Directory navigation in the Quipu X.500 system

Paul Barker
Colin J. Robbins

12th October 1989

Abstract

OSI Directory Services have recently been standardised according to the X.500 / IS 9594 standard. This first part of this paper gives a brief overview of the Directory Services model.

Quipu [1][2] is one of the first implementations of the X.500 standard and has been developed at UCL. [1] Quipu is publicly available as part of the ISODE [3] package.

One of the key aspects of Directory Services not fully specified in the standard is that of managing the distribution of the Directory. The approach taken by the Quipu system in representing Directory knowledge and handling Directory navigation across heterogeneous networks is described here. The issues raised by this are discussed.

KEYWORDS: OSI, Directory, Quipu, Distributed System, Navigation.

1 Introduction

The first part of this paper introduces OSI Directory Services. These have recently been standardised according to the CCITT X.500 recommendations / ISO 9594 International Standards [4]. This paper gives a very brief overview of the standards framework.

The remainder of the paper discusses the approach taken by Quipu with regard to directory navigation. The discussion focusses on how Quipu attempts to provide a robust and efficient service, given less than fully reliable, heterogeneous networks.

[1]The work was originally funded by under the European Strategic Program for Research into Information Technology (ESPRIT). Quipu was developed as a deliverable for INCA, project 395. Quipu is currently funded by the U.K. Joint Network Team.

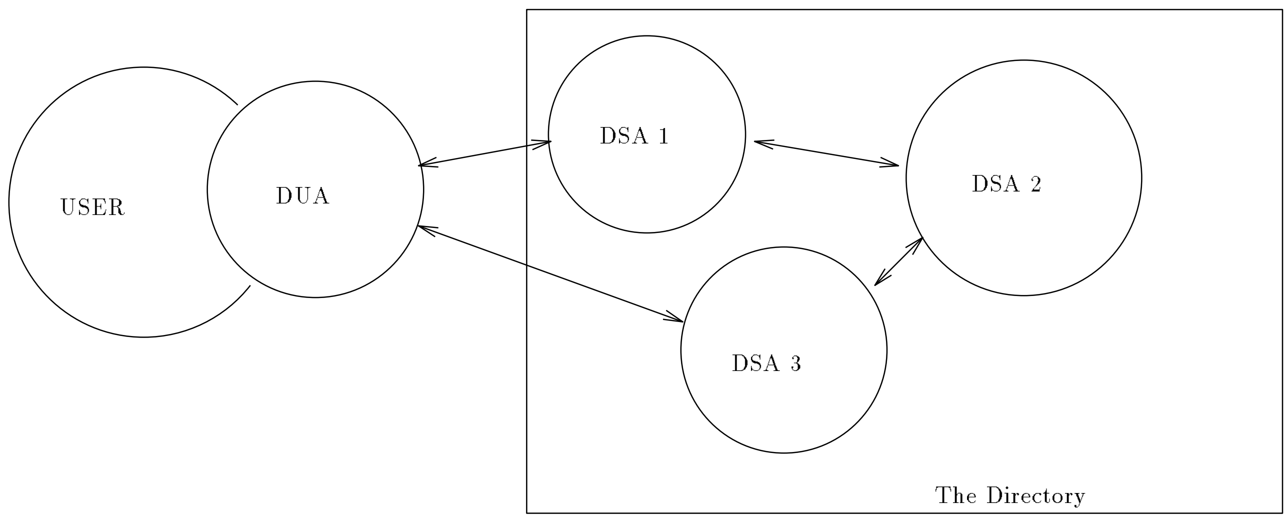

Figure 1: Directory Object Model

2 Overview of Directory Services model

The OSI Directory is intended to support human user querying, allowing users to find, inter alia, telephone and address information of organisations and other users.

It is also intended to support electronic communication such as message handling systems and file transfer. The Directory provides name to address mapping to support, for example, OSI presentation address look-ups. Message handling systems will be provided with support for user-friendly naming, security and distribution lists.

2.1 Directory characteristics

In essence the Directory is a database with certain key characteristics.

1. The Directory is intended to be very large and highly distributed. It is anticipated that the Directory will be distributed largely on an organisational basis.

2. The Directory is hierarchically structured, the entries being arranged in the form of a tree. Entries near the root of the tree will usually represent objects such as countries and organisations, entries at or near the leaves of the tree will represent people, equipment or application processes.

