        Connection-less Lightweight X.500 Directory Access Protocol

Status of this Memo

X.500

   The protocol described in this document is designed to provide access
   to the Directory while not incurring the resource requirements of the
   Directory Access Protocol (DAP) [3]. In particular, it is aimed at
   avoiding the elapsed time that is associated with connection-oriented
   communication and it facilitates use of the Directory in a manner
   analagous to the DNS [5,6]. It is specifically targeted at simple
   lookup applications that require to read a small number of attribute
   values from a single entry. It is intended to be a complement to DAP
   and LDAP [4]. The protocol specification draws heavily on that of
   LDAP.

1. Background

   The Directory can be used as a repository for many kinds of
   information. The full power of DAP is unnecessary for applications
   that require simple read access to a few attribute values.
   Applications addressing is a good example of this type of use where
   an application entity needs to determine the Presentation Address
   (PA) of a peer entity given that peer's Application Entity Title
   (AET). If the AET is a Directory Name (DN) then the required result
   can be obtained from the PA attribute of the Directory entry
   identified by the AET. This is very similar to DNS.

Use of DAP to achieve this functionality involves a significant
number of network exchanges:

```
 _____
|_#_|_____Client_(DUA)_____DAP_____Server_(DSA)_____|
|  1|  N-Connect.request          ->                              |
|  2|                             <-     N-Connect.response       |
|  3|  T-Connect.request          ->                              |
|  4|                             <-     T-Connect.response       |
|   |  S-Connect.request,                                         |
|   |  P-Connect.request,                                         |
|   |  A-Associate.request,                                       |
|  5|  DAP-Bind.request           ->                              |
|   |                                    S-Connect.response,      |
|   |                                    P-Connect.response,      |
|   |                                    A-Associate.response,    |
|  6|                             <-     DAP-Bind.response        |
|  7|  DAP-Read.request           ->                              |
|  8|                             <-     DAP-Read.response        |
|   |  S-Release.request,                                         |
|   |  P-Release.request,                                         |
|   |  A-Release.request,                                         |
|  9|  DAP-Unbind.request         ->                              |
|   |                                    S-Release.response,      |
|   |                                    P-Release.response,      |
|   |                                    A-Release.response,      |
| 10|                             <-     DAP-Unbind.response      |
|   |  T-Disconnect.request,                                      |
| 11|  N-Disconnect.request       ->                              |
|   |                                    T-Disconnect.response,   |
| 12|                             <-     N-Disconnect.response    |
|___|_____|
```

This is 10 packets before the application can continue, given that it
can probably do so after issuing the T-Disconnect.request. (Some
minor variations arise depending upon the class of Network and
Transport service that is being used; for example use of TP4 over
CLNS reduces the packet count by two.) LDAP is no better in the case
where the LDAP server uses full DAP to communicate with the
Directory:

```
 _____
|__#_|___Client_____LDAP_____LDAP_server_____DAP_____DSA_____|
|  1 |  TCP SYN       ->                                              |
|  2 |                <-     TCP SYN ACK                              |
|  3 |  BindReq       ->                                              |
|  4 |                       N-Connect.req       ->                   |
|  5 |                                           <-     N-Connect.res |
|  6 |                       T-Connect.req       ->                   |
|  7 |                                           <-     T-Connect.res |
|  8 |                       DAP-Bind.req        ->                   |
|  9 |                                           <-     DAP-Bind.res  |
| 10 |                <-     BindRes                                  |
| 11 |  SearchReq     ->                                              |
| 12 |                       DAP-Search.req      ->                   |
| 13 |                                           <-     DAP-Search.res|
| 14 |                <-     SearchRes                                |
| 15 |  TCP FIN       ->                                              |
| 16 |                       DAP-Unbind.req      ->                   |
| 17 |                                           <-     DAP-Unbind.res|
| 18 |                       N-Disconnect.req    ->                   |
| 19 |                                           <-     N-Disconnect.res|
|____|_____|
```

Here there are 14 packets before the application can continue. Even
if the LDAP server is on the same host as the DSA (so packet delay is
negligible), or if the DSA supports LDAP directly, then there are
still 6 packets.

```
 _____
| #|    Client      LDAP    LDAP server|
|__|_____|
| 1| TCP SYN        ->                 |
| 2|                <-      TCP SYN ACK|
| 3| BindReq        ->                 |
| 4|                <-      BindRes     |
| 5| SearchReq      ->                 |
|_6|_____<-_____SearchRes__|
```

This protocol provides for simple access to the Directory where the
delays inherent in the above exchanges are unacceptable and where the
additional functionality provided by connection-mode operation is not
required.

2. Protocol Model

CLDAP is based directly on LDAP [4] and inherits many of the key
aspects of the LDAP protocol:

- - Many protocol data elements are encoding as ordinary strings
     (e.g., Distinguished Names).

- - A lightweight BER encoding is used to encode all protocol
     elements.

It is different to LDAP in that:

- - Protocol elements are carried directly over UDP or other
     connection-less transport, bypassing much of the
     session/presentation overhead and that of connections (LDAP uses
     a connection-mode transport service).

- - A restricted set of operations is available.

The definitions of most protocol elements are inherited from LDAP.

The general model adopted by this protocol is one of clients
performing protocol operations against servers. In this model, this
is accomplished by a client transmitting a protocol request
describing the operation to be performed to a server, which is then
responsible for performing the necessary operations on the Directory.

Upon completion of the necessary operations, the server returns a
response containing any results or errors to the requesting client.

