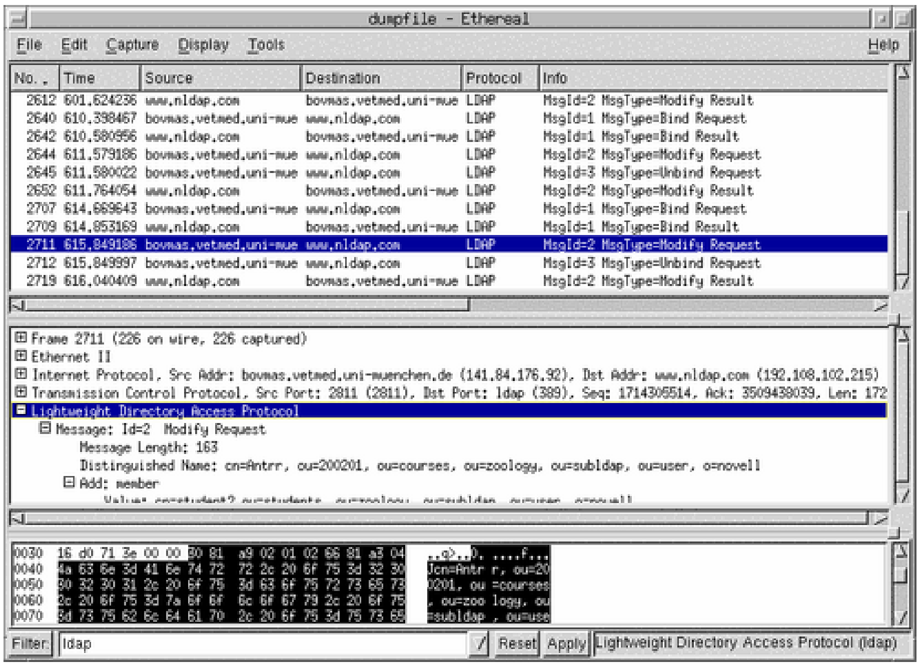
[3]RFC 1777 (Version 2), 2251 (Version 3)

[4]http://www.openldap.org/

[5]previously: Novell Directory Services (NDS), http://www.novell.com/products/edirectory/

[6]http://www.microsoft.com/windows2000/technologies/directory/ad/default.asp

[7]RFC 1798



```
No.  Time       Source                Destination           Protocol  Info
2612 601.624236 www.nldap.com         bovmas.vetmed.uni-mue LDAP      MsgId=2 MsgType=Modify Result
2640 610.398467 bovmas.vetmed.uni-mue www.nldap.com         LDAP      MsgId=1 MsgType=Bind Request
2642 610.580956 www.nldap.com         bovmas.vetmed.uni-mue LDAP      MsgId=1 MsgType=Bind Result
2644 611.579186 bovmas.vetmed.uni-mue www.nldap.com         LDAP      MsgId=2 MsgType=Modify Request
2645 611.580022 bovmas.vetmed.uni-mue www.nldap.com         LDAP      MsgId=3 MsgType=Unbind Request
2652 611.764054 www.nldap.com         bovmas.vetmed.uni-mue LDAP      MsgId=2 MsgType=Modify Result
2707 614.669643 bovmas.vetmed.uni-mue www.nldap.com         LDAP      MsgId=1 MsgType=Bind Request
2709 614.853169 www.nldap.com         bovmas.vetmed.uni-mue LDAP      MsgId=1 MsgType=Bind Result
2711 615.849186 bovmas.vetmed.uni-mue www.nldap.com         LDAP      MsgId=2 MsgType=Modify Request
2712 615.849997 bovmas.vetmed.uni-mue www.nldap.com         LDAP      MsgId=3 MsgType=Unbind Request
2719 616.040409 www.nldap.com         bovmas.vetmed.uni-mue LDAP      MsgId=2 MsgType=Modify Result

⊞ Frame 2711 (226 on wire, 226 captured)
⊞ Ethernet II
⊞ Internet Protocol, Src Addr: bovmas.vetmed.uni-muenchen.de (141.84.176.92), Dst Addr: www.nldap.com (192.108.102.215)
⊞ Transmission Control Protocol, Src Port: 2811 (2811), Dst Port: ldap (389), Seq: 1714305514, Ack: 3509438039, Len: 172
⊟ Lightweight Directory Access Protocol
   ⊟ Message: Id=2  Modify Request
         Message Length: 163
         Distinguished Name: cn=Antrr, ou=200201, ou=courses, ou=zoology, ou=subldap, ou=user, o=novell
      ⊟ Add: member
            Value: cn=student2 ou=students, ou=zoology, ou=subldap, ou=user, o=novell

0030  16 d0 71 3e 00 00 30 81  a9 02 01 02 66 81 a3 04   ..q>..0. ....f...
0040  4a 63 6e 3d 41 6e 74 72  72 2c 20 6f 75 3d 32 30   Jcn=Antr r, ou=20
0050  30 32 30 31 2c 20 6f 75  3d 63 6f 75 72 73 65 73   0201, ou =courses
0060  2c 20 6f 75 3d 7a 6f 6f  6c 6f 67 79 2c 20 6f 75   , ou=zoo logy, ou
0070  3d 73 75 62 6c 64 61 70  2c 20 6f 75 3d 75 73 65   =subldap , ou=use
```

