

Lightweight Directory Access Protocol (LDAP)

Refs:

- Netscape LDAP server docs
- U. of Michigan LDAP docs
- www.openldap.org docs
- RFCs: 1777, 1773, 1823, ...

Netprog: LDAP

1

Directory Services

- A "directory" service is a network accessible database:
 - Small amount of information in each request/reply.
 - Limited functionality (as compared to a complete database system)
 - Updates (changes) are much less frequent than queries.

Netprog: LDAP

2

Directories

- Some typical examples include:
 - telephone directories
 - lists of addresses (email, network, P.O., etc)
- Each record is referenced by a unique key:
 - given a name, look up a phone number
 - given a name, look up an email address

Netprog: LDAP

3

Applications

- Some applications simply provide a *front-end* to a directory service.
 - Electronic phone book.
- Some applications use a directory service to store configuration information, auxiliary databases, etc.

Netprog: LDAP

4

Information Structure

- Typically, the information in a directory is structured hierarchically (but it doesn't have to be).
- The structure of the data (the hierarchy) is often useful in finding data and provides some (minimal) relationship between records.

Netprog: LDAP

5

Example: DNS

The Domain Name System is an example of a directory:

- hierarchical structure
- for each item there is a unique key (the hostname) and a number of attributes:
 - IP address
 - Mail exchanger
 - Host information
 - etc...

Netprog: LDAP

6

X.500

- X.500 is a Directory Service that has been used for a while:
 - Based on O.S.I. Protocol Stack
 - requires upper layers (above transport) of the OSI Stack
 - *Heavyweight* service (protocol).

Netprog: LDAP

7

LDAP

- A number of *lightweight* front-ends to X.500 have been developed - the most recent is LDAP:
 - Lightweight Directory Access Protocol
 - Based on TCP (but can be mapped to other protocols).
 - 90% of the functionality of X.500
 - 10% of the cost

Netprog: LDAP

8

LDAP & U. of Michigan

- LDAP originated at the University of Michigan.
- LDAP can be used as a front-end to X.500 or stand-alone.

- LDAP is now available commercially from a number of sources (including Netscape)

Netprog: LDAP

9

LDAP definition

- RFC 1777:
 - data representation scheme
 - defined operations and mapping to requests/response protocol.
 - RFC 1823: Application Programming Interface (has become a standard)
- API provided – no sockets programming required!

Netprog: LDAP

10

LDAP Data Representation

- Each record has a unique key called a *distinguished name* (**dn** for short).
- A distinguished name (RFC 1779) is meant to be used by humans (not just computers).
- Each **dn** is a sequence of components.
 - Each component is a *string* containing an attribute=value pair.

Netprog: LDAP

11

Example DN

CN=Dave Hollinger,
OU=Computer Science,
O=Rensselaer Polytechnic Institute,
C=US

Typically written all on one line.

Netprog: LDAP

12

Hierarchy

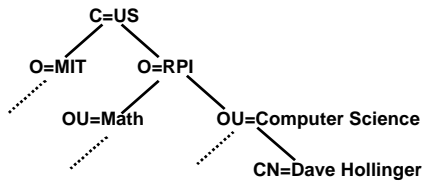
- Like Domain Names, the name can be interpreted as part of a hierarchy.
- The last component of the **dn** is at the highest level in the hierarchy.

CN=Joe Integral, OU=Math, O=RPI, C=US

Netprog: LDAP

13

Sample Hierarchy



Netprog: LDAP

14

Component Names

- The components can be anything, but there is a standard hierarchy used (for a *global* LDAP namespace):

C	<i>country name</i>
O	<i>organization name</i>
OU	<i>organizational unit</i>
CN	<i>common name</i>
L	<i>locality name</i>
ST	<i>state or province</i>
STREET	<i>street address</i>

Netprog: LDAP

15

Relative DNs

- Relative Distinguished Names are the individual components of a Distinguished Name (interpreted as relative to some position in the hierarchy).
- For example, the **RDN** "ou=Math" falls in the hierarchy below "o=RPI, c=US".