Note that, although servers are required to return responses whenever
such responses are defined in the protocol, there is no requirement
for synchronous behaviour on the part of either client or server
implementations: requests and responses for multiple operations may
be exchanged by client and servers in any order, as long as servers
eventually send a response for every request that requires one.

Also, because the protocol is implemented over a connection-less
transport service clients must be prepared for either requests or
responses to be lost. Clients should use a retry mechanism with
timeouts in order to achieve the desired level of reliability. For
example, a client might send off a request and wait for two seconds.
If no reply is forthcoming, the request is sent again and the client
waits four seconds. If there is still no reply, the client sends it
again and waits eight seconds, and so on, until some maximun time.
Such algorithms are widely used in other datagram-based protocol
implementations, such as the DNS. It is not appropriate to mandate a
specific algorithm as this will depend upon the requirments and
operational environment of individual CLDAP client implementations.

It is not required that a client abandon any requests to which no
response has been received and for which a reply is no longer
required (because the request has been timed out), but they may do
so.

Consistent with the model of servers performing protocol operations
on behalf of clients, it is also to be noted that protocol servers
are expected to handle referrals without resorting to the return of
such referrals to the client. This protocol makes no provisions for
the return of referrals to clients, as the model is one of servers
ensuring the performance of all necessary operations in the
Directory, with only final results or errors being returned by
servers to clients.

Note that this protocol can be mapped to a strict subset of the
Directory abstract service, so it can be cleanly provided by the DAP.

3. Mapping Onto Transport Services

This protocol is designed to run over connection-less transports,
with all 8 bits in an octet being significant in the data stream.
Specifications for two underlying services are defined here, though
others are also possible.

Within a searchResponse all but the last SearchResponse has choice
'entry' and the last SearchResponse has choice 'resultCode'. Within
a searchResponse, as an encoding optimisation, the value of the
objectName LDAP DN may use a trailing '*' character to refer to the
baseObject of the corresponding searchRequest. For example, if the
baseObject is specified as "o=UofM, c=US", then the following
objectName LDAPDNs in a response would have the indicated meanings

```
        objectName returned    actual LDAPDN denoted
        _____
        "*"                    "o=UofM, c=US"
        "cn=Babs Jensen, *"    "cn=Babs Jensen, o=UofM, c=US"
```

4.1. Errors

The following error code is added to the LDAPResult.resultCode
enumeration of [4]:

                          resultsTooLarge   (70),

This error is returned when the LDAPMessage PDU containing the
results of an operation are too large to be sent in a single
datagram.

4.2. Example

A simple lookup can be performed in 4 packets. This is reduced to 2
if either the DSA implements the CLDAP protocol, the CLDAP server has
a cache of the desired results, or the CLDAP server and DSA are co-
located such that there is insignificant delay between them.

```
  _____
 |_#|___Client_____CLDAP_____CLDAP_server____DAP_____DSA_____|
 | 1|  SearchReq     ->                                            |
 | 2|                      DAP-Search.req   ->                     |
 | 3|                                       <-    DAP-Search.res|
 | 4|              <-      SearchRes                             |
 |__|_____|
```

5. Implementation Considerations

The following subsections provide guidance on the implementation of
clients and servers using the CLDAP protocol.

5.1. Server Implementations

   Given that the goal of this protocol is to minimise the elapsed time
   between making a Directory request and receiving the response, a
   server which uses DAP to access the directory should use techniques
   that assist in this.

   - - A server should remain bound to the Directory during reasonably
        long idle periods or should remain bound permanently.

   - - Cacheing of results is highly desirable but this must be
        tempered by the need to provide up-to-date results given the
        lack of a cache invalidation protocol in DAP (either implicit
        via timers or explicit) and the lack of a dontUseCopy service
        control in the protocol.

   Of course these issues are irrelevant if the CLDAP protocol is
   directly supported by a DSA.

5.2. Client Implementations

   For simple lookup applications, use of a retry algorithm with
   multiple servers similar to that commonly used in DNS stub resolver
   implementations is recommended. The location of a CLDAP server or
   servers may be better specified using IP addresses (simple or
   broadcast) rather than names that must first be looked up in another
   directory such as DNS.

6. Security Considerations

   This protocol provides no facilities for authentication. It is
   expected that servers will bind to the Directory either anonymously
   or using simple authentication without a password.

7. Bibliography

   [1] The Directory: Overview of Concepts, Models and Service. CCITT
       Recommendation X.500, 1988.

   [2] The Directory: Models. CCITT Recommendation X.501 ISO/IEC JTC
       1/SC21; International Standard 9594-2, 1988.

   [3] The Directory: Abstract Service Definition. CCITT Recommendation
       X.511, ISO/IEC JTC 1/SC21; International Standard 9594-3, 1988.

   [4] Yeong, W., Howes, T., and S. Kille, "X.500 Lightweight Directory
       Access Protocol", RFC 1487, Performance Systems International,
       University of Michigan, ISODE Consortium, July 1993.

    [5] Mockapetris, P., "Domain Names - Implementation and
        Specification", STD 13, RFC 1035, USC/Information Sciences
        Institute, November 1987.

    [6] Mockapetris, P., "Domain Names - Concepts and Facilities", STD
        13, RFC 1034, USC/Information Sciences Institute, November 1987.

8. Acknowledgements

9. Author's Address

    Alan Young
    ISODE Consortium
    The Dome, The Square
    RICHMOND
    GB - TW9 1DT

    Phone: +44 81 332 9091
    EMail: A.Young@isode.com
    X.400:   i=A; s=Young; o=ISODE Consortium; p=ISODE; a=MAILNET; c=FI