USENIX Association

Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

Gaining the Middleground: A Linux-based Open Source Middleware Initiative

Dr. Greg Wettstein Ph.D., *North Dakota State University*
Johannes Grosen M.S., *North Dakota State University*

August 25, 2000

Abstract

Central to the development of the Internet, and the resulting business paradigm shifts, has been the adoption of standards which have provided the necessary framework for computer systems to cooperate and exchange information. The next major step necessary to support the continuation of this process is the development of software protocols and tools which provide the ability to manage the delivery of services to users.

This paper describes a Linux-based middleware initiative which has been deployed at North Dakota State University to support a Category of Service information delivery system for the North Dakota Higher Education Computing Network. The system was designed to support a computing model where the majority of the service infrastructure is supplied by Linux servers and which is extensible to the 11 state institutions comprising the HECN.

The paper begins with a discussion of middleware and its importance in the development of next-generation information systems. Included in this discussion are the topics of identification, authentication and authorization and the requirement for them in a modern middleware solution. Central to this topic will be a discussion of the potential threat that proprietary middleware solutions pose to the continued penetration of Open Source efforts into the enterprise computing model.

The paper then discusses in detail the design and architecture of the User Services Management System which implements the middleware solution at NDSU. The three fundamental components discussed will be the User Services DataBase, LDAP meta-directory services and the User Account Management System .

Discussion of the system architecture will include details of how standard services have been modified to operate under middleware control.

The paper will conclude with a discussion of the implications of this work. Also included is a discussion of the next generation of the system architecture.

Introduction

North Dakota State University (NDSU) is one of eleven colleges and universities in the North Dakota University System. With an enrollment of 10,000 students, NDSU is a land-grant institution offering several Phd programs and home to several internationally recognized researchers in the fields of chemistry, engineering, and agriculture. NDSU is a member of Internet2 and a partner in the Great Plains Network, a NSF-funded regional network connecting six mid-western states and supporting earth system sciences research. The authors are employed by the NDSU campus computer center, Information Technology Services, (ITS). Dr. Wettstein is the senior system administrator and Mr. Grosen is the associate director for networking and multi-user computing systems. ITS provides desktop computing support and services, training, networking, and other information technology to the NDSU campus. ITS is also home to the SENDIT network, a K-12 technology services initiative for the state of North Dakota.

As a rural state with a small population, North Dakota has faced and continues to face many challenges in deploying information technology services. In the mid

1970's, university system administrators conceived of the notion of tackling some of these problems by centralizing as many services as possible and deploying a communications infrastructure to provide access. The resulting system of services was named the Higher Education Computing Network (HECN). Initially, the HECN concentrated on providing administrative computing services and the University of North Dakota was designated the administrative computing host site. In the late 70's it was recognized that technology was becoming an important component in academics and, in 1980, NDSU was charged with the responsibility of centrally providing academic computing services. Initially, the infrastructure consisted of a single IBM mainframe computer. Over time, it came to include multiple servers and services such as email, domain name services, USENET news, Unix shell access, etc. When the mainframe system was removed from service four years ago, the model for delivering services remained essentially the same; users accessed services via interactive logins over the state's wide area networks.

As desktop PCs became more ubiquitous and the advent of graphical interfaces made these applications more accessible to the non-technical, users began to prefer using their desktop software over that available on the centralized servers. Concurrently, a survey of HECN users and administrators revealed that there was consensus on the need for a unified electronic "phonebook" or directory. The survey also revealed the desire for a standardized scheme for email addresses. Two criteria were identified: first, the recipient portion of the email address should be based, as closely as possible, on the person's full name and, second, the dependency on hostnames for mail servers should be replaced with DNS domain names and MX records. These observations and requests became input into a major re-design of the infrastructure for providing HECN academic computing services.

During the design process, additional technical requirements were developed. Key among these was moving to a client-server architecture using micro-servers running Linux. The old server infrastructure was based on several Unix hardware and operating system combinations. This requirement would allow us to reduce the heterogeneity in the server infrastructure, simplify management, ease development, and lower costs.

Another technical requirement was to move to a Category of Service (CoS) model. In the "old" model, users were assigned Unix logins which gave them access to all services (email, printing, etc.) supported on the server. Servers were, in effect, equivalent to certain standard sets of services and users were assigned according to their needs. In some cases users were assigned to multiple servers to provide them with all the services they required. In the client-server model, Unix logins could no longer be used. Thus, a new way of creating and assigning sets of services for users was required.

