Search Operations on Distributed Directories

Michael Bauer and Arin Zahalka
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract

The ISO's X.500 Directory service provides agents with information about entities within a distributed environment. The standard identifies information (entry data) to be maintained by the directory as well as information (knowledge) that is used by the directory agents to facilitate cooperative activities, such as searching, updating and manipulating entry data. It does not address the implementation of distributed algorithms for these operations. This paper discusses different types of knowledge and algorithms for distributed search. The knowledge can be categorized into two types: that dealing with network interconnection or topology and that dealing with directory information.

1 Introduction

The wide availability of relatively inexpensive communication services and computer hardware has made distributed computing seem viable and able to provide many advantages: increased availability and access to resources, increased reliability and better performance. An integral part of a distributed environment is a Distributed Directory Service (DDS). Such a service aids users, processes and applications in identifying and locating objects or entities in a distributed environment [6]. Entities include persons, groups, hosts, organizations, processes and applications. The DDS also manages the information pertinent to these entities. Thus, the DDS acts as both an information provider and an information manager, maintaining entity information in the face of changes to users, nets, works, organizations and services [3]. It can be used by any application which interacts with named entities in the distributed environment. These applications include e-mail, telematic services, file management systems, file transfer protocols, and security and authentication procedures.

Some of the major work on distributed directories is the outcome of standardization efforts in the area of communications. The ongoing X.500 standard is the culmination of efforts to standardize directory services. One aspect not addressed by this standard is the specification of distributed algorithms to perform directory operations such as Searching, Adding, Deleting, Updating and Modifying Directory entries. This paper focuses on one of the fundamental operations, namely, search.

1.1 An Overview of X.500

The X.500 Directory Standard specifies a Directory Service which provides and manages information about entities. This Directory Service and its information is expected to be distributed physically, yet unified functionally. Although the upcoming 1992 X.500 draft [5] does not yet meet all of the standard's intended requirements, it does specify, among other things, the nature of the information to be stored in the Directory, limited distributed operation functionality and the protocols necessary for communication between nodes in order to perform directory operations.

The X.500 Directory service provides entities within a distributed environment with information about other entities. These entities are represented by entries in a hierarchical name space, the X.500 Directory Information Tree (DIT). Entries are placed in the DIT according to the organizational relationships between the real-world entities which they represent. All the entities and information held by the directory is called the Directory Information Base (DIB). It is assumed that updates to the DIB will be much less frequent than queries and that some service will be provided regardless of network partitioning resulting from events such as non-local site failures. This assumption of query type and frequency is based upon the applications anticipated to use the Directory service. The query type and frequency anticipated is a
major difference in the requirements of a Directory service and a distributed database system.

A DIT entry contains information about the entity that it represents. This entity information is kept in the form of attributes whose types and structures are governed by a set of rules, or integrity constraints, called schemas. Each entry in the DIT is labelled by a Relative Distinguished Name (RDN) composed from an attribute or a set of attributes of that entity and which is unique amongst the other entries who are children of the same parent entry. In this way, each entry has a globally unique name which is composed of the concatenated sequence of relative distinguished names in the path from the root of the DIT to the entry itself. This globally unique name is called the entry's Distinguished Name (DN).

The X.500 Abstract Service Definition [4] defines abstract ports and operations which provide the user functionality for retrieving, searching and modifying directory information. These operations form the Directory Access Protocol. The current X.500 standard supports the Read, Compare, List, Search and Abandon interrogation functions and the basic manipulation functions – Add, Remove, Modify entry and Modify RDN. The Directory service’s modification functionality is limited in that it does not support the arbitrary creation and deletion of non-leaf entries.

The Directory service (including the DIT) is distributed over physically separated entities (such as computer nodes) called Directory System Agents (DSA’s). This distribution is transparent to the user. Each user or user-process is represented by a Directory User Agent (DUA) which is responsible for querying or interactively interrogating the directory.

1.2 Directory Search

Given a distributed directory, there is a need to support operations such as search and update. The search operation is fundamental in locating, retrieving, manipulating and, in general, utilizing the information available at the DSAs. Several kinds of distributed search operations are possible within the scope of X.500: search for entry with specified distinguished name or other attributes, search for entry matching a filter, or search for entry with an incompletely specified DN (e.g., an RDN). The basic search operation requires that the distinguished name of the entry desired be provided. Each site has a directory agent (DSA) and all agents collectively cooperate to provide the distributed search service throughout the network.