LDAP makes use of ASN.1/BER encoding. A tcpdump (*tcpdump -s 0 -w filename*) of an LDAP session can be inspected with a packet analyzer such as Ethereal[8] that also understands ASN.1/BER and LDAP application classes. On the screeshot you see a *modify* request of LDAP version 3.

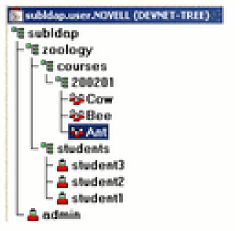
3 A sand box

Both OpenLDAP as well as eDirectory a freely available (free as speech in the former, free as beer in latter case). As we just want to test clients, the most simple thing to do is to use the test site provided by Novell[9].

[8] http://www.ethereal.com/

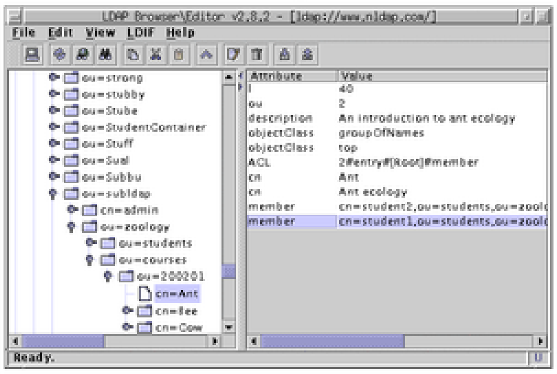
[9] http://www.nldap.com/, in nldapaccount.txt you will find complete access data (*cn/userPassword*) for the current account. If you like to set up an account yourself, just follow the links *eDirectory Account Adminstration->create NDS Container & its admin user*.

Nwadmin



```
subldap.user.NOVELL (DEVNET-TREE)
subldap
├ zoology
│ ├ courses
│ │ └ 200201
│ │   ├ Cow
│ │   ├ Bee
│ │   └ Ant
│ └ students
│   ├ student3
│   ├ student2
│   └ student1
└ admin
```

LDAP Browser/Editor



LDAP Browser\Editor v2.8.2 - [ldap://www.nldap.com/]

```
ou=strong
ou=stubby
ou=Stube
ou=StudentContainer
ou=Stuff
ou=Sual
ou=Subbu
ou=subldap
  cn=admin
  ou=zoology
    ou=students
    ou=courses
      ou=200201
        cn=Ant
        cn=Bee
        cn=Cow
```

Attribute	Value
l	40
ou	2
description	An introduction to ant ecology
objectClass	groupOfNames
objectClass	top
ACL	2#entry#[Root]#member
cn	Ant
cn	Ant ecology
member	cn=student2,ou=students,ou=zoolc
member	cn=student1,ou=students,ou=zoolc

Ready.

