ROLE OF LDAP IN ENTERPRISE

Amrita Shukla
Lecturer, Electronics & Telecomm. Engg.
U.I.T. RAIPUR[C.G.]

Dr.A.S.Thoke
Prof. & H.O.D., Electrical Engg.
N.I.T. RAIPUR[C.G.]

Ms. Smita Shukla
Lecturer, Electrical Engg. Deptt.
N.I.T. RAIPUR[C.G.]

ABSTRACT

Lightweight Directory Access Protocol (LDAP) is a fast growing technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that needs to be available for today's distributed systems and services. After a fast start, it can be assumed that LDAP has become the de facto access method for directory information, much the same as the Domain Name System (DNS) is used for IP address look-up on almost any system on an intranet and on the Internet. LDAP is currently supported in most network operating systems, groupware and even shrink-wrapped network applications.

Introduction

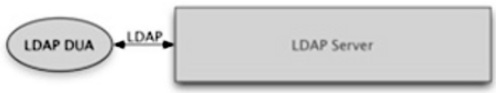
People and businesses are increasingly relying on networked computer systems to support distributed applications. These distributed applications might interact with computers on the same local area network (LAN), within a corporate intranet, or anywhere on the worldwide Internet. To improve functionality, ease of use and to enable cost-effective administration of distributed applications information about the services, resources, users, and other objects accessible from the applications needs to be organized in a clear and consistent manner. Much of this information can be shared among many applications, but it must also be protected to prevent unauthorized modification or the disclosure of private information. Information describing the various users, applications, files, printers, and other resources accessible from a network is often collected into a special database, sometimes called a directory. As the number of different networks and applications has grown, the number of specialized directories of information has also grown, resulting in islands of information that cannot be shared and are difficult to maintain. If all of this information could be maintained and accessed in a consistent and controlled manner, it would provide a focal point for integrating a distributed environment into a consistent and seamless system.

The Lightweight Directory Access Protocol (LDAP) is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory. LDAP is gaining wide acceptance as the directory access method of the Internet and is therefore also becoming strategic within corporate intranets. It is being supported by a

M. Iskander (ed.), Innovations in E-learning, Instruction Technology, Assessment, and Engineering Education, 1–6.

growing number of software vendors and is being incorporated into a growing number of applications.

Old Model



New Model

Fig. 1 Old And New Model of LDAP

What is a Directory?

A directory is a listing of information about objects arranged in some order that gives details about each object. Common examples are a city telephone directory and a library card catalog. For a telephone directory, the objects listed are people; the names are arranged alphabetically, and the details given about each person are address and telephone number. Books in a library card catalog are ordered by author or by title, and information such as the ISBN number of the book and other publication information is given.

In computer terms, a directory is a specialized database, also called a data repository, that stores typed and ordered information about objects. A particular directory might list information about printers (the objects) consisting of typed information such as location (a formatted character string), speed in pages per minute (numeric), print streams supported (for example PostScript or ASCII), and so on. Directories allow users or applications to find resources that have the characteristics needed for a particular task. For example, a directory of users can be used to look up a person's e-mail address or fax number. A directory could be searched to find a nearby PostScript color printer. Or a directory of application servers could be searched to find a server that can access customer billing information. The terms white pages and yellow pages are sometimes used to describe how a directory is used. If the name of an object (person, printer) is known, its characteristics (phone number, pages per minute) can be retrieved. This is similar to looking up a name in the white pages of a telephone directory. If the name of a particular individual object is not known, the directory can be searched for a list of objects that meet a certain requirement. This is like looking up a listing of hairdressers in the yellow pages of a telephone directory. However, directories stored on a computer are much more flexible than the yellow pages of a telephone directory because they can usually be searched by specific criteria, not just by a predefined set of categories.

