Directory Services

Justin Groce
Systems Programming
jbgroce21@tnech.edu

ABSTRACT

Directory Service (DS) is a service that allows devices to “talk” to each other. It does this by keeping a database that maps names of resources of a network to their address. Users can them connect to the DS servers to retrieve information. The servers can also store information deemed useful for the users or applications that will be connecting to the database. DS is based on a standard called X.500.

DS also defines the protocol in which a user or application is to talk to the server. This connection with the server could be to find the address associated with a resource attached to the network. To get this address, the user or application needs only to know the name of the resource.

The fact that the user needs only to know the name of a resource greatly eases the process of connecting to a resource for a human user, as humans use names to talk to and about people and places on a daily basis.

Keywords

DS, Directory Services, Network.

1. What is Directory Services

Directory Services (DS) is a service used to effectively and efficiently manage resources within a network. That network can be very small or global. Any network needs DS. DS can be thought of as a database of resources contained in a specified network. Those resources are objects such as printers, servers, network users, and applications [1].

DS serves as a map of the names of each resource to its individual address. The purpose of this is to make the process of finding a resource simple for users. Users, as humans, use names when referring to people or places. In the same way, users find it easier to refer to the objects in the network by their names. However, not only users make use of DS. Applications are also able to use DS to retrieve addressees. One example of an application that can use the database in DS is an email application. When a user sends an email, that user addresses the email to the name of the recipient. The application then uses the DS database to get the network address by knowing the name [1].

By X.500 guidelines, a global DS can be easily broken up into smaller and smaller networks. Starting at a very small network, each DS is responsible for its local names and addresses. Then each local DS can get combined into a lager DS until every network is connected. This keeps things running quickly and efficiently [2].

Another helpful task that DS provides for a network is to define a name space for that network. A name space is a group of rules that defines how the resources in a network are to be named within the name space. A name space is needed to ensure that all the names of the resources are unique and that there is a one-to-one mapping of resources to names. The DS serves as a name space by keeping a map of each name to the corresponding network address. This allows for the network address of a resource to change while its name remains the same within the DS. This is accomplished by changing the address once in the DS database. The DS will then give out the new address when the user looks up the old name [1].

If networks did not have DS, the user would have to know the network address of every resource they wanted to connect to. This is highly unreasonable, especially within a very large network.

The DS database can also store information useful to users such as network settings and configurations. In this way, DS has become a distributed global database of sorts.

2. Implementation

By applying the X.500 standard, a specific protocol is followed for applications to connect to the Directory Service (DS) database. This protocol is called Directory Access Protocol (DAP) [4].

A modification to DAP to simplify the process is the Lightweight Directory Access Protocol (LDAP) [3]. For the remainder of the section, LDAP will be discussed. This is because LDAP is basically a simplified version of DAP. Therefore, LDAP follows much the same principles that DAP follows.

LDAP takes out much of the protocol in DAP that is considered to be unessential to the core needs of a DS. LDAP gives users or applications a way to know how to access the addresses they need to connect to resources on a network they are directly or indirectly connected to. When designers implemented LDAP, they created separated servers that would index all the resources they knew about individually. They also know how to find the requested information if it is not locally stored. LDAP doesn’t tell users how to design their systems in order to use LDAP. Instead, LDAP is simple a language in which applications know how to get information from LDAP enabled servers. This concept also works with server to server applications [3].

One possible way of getting information from an LDAP server is requesting a specific string. The server then filters its data with the string to find possible data of interest to the requester [3].

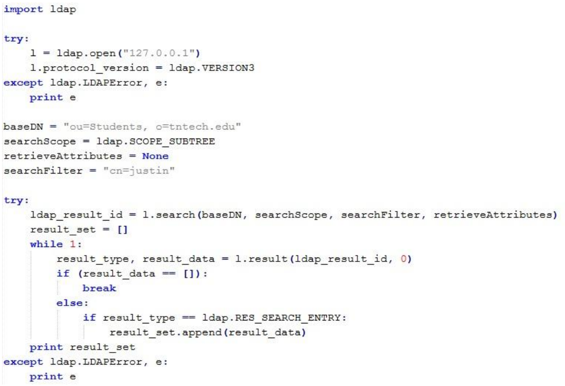
Of course with security being of the utmost importance in a system that gives out addresses of resources, LDAP provides security by using permissions. These permissions are a way for the administrator of a particular LDAP server to set up who can and who cannot gain access to the server. LDAP administrators can also choose to keep some records hidden from users or applications that connect to the server. This is done by setting the data as private. Much like programming languages can set variables to private to restrict other classes from accessing and changing information, private data in LDAP servers can only be accessed by those chosen by the administrator. However, making data private is an optional feature in LDAP [3].

LDAP is set up to hold data types chosen from a base list of data types. This is called a schema. When a schema is chosen, attributes are given to the data type [3]. For example, a possible data type entry would be “student.” The student data type could have as many attributes as the administrator wanted to keep track of. These attributes could be things such as “name,” “grade,” “gradePointAvg,” and “schoolName.” The attributes of a data types are given back to the user or application that request information about the data type.

Information can be retrieved from an LDAP server using a variety of different methods. As long as the client used the LDAP language, it can connect and receive information. Some common languages used to implement an LDAP client are python and ruby. Sample code can be found using a quick Google search.

The important thing to remember is the “bind” call. This is what allows access to the server.

A simple example in python is as follows.



```
import ldap

try:
    l = ldap.open("127.0.0.1")
    l.protocol_version = ldap.VERSION3
except ldap.LDAPError, e:
    print e

baseDN = "ou=Students, o=tntech.edu"
searchScope = ldap.SCOPE_SUBTREE
retrieveAttributes = None
searchFilter = "cn=justin"

try:
    ldap_result_id = l.search(baseDN, searchScope, searchFilter, retrieveAttributes)
    result_set = []
    while 1:
        result_type, result_data = l.result(ldap_result_id, 0)
        if (result_data == []):
            break
        else:
            if result_type == ldap.RES_SEARCH_ENTRY:
                result_set.append(result_data)
    print result_set
except ldap.LDAPError, e:
    print e
```

3. Conclusion

Directory Service (DS) has become one of the best and largest distributed global databases. By implementing the Directory Access Protocol (DAP) or the smaller Lightweight Directory Access Protocol (LDAP), DS maintains data about resources within networks efficiently.

DS allows humans to use applications or to connect to resources in any network they are connected to with the ease of using names. Humans are already used to referring to objects by their names. Many go as far as naming things such as their cars or their lucky bowling ball. Therefore, when using a computer, humans would much rather connect to “my printer” than 154.234.0.128. DS provides this service for the user. Applications also are able to take advantage of using DS servers to connect to the database holding the information they need.

4. REFERENCES

[1] McLain, Nancy. "What Is a Directory Service?" (Nov 2010) http://support.novell.com/techcenter/articles/anp20000501.html

[2] Weider, C. and J. Reynolds. "Executive Introduction to Directory Services Using the X.500 Protocol" (Nov 2010) http://delivery.acm.org.ezproxy.tntech.edu/10.1145/rfc_fulltext/RFC1308/rfc1308.txt?key1=RFC1308&key2=6915829821&coll=DL&dl=ACM&CFID=110046732&CFTOKEN=15309991

[3] "What is LDAP?" (Nov 2010) http://www.gracion.com/server/whatldap.html

[4] "X.500 Overview" (Nov 2010) http://download.oracle.com/javase/jndi/tutorial/ldap/models/x500.html