For the deployment and inter-operation of DSAs it is important to understand what kinds of knowledge (meta-information, facts, assumptions, constraints etc.) can be helpful in locating the requested entry. Since a DSA may receive a search request for an entry which it does not hold, it must have some knowledge to enable it to interact with other DSA’s and send the search request, either directly or indirectly. The use of different kinds of knowledge are examined in this paper: Topological Knowledge and Directory Knowledge. Topological Knowledge encompasses knowledge that pertains to interconnections among agents and sites. DIT Knowledge covers the types of knowledge that deal specifically with the DIT, including its partitions and the relationships among them. Algorithms using this knowledge are discussed and are based on the following assumptions about the network and each cooperating agent: 1) the message service is reliable, 2) the agents are autonomous except for message exchange used in seeking information, 3) each agent executes the same algorithm for processing requests, and 4) all agents use the same type of knowledge.

2 Connectivity Knowledge

In order for any kind of effective communication to take place, each agent must have some knowledge about the topology of the network in which it exists. The most basic knowledge an agent needs is about the agents to which it has a communication link. Under these circumstances, since the network is connected, a message from one agent can reach any other agent in the network. This type of Topological Knowledge is called Adjacent Connectivity Knowledge.

Let \( N = \langle S; C \rangle \) be a network where \( S \) is the set of sites, each with a directory agent, and \( C \) is the graph of connections among the sites. An edge \( S_i, S_j \in C \), \( S_i, S_j \in S \), exists if the agent at site \( S_i \) is “logically connected” to the agent at \( S_j \). That is, \( S_i \) has a PSAP (Presentation Service Access Point) for \( S_j \) (or equivalent).

Definition 1 • Adjacent Connectivity Knowledge. For each site \( S_i \in S \), if \( (S_i, S_j) \in C \), then let \( S_j \in ADJ_i \) – the set of adjacent sites.

Using this knowledge, each agent that receives a request and does not have the answer, must forward the request. There are several ways in which to do this including a broadcast or multicast in parallel to a site’s neighbors [1]. A depth-first strategy is employed by the algorithms discussed here (see [7] for more details).
The agents on the sites of the network cooperate by exchanging messages. Since the only knowledge available to these agents is Adjacent Connectivity Knowledge, they must rely on messages sent between the "adjacent" sites. Each message relating to the same search request carries a unique identifier (assigned by the originator) so that associated replies can be determined. In order for the originating site to be able to receive the response, the identity of the originator of a particular request is included in all messages associated with that request. Whenever a site sends a message, it includes its identity. Four types of messages are possible in relation to any Search request: a REQUEST message (carrying the request itself), a REJECT message (indicating that the site has previously received the same request), a YES message (carrying the information requested), and a NO message (indicating that the site is unable to obtain the answer).

Because this knowledge concerns only the interconnection topology of the network of agents and not what directory information is stored on the sites, the search for the entry with the purported distinguished name is done by checking the sites along the paths of the network which the request will travel. This means that the answer will be obtained when the request eventually encounters the site which holds the desired entry. A flooding approach [1] may also be employed, though more messages will be needed [2].

In terms of the number of messages involved, the worst case scenario for a request occurs when the answer does not exist in the network. This is due to the fact that each agent in the network must receive the request in order for it to be finally determined that the answer does not exist. In this case, each connection in the network will carry exactly two messages related to the request. Thus, if NC is the number of connections in the network, then 2NC is the maximum number of messages that will be generated for a request. If n is the number of sites, then one can show that the worst case is n(n - 1) messages [7].

3 Search with DIT Knowledge

Since Adjacent Connectivity Knowledge deals only with the logical connections and not the DIT, the previous algorithm is unable to selectively direct the search to a site which may contain the desired entry. Additional knowledge of the DIT information that the sites contain is required.

The DIT is partitioned and these partitions are located on the different sites of the network. Note that a site may be allocated more than one partition or it may be allocated no partitions. The following captures these basic notions and can be made more precise [7]:

Definition 2 - A Distributed Directory consists of a DIT, a set of partitions MP of the DIT which have been allocated to the sites of a network N by a mapping DD which ensures that each partition resides on exactly one site.