Once you enter this sandbox via a GUI such as the *LDAP browser/editor*[10], or Novell's native tools (nwadmin, Console1), you can easily grasp the simple structure of the sandbox. There a zoology courses that can be registered by students (Register/Unregister). Alternatively, via the command line, one can easily generate a an LDIF file (that is also human readable):

```
    ldapsearch -vLx -b "ou=subldap, ou=user,
    o=novell" -h www.nldap.com
    "(objectclass=*)" > subldap.ldif
```

4 Simple LDAP clients from the command line interface

4.1 Installation

Java: We have used the Blackdown port[11], preferably Java 1.4.1[12]. The next step is to get a copy of the Java Novell Developer Kit[13], but the only file we really need is *ldap.jar*, that we install to */usr/lib/j2sdk1.4.1/jre/lib/ext/* (or adapt the class path alternatively).

Python: We use Python 2.1.3 and the python-ldap library[14] as of release 2.0.0pre06 (that one goes to */usr/lib/python2.1/site-packages*).

Server; OS: Additionally to the sandbox that we have already described, we used an eDirectory 8.6.2 auf Netware 5.1[15] as directory server, python and java middleware had been installed on a Debian Woody GNU/Linux 2.4.19 i686.

[10]http://www.iit.edu/ gawojar/ldap/

[11]http://www.blackdown.org/

[12]For Java 1.3 one has to supplement the Java Secure Socket Extension (JSSE) 1.0.2 into the Java extension directory; JSSE is already included in Java 1.4.

[13]http://developer.novell.com/ndk/jldapunx.htm, 15 November 2002, also linked from http://www.openldap.org/jldap/. Documentation and examples also can be found in that package; extensive printed documentation is also available in [Har00]).

[14]http://python-ldap.sourceforge.net/

[15]eDirectory also runs on AIX, Linux, Solaris, Windows.

4.2 View course list (anonymously)

We want to write a client that anonymously views the content of the course directory, so we have the following tasks:

- initialize LDAP connection
- send LDAP search request to server
- print the list of results *res*
- close connection

Let's begin doing this in python (ListCourses.py):

```
#!/usr/bin/python
import ldap
l=ldap.initialize("ldap://www.nldap.com:389")
res = l.search_s("ou=200201, ou=courses,
  ou=zoology, ou=subldap, ou=user,
  o=novell", ldap.SCOPE_ONELEVEL,
  "objectclass=*")
for r in res:
  print r[0]
c.unbind_s()
```

Giving us (obviously):

```
blasum@promitheas:$ ./ListCourses.py
cn=Ant,ou=courses,ou=zoology,ou=user,o=NOVELL
cn=Bee,ou=courses,ou=zoology,ou=user,o=NOVELL
cn=Cow,ou=courses,ou=zoology,ou=user,o=NOVELL
```

Note: "_s" used in the commands chooses the synchronous connection mode (client waits until data has arrived). In the JLDAP API the choice of synchronous/asynchronous is simply done by method overloading. The java client that here, as well as in all other examples, also acts synchronously, looks like this (ListCourses.java):

```
import java.io.*;
import com.novell.ldap.*;
public class ListCourses {
  public static void main (String [] args)
    throws LDAPException {
    LDAPConnection c=new LDAPConnection ();
    c.connect ("www.nldap.com", 389);
    LDAPSearchResults res = c.search(
      "ou=200201, ou=courses, ou=zoology,
      ou=subldap, ou=user, o=novell",
      LDAPConnection.SCOPE_ONE, null, null,
      false);
    while (res.hasMore()) {
    System.out.println(res.next().getDN());
    c.disconnect();
```

```
        }
      }
    }
```