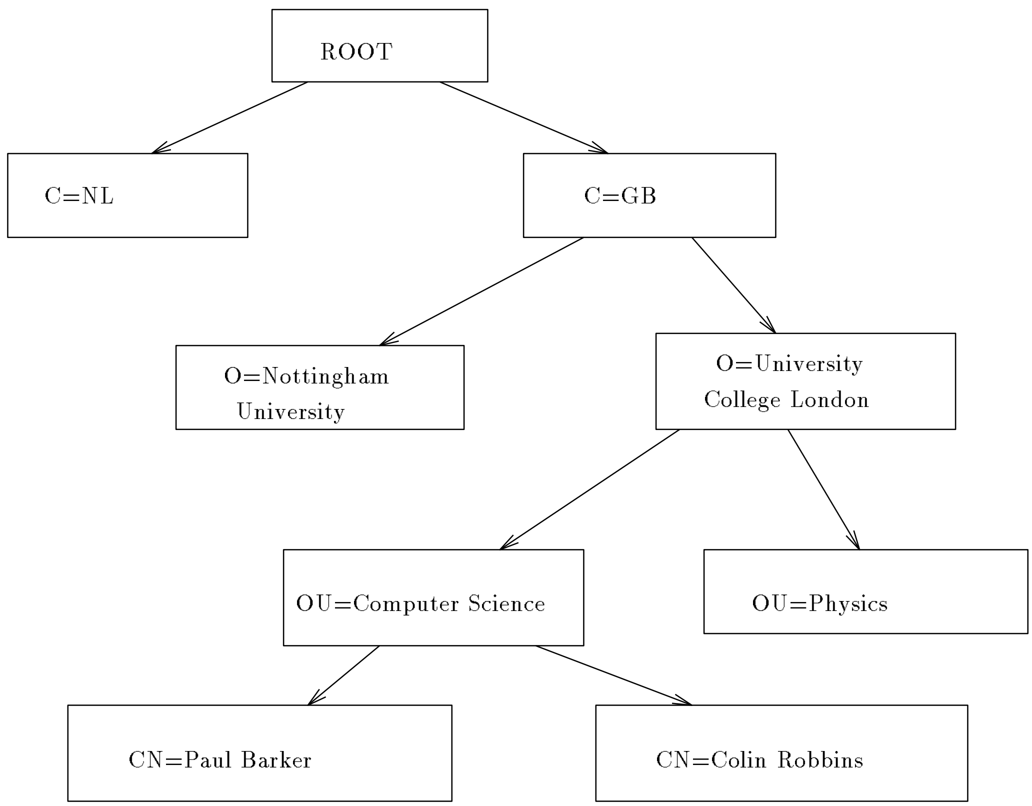

Figure 2: Hypothetical DIT

3. Read and search operations will dominate over modification operations.

4. Temporary inconsistencies in the data are acceptable. This greatly facilitates the replication of data in the Directory by obviating concerns about record locking and atomic operations.

2.2 Directory Object Model

The Directory can be decomposed into objects as in figure 1. A user accesses the Directory by means of a *Directory User Agent* (DUA). The DUA communicates with the Directory by using *Directory Access Protocol* (DAP).

The Directory comprises a collection of *Directory System Agents* (DSAs). Each DSA has an associated database which holds some portion of the global database. The DSAs cooperate to provide the Directory Service. The DSAs communicate with each other by using *Directory System Protocol* (DSP).

The distributed operation of the Directory is implemented by using one or more of the following modes of operation:

- Chaining is where a DSA passes an operation onto a further DSA, awaits the response, and passes it back to the initiating DUA.
- Referral is where a DSA returns a reference to another DSA back to the initiating DUA or DSA. This reference consists of the name of another DSA which the operation might be passed to.
- Multicasting is where a request is broadcast to several DSAs, which may then collectively resolve the request.

The combination of these modes of operation used by the Quipu implementation are discussed later.

2.3 Structure of the Directory

It was noted earlier that the Directory is organised hierarchically in the form of a tree. The Directory database is usually referred to as the *Directory Information Tree* (DIT).

The overall structure of the DIT is shown in figure 2.

