Solaris ONC

Network Information Service Plus (NIS+)

A White Paper

Portions of this paper were previously published in the proceedings of Sun User Group, United Kingdom (SUG UK), 1991.

Table of Contents

Network Information Service Plus (NIS+)

Chuck McManis, Saqib Jang

Executive Summary

The enormous growth in network computing over the past few years presents significant challenges for distributed system administrators, end users, and application developers. Distributed networks have become larger, typically consisting of interconnected subnetworks spanning multiple sites. Management of these networks, with tasks such as addition, relocation, and removal of network resources, including hosts, applications, and printers, has taken on added complexity. The growth in size of distributed systems presents complex requirements for applications and end users to transparently access resources across a network.

To a large extent, such challenges can be addressed through the deployment of a robust, high-performance, network-accessible repository of distributed system resources commonly called an enterprise naming service.

This paper provides an architectural overview of SunSoft's™ Network Information Service Plus (NIS+), an enterprise naming service for heterogeneous distributed systems.

Introduction

NIS+ replaces ONC™ Network Information Service (NIS) and is designed to address the management and resource location requirements for heterogeneous distributed systems of the 90s. It is a repository of user-friendly names and attributes of network resources such as hosts, applications, users, and mailboxes. Clients of NIS+, both applications and users, can efficiently look up information on network resources and access these resources in a location-independent way. NIS+ also plays a critical role in the efficient operation and administration of distributed systems by acting as a central point for addition, removal, or relocation of resources.

The NIS+ naming service was designed to address the following goals:

- To prevent unauthorized access to network resources and act as the platform for distributed system security.
- To scale effectively from very small to very large networks, consisting of tens of thousands of systems.
- To provide the ability to easily administer from very small to large networks, spanning multiple sites.
- To provide for autonomous administration of subnetworks.
- To provide highly consistent information.

NIS+ Overview

NIS+ works with the ONC distributed computing platform, which is an integral part of the Solaris™ environment. Solaris 2.0 is comprised of SunOS™ 5.0, enhanced ONC, NIS+, OpenWindows™ V3, DeskSet™ V3, and OPEN LOOK®. A network computing standard that enables Solaris-based systems to connect to any other proprietary or standard system, ONC permits users to access information throughout the network and make use of all the computing power in the user organization, regardless of location. ONC is supported on all major hardware platforms, from PCs to mainframes. With an installed base of over 1.3 million systems, ONC is unrivalled as the industry standard for distributed computing in heterogeneous networks.

NIS+ is a client/server application built atop the ONC Transport-Independent Remote Procedure Call (TI-RPC) interface. RPC applications are also clients of the NIS+ name service. Since the RPC client and server components of an application may be located on arbitrary machines in a large network, RPC clients use the name service to locate and bind to RPC servers in a flexible and high-performance way.

NIS+ Enhancements

NIS+ includes enhancements which fall into three general categories: the structure of the global namespace[1], the structure of data within maps, and the authentication and authorization models associated with the namespace and the data that it contains.

1. Refer to Glossary of Terms at the end of this paper for definition of *namespace* and other terms.

First, NIS+ includes support for hierarchical domain names, which is a better model for a namespace of many smaller domains. This model has been very successfully used by the Domain Name System that is currently deployed on the Internet and brings other benefits to the system as well. The most important benefit is that hierarchical domains provide a structure upon which to hang a distributed authority mechanism. Other advantages of hierarchical names are that given some basic information, there are well-known mechanisms for locating nodes within a tree, and a mechanism for generating globally unique names that can be based on using the domain name as part of the name.

Secondly, NIS+ includes a new database model, which consists of two parts. The first part is a record containing the schema for the database that is stored in the namespace. The second part is the database itself, which is managed by the same server that serves the portion of the namespace where the schema record was found. To keep the databases simple and the representation of the schema manageable, only two properties of the database were chosen to be part of the schema. These properties are the number of columns in a database, and an indication for each column as to whether or not it should be used as one of the indexes for the database records. Additionally, for columns that were determined to be searchable, a flag is present to specify whether or not the case of the characters should be considered when searching the database with this index. Using this model, NIS `dbm` databases are described as two-column databases, with the first column searchable and case sensitive. Figure 1 shows a graphical representation of how these hierarchical domains and databases, called tables, are related in a global namespace.