Disgression: If we eager to write python-like, but to have it interpreted by Java, one can use the jython interpreter[16]. ListCourses.jy then (a nice mixture of Python and LDAP) has this flavor:

```
from com.novell.ldap import *
c = LDAPConnection ()
c.connect ("www.nldap.com", 389)
res = c.search("ou=200201, ou=courses,
  ou=zoology, ou=subldap, ou=user,
  o=novell",
  LDAPConnection.SCOPE_ONE, None, None, 0)
while res.hasMore():
   print res.next().getDN()
c.disconnect()
```

Observation: Running java took 1.0 seconds, its python counterpart consumed 0.14 seconds; jython got the job done in 4.5 seconds (+ ca. 0.5-1 seconds response time for the transcontinental LDAP query). The interfaces of JLDAP and python-ldap look very similar, with python-ldap being somewhat leaner.

4.3 Change subscription data (as administrator)

We want to use an authenticated connection to modify entries in the directory. This boils down to the following steps:

- initialize LDAP connection
- authenticate to server
- send *modify* request (consisting of a list of modifications)
- close connection

Here it is in python (Register.py):

```
import ldap
import sys
if len (sys.argv) != 3:
  print "Usage: Register.py course student"
  sys.exit(1)
ldap.set_option (ldap.OPT_PROTOCOL_VERSION, ldap.VERSION3)
c=ldap.initialize("ldap://www.nldap.com:389")
c.simple_bind_s("cn=admin, ou=subldap,
  ou=user, o=novell", "secret")
c.modify_s ("cn=" + sys.argv[1] + ", " +
  "ou=200201, ou=courses, ou=zoology,
   ou=subldap, ou=user, o=novell",
```

[16]http://www.jython.org/

```
  [(ldap.MOD_ADD, "member", "cn=" +
  sys.argv[2] + "," + "ou=students,
  ou=zoology, ou=subldap, ou=user,
  o=novell")])
c.unbind_s()
```

The inverse operation looks like Unregister.py, except for that here we replaced *MOD_ADD* by *MOD_DEL*. The java versions Register.java and Unregister.java also just differ in *ADD* vs *DELETE*.

Observations: time consumption is 1.2 seconds for JLDAP, 0.11 seconds for python-ldap. When using eDirectory one needs to set the LDAP protocol version explicitly to 3 (otherwise *simple_bind_s* silently fails in python-ldap, whereas JLDAP has set vesion 3 as its default). When *simple_bind_s* succeeds, but the subsequent *modify* request fails, we get an exception (to keep it simple, we do not handle it here).

4.4 Squeezing in an abstraction layer: *ChangeManager*

We don't really want to hardwire library specifics into our application, thus we have an written an abstraction layer (ChangeManager.java / ChangeManager.py) that wraps around LDAP commands (below the presentation logic layer):

```
Python 2.1.3 (#1, Sep 12 2002, 00:24:18)
[GCC 2.95.4 20011002 (Debian prerelease)] on linux2
>>> import ChangeManager
>>> c = ChangeManager.ChangeManager()
>>> courses = c.getCourseList('03')
```

By hiding the entire LDAP interaction into the ChangeManager one should be able to e.g. switch to an entirely different (e.g. relational) data repository.

The entire architecture is thus three-tiered: LDAP/eDirectory as repository, an intermediate layer (ChangeManager) for logic and zope DTML or java servlets for the presentation layer. If you are so inclined, you could roughly map this to the Publisher/Subscriber pattern (e.g. [Gam94]) with subscribers/observers embodied in command line or web interface interaction and the publisher residing in the directory.

5 Presentation layer

5.1 Zope DTML

Naive approach: Using zope 2.5.1 (running on python 2.1) we (quite brutally) copied (well, symlinked) our servlet Courses.py to */usr/lib/zope/Extensions*, and dumpled the infrastructure (all other python classes) to */usr/lib/python2.1/.* The zope instance Zope-Instanz now houses a DTML page (index.html) that via

```
<dtml-var expr="doGet( term=REQUEST['term'],
course=REQUEST['course'], student=REQUEST['student'],
password=REQUEST['password'],
raction=REQUEST['raction'])">
```

calls a zope *External Method* (*doGet*), that represents a reference to the method *doGet* in Courses.py. If we want changes in Courses.py to take effect, we have to restart zope.