One type of DIT Knowledge deals with the organization of the DIT, its partitioning, and the location of a partition on the sites of the the network. Since the entries are organized hierarchically in the DIT, the search for an entry would involve traversing the DIT by relying on the hierarchical name structure. Such a traversal requires knowledge of a partition's hierarchical placement in the DIT. That is, each site must have knowledge of the predecessor and successor partitions of each of its own partitions. We call this type of knowledge DIT Partition Knowledge. This knowledge is part of the knowledge assumed by the X.500 standard.

Definition 3 - DIT Partition Knowledge. Each site $S_i \in S$ has a set of predecessors $PRED_i$, where for each partition $P_j \in MP$ residing on $S_i$, there is a predecessor $Pred_j \in PRED_i$, $Pred_j = (PN_x, S_y)$ such that:

1. $PN_x$ is the distinguished name representing some partition $P_x \in MP$.
2. $P_x$ resides on some site $S_y$; $P_x$ is the called the predecessor partition of $P - j$ and $S_y$ is called a predecessor site of $S_i$.
3. $Pred_j = \emptyset$ if the distinguished name of $P_j$ is the DIT root.

Similarly, each site $S_i$ has a set of successors SUCCi.

With DIT Partition Knowledge it is possible to traverse the DIT because the predecessor and successor partitions and sites are known. In order to find an entry, the partitions are traversed from site to site until the desired entry is found. Note that since a connection between partitions does not necessarily imply a connection between the respective sites that these partitions reside on, it may be the case that a successor or predecessor partition resides on a non-neighboring site. Unfortunately, DIT Partition Knowledge by itself is not particularly useful. It can be shown [7] that in the worst case performs only as well as Adjacent...
Connectivity Knowledge, although in actual practice it might prove to be more useful.

Since the DIT is arranged according to an organizational and/or functional hierarchy, it is intuitively logical that an organization have connections with sites that are underneath its power structure. Such sites contain the lower-level partitions which correspond to the children organizations, e.g., departments. An obvious constraint would be to ensure that site interconnections, at a minimum, reflect relationships among the DIT partitions. The constraint on the network dictates that a site have a connection with each of its successor sites.

**Definition 4 • Successor Site Connectivity.** For each \( P_j \in MP \) residing on \( S_i \in S \) the following holds: For each of \( P_j \)'s successor partitions \( P_k \in MP \), let \( P_k \) reside on some site \( S_w \in S \). Then \((S_i, S_w) \in C \).

That is, the connections between partitions coincides with the connections between sites. This constraint guarantees that a site has a connection with each of its successor sites. This also implies that a site also has a connection with each of its predecessor sites.

It is evident that in order for a network under the constraint of Successor Site Connectivity to maintain the imposed connections, a potentially tremendous amount of maintenance overhead is possible. It might also appear that this constraint would impose a potentially great amount of inter-site communication once a connection between two sites is established, e.g. as a result of the alteration of the DIT. The study of distributed algorithms relying on the combination of DIT Partition Knowledge and Successor Site Connectivity is important because it is the minimum knowledge assumed by the X.500 standard.

The combination of partition knowledge and site connectivity provides an alternative distributed search algorithm. Traversing the DIT from partition to partition corresponds to sending a search request to an "adjacent" site. Since every site visited is "adjacent" to the previous site visited, the search request will travel along one path until it reaches the site with the desired entry or it can be determined that no site contains the entry. The longest possible path that the search request can take from the originating site to the site with the entry is \((n - 1)\) arcs long. It would also take \((n - 1)\) arcs to return a response to the originating site. Thus the maximum number of messages that must be propagated is \(2(n - 1)\).

### 4 Summary and Conclusions

The current X.500 standard does not specify global distributed algorithms to perform operations on the Distributed Directory. Several types and combinations of knowledge have been discussed and their impact and use in searching X.500 directories. Topological Knowledge, as defined by Adjacent Connectivity Knowledge, only provides for a depth-first search. DIT Partition Knowledge coupled with Successor Site Connectivity, essentially the minimal knowledge assumed by X.500, provides for an efficient structured search. DIT Partition Knowledge alone, however, leads to a less efficient search.

Future work includes studying algorithms for searching for entries matching any specified attributes and entries matching partially specified attributes (e.g. given an RDN instead of a DN). Protocols and algorithms must be developed to administer, update and maintain the knowledge available to the agents. The maintenance overhead and complexity are also issues requiring further study. The performance and behaviour of the algorithms under failures must also be studied.

### References