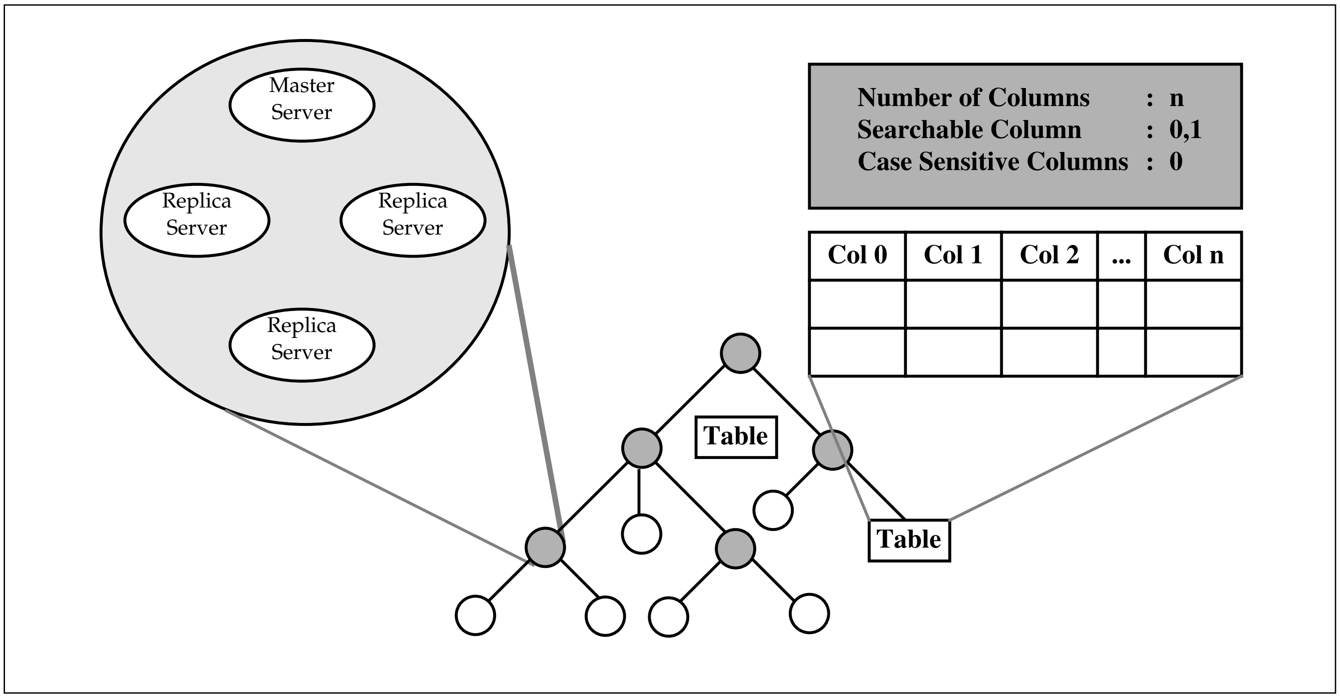

Figure 1 Graphical Representation of the New NIS+ Namespace

Thirdly, there are two classes of changes that have been made to the NIS authentication and authorization model. One is the ability to authenticate access to the service and thus discriminate between accesses that are allowed to members of the community versus other network entities. The other is an authorization model to allow specific rights to be granted or denied based on this authentication. Authentication is provided by mechanisms available to all users of ONC RPC. This allows the server to ensure the identity of principals making requests and clients to ensure the identity of the server answering the request. A second change is the need for the information to have an authorization model associated with it. This is accomplished by using an authorization model similar to the UNIX® file system model. This model specifies that each item in the namespace has a set of access rights associated with it, and that these rights are granted to three broad classes of principals: the owner of the item, a group owner of the item, and all other principals. The specific access rights are different than the traditional read, write, and execute rights that the file system grants owing to the nature of information services. Also, it is useful to include a class of clients which are not authenticated, the “nobody” principal. The authentication and authorization mechanisms incur a finite performance penalty when they are in effect. By allowing the client to specify access rights to unauthenticated principals, we allow them to choose better performance at the cost of weaker security. Finally, access rights to individual rows and columns within the database are also provided. This provides a desirable property, that we can maintain the privacy of particular fields in a record, such as the encrypted password, while still giving unrestricted access to the other fields of database records.