Integration as Zope component (”product”): In the product Courses we leave the LDAP connection open after usage and reopen it only if it does not persist any longer (self.c == None). We will see that this technique (learned from the LDAPUserFolder zope product) will save TLS/SSL handshakes.

5.2 Tomcat Servlets

The most popular provider of java CGI functionality for the web are servlets or JSPs. We use tomcat 4.1.18 for this. During installation and development it is easy to make tomcat debug friendly (via debug levels and reloadable in server.xml), one should reset this in the production environment. Depending on the java security mode defaults, it may be necessary to tweak *SocketPermissions* in policy files the default (that was necessary in the debian tomcat package, but not necessary when directly downloading from jakarta.apache.org). Courses.java is the complete servlet.

6 TLS/SSL

Having done the proof of concept it is wise to secure the whole thing, this entails both the communication between web server and browser on the one hand, as well as the communication between web server and LDAP resitory (eDirectory server).

SSL has originally been developed and specified by Netscape, this was rebaptized TLS when further standard development moved to an eponymous IETF working group, due to their similarity both protocols are often mentioned together [Re01]. In current practice, during the handshake, most clients offer the server a dozen or more transmission options, whereof the server picks one to its taste. The growing popularity if TLS is demonstrated by the fact that TLS now is preference of both OpenLDAP *slapd* 2.1.12 as well as eDirectory 8.7 (whereas eDirectory 8.6.2 still goes for SSL).

The assigned port for LDAP over TLS/SSL is 636. First we establish a TLS/SSL connection; once the handshake has succeeded, we can use TLS/SSL *data* transmissions to run the LDAP protocol. It is just the same idea as using HTTPS protokoll (on port 443)[17].

6.1 Web server ↔ Browser

Although tomcat and zope both offer SSL support, we chose to hide both behind apache’s SSL module (*lib_modapache_ssl*), one way to do this is the *ProxyPass* instruction (rationale: one single point for web server certificates, the apache SSL module has been widely used).

6.2 Web server ↔ LDAP server

Background

To rule out man-in-the-middle attacks, the X.509 certificate of the *certification agency that had once certified the LDAP server’s public key* (CA certificate; this is exactly

[17]Slightly misleading to the aspiring TLS/SSL LDAP adept is the fact that there is also an RFC 2830 for TLS initialization *via already initialized* unencrypted LDAP sessions, that RFC is however rarely implemented.

the certificate of the CA that has signed the LDAP server certificate being transmitted in the second step of the handshake) has to be transferred to the client manually once [Mar02], when doing this mind that:

- In the LDAP server certificate for the *Subject name (cn)* use the FQDN (fully qualified domain name) of the LDAP server (same idea as with the certificates used for HTTPS, apparently this is not default neither in Novell nor the *openssl* tools).
- The CA certificate (e.g. *Institutional CA* in an eDirectory) must be exported to a file (Novell's *b64* format is the same as *PEM* in OpenLDAP tools), it then can be presented to the client in a suitable format.

For inspecting TLS/SSL transactions ("did we use encryption or not; what protocol are we actually using ?") the most useful tools were: *ssldump*[18] (you see all TLS/SSL transactions captured by on a network interface), *ldapsearch -d 7*[19] (verbose, once debug level is high enough, same applies for slapd). X.509 certificates can either be watched in the output of these debuggers or statically using "*openssl x509 -noout -text -in cert.pem*".

Python client

Since version 2.0.0pre06 python-ldap also runs over TLS/SSL (in the handshake the openssl negiotiated *RSA with 3DES EDE CBC SHA* with a default eDirectory 8.6/8.7. Using

```
ldap.set_option (ldap.OPT_X_TLS_CACERTFILE,
  "nldapcacert.pem")
```

we tell the client, where to find the CA certificate.