The hypothetical entries illustrate the hierarchy of the Directory and how entries are named within the Directory. At each point in the Directory, entries are differentiated by unambiguous *Relative Distinguished Names* (RDNs). Thus, in figure 2, "C=GB" and "C=NL" are RDNs under the root of the DIT. An entry's *Distinguished Name* is derived by concatenating all the RDNs of the entries from the root of the tree to the entry itself.

Each entry may be further decomposed as shown in figure 3.

An entry comprises a set of attributes, which in turn consist of a type and a value or set of values. It should be noted that an entry's distinguished name is merely a special attribute type-value pair. For example, an entry for a human being will have, inter alia, an attribute type *Common Name.* This attribute will often be multi-valued. The Common Name attribute for Steve Kille's entry might take the values "Steve Kille", "Stephen E. Kille" and "S. Kille" with "Steve Kille" being the distinguished value.

2.4 Accessing the Directory

A user makes use of the Directory by means of the *Directory Abstract Service.* The services provided are grouped into three *ports,* the read port, the search port and the modify port.

The read and search ports provide querying facilities onto the Directory. It is possible to read or compare an entry identified by its distinguished name.

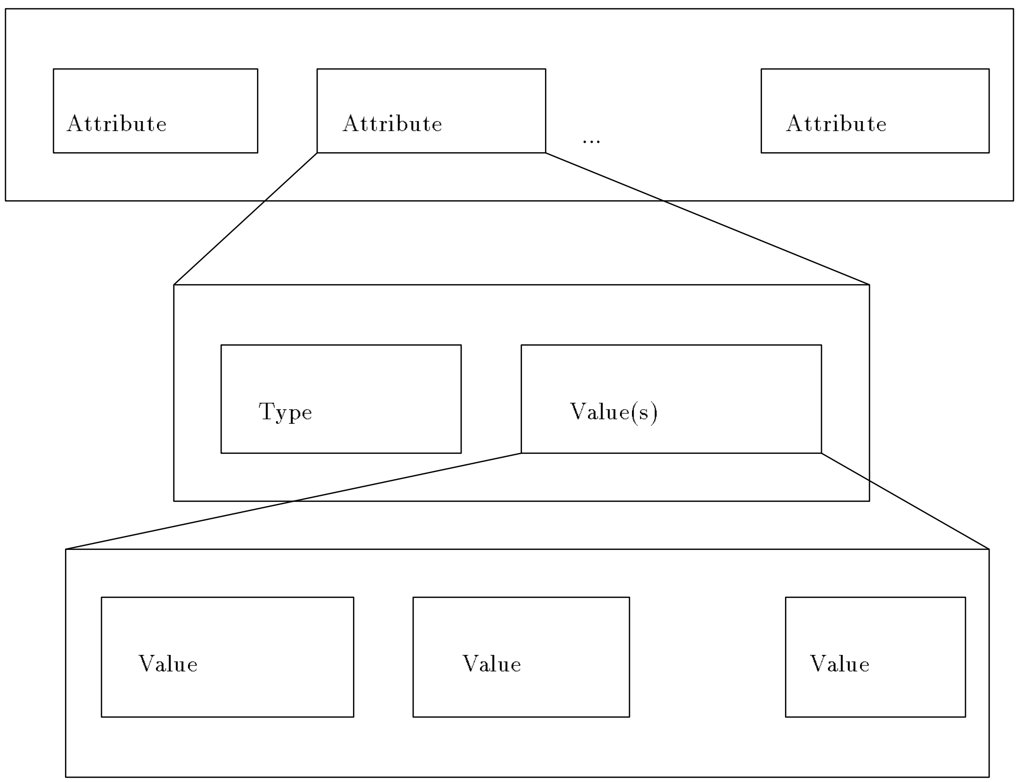

Figure 3: Structure of an entry

The powerful search operation allows querying of entire sub-trees, returning specified attributes for all entries which satisfy the criteria specified in the search arguments. This allows entries to be identified by attributes other than just the distinguished name and thus provides users with a highly flexible mechanism for identifying entries and retrieving information from the Directory.

Modification operations allow the addition and removal of entries from the DIT, the amendment of entries and the renaming of entries.