Netprog: LDAP

16

DN usage

- A distinguished name is a key used to access a record.
- Each record can contain multiple attribute/value pairs. Examples of attributes:

phone number	email address
title	home page
public key	project 3 grade

Netprog: LDAP

17

ObjectClass

- A commonly used attribute is "objectClass".
- Each record represents an object, and the attributes associated with that object are defined according to *it's objectClass*
 - The value of the objectClass attribute.

Netprog: LDAP

18

Object Type examples

- Examples of objectClass:
 - organization (needs a name and address)
 - person (needs name, email, phone & address)
 - course (needs a CRN, instructor, mascot)
 - cookie (needs name, cost & taste index)

Netprog: LDAP

19

Defining ObjectClass types

- You can define what attributes are required for objects with a specific value for the objectclass attribute.
- You can also define what attributes are allowed.
- New records must adhere to these settings!

Netprog: LDAP

20

Multiple Values

- Each attribute can have multiple values, for example we could have the following record:

```
DN: cn=Dave Hollinger, O=RPI, C=US
CN: Dave Hollinger
CN: David Hollinger
Email: hollingd@cs.rpi.edu
Email: hollid2@rpi.edu
Email: satan@hackers.org
```

Netprog: LDAP

21

LDAP Services

- Add, Delete, Change entry
- Change entry name (dn).
- Searching (the primary operation)
 - Search *some portion* of the directory for entries that match some criteria.

Netprog: LDAP

22

Authentication

- LDAP authentication can be based on simple passwords (cleartext) or Kerberos.
- LDAP V3 includes support for other authentication techniques including reliance on public keys.

Netprog: LDAP

23

LDAP Requests

- bind/unbind (*authentication*)
- search
- modify
- add
- delete
- compare

Netprog: LDAP

24

LDAP Protocol Definition

- The protocol is defined in RFC 1777 using ASN.1 (abstract syntax notation) and encoding is based on BER (Basic Encoding Rules) - all very formal.
- All requests/responses are packaged in an "envelope" (headers) and include a messageID field.

Netprog: LDAP

25

Example - LDAP bind request

Bind request must be the first request.

BindRequest =

```
[Application 0] SEQUENCE {
  version  INTEGER (1..127),
  name     LDAPDN,
  authentication CHOICE {
    simple   [0] OCTET STRING,
    krbv42LDAP[1] OCTET STRING,
    krbv42DSA [2] OCTET STRING
  }
}
```

Netprog: LDAP

26

Other Requests

- Search/modify/delete/change requests can include maximum time limits (and size limits in the case of search).
- There can be multiple *pending* requests (each with unique messageID).
 - Asynchronous replies (each includes messageID of request).

Netprog: LDAP

27

Search Request Parameters

base **scope**
size **time**
attributes **attrsonly**
search_filter

Netprog: LDAP

28

Search Parameter: Base

- The base is the DN of root of the search.
- A server typically serves only below some subtree of the *global* DN namespace.
 - You can ask the server to restrict the search to a subtree of what it serves.

Netprog: LDAP

29

Search Parameter: Scope

- *base* – search only the *base* element.
- *onelevel* – search all elements that are children of the *base*.
- *subtree* – search everything in the subtree *base*

Netprog: LDAP

30

Search Parameter: Time

Limit on number of seconds the search can take.

Value of 0 means "no limit".

Netprog: LDAP

31

Search Parameter: Size

Limit on the number of entries to return from the search.

A value of 0 means no limit.

Netprog: LDAP

32

Search Parameter: Attributes

A list of attributes that should be returned for each matched entry.

NULL mean "all attributes"

Attribute names are strings.

Netprog: LDAP

33

Search Parameter: Attrsonly

a flag that indicates whether values should be returned

- TRUE: return both attributes and values.
- FALSE: return just list of attributes.

Netprog: LDAP

34

Search Parameter: Filter

a search filter that defines the conditions that constitute a match.

Filters are text strings.

There is an entire RFC that describes the syntax of LDAP filters. (RFC 1558)

Netprog: LDAP

35

Search Filters

- Restrict the search to those records that have specific attributes, or those whose attributes have restricted values.