The CoS model was the answer to this problem. Services would no longer be tied to any particular server. Each user would be assigned a single "login" with a single password. Each service would be separate from all other services and enabled or disabled without affecting any other services for that user. Custom sets of services could be created for users without the need for multiple server logins.

Two fundamental problems arose out of these requirements:

- how to identify, authenticate, and authorize access to services
- how to implement and manage the CoS model

These problems led us to the discovery of the importance of *middleware* as a basic component of the IT architecture we were developing.

The Necessity for Middleware

The requirements and design process described above began in 1997. At that time, the notion of middleware was not widely known in the field of IT. Instead, middleware was understood by its major components: *identification*, *authentication*, and *authorization* (IAA). These components must be clearly understood before the importance and role of middleware can be understood.

Identification, Authentication and Authorization

A key component in a distributed, client-server-based network is the ability to reliably and consistently identify users. It is extremely desirable (at least from a user's perspective) if there is a single identifier that universally identifies them throughout the enterprise. This latter goal has typically been very difficult to implement because, in a heterogenous network, servers and/or services may have different requirements for an identifier. In addition, in larger enterprises, the network may be composed of several administrative subdomains making consistency an administrative challenge at the very least. Ideally, there should be a single external identifier (user id) and some means of mapping that identifier, if necessary, to an internal, invariant identifier. This solves the problem of the heterogenous environment and permits the user id to be changed by the user without affecting the internal identifier.

Once we have the means to identify users the next challenge is to verify that the identity presented is being used by the person it is assigned to. This is authentication; the process of identification verification. Traditionally, authentication has been accomplished by the use of an identifier and password pair. The user id uniquely identifies the user to the system and the password can be used to verify the identity of the user. As with the identifier, it is desirable to have a single password that can authenticate the identifier throughout the enterprise. It is also important that every effort is made to insure that the use of the password on the network is secure from "snooping" and other attacks.

Most discussions stop at this point; we have an identifier and the means to authenticate the use of that identifier. However, in this model, with a single identifier and password for every user available to every server on the network, how do we determine whether to allow access to a service? Traditionally, authorization is implied by a valid identifier and password as in a Unix login. If a user can login to the server they have access to whatever services are available on that server. In the distributed computing model, this is no longer enough. We have to be able to determine whether a valid identifier is authorized to access a service and the terms under which the access can occur. Thus, a complete solution is only possible when we have all three components; identification, authentication, and authorization.

The Role of Middleware

Middleware is a term which refers to the set of services composed of IAA, APIs, and management systems which support the needs of a distributed, networked computing environment. As the name "middleware" suggests, it is, using the hierarchical model, a layer of services which sits between applications and the services they access. An excellent overview of middleware can be found in RFC 2768[1]. The RFC's authors admit that even today, middleware is still not a well-defined term but the above definition is consistent with the RFC's and is sufficient for the purposes of this paper.

Middleware, at first glance, seems somewhat innocuous or invisible, especially when it works well. This tends to disguise the importance of this layer, however. Middleware is a critical component in the CoS model. It is the middleware which supports the creation and removal of users, provides the ability to verify an identity, manages the addition and removal of services associated with user ids, and even the addition and removal of services available to users. Within the enterprise, a good middleware solution makes the delivery of services transparent to users. There is no need to know which server is providing a given service and a single login/password provides access to all of the enterprise's services, subject to whatever authorization policies have been established. In the business-to-business world, where multiple enterprises establish electronic relationships, middleware will perform the same function but it is complicated by the fact that there are currently no middleware standards in place.

The IT community today certainly recognizes the importance of middleware. The Internet2 community has formed a working group to investigate middleware. Companies like Microsoft (Active Directory) and Novell (NDS) have committed huge amounts of money and, in effect, bet their corporate futures on gaining adoption of their middleware implementations as standards. This makes middleware, in the authors' opinion, the single most important component in future networked computing environments. The Open Source community, and the Linux community in particular, must recognize its importance.