Then we just have to adapt the LDAP URL, i.e. (RegisterSSL.py):

```
c = ldap.initialize ("ldaps://www.nldap.com:636")
```

instead of previously (Register.py):

```
c = ldap.initialize ("ldap://www.nldap.com:389").
```

To make the whole thing really work, otherwise we see:

```
ldap.SERVER_DOWN: {'desc': "Can't contact LDAP server"}
```

make sure that:

- The openldap libraries installed on the client have to support TLS/SSL (e.g. for a debian *stable* supplement *libldap2-tls* from *testing*).

[18]http://www.rtfm.com/ssldump/
[19]http://www.openldap.org/

Java client

Using the *keytool* in a we typically store the the CA certificate in a file in the keystore format, that is then read in RegisterSSL.java by a call to *java.security.Security.addProvider()*[20]: Looking at the protocol's handshake a default JSSE seems to prefer *RSA with RC4 MD5* (optimized for speed).

7 Discussion

7.1 Accessibility of eDirectory via LDAP

The LDAP interface of eDirectory [Ser02] seems to be mostly conformant to the LDAPv3 User Schema (RFC 2256); a (meaningful) deviation was that the *userPassword* attribute could be evaluated only via *compare* operations, but not with read operations[21], otherwise we did not encounter any deviations.

A challenge in using eDirectory via the LDAP interface as compared to Novell's more native interfaces (DirXML, NJCL java libraries) is given in that LDAP as a lightweight protocol does not support transactions, whereas eDirectory sometimes usess non-atomic operations (e.g. group membership in eDirectory is both an attribute of the group as well as an attribute of the user). Our response was to define group membership (=course subscription) as a group attribute only, this can be done without problems.

When we used group membership we also ran into one instance where the mapping of the eDirectory was not yet stable: in version 8.6.1 group membership was the attribute *member*, in version 8.6.2 it was *uniqueMember* (both are RFC 2256 compatible).

7.2 Performance and Usability

Simulating 400 concurrent users for 60 seconds under *siege*[22] we obtained the following performance (399MHz PII, 796 bogomips, 128 MB RAM):

Apache directly	25-35 transactions/sec.
Apache+Java+Tomcat (w/o TLS/SSL)	10-12 transactions/sec.
Apache+Zope+Python (w/o TLS/SSL)	10-12 transactions/sec.
Apache+Java+Tomcat (with TLS/SSL, no pooling)	3.2 transactions/sec.
Apache+Zope+Python (with TLS/SSL, no pooling)	3.2 transactions/sec.
Apache+Zope+Python (with TLS/SSL, pooling)	10-12 transactions/sec.

The time differential we measured due to the initial loading of the java virtual machine flattens out when one uses an up-to-date version of tomcat (such as 4.1.18, whereas the java+tomcat combination had performed very poorly when using the less recent tomcat version 4.0.6, where we had a performance worse by facter of 2 to 8).

Obviously, performance of *with TLS/SSL* is improved (an approximation of the value *with TLS/SSL* to those *without TLS/SSL* is to be expected) if, instead of opening a new connection for each TLS/SSL transaction between web server and LDAP server we just reuse a connection that has been opened once (this kind of pooling was achieved by choosing the implementation as a zope product leaving connections open).

[20]Following the hints by Susan Perrin 04 June 2002 on novell.devsupp.ldap_j, 04 June 2002.

[21]Andy Chalarambous 16 Oct 2001 on novell.support.ds.nds-general

[22]http://www.joedog.org/siege/index.shtml

Taking into account these uniform performance measurements, our juxtaposition of library interfaces (api.txt), may both be helpful to persons having done their prototype in java and now wanting to do it python (as we did) or for those migrating into the converse direction. The similarity of the ldap libraries is probably due to the explict specification of a C API in RFC 1823; when using the JLDAP API the price we had to pay for more fine-grained typing (e.g. searches on attributes return *com.novell.ldap.AttributeSet* instances in JLDAP, whereas python-ldap just gives us hashes of lists) were more type conversions in our ChangeManager.