`"objectclass=*" match all records`

`"cn=*dave*" matches any record with "dave" in the value of cn`

Netprog: LDAP

36

Complex Filters

- You can combine simple filters with boolean &, | and !:

```
(&(cn=*da)(email=*hotmail*))
```

```
(&(!(age>=18))(drinks=yes))
```

```
(|(grade>=90)(cookies>10))
```

Netprog: LDAP

37

Search Reply

- Each search can generate a sequence of Search Response records:
 - Distinguished Name for record
 - list of attributes, possibly with list of values for each attribute.
 - Result code
- LDAP includes an extensive error/status reporting facility.

Netprog: LDAP

38

Other Requests/Responses

- The other requests and responses are detailed in RFC1777.
- However, to write a client we don't need to know the details of the protocol, there is an API (RFC 1823) and library available!

Netprog: LDAP

39

LDAP API

- There are actually a couple of well-established APIs:
 - the original (RFC 1823) from U. of Michigan.
 - Netscape has one.
- In both cases we are spared the details of the protocol, we just call some subroutines.
- The socket stuff is handled for us.

Netprog: LDAP

40

Writing a client

1. Open connection with a server
2. Authenticate (or authenticate if you must).
3. Do some searches/modification/deletions.
4. Close the connection

Netprog: LDAP

41

Opening a connection

```
int ldap_bind(  
    LDAP *ld,           connection handle  
    char *dn,           who you are (your dn)  
    char *cred,         your credentials  
    int method)        which kind of authenticaton
```

return value is `LDAP_SUCCESS` on success or else
`ldap_errno` is set to indicate the error.

Netprog: LDAP

42

Simple bind

There are actually a bunch of `ldap_bind` functions, this is the simplest:

```
int ldap_simple_bind(  
    LDAP *ld,          connection handle  
    char *dn,          who you are (your dis. name)  
    char *passwd)      your ldap password
```

The sample LDAP server on `monte.cs.rpi.edu` is set up so you don't need a password (or dn) to do anything. :

```
ldap_simple_bind(l,NULL,NULL);
```

Netprog: LDAP

43

Synchronous vs. Asynchronous

- Synchronous calls all end in "_s"

```
ldap_simple_bind_s(l,NULL,NULL);
```

- Easier to use (returns the result right away).

Netprog: LDAP

44

Simple Search Query

```
int ldap_search_s(  
    LDAP *ld,          connection handle  
    char *base,        dn of the base of the search  
    int scope,         scope of the search  
    char *filter,      search filter  
    char *attrs[],     list of attributes to return  
    int attrsonly,     flag - return no values?  
    LDAPMessage **res) result of query
```

Netprog: LDAP

45

Search Scope

- **LDAP_SCOPE_BASE**: search only the base for a match.
- **LDAP_SCOPE_ONELEVEL**: search only one level below the base.
- **LDAP_SCOPE_SUBTREE**: search the entire subtree below the base.

Netprog: LDAP

46

Search Filters

- LDAP search filters are described in RFC 1960.
 - attribute=value pairs with support for boolean connectives and relational operators
- Examples:

```
"(objectclass=*)"
```

```
"(&(objectclass=Cookie)(tasteindex>=30))"
```

Netprog: LDAP

47

Example Search

```
ldap_search_s(1,  
  "course=Netprog, school=RPI",  
  LDAP_SCOPE_SUBTREE,  
  "(cn=Joe Student)", NULL, 0, &mesg);
```

On success, mesg is a pointer to the result. To access the records in the result you have to use more of the LDAP library.

Netprog: LDAP

48

Search Results

The result is a list of records - you do something like this to scan the list:

```
LDAPMessage *p; char *dn;
for (p=ldap_first_entry(l,msg);
     p != NULL;
     p=ldap_next_entry(l,p)) {
    dn = ldap_get_dn(l,p);
    printf("dn: %d\n",dn);
}
```

Netprog: LDAP

49

Attributes of each entry

- Extracting the attributes (and values) from each entry is similar - step through a list of attributes using:

```
ldap_first_attribute()
ldap_next_attribute()
```

- Example code in RFC 1823!!!

Netprog: LDAP

50