A proprietary solution such as Active Directory or NDS will irreparably harm the Open Source movement. The success of Linux in the enterprise has largely been because of its adherence to open standards and availability of source code. This has permitted managers to easily de-

[1]http://www.ietf.org/rfc/rfc2768.txt?number=2768

ploy Linux into existing enterprise IT architectures where open standards exist. If the middleware standard isn't open the implications are obvious. For example, consider the SMB file protocol and the open-source Samba software. With the introduction of Microsoft Windows 2000 and the incorporation of Active Directory into file and print services, Samba's viability is immediately threatened. This scenario will repeat itself over and over unless the middleware is standards-based and open.

An Open Source Middleware Marriage

The Open-Source community provided the important infrastructure for the development of the CoS middleware solution described by this paper. In the current implementation the three basic components of the IAA process are handled by the LDAP directory server provided by the OpenLDAP[2] project and by the MIT Kerberos[3] authentication system.

The success of this middleware implementation was a function of having access to source based tools which implemented commonly accepted standards and protocols for identification, authentication and authorization of users for information technology services.

LDAP Directory Services.

The most central component of a middleware initiative is the ability to identify a person or entity to which services are being provided. The Lightweight Directory Access Protocol (LDAP) provides both an open protocol as well as an open-source solution for implementing the identification component of the process.

The directory services component of the middleware solution is essentially a database of all objects within an enterprise information delivery system. This notion of managing a computing infrastructure through a network accessible database or directory system is referred to as 'directory enabled computing'. Proper implementations of directory systems provide the advantage of centralizing the tracking of resources as well as decreasing administrative costs by centralizing the control and management of service delivery.

Directory systems are based on representing service entities and computing resources in a hierarchical tree form. The most critical element in the implementation of a directory is the requirement that each user be granted a unique identifier which optimally should be considered immutable throughout the users lifetime in the organization. This identifier forms the basis for the Distinguished Name (DN) in the directory, essentially the equivalent of a database "key." This characteristic is particularly important in the implementation of digital signature systems since digital certificates are branded with the DN of the user which the certificate will authenticate. As discussed in the previous section, this requirement also makes directories ideal for supporting identification in a middleware solution.

Implementation of an enterprise directory system will become increasingly important to the effective delivery of services in a highly networked enterprise computing environment. Decreasing system administration costs is considered to be a fundamental benefit of directory systems. In the future, the role of directories will become critical in implementing strategies such as organization-to-organization partnering and in managing concepts such as organizational role playing. Role playing refers to the notion that an individual may have multiple functional roles in a organization. Often, it will be desirable to vary a user's authorizations based on the role they are assuming at a given time. Directory services will also become a critical component in monitoring and delivering quality of service guarantees and as an integral component in developing policy based computing initiatives.

An enterprise directory system is critical to the successful implementation and deployment of digital signatures based on PKI or X.509v3 certificate technology. Federal initiatives are underway which will require that organizations wishing to do business with government agencies use digital certification to authenticate transactions and decrease administrative costs by reducing paper-based accounting and transaction schemes.

One of the foremost goals in implementing directory services for this middleware project was to provide the mechanism for implementing a single sign-on identifier system for all users. Included in this goal was the desire

[2] http://www.openldap.org/
[3] http://web.mit.edu/kerberos/www/

to unify the email address with the identifier token used to access other services. Implementing this mechanism enables all users to be provided with a single identifier which is not only the preamble for their email address (component in front of the @ sign) but also their login name or identifier for all other services.

The canonical identification token is referred to as the Information IDentification or IID. This token is formed using an underscore to separate the users first and last names. A tie breaking algorithm was developed to insure that the IID was unique across the entire user domain. In addition to the IID each user is granted an invariant identifier which is used to uniquely identify the user throughout the lifetime of the organization. This invariant identifier is used to construct the Kerberos authentication principal and also serves as the machine identifier for the user. The LDAP directory thus provides the mechanism for mapping an external identification element into the identity of a user which in turn provides access to the information elements (attributes) of that user.

This solution offers a number of advantages to the traditional scheme of granting a user a UNIX style userid. First of all the IID is not limited to eight characters which in turn allows the IID to be more expressive of the users actual identity. This, for example, allows email addresses to be constructed which resemble the users actual name. A second and major advantage is that this mapping allows the external representation of the user to be changed without affecting the actual identification of the user within the system. This allows account creation to occur once without the need to modify machine accounts and/or configurations if the user should wish to change their external identification.