Differences Between Directories and Databases

A directory is often described as a database, but it is a specialized database that has characteristics that set it apart from general purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) much more often than they are updated (written). Hundreds of people might look up an individual's phone number, or thousands of print clients might look up the characteristics

of a particular printer. But the phone number or printer characteristics rarely change. Because directories must be able to support high volumes of read requests, they are typically optimized for read access. Write access might be limited to system administrators or to the owner of each piece of information. A general purpose database, on the other, hand needs to support applications such as airline reservation and banking with high update volumes. Because directories are meant to store relatively static information and are optimized for that purpose, they are not appropriate for storing information that changes rapidly. For example, the number of jobs currently in a print queue probably should not be stored in the directory entry for a printer because that information would have to be updated frequently to be accurate. Instead, the directory entry for the printer could contain the network address of a print server. The print server could be queried to learn the current queue length if desired. The information in the directory (the print server address) is static, whereas the number of jobs in the print queue is dynamic. Another important difference between directories and general purpose databases is that directories may not support transactions (some vendor implementations, however, do). Transactions are all-or-nothing operations that must be completed in total or not at all. For example, when transferring money from one bank account to another, the money must be debited from one account and credited to the other account in a single transaction. If only half of this transaction completes or someone accesses the accounts while the money is in transit, the accounts will not balance. General-purpose databases usually support such transactions, which complicates their implementation.

Because directories deal mostly with read requests, the complexities of transactions can be avoided. If two people exchange offices, both of their directory entries need to be updated with new phone numbers, office locations, and so on. If one directory entry is updated, and then other directory entry is updated there is a brief period during which the directory will show that both people have the same phone number. Because updates are relatively rare, such anomalies are considered acceptable. The type of information stored in a directory usually does not require strict consistency. It might be acceptable if information such as a telephone number is temporarily out of date. Because directories are not transactional, it is not a good idea to use them to store information sensitive to inconsistencies, like bank account balances. Because general-purpose databases must support arbitrary applications such as banking and inventory control, they allow arbitrary collections of data to be stored. Directories may be limited in the type of data they allow to be stored (although the architecture does not impose such a limitation). For example, a directory specialized for customer contact information might be limited to storing only personal information such as names, addresses, and phone numbers. If a directory is extensible, it can be configured to store a variety of types of information, making it more useful to a variety of programs.

Another important difference between a directory and a general-purpose database is in the way information can be accessed. Most databases support a standardized, very powerful access

method called Structured Query Language (SQL). SQL allows complex update and query functions at the cost of program size and application complexity. LDAP directories, on the other hand, use a simplified and optimized access protocol that can be used in slim and relatively simple applications.

Because directories are not intended to provide as many functions as general-purpose databases, they can be optimized to economically provide more applications with rapid access to directory data in large distributed environments. Because the intended use of directories is restricted to a read-mostly, no transactional environment, both the directory client and directory server can be simplified and optimized.

Directory Clients and Servers

Directories are usually accessed using the client/server model of communication. An application that wants to read or write information in a directory does not access the directory directly. Instead, it calls a function or application programming interface (API) that causes a message to be sent to another process. This second process accesses the information in the directory on behalf of the requesting application. The results of the read or write are then returned to the requesting application (see Figure 2).

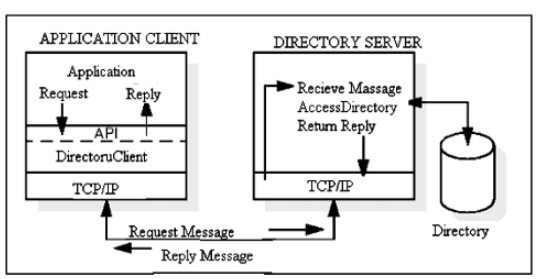


Figure 2. Directory Client / Server System

The request is performed by the directory client, and the process that looks up information in the directory is called the directory server. In general, servers provide a specific service to clients. Sometimes a server might become the client of other servers in order to gather the information necessary to process a request. A directory service is only one type of service that might be available in a client/server environment. Other common examples of services are file services, mail services, print services, Web page services, and so on. The client and server processes might or might not be on the same machine. A server is capable of serving many clients. Some servers can process client requests in parallel. Other servers queue incoming client requests for serial processing if they are currently busy processing another client's request. An API defines the programming interface a particular programming language uses to access a service. The format and contents of the messages exchanged between client and server must adhere to an agreed upon protocol. LDAP defines a message protocol used by directory clients and directory servers. There is also an associated LDAP API for the C language and ways to access LDAP from writhing a Java application. The client is not dependent upon a particular implementation of the server, and the server can implement the directory however it chooses.