2.5 Other aspects Of Directory Services

There are many aspects of the Directory Services standard which cannot be described in detail. Such aspects include:

- Access control
- Authentication
- Service controls
- Schemas
- Use of OSI

3 Distributed Operations in Quipu

The remainder of the paper focusses on the issue of distributed operations. As the Directory is widely distributed, *knowledge* must be maintained of how the DIT is distributed amongst the collection of DSAs which comprise the Directory. The standard does not specify how this knowledge should be represented in the Directory. The approach followed by Quipu is discussed.

It was noted earlier that the model allows for several modes of interaction between DSAs as they cooperate to service requests made by DUAs; namely chaining, referral and multicasting. The approach used by Quipu is discussed, with particular reference to the problem of coping with the situation where the DIT is fragmented into DSAs on disjoint networks.

3.1 Directory Service requests requiring distributed operation

When considering the effects of directory distribution, there are four possible results which can result from a request being presented by a DUA to the DSA at the directory access point.

i) The request may be satisfied locally.

ii) The "local" DSA may be able to determine that the request cannot be serviced by any DSA. The directory knowledge indicates that the entry required would be held in that DSA if such an entry existed. In this case the DSA would return a *name error* to the DUA.

iii) A request is made to the local DSA which requires navigating down to a sub-tree not held locally. A set of references is acquired indicating other DSAs which might be able to satisfy the request.

iv) A request is made which requires navigating to a higher point in the tree than that held locally. The addresses of DSAs nearer the root must be found from local tables.

The rest of the paper discusses how Quipu proceeds in cases iii and iv above.

3.2 Representing directory knowledge

Case iii above requires the existence of knowledge information. This is information which a DSA has about which entries it holds and how to locate other entries in the Directory. The standard does not specify how or where this knowledge is stored. Quipu takes the approach that the OSI Directory itself should be used, and stores the knowledge in the DIT.

The first step in storing the knowledge is to give every DSA in the directory an entry in the DIT which contains information about the DSA. For example the DSA holding the data for University College London has the distinguished name "(country=GB, commonname=Vicuna)", and has the following attributes:-

```
presentationAddress= Internet=128.16.8.50+50987 | X121=23421920030045
description= A wild animal of the Alpaca family,
description= DSA running on vs1 holding full UCL bit of tree.
supportedApplicationContext= x500DAP & x500DSP & QuipuDSP
commonName= Vicuna
objectClass= quipuDSA & dSA & applicationEntity & top
```

The first thing to notice is the name. It is a Quipu convention that all Quipu DSAs are named after endangered South American wildlife. Quipu was originally developed under the aegis of the ESPRIT project, INCA.

The above entry enables a DSA to determine the address or addresses of other DSAs. However, a DSA still needs to determine which DSA to contact to answer a particular request. Quipu DSAs achieves this by requiring that every non-leaf object has a "masterDSA" attribute, the value of which is the DN of the DSA to contact. It is important to note that Quipu makes an important simplification of the model in this respect. It is assumed that if an entry is held in a DSA, then all sibling entries are held in that DSA. This assumption allows for a relatively straightforward replication mechanism based on Quipu's getedb mechanism. This is discussed in [1].

In addition, Quipu has added the concept of *slave* DSAs to the model. These are DSAs which hold a shadow copy of some data, and are prepared to answer requests regarding that data. Thus a non-leaf entry may have "slaveDSA" attributes which give the DNs of DSAs that hold such data.

A Caveat on naming DSAs

> Using this approach, care must be taken to name the DSAs high enough in the DIT to prevent looping. For example, consider a DSA holding the subtree for "(country=GB, organisation=University College London)" which is named "(country=GB, organisation=University College London, commonname=Vicuna)". If an operation attempted to list the subordinates of "(country=GB, organisation=University College London)", a referral would have to be made to the DSA "(country=GB, organisation=University College London, commonname=Vicuna)". This would require the entry "(country=GB, organisation=University College London, commonname=Vicuna)" to be read by the DSA. To read this entry, the DSA would have to know how to navigate to "(country=GB, organisation=University College London)" – but does not know how to do that, without seeing the "(country=GB, organisation=University College London, commonname=Vicuna)" entry! Thus a (detectable) loop has been created. To avoid this, DSAs should be named at the same level, or higher, in the DIT as the entries they are holding. This has the effect that there are lots of DSAs named at the higher levels of the DIT.