The Open-Source middleware solution described by this paper is heavily reliant on directory services to provide not only a repository of user information but also as the central resource for the authorization of services. Classic middleware solutions within the UNIX environment have typically used directories to manage user information such as numeric ID's, personal information (GECOS) and group relationships. A central goal in the development of this solution was to also facilitate the concept of service authorization. Each individual object describing a user entity has an attribute associated with it which describes whether or not the user is authorized to receive a given category of service. In the current implementation a very simplistic methodology is used to authorize services.

Each service category is specified by a tag field within the services attribute. The basic level of services provided by the system consists of authentication services, email access, remote configuration management, dial-up (modem) access and USENET news reading privileges. Absence of an attribute tag indicates that the service is not granted to the user, a lower case tag is used to indicate that the service has been disabled and an uppercase tag denotes that the user is authorized for the service. The following figure demonstrates three users with different states of access to email services:

No email service	services: USENET
Email disabled	services: USENET email
Email enabled	services: USENET EMAIL

This scheme provides support personnel with the means of determining the services granted to any individual user and the status of that access. The ability to disable a service through the directory provides a mechanism for suspending services without disturbing the actual instantiation of a service. For example, with email services, the delivery of mail to the user will continue even if access to the message store is suspended.

Kerberos Authentication

The second pivotal element of an effective middleware solution is the authentication of the users who are identified via the LDAP directory system. The Kerberos authentication system was implemented to provide this component for the CoS information delivery system.

Implementation of an LDAP directory system simply provides a mechanism for identifying a user within the organization and associating the user with a set of attributes describing the user and their role in the organization. An equally important part of this process is the guarantee that the user who is interacting with the system is actually the user identified in the directory system.

In some directory implementations the authentication process is conducted by storing authentication tokens such as passwords as an attribute associated with a user's directory object. Other schemes rely on the use of digital certificates to authenticate the user to the database via the

Secure Socket Layer (SSL) protocol. The Kerberos authentication system was chosen to provide this important functionality for a number of reasons. First and foremost was the desire to separate the authentication and identification components of the middleware process. A second imperative was to leverage the extensive security experience of the Kerberos team in developing a scheme for strong network based authentication. A final rationale was the ongoing concern of the design team with the issue of user credentialling including the imposition of finite time limitations on the authentication and authorization periods.

The Kerberos system is based on the notion of a Trusted Third Party authentication scheme sometimes referred to as a shared secret system. Each entity authenticated by the system is referenced by an alpha-numeric tag known as a principal. A secret key is associated with each principal maintained by the authentication database. A user authenticates to the system by encrypting a request for authentication. Successful decryption of the request validates the user.

Additional security guarantees are provided by insuring that a server providing services to the user is indeed a legitimate server. This functionality prevents security attacks such as IP spoofing and DNS contamination. The authentication server provides the user with a token which is encrypted with a secret known only by the server dispensing a particular service. The server must successfully decrypt this token for the service connection to be properly authenticated.

In addition to authentication the Kerberos system also provided this middleware solution with the resources needed to implement a single-signon system for service access. Web based tools accessed through an SSL secured WEB server provide support and administrative staff a mechanism for managing user passwords. An optimum implementation of Kerberos authentication requires that passwords never be allowed to travel unencrypted on the network. Current client limitations precluded attaining this optimum environment. The use and development of Kerberos provides a solid foundation for developing and strengthening the local security infrastructure as client support matures.

A significant result of this project was the implementation of strong Kerberos authentication and encryption in the Open LDAP directory server. Current implementations of the server include Kerberos IV authentication but did not include support for the current Kerberos 5 release. Ongoing work is being conducted to implement the notion of service classes functioning as authentication entities with respect to service objects within the LDAP directory. This work will be extended as the policy/authorization engine (discussed later) is integrated with identification and authentication services.

The ability of Kerberos V to support separate authentication realms was leveraged heavily in this project. Currently six security realms are supported by this implementation. Support for cross-realm authentication was used to implement inter-operability between servers specific to each organizational unit. The separation of the authentication realms also promoted individual autonomy for the participating organizations. User level administrators within one security realm do not have access to security information in another organization's realm. The separation of the authentication realms also reduces the potential impact in the event of a compromise of one of the Kerberos key management servers.

The ability of the LDAP directory server to identify users merges naturally and symbiotically with the separate authentication realms. Authentication and authorization tools developed as part of this middleware solution use information attributes from the directory server to determine which security realm should be used to authenticate the user. This allows servers supporting separate organizational units to provide services on a cooperative basis with provisioning of services controlled through the directory system.