Directory-Enabled Applications

A directory-enabled application is an application that uses a directory service to improve its functionality, ease of use, and administration. Today many applications make use of information that could be stored in a directory. For example, consider a group calendar application that is used to schedule

meetings of company personnel in different conference rooms. In the worst case, the calendar application does not use a directory service at all. If this were the case, a user trying to schedule a meeting would have to remember the room number of every conference room that might be appropriate for the meeting. Is the room big enough, does it have the necessary audio and video equipment, and so on? The user would also have to remember the names and e-mail addresses of every attendee that needs to receive a meeting notice. Such an application would obviously be difficult to use. If conference room information (size, location, special equipment, and so on) and personnel information (name, e-mail address, phone number, and so on) could be accessed from a directory service, the application would be much easier to use. Also, the functionality of the application could be improved. For example, a list of all available conference rooms meeting the size and equipment requirements could be presented to the user. But the developers of directory-enabled applications are faced with a problem. What if they cannot assume that a directory service will exist in all environments? If there is a directory service it might be specific to a certain network operating system (NOS), making the application non-portable. Can the existing directory service be extended to store the type of information needed by the application? Because of these concerns, application developers often took the approach of developing their own application-specific directory.

Conclusion

Lightweight Directory Access Protocol (LDAP) is a fast growing technology for accessing common directory information. LDAP has been embraced and implemented in most network-oriented middleware. As an open, vendor-neutral standard, LDAP provides an extendable architecture for centralized storage and management of information that needs to be available for today's distributed systems and services.

Another very important role for LDAP in Active Directory is that of cross-platform access interface. LDAP is not tied to a particular platform as is ADSI, which is COM-based and therefore tied to Windows. That means applications and scripts can be written to access and manage Active Directory from virtually any platform. This is a huge improvement over Windows NT4 where access was limited to the Win32 API, which for the most part can only be used from the Windows platform. LDAP allows companies that have a non–Windows-based enterprise management infrastructure the ability to populate, maintain, and monitor Active Directory from the platform of choice.

With LDAP Version 3, a solid foundation for a directory service infrastructure for the Internet was built. As we have seen in previous chapters, most vendor implementations are based on this version or have most features of Version 3 incorporated. But there is still room for enhancements, for example in areas of API support for other program languages, like Java. To define these standards, members of the Internet Engineering Task Force (IETF) work on and submit draft proposals that eventually might become Request for Comments (RFCs). The RFCs describe the idea and the implementation of the major design and technologies for the new functions and features. We describe

some important proposed enhancements in the next section. Although there is not a specific section in this book devoted to it, it should be mentioned that the vendor products will of course be further developed and enhanced, namely in areas such as improved functionality, manageability, and performance. For example, client-side caching could be implemented to improve performance remarkably, especially on multi-user client systems with heavy directory access. Graphical management tools can be added, or existing GUIs may be improved that allow easy configuration and contents management. As LDAP matures to a de facto standard, it will eventually replace proprietary directory services in vendor products and other standardized middleware solutions, such as the Distributed Computing Environment (DCE). DCE makes heavy use of a directory service and currently uses its own, specific implementation, called Cell Directory Service (CDS).

References

[1.] S. Bradner, "Key Words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[2.] J. Klensin, R. Catoe, P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 1997.

[3.] J. Myers, "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[4.] Wahl, M., Howes, T., Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, August 1997.

[5.] Wahl, M., Coulbeck, A., Howes, T., Kille, S., "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[6.] Paul J. Leach, Chris Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

[7.] Ed Reed, "LDAP Subentry schema", draft-ietf-ldup- subentry-08.txt, April 2001.

[8.] K. Zeilenga, "LDAP Password Modify Extended Operation", RFC 3062, February 2001.

[9.] Understanding and Deploying LDAP Directory Services Timothy A. Howes, Mark C. Smith and Gordon S. Good Macmillan Network Architecture and Development Series. Implementing LDAP

[10.] Mark Wilcox Wrox Press Ltd Perl for System Administration

[11.] David N. Blank-Edelman O'Reilly