NIS+ Features

The highlights of the NIS+ naming service’s capabilities are:

- Support of hierarchical namespace enables simplified distributed management. The namespace can be divided into domains with each domain managed on an autonomous basis.
- Comprehensive and flexible security mechanisms prevent unauthorized access to network resources. Authentication facilities allow verification of a client’s identity, while authorization is provided to ensure the client has rights to perform desired name service operation.
- Partitioning of name service information provides improved scalability. The unit of partitioning is the directory which contains object information for each namespace domain.
- Replication, on a per-directory basis, allows high availability/reliability. Each domain has a master directory to which all updates are applied and a number of slave replicated directories. Fast consistency between directory replicas enables support of rapidly changing network environments.
- Complete and consistent suite of administrative, namespace and information base functions with programmatic interfaces enables simplified access to and management of (subject to security constraints) name service. Administrative operations can be performed remotely and flexibly. Namespace and information base operations allow applications and users to efficiently access and modify information on network objects.
- Compatibility to allow NIS-based applications and environments to migrate smoothly to NIS+.

Who Benefits?

System Administrators

NIS+ is a powerful tool for simplified administration of heterogeneous distributed systems. As the size of such systems grows and the requirements for decentralized administration emerge, a multidomain hierarchical namespace can be created. Assignment of names and modification of information on network resources within each domain can be decentralized. Further, the hierarchy can correspond to organizational (with each domain representing a functional group, for example, Engineering or Marketing) or logical hierarchies, allowing administrators to use intuitive schemes for organizing their namespace.

Comprehensive security schemes benefit administrators by ensuring that the name service is protected from unauthorized access. This is critical since the name service functions as the primary means for administrators to add, remove, or modify network resources. The security functionality is flexible as administrative groups can ensure that name service information under their control is protected from access and modification by principals outside of that organization.

Partitioning allows administrators to continue using NIS+ as the platform for distributed management as networks grow in size. As multiple domains are created, the overall namespace can have unbounded growth, yet the size of an individual directory remains within bounds. Replication of individual directories, on a master/slave basis, allows coping with computer and communication link failures. In addition, NIS+ provides for fast transfer of updates from master to slave servers, allowing authorized administrators to rapidly add, remove, or relocate network resources.

NIS+ also includes a set of functions for flexible and easy administration of the name service itself. This includes functions for starting and stopping NIS+ servers, replicating and partitioning operations, and setting security levels. NIS+ functions support a corresponding Application Programming Interface (API) designed to allow NIS+ to be the platform for next-generation administrative applications from SunSoft and Independent Software Vendors (ISVs).

Application Developers

Application developers can use the NIS+ API in a number of ways. First, applications can use the API to lookup information on network resources. The SunOS 5.0 system uses NIS+ as the repository for storage of information on hosts, passwords, and users for administrative purposes. Applications can use the NIS+ API to access this information in a high-performance fashion. Second, NIS+ can be used as a secure repository of network-accessible application-specific data. Improved consistency and the read/write capability within NIS+ API enable the service to be used to store and modify application-specific information. For example, the OpenWindows V3 Calendar Manager uses NIS+ to store group schedule information that authorized group members can access and modify to schedule meetings.

Finally, new administrative applications can be developed that run atop the API and take advantage of its simplicity and consistency in providing access to network resources. NIS+ will serve as the platform for future administrative applications from SunSoft. NIS+ also provides full compatibility with the NIS (i.e., `yp_xxx`) programming interface to allow ease of application migration.

End Users

NIS+ also provides significant advantages for end users. The security functionality within NIS+ enables users to trust network communications and to protect sensitive information from unauthorized access. Users' productivity increases as applications transparently locate resources by accessing the central data storage facility with NIS+. The simplicity and completeness of the NIS+ interface enables users to take on a greater proportion of administrative tasks, which is particularly advantageous where administration resources can't keep pace with constantly growing distributed systems.

NIS+ Architecture

NIS+ provides two types of services to clients. The first is a *name service* that maps names, such as domain names, to their respective servers. The second is a *directory service* where the desired information itself is returned, rather than a pointer to it, such as the UNIX password record.