7.3 TLS/SSL for python and zope

In a previous article with a similar point of view as ours [Ri01a] then non-availability of LDAPv3 and this LDAP over TLS/SSL was still a problem for python programmers. This has been solved, since Michael Ströder introduced LDAPv3 support into in die python-ldap-2.0.0pre06 in summer 2002. As most zope applications for LDAP (such as *LDAPUserFolder*, *CMFUserFolder*, *LDAPDirectoryManager*, *LDAPAdapter*, *ZopeLDAP*) use python-ldap (ldaptor would be an alternative), the use of TLS/SSL is possible, once the user updates the openldap and python-ldap library (see [Chou03]) and provides all the CA certificates needed for the connection.

7.4 A tribute to “Directories vs databases”

Concerning the question “Directory or database?” our use case is on the borderline: because course selections fit well into existing LDAP infrastructure (a simple addition of groups and membership attributes in an existirng directory, e.g. no changes to the schema were made) and only lightweight data has been stored the use of a directory can be justified (similar to [Ri01b]); this not necessarily needs to hold for more complicated and rather relational use cases [Koehn02, Ma01]. The decision to implement this use case via a directory was also motivated to gain experience in the accessibility of LDAP repositories which may be useful for more typical tasks (authentication, certificate management etc).

8 Acknowledgments

The use case is due to Prof Klaus Osterkorn; thanks also go to Christoph Wunder and Helmut Naughton for some“SSL sessions”.

9 Sources, Addenda und Corrigenda

http://www.blasum.net/holger/wri/comp/net/7appl/ldap/bochum2003/talkback/.

References

[Chad96] David Chadwick, Understanding X.500 - The Directory, http://www.isi.salford.ac.uk/staff/dwc/Version.Web/Contents.htm.

[Chou03] TC Chou, Setting LDAP, 2003, http://zope.slat.org/Members/tcchou/index_html/setting_ldap.

[Gam94] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns, Addison-Wesley 1994.

[Har00] Robert G. Harrison, Jim Sermersheim, and Steve Trottier, LDAP Developer's Guide, Novell Press 2000.

[Kla01] Norbert Klasen, Directory Services for Linux in comparison with Novell NDS and Microsoft Active Directory, Diploma Thesis, RWTH Aachen 2001, http://www.daasi.de/staff/norbert/thesis/html/ .

[Koehn02] Kristian Köhntopp, Directory Services vs. Relationale Datenbanken, 2002, http://www.koehntopp.de/kris/artikel/dir-vs-rel/.

[Ma01] Vikas Mahajan, Directory, Database, or Both?, http://www.ldapzone.com/general_interest.html#2.

[Mar02] Franck Martin, SSL Certificates HOWTO, http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/SSL-Certificates-HOWTO.html.

[Ra] Raphinou, ldap installation with ssl, http://www.raphinou.com/ldaps/LDAP-SSL.HOWTO.

[Re01] Erich Rescorla, SSL and TLS, Addison-Wesley 2001.

[Ri01a] Mike Richichi, How to Use Perl, Python, and PHP to Access NDS eDirectory 8.5 via LDAP, Novell Appnotes, http://developer.novell.com/research/appnotes/2001/august/05/a010805.pdf, http://developer.novell.com/research/appnotes/2001/august/05/apv.htm.

[Ri01b] Mike Richichi, ATTIC: a case study of directory-enabled course management, Proceedings of 29th ACM SIGUCCS Conf, 2001, pp. 258 - 261.

[Ser02] Jim Sermersheim, LDAP Schema for NDS, 2002, http://www.ietf.org/internet-drafts/draft-sermersheim-nds-ldap-schema-03.txt.