When an operation cannot be satisfied locally, a list of DSAs which either master or shadow the information will be generated by looking at these attributes. We will now consider how Quipu chooses DSAs from this list to resolve the request.

3.3 DSA selection criteria

It will be seen that randomly selecting a DSA from a list of possible DSAs is not an optimal strategy. The reasons for this are discussed below. Quipu uses a number of criteria when establishing which DSA it will forward a request to. Rather than picking a single "best" DSA, Quipu sorts the list of DSAs into an order of preference. A simple insert sort algorithm is used which successively compares pairs of DSAs to see which is the "best".

It is worth noting here the reason why Quipu sorts the list of DSAs rather than merely selecting the best DSA. As will be explained in some detail shortly, a Quipu DSA is able to make some assumptions about another DSA's behaviour if it knows that it is a Quipu DSA. The semantics of X.500 dictate that a subordinate reference contains a single DSA if a request cannot be satisfied at a given DSA. However, the syntax of X.500 allows more than one DSA to be named in a continuation reference. Quipu sometimes takes advantage of this when communicating with other Quipu DSAs, by passing a *Quipu-Specific Subordinate Reference* (QSSR) which references multiple DSAs. QSSRs cannot always be used as some requests, for example modification operations, and operations which specify the "don't use copy" service control, must be directed at the sole master DSA. In these cases a standard subordinate reference is used.

This section discusses the criteria which are used. The order of discussion indicates the weight given to the criteria. The less important criteria are only used if no preference can be deduced from the more important.

3.3.1 DAP only DSAs

DSAs which do not support DSP impose referral mode when other considerations might tend to favour chaining. This restriveness means that such DSAs are not favoured and any such DSAs are placed at the bottom of the sorted DSAs list.

3.3.2 Prefer a Quipu DSA

The first choice it to select a Quipu DSA. The main reason for this is that the DSAs can then talk over their own application context (rather than the standard X500 DSP context), which allows them to make a few simplifying assumptions, e.g. QSSRs (although the protocol used is the same).

This is represented in the Directory by a DSA having the attribute type *Supported Application Context* with a value "quipuApplicationContext".

3.3.3 Prefer a reliable DSA

Experience with Quipu-5.0 in which a DSA was chosen effectively at random (but for the same query the same "random" DSA would be picked!) showed that the network connections to some DSAs were much more unreliable than others. As a result, a lot of time was spent attempting associations that were almost certain to fail. Thus a mechanism has been introduced which attempts to identify reliable DSAs.

To make this choice every DSA holds the following information on each other DSA it tries to contact:

- Distinguished name of DSA
- Time of last attempt
- Time of last success
- No. of failures since last success

Every time an association is attempted to a DSA, its DSAInfo is found, and the *lastAttempt* field is set to the current time. If the association succeeds *lastSuccess* is set to the current time, and *failures-since-last-success* is set to zero. If the association fails *failures-since-last-success* is incremented.

The notion of *recent* success or failure is used to decide which DSA to prefer. "Recent" is in practice the value of the tailorable variable selected to age the cache of connectivity information. It is not at present clear what the optimum timeout period is for aging this information. This area requires further experimentation.

The following algorithm is then used to select the more reliable DSA.

If both DSAs have been accessed successfully recently, prefer a DSA which has suffered no recent communication failures. If either communication with both DSAs has failed recently, or neither DSA has a record of failure, then some other DSA selection criterion must be used. No attempt is made to discriminate between DSAs on the basis of how recently the successes or failures occurred.

If only one of the DSAs has been successfully contacted recently, prefer that DSA unless it also has a record of recent failure. In the case of a recent failure, prefer the other DSA, unless it also failed recently in which case no discrimination can be made.

If neither DSA has been contacted successfully recently, some other criterion must be used to choose between the DSAs.

3.3.4 Prefer a close DSA

A close DSA may be preferable for a number of reasons. Network charges may be lower, or non-existent, for proximate DSAs. Physically close DSAs may often be connected by networks offering greater bandwidth. Physically close DSAs may be separated by fewer gateways than DSAs separated by great distances.

The following sections suggest 3 ways a *close* DSA may be selected.