KerDAP API

The preceding discussion notes the synergy that occurs when elements of identification and authentication are merged. An important component of this middleware solution was the development of an Application Programming Interface (API) which allows common Linux based applications to leverage this middleware-enabled computing architecture. The simplistic programming library developed is referred to as *KerDAP* reflecting the union of the two open source solutions.

In order to be effectively managed through this system, all applications which implement user services needed to be modified to take advantage of the middleware. The goal of the KerDAP API is to encapsulate the mechanics of directory lookups and Kerberos authentication into simple function calls which can be easily integrated into the authentication structure of the open source applications which were used to implement the services.

The current library exports the following four functions:

1. char * IID_Login(char * iid, char *password)
2. char * IID_Service_Login(char *iid, char *password, char *service)
3. char * IID_To_Uid(char *iid)
4. char * Get_Uid(void)

The first function carries out simple identification and authentication when given the canonical identifier (IID) and a password. The second function implements the triad of identification, authentication and authorization (IAA) when given an identifier, a password and a service attribute. In both cases a NULL pointer is returned if the process fails and a character pointer to the machine representation (POSIX uid) on success.

The third and fourth functions are utility functions which are useful when middleware support is enabled in applications. The third function simply carries out the identification process and returns the POSIX userid which the canonical identifier (IID) maps to. The fourth function provides a mechanism for retrieving the value after a successful call of one of the first three functions. All functions cache the POSIX userid in static storage if a mapping is successful.

There are situations where an application needs to be middleware-enabled but source code is unavailable. Other applications implement authentication using an external mechanism. In these cases, the API is of no use and another method is required. KerDAP provides the *kerdauth* command-line utility for these situations. This utility has proven to be particularly useful in a wide variety of CGI applications where middleware support is required. It has also been used as a supplemental authenticator for Squid proxy services as well as user authentication for INND (USENET news). The following usage table summarizes the simplistic character of the application:

Option	**Action**
-A	Authorize mode
-K	Authenticate mode
-e	IID to authenticate or authorize
-h	Print usage
-s	Service to authorize for
-v	Verbose mode

The *kerdauth* utility operates in either authorization or authentication mode. In both modes the utility reads the user password on standard input. This minimizes the potential for security issues related to passing passwords on the command line. In both modes the canonical user identifier (IID) is passed via the -e switch on the command line. In authorization mode the -s switch is used to specify the service which authorization is requested for.

When used as an external authenticator the *kerdauth* utility returns results via the exit status from its execution. A return code of 0 indicates the authentication or authorization was successful. A non-zero return code is used to indicate a failure condition. Specifying the verbose mode causes the application to print out the results of the authentication or authorization procedure.

Middleware Management - User Services Management System

One of the primary yardsticks for measuring the success of a middleware-enabled computing architecture is an increase in manageability of the information delivery infrastructure. The overall goal of this middleware project was a complete end-to-end solution which provided a seamless architecture for the management and tracking of user services and the ultimate instantiation of the services on a target server.

The multi-component system architecture implementing this solution is referred to as the User Services Management System (USMS). This system is unique in that it provides a complete open-source implementation capable of tracking and managing enterprise information services. Of particular importance is the modular and extensible design which allows the system to expand without organizational or geographical limitations. The following sections discuss the essential components of the implementation.

User Services DataBase

Two strategies exist for the management of service entities within a middleware solution:

1. Canonical directory services.
2. Meta-directory services.

In a canonical directory services implementation all user and service information is tracked in the hierarchical enterprise directory. Classical commercial middleware solutions such as Novell Netware's NDS and Microsoft Active Directory implement this type of solution. A meta-directory implementation uses a separate repository of information and provides a scheme for propagating the data objects into one or more directory servers.

An important initial design decision in this middleware solution was to implement a meta-directory strategy for tracking users and services. Directory server systems such as LDAP tend to be optimized for the rapid retrieval of information elements given a system of filtering constraints or search rules. The most important and fundamental limitation of these systems is that they provide no relational data services nor do they provide important data preservation features such as referential integrity or transaction guarantees. For these reasons the decision was made to implement the management of users and services in a relational database system and to propagate the directory objects through a meta-directory update system.