NIS+ Naming Model

The naming model used by NIS+ is a graph structured as a singly rooted tree. Within this graph each vertex represents one NIS+ *object,* each of which may have several children associated with it but only one parent. There are six types of NIS+ objects defined: *directory, table*, *group*, *link*, *entry,* and *private*. Directory objects identify a database of NIS+ objects. Objects within that database are represented as children of the directory object. A directory object and all of its children is an NIS+ *domain.* An NIS+ directory that is a child of another NIS+ directory is a *subdomain*, and all domains below the root directory are the NIS+ *namespace.*

An NIS+ object name consists of several labels, each separated from the next by a dot (".") character. The rightmost label is closest to the root of the namespace. Labels that contain the dot character are quoted. These names are shown graphically in Figure 2. Names that end with the dot character are said to be *fully qualified*, whereas names which do not are said to be *partially qualified*. Names that identify objects in the namespace are called NIS+ *regular names*.

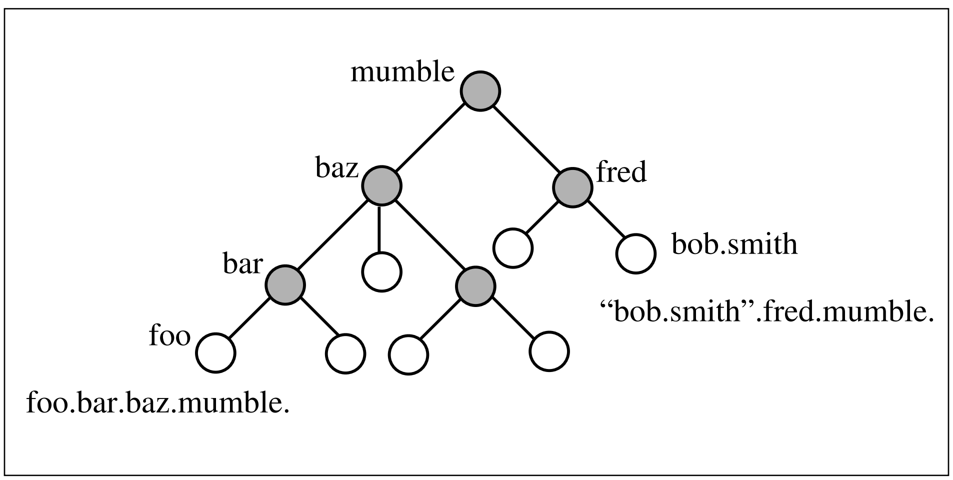

Figure 2 Construction of a Multipart NIS+ Name

NIS+ Database Model

Table objects in the NIS+ namespace identify databases called *tables*. These databases are called tables because the model used is that of a columnar table. The object contains the schema for the database it identifies. This schema specifies the number of columns in the table and identifies which columns can be searched with an NIS+ query.

Rows within an NIS+ table are identified by a compound name syntax called *indexed names*. These names contain a search criterion and a regular name. The regular name portion identifies the NIS+ table to search. The search criterion identifies the rows of interest by specifying what value one or more searchable columns must contain to satisfy the search. The search criterion consists of an open bracket "[" followed by zero or more attributes of the form *column_name* **=** *column_value* followed by a close bracket character "]". Attributes in the search criterion are separated by commas, and the search criterion and the name of the table to apply it to are also separated by commas. An example of these compound names is shown in Figure 3. In the first example, the name **[manager=susan],employees.widget.com.** selects only the fourth row which satisfies this search. In the second example, the name **[manager=george],employees.widget.com.** selects the first three rows which all satisfy the search criterion. Finally the name **[manager=george, name=bob],employees.widget.com.** would return only the first row, because only that row satisfies the complete criterion. The null search criterion, *[ ]*, selects all of the rows.

Table name: employees.widget.com

name	manager	department	division
bob	george	engineering	electronics
mary	george	engineering	electronics
jane	george	engineering	electronics
sam	susan	sales	electronics

[manager=susan], employees.widget.com.