Clearly it is preferable to choose a DSA on the same local area network, or using the preferred network type if possible. To make this decision, we need a method of addressing DSAs on different networks, that is:

i) compatible with the standard, that is it can be stored in the "presentation address" attribute of a DSA;

ii) can supply sub network details.

OSI purists may well be alarmed at this point. Network layer details should be hidden from applications. NSAPs should not contain routing information.

However, at present, real users do not use OSI network services. Network services are currently provided largely by TCP/IP and X.25 (1980) networks. These network domains are themselves not fully connected. TCP/IP is often used on LANs which are not connected to the Internet. X.25 domains exist, such as the U.K.'s JANET, which are not fully connected to the international X.25 networks.

Until OSI network services are available to and used by almost all users, a work-around solution is required. Kille [5] has defined a mapping between the various network address spaces and OSI presentation addresses. This uses a part of the Telex address space to hold the encoded addresses.

Every DSA has a distinguished name and this can be used to select a potentially close DSA. For example, if our DSA is called "(country=GB, Organisation=University College London, commonname=Tamarin)", and we have a references to DSAs "(country=US, commonname=Fruit Bat)", and "(country=GB, commonname=Vicuna)", then on the basis of distinguished names, "(country=GB, commonname=Vicuna)" is *likely* to be closer.

DSAs may be managed in Directory Management Domains (DMD) for accounting purposes. If a DSA is in the same DMD as the requestor, then if may be best to use this DSA in preference to a DSA in a different domain.

Quipu does not currently use this as a selector, as the concept of DMD has not been utilised fully in current pilot exercises, thus the selector would always return "no difference" when comparing two DSAs.

3.3.5 Need for experimentation

How successful this algorithm is in practice remains to be seen. Quipu-6.0, which attempts to make the above decisions, is about to be released. However successful the algorithm proves to be, one may be fairly confident that the method is better that a random selection.

3.4 Chaining, referrals, multicasting

Having decided which DSA or DSAs to contact to follow references, the decision of which intercation model to use still has to be made. This is now considered.

Quipu has a basic framework for interaction between DUA and DSA, and between two DSAs. We will see later that are several situations which force departure from this model.

The basic model is as follows:

> If the first DSA contacted cannot satisfy a request, it chains that request on to a second DSA. If the second DSA cannot satisfy the request it sends a referral back to the initial DSA which then chains the request to the referenced DSA. From the viewpoint of the DUA, the model is one of chaining. From the viewpoint of the first DSA, the model is one of referral.

The advantages of this model are as follows:

i) The work of the DUA is simplified by placing a heavy onus on the DSA at the DUA's access point. All references are followed by the DSA. The DUA only needs a single access point onto the Directory.

ii) A corollary of the access point DSA shouldering the burden of chasing referrals is that the DSA is able to cache all the information that it acquires from other DSAs. Caching can dramatically improve performance for all the DUAs and DSAs which communicate with that DSA. Obviously care needs to be exercised as the cache ages and caches have to be purged periodically. Great care also needs to be taken that access control mechanisms are not circumvented by the use of caching.

iii) The DUA only needs to be on the same network as its access point DSA. Full connectivity with the Directory can be achieved so long as that DSA can contact other DSAs by chaining or referral. It should be noted that this problem can be circumvented by the use of transport service bridges.

iv) The model is a good basis for a charging policy. The best model for charging would be one of DUA referral where all charges could be assigned to the originating DUAs. For reasons already discussed this is not the best model for a variety of other reasons. The DSA referral model provides a reasonable, second best approach. All DUA requests which generate requests across wide area, chargeable networks, are initiated by a single DSA which represents the DUA. It is clearly very difficult to administer a charging policy for any model which allows for a substantial amount of chaining. To cope with this problem, Quipu in fact allows a DSA to "defend" itself against chaining requests by setting a "dsp_chaining" variable to "off".

v) The DSA referral model allows more control over an operation and may be beneficial if some DSAs are not highly reliable. Under the chaining model, if knowledge is fairly minimal, unreliable DSAs may cause part of the DIT to become detached and unreachable. Under the DSA referral model, a local Directory administrator can try and guard against this by ensuring that considerable knowledge is held locally.

It should be noted that the above model cannot always be adhered to. The following reasons all require a different approach.

Service controls might, for example, be set such that chaining is prohibited whereas the model indicates that a DSA should chain.