This approach also offered flexibility for future integration into Enterprise Resource Planning (ERP) and data warehousing projects which are under development by the university system. As was noted previously, middleware management systems will be an essential component of PKI deployments and will need to support and implement the notion of organizational role playing. Integration of service management middleware solutions with administrative and enterprise computing systems will provide useful synergies for implementing a seamless information access and delivery architecture.

The actual implementation of the USDB consists of a relational database and application software that is responsible for populating, updating and managing the system. The relational database component is implemented with Oracle and the application software is written in PERL. A design mandate was to implement the application interface to the relational database with the modular DBI::DBD system. An additional constraint was to implement the database using ANSI standard SQL and datatypes. The overall goal was to isolate the application software from the database implementation so as to allow an alternate database to be 'plugged' in as the backing store.

The database implements a series of tables which track users, services and hosts which implement service delivery. The application layer implements the notion of creating a 'binding' which is a tuple relating a user, service and server. The application software provides an implementation of a rules structure which allows a common service such as EMAIL to be bound to different servers based on parameters such as the organizational unit (OU) of the person receiving the service.

The entire system inter-operates with a mainframe computer system which supports the administrative software systems for the university system. A subset of the user's information is exported from this system to the USDB which provides for essentially real-time updates of the middleware data elements.

A WEB based application is provided for user interaction. This application allows a user to apply for additional services and change various characteristics of their service profile. The system also implements an acceptable use quiz which is mandated by the legal department of the university to insure that users understand and can be held responsible to the information ethics policy of the university system.

LDAP Account Management System

The second major component of the USMS is the meta-directory update system which is referred to as LAMS. This system is responsible for propagating directory updates from USDB to the LDAP directory servers. LAMS also provides a command line interface for making changes in the directory across the master and replicate LDAP servers. The following table summarizes the list of operational modes for LAMS:

Action	Description
Add	Insert a DN
Kill	Remove a DN
Delete	Remove an attribute of a DN
Update	Modify a set of attributes for a DN
Modify	Modify attributes of a series of DN's
Query	Lookup and display a DN

The USDB maintains a set of rules which map particular record fields from the relational databases into various attribute elements for each directory object. When the USDB determines that one of the LDAP exported attributes has changed the LAMS system is called to update the directory object. While LAMS operates on the directory objects and their associated attributes, all requests for update and modification services are done via the user's canonical identification IID. The input for requesting changes is made through ASCII files coded in the Lightweight Directory Interchange Format (LDIF).

All changes are propagated from the USDB into the LDAP directory servers in the amount of real-time afforded by the administrative systems that ultimately serve as the authoritative source of user information. Error messages from the update and replication process are posted back to the administrative team via e-mail so that remedial action can be taken to correct errors. The instances of manual intervention is generally quite low. The most common errors arise from incorrect user reference data which typically requires intervention at the USDB level or higher.

The meta-directory update system can be globally disabled so that updates are not propagated to the master and replicate servers. This feature is useful from a system administrations perspective when there is a desire to hold all the directory servers in a known state. The USDB holds the last modification time for an object as well as the last propagation time. After directory updates are re-enabled all changes to the meta-directory information since the last update are propagated into the directory servers.

The USDB also supports the ability to generate a complete LDAP directory load in LDIF format. This file is suitable for building an entirely new LDAP directory server database representing the current state of information under management by the middleware structure. This feature is useful from a disaster recovery perspective as well as for re-synchronizing all directory servers to a known state.

The low cost of the Intel architecture and the OpenLDAP directory server software makes running multiple directory servers economical which in turn yields important benefits from an administrative perspective. The data center implementing this middleware solution uses one primary LDAP server and two replicates. The LDAP connections are mediated through a fourth server running TCP/IP port redirector software. This provides for load-balancing as well as the ability to remove directory servers from the active service rotation. This feature provides an easy mechanism for synchronizing the directory databases while maintaining a high availability profile for directory services.

User Account Management System

The final component of the services management system is responsible for the instantiation of the service on a server and is known by its acronym UAMS. This subsystem is responsible for serving as the bridge between the USDB and the actual servers providing end-user services.

An important concept in this services oriented solution is the notion of 'binding' a particular service entity to a host. A service entity is most easily understood to be a user with a specific service such as electronic mail. The USDB application layer provides a mechanism where a number of hosts can be defined as candidates for providing a particular service and then placed into a pool. At the time of service creation the binding process selects a candidate server using either a round-robin or weighted averages algorithm and 'binds' the user to that server. The UAMS layer is responsible for taking this binding request from the USDB and carrying out the actions needed to prepare the server to provide the service for the user.