[manager=george],employees.widget.com

Figure 3 Indexed Names Selecting Entries From a Table

The set of names that NIS+ will accept can be defined by the grammar shown in Figure 4. This grammar defines the terminal characters dot (.), comma (,), open bracket ([), close bracket (]), and equals (=). Generally, it is inadvisable to put terminal characters in the strings that make up the NIS+ names. However, should that be necessary, a quoting mechanism based on the double quote (“) character is provided. All characters between two double quote characters are not scanned. The double quote may itself be quoted by placing two double quote characters adjacent to each other. These two characters are treated as a single instance of the double quote character.

```
NAME                ::= <REGULAR NAME> | <INDEXED_NAME>
REGULAR_NAME        ::= . | <STRING> . | <STRING> . <REGULAR_NAME>
INDEXED_NAME        ::= <SEARCH_CRITERION> , <REGULAR_NAME>
SEARCH_CRITERION    ::= [ <ATTRIBUTE-LIST> ]
ATTRIBUTE_LIST      ::= <ATTRIBUTE> | <ATTRIBUTE> , <ATTRIBUTE -LIST>
ATTRIBUTE           ::= <STRING> = <STRING>
STRING              ::= see note
```

Note: ISO Latin 1 character set, the initial character in the string may not be either ‘@’ or ‘-’; embedded terminal characters must be protected by double quotes.

Figure 4 NIS+ Name Grammar

In addition to the schema, the table object holds other properties associated with the database. These include the tables “type,” a concatenation path, and a separator character that clients printing entries from the table use to separate the data from each column in the output. All of these properties are shown graphically in Figure 5.

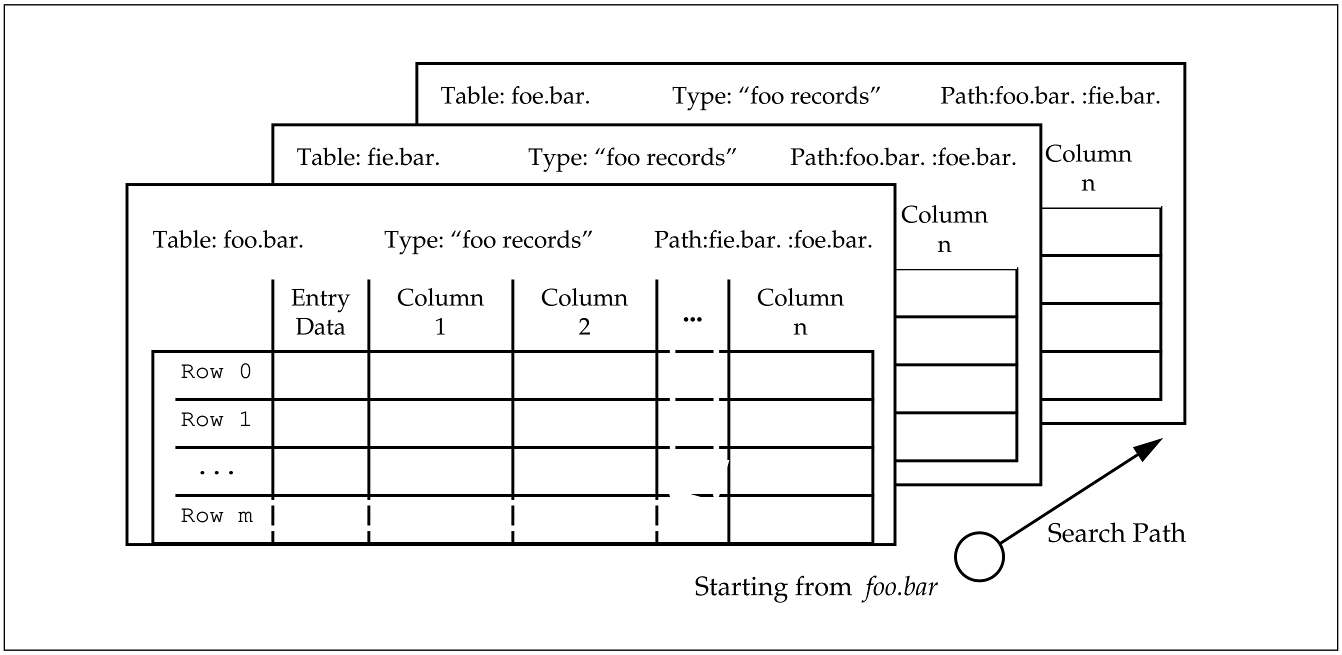

Figure 5 Graphical Representation of an NIS+ Table

The type property is a string that the server uses to prevent adding to a table entries that are not compatible with the other entries in the table. This string is compared to the type string of the entry being added and, if they do not match, the update is rejected. The server also compares the number of columns in the entry being added to the number of columns in the table, as well as their type (binary or text) to minimize mistakes that could corrupt the database.