Some DSA implementations only support DAP although supporting DSP is a requirement of the standard. If such DSAs are referenced when a request cannot be satisfied locally, the request can only be pursued further by DUA referral.

It is a fact of life, as noted earlier, that DSAs will be run on disjoint networks. Ensuring that the Directory does not itself become disjointed requires cognisance of the underlying networks when assessing whether to chain or refer a directory request.

A basic assumption of the model is that DSAs should trust each other. However, such trust can in practice only be based on DSAs being able to authenticate each other. Quipu does not currently use authentication between DSAs for the following reasons: simple authentication is regarded as being too simple to forge to be worthwhile; strong authentication is time-consuming and requires a framework to manage the requisite certificates. However, the performance of the encryption algorithms has been considerably improved of late and strong authentication is being actively considered for the next release of Quipu. For this reason, Quipu will not perform modification operations over DSP. Thus DSAs must send referrals back to a DUA, whatever the model suggests is the preferred mode of interaction.

3.5 Chaining preferred

If the above model indicates that chaining is preferred, the following algorithm is then used to finally select a DSA to contact:

The ordered list of DSAs is searched to see if there is a connection already open to any of the DSAs. If there is, the request is forwarded to the first such DSA on the list.

If the DSA does not have a connection open to any of the DSAs in the list, the DSA tries to open a connection to each DSA in turn until a connection attempt is successful.

If all connection attempts fail, the DSA then tries to send a referral. The DSA attempts to select the first DSA in the list which appears to be on a compatible network.

If none of the DSAs in the list appear to be on a compatible network, a referral is sent back which names the first DSA in the list.

3.6 Referral preferred

If the model indicates that referral is preferred, the following procedure is used. It should be noted that the network compatibility testing is crucial in creating a widely distributed Directory spread over heterogeneous networks.

The DSA searched the list for any DSA with apparent network compatibility with the calling DUA or DSA.

If any DSA is found which appears to be on compatible network, then if the initiating DSA is a Quipu DSA, the list of references are returned to that DSA. If the initiator is not a Quipu DSA, then only the single, "network compatible" reference is returned. If the initiating DSA is a Quipu DSA and receives a list of DSAs, the procedure described earlier to sort the DSAs into an order of preference, is followed. The initiating DSA must discard any earlier lists of DSAs it had compiled or received earlier while trying to complete the operation, on receipt of a further list of references.

If no DSA in the list appears to on a network compatible with the originator, then an attempt is made to chain the operation, service controls permitting. If chaining fails, a referral is sent to the originating DSA.

3.7 Multicasting

In general, Quipu does not need to multicast because of the assumption that all sibling entries are held within a single DSA. However there are two occasions when Quipu makes use of multicasting.

i) Subtree searches, where the subtree is held in multiple DSAs.

ii) When following a non-specific subordinate references, generated by non-Quipu DSAs.

4 Conclusions

The approach used by Quipu to store OSI Directory knowledge has been described. It seems prudent that this information should be stored in the Directory itself.

The Directory is distributed across a large number of DSAs. These DSAs may reside on disjoint networks. The approach taken by Quipu to solve this problem has been discussed.

Some replication of the Directory will tend to improve the Directory's resilience to DSA failure and may also improve performance generally. The mechanisms used by Quipu to determine which DSA to forward requests to has been described.

Some differences between the standard X.500 model and the Quipu implementation have been noted. Quipu takes account of DSAs being situated on disjoint networks. Furthermore, Quipu tries to provide a robust and efficient service by noting DSA reliability, connectivity and proximity.

5 References

[1] "The design of Quipu", Stephen E. Kille, Research Note RN/89/19, Department of Computer Science, University College London, March 1989.

[2] "The QUIPU Directory Service", Stephen E. Kille, IFIP WG 6.5 Conference on Message Handling Systems and Distributed Applications, pp173-186. North Holland Publishing, October 1988.

[3] "The ISO Development Environment Users's Manual (Version 5.0)", Marshall T. Rose, The Wollongong Group, Palo Alto, March 1989.

[4] "The Directory - Overview of Concepts, Models and Services", X.500 and ISO 9594, 1988.

[5] "An interim approach to the use of network addresses", Stephen E. Kille, Research Note RN/89/19, Department of Computer Science, University College London, March 1989.