The USDB application layer also supports the notion of so called "host-less" services. These are services which only require KerDAP API services and do not actually require account creation on the server. Two examples of such services are USENET news service and authenticated WEB proxy service. In both cases the *kerdauth* binary is used to provide simple authentication and authorization services via the external authenticator mechanism provided by INND and the Squid proxy server. The only action needed to instantiate the service is the presence of the user in the LDAP directory with an appropriate services tag and a Kerberos principal for authentication.

The UAMS system is designed around the paradigm of queueing requests for user services. Queued requests are stored in separate spooling areas, one for each server. The system provides for locking and arbitration of access to the queue. Requests for service instantiation are stored until the queue is "processed" either by an administrator or by an automated mechanism. Processing of the requests can be carried out either on a per host basis or globally for all hosts which have entries in their request queue. The process of "running" a queue involves reading each request and sending that request via an authenticated and encrypted session to the target host. The results of the requests (success or failure) are appended to the service request and are used to replace the request queue. This mechanism provides for historical accounting of the success or failure of the requests. The UAMS system provides tools for an administrator to review both pending and processed requests.

The actual syntax of the service request are very simplistic. The following tables contain the input for a series of pending requests and the resulting output:

```
imap1.domain:Bull_Dozer:EMAIL:create
imsp1.domain:Bull_Dozer:CONFIG:create
kdc1.domain:Bull_Dozer:KERBEROS:create
imap1.domain:Back_Hoe:EMAIL:delete
imap2.domain:Road_Grader:EMAIL:modify:quota
```

```
imap1.domain:Bull_Dozer:EMAIL:create:OK
imap1.domain:Bull_Dozer:CONFIG:create:OK
imap1.domain:Bull_Dozer:KERBEROS:create:FAILED
imap1.domain:Back_Hoe:EMAIL:delete:OK
imap2.domain:Road_Grader:EMAIL:modify:quota:OK
```

The actual instantiation of the services is carried out by UAMS clients which are installed on the target hosts. The entire UAMS system is implemented with Bourne Shell scripts and relies on the notion of function inheritance to implement the actual service classes.

The basic UAMS client implements a library of common functionality including user account creation via the *useradd* utility. Each service category is implemented by sourcing in a module which has the option of either replacing existing functions with modified versions or using the functionality provided by the base library. This inheritance mechanism also provides the ability to import functionality from host and domain specific service modules. This system allows modules to be written which support enterprises with the need for tailoring services on the basis of organizational unit needs. While the UAMS client itself is implemented in Bourne shell the actual service instantiation modules can include any tools and/or languages available on the target platform. Pre- and post-processing hooks are also implemented to handle any setup or departure requirements needed for a sequence of service management requests.

Within this middleware solution the LDAP directory is defined as the official API for obtaining any and all information relevant to the implementation and delivery of services. This requirement enables the simplistic UAMS syntax. The UAMS client library provides a number of convenience functions for interrogating the LDAP directory and returning data attributes needed to implement the service instantiation process. This requirement has two implications: first of all, LAMS and UAMS invocations must be synchronized to insure that service creation occurs after the meta-directory system has populated or updated the LDAP directory servers. The second implication is that this requirement imparts an additional degree of security since an intruder would need to compromise the directory servers in order to effectively coerce the system into creating invalid or modified accounts.

KerDAP Enabled Applications

The primary goal of this middleware project was to implement authenticated and differentially authorized services to a wide variety of systems. The following services are currently implemented using the KerDAP API and under management of the USMS:

1. Generic host logins.
2. USENET news reading.
3. TACACS terminal server access.
4. WEB (Squid) proxy services.
5. IMAP email.

6. IMSP remote configuration management.
7. FTP file services.
8. Shared Message Block (SMB) file services.
9. WEB forms and applications.
10. User and system administration and management tools.

In almost all cases only minimal alterations to the sources were needed to implement the triad of IAA.

LDAP directory services were also heavily leveraged as a component of the clustering and high-availability strategies implemented in the data center. The most notable example of this is the use of IMAP and IMSP redirection systems. LDAP support was added to the open-source *perdition*[4] software as an alternative "database" for determining which server implements the IMAP message store and IMSP accounts for a user with email services. The mail destination attribute, which serves as the source for IMAP and IMSP redirections, also provides the basis for mail routing through the LDAP-enabled sendmail hubs which handle incoming mail for all organizational units serviced by this solution.