The concatenation path, or search path, is a string containing the list of tables to search if the search criterion in the indexed name doesn't match any entries in the table. This search path allows all of the tables identified within it to appear to the client as one large table. When entries from several tables in the path all match the same search criteria, the first ones found are the ones returned to the client.

In addition to the column data, each row in a table contains some specific information about itself in the column labelled *Entry Data*. This entry data consists of the name of the owner for this row, a group owner, a time-to-live value for the row, and a set of access rights for this row. The entry data is combined with the data from the columns when a row is returned in the form of an entry object. This data is present for all rows, but is neither searchable nor explicitly identified as a column in the table object.

NIS+ Object Model

NIS+ objects are implemented as variant records. All objects share a common set of properties which include an owner name, group owner name, access rights, object identifier, and time-to-live values. Additionally, each type of object has a variant part that contains specific information for that type of object. Directory objects contain the names, addresses, and authentication information for machines that serve a domain. Table objects contain the schema for NIS+ tables. Group objects contain a list of principals that are members of that group. Link objects contain the name of another object, and entry objects contain data from an NIS+ table. A diagram of these objects is shown in Figure 6. The shaded portion represents properties that are unique to a particular type of object. The top portion show properties that are common to all objects.

DIRECTORY	TABLE	GROUP	LINK	ENTRY	
iod name domain owner group access rights time to live					Common Properties
directory name directory type servers(s) master replica(s) ...	table type number of columns column description(s) column 0 column 1	group flags group members(s) member 0 member 0 ...	link name link type	entry's type entry data datum 0 datum 1 ...	Variant Properties

Figure 6 Format of NIS+ Objects

NIS+ Authorization Model

The NIS+ authorization model has four different access rights that can be granted to four classes of NIS+ principal. The four classes of principal are: the *owner* of an object; the *group owner*, which is a set of principals that is specified by a group object; the set of authenticated principals that are known to the NIS service, called collectively the *world*; and the set of unauthenticated principals called collectively the *nobody* principal.

The four access rights that are grantable are read, modify, create, and destroy. Read conveys the right to read the contents of objects, directory objects, and table objects. Modify conveys the right to change the attributes of an object such as the owner, group owner, time-to-live, and so forth. For directory and table objects, create conveys the right to add objects to the namespace controlled by that object, either a domain or a database, depending on the object's type. Destroy conveys the right to remove objects from directories or tables. Domains and tables have additional access rights masks that allow rights to be granted with a finer degree of granularity.

The NIS+ service uses authentication information extracted from the RPC messages it receives to identify the NIS+ principal making the request. ONC/RPC messages have attached to them authentication information in the form of an authentication *flavor*. Each flavor of authentication contains a unique principal name. The service uses this name and the authentication type to search an NIS+ table named *cred.org_dir.*<*domain*>. In that table, there is an entry that contains the flavor-specific name and the NIS+ principal name. This NIS+ defined name is used in the authorization mechanism rather than the authentication flavor-specific name. This mapping allows the NIS+ service to accept different flavors of authentication information and continue to work correctly.

The authorization model for tables is somewhat more sophisticated than that of the namespace. Access rights in tables can be granted on a per table, per entry, and per column basis. Each level supersedes all subsequent levels. Thus, if read access is allowed for the table, it is allowed for every row and every column in the table. If read access is not allowed for the table but is allowed for a particular row, then it is allowed for all columns in that row. Finally, if read access is not allowed for the table and not allowed for the row but is granted for a particular column, then that column's value is returned. This is shown graphically in Figure 7. Each mask serves to narrow the authorization to a finer grain. In the figure, the entire table is readable only by the owner of the table.

However, all of Row 1 is readable by the group owner of that row. Finally, Column 1 of Row 1 is readable by all NIS+ principals. The modify right may be similarly granted.

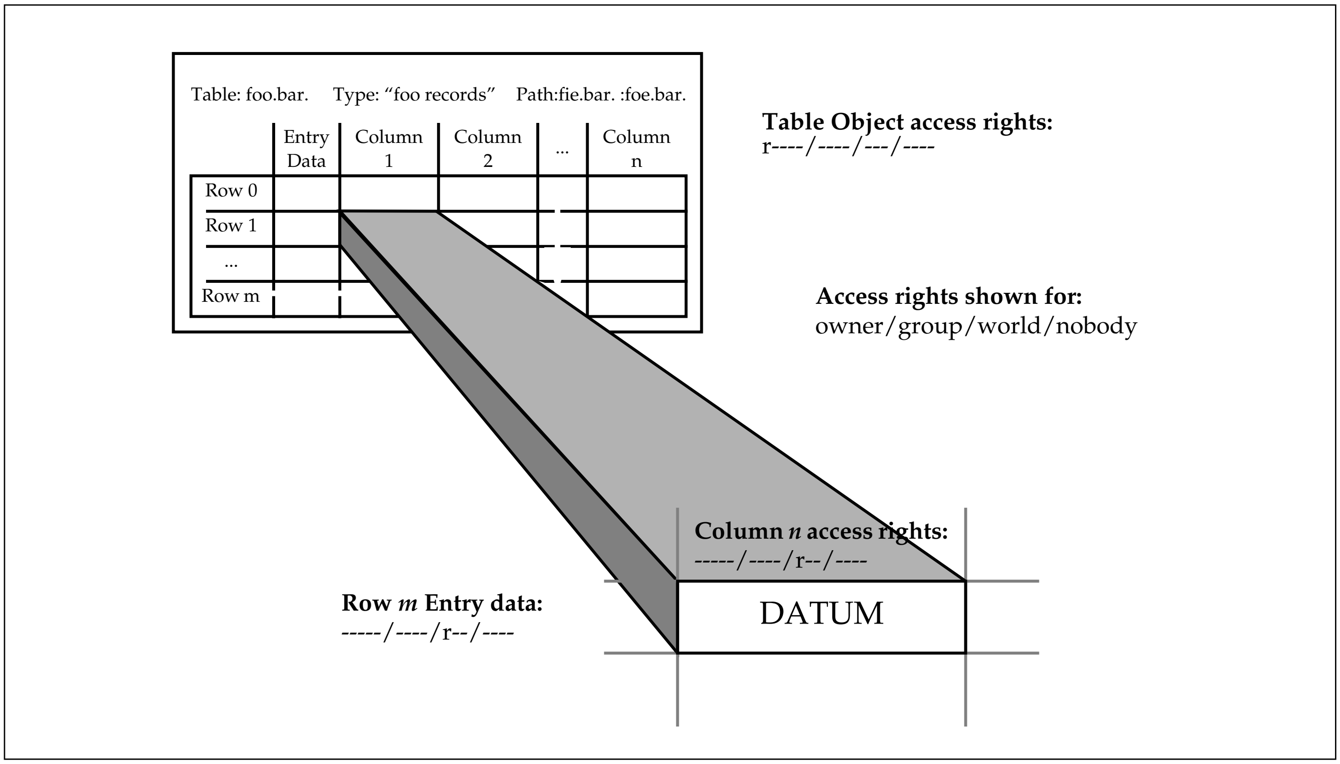

Figure 7 Access Rights for Tables Showing Narrowing of Access Rights

The service enforces read access by censoring the information that the requesting principal is not allowed to read. When one or more, but not all, columns of an entry are protected, the protected columns have the sensitive data replaced by the “*NP*” string. Entries that are completely protected are simply removed from the list of entries that are returned. If no access to the table is allowed, an error of NIS_PERMISSION is returned.

NIS+ Replication Model

Like the existing NIS service, NIS+ domains are replicated using a master and slave model. However, the replication is done on a per domain basis with each internal node in the hierarchy having its own master and replicas. Further, all changes to either a domain or a table within that domain are logged on an event basis, and it is these change events that are propagated to replicas rather than to complete databases.

Implementation of NIS+ Service

The components of the NIS+ design which implement these models are shown in Figure 8. The highlighted components are currently part of the SunOS system and are not considered to be part of this design. In Figure 8, each box represents a set of interfaces. Dependencies are not strictly linear since the location mechanism may be required to lookup directory objects within the NIS+ namespace, and this causes it to make use of the NIS+ client library interfaces.