Future Initiatives

KerDAP is considered to be in a developmental phase of implementation. The USMS has demonstrated its ability to decrease administrative costs and to more precisely control the delivery of services to the user community. Experience with this initial implementation has led to plans for a number of important modifications to the middleware architecture.

First and perhaps foremost are plans for the implementation of an authorization server. Just as USDB, LAMS and UAMS provide the three components of IAA for the management system, the authorization server will couple with LDAP and Kerberos to complete the triad of IAA for KerDAP-enabled applications. The KerDAP library will be extended with a set of functions which will allow each service request to query an authorization server to determine whether or not access to the service is authorized at a particular instant in time. In this scheme the LDAP server assists the authorization process by supplying a unique service token for each service which has been bound to a service entity. This token is passed to the authorization server which than authorizes access to the service based on a set of rules which are specified generically for the service and more specifically for a particular service entity.

[4]http://perdition.sourceforge.net/

It is anticipated that this authorization server will allow the centralized management of IP access controls and other administrative functionality. Most importantly it will provide a mechanism for establishing finite service lifetimes and implementing the notion of organizational role playing for service entities within the enterprise structure. It is anticipated that this authorization server will play an important role either as a replacement or delivery mechanism for X.509v3 attribute certificates.

A second important area of anticipated development is in the merging of shared message block file (SMB) services with middleware services. Integration of the Samba file server with the middleware solution is anticipated to make this file sharing protocol more competitive with the management and administrative advantages of currently popular commercial alternatives. Initial work beyond simple authentication and authorization is focusing on allowing share configuration and access information to be obtained from the LDAP directory rather than host specific configuration files. A second and longer term project is to implement the notion of a Samba fanout server which would automatically redirect SMB connection requests based on user identification and the name of the requested file share.

Another important area of development is the delivery of KerDAP IAA services through the Pluggable Authentication Module (PAM) system. The PAM system is currently seeing widespread use throughout the Open-Source community. The goal of these efforts is to minimize the amount of modification needed at the source level of the service delivery applications.

Current trends of inter-operability in the network directory arena are being closely monitored as well. There are a number of efforts focusing on the use of XML and its derivatives to support the propagation of information from meta-directory systems into server directory databases. Of note is industry led work on the Directory Services Markup Language (DSML). If standards efforts prevail the goal would be to implement the functionality of LAMS via DSML which would provide this middleware solution with a mechanism for propagating and replicating directory information into commercial as well as open-source directory server solutions.

Conclusions

The incredible success of the INTERNET, the increasing demand for mobile computing, and resultant business model paradigm shifts will demand middleware based IT architectures. The choice of a middleware solution will affect all other applications in the enterprise. Application and server systems will need to participate in and integrate with the middleware. Failure to do so will preclude their use within the framework of enterprise computing.

The Open Source and Linux communities must recognize this and respond with an open, standards-based solution to insure the continued viability of their service delivery platform. The success of the open source model to date indicates that it works and that, if an open middleware solution is developed, the INTERNET community will embrace it.

The success of KerDAP and USMS suggests that the notion of a viable middleware strategy based on an open system architecture and open source software is realistic. The authors are pleased with the interest in their solution and invite interested parties to contact them via email at ker_DAP@ndsu.nodak.edu. Copies of this paper are available via the web at http://www.ndsu.nodak.edu/servergroup/kerDAP/.

Acknowledgements

Any significant achievement is invariably a success due to individual contributions through a team effort. USMS and KerDAP is no exception to this rule.

The authors would like to first of all thank Ms. Bonnie Neas, Director of ITS at NDSU, for creating an environment which nourishes and encourages this type of effort in a research/production environment.

Special acknowledgements go to Mr. Dick Jacobson and Mr. Dale Summers. The development of the USDB and its integration with the administrative parent systems is the result of their programming and organizational talents. Success of any middleware initiative is dependent on access to individuals such as these that have a thorough knowledge of an organization, its operation and the policy implications of implementing effective middleware strategies. Their efforts were aided by Mr. Mick Pytlik and his administrative services staff which worked closely with them in identifying successful integration strategies.

Finally a special thanks to the tireless legions of Open-Source programmers. Their efforts toward the common, collective good provided the tools and the sandbox without which this project would not have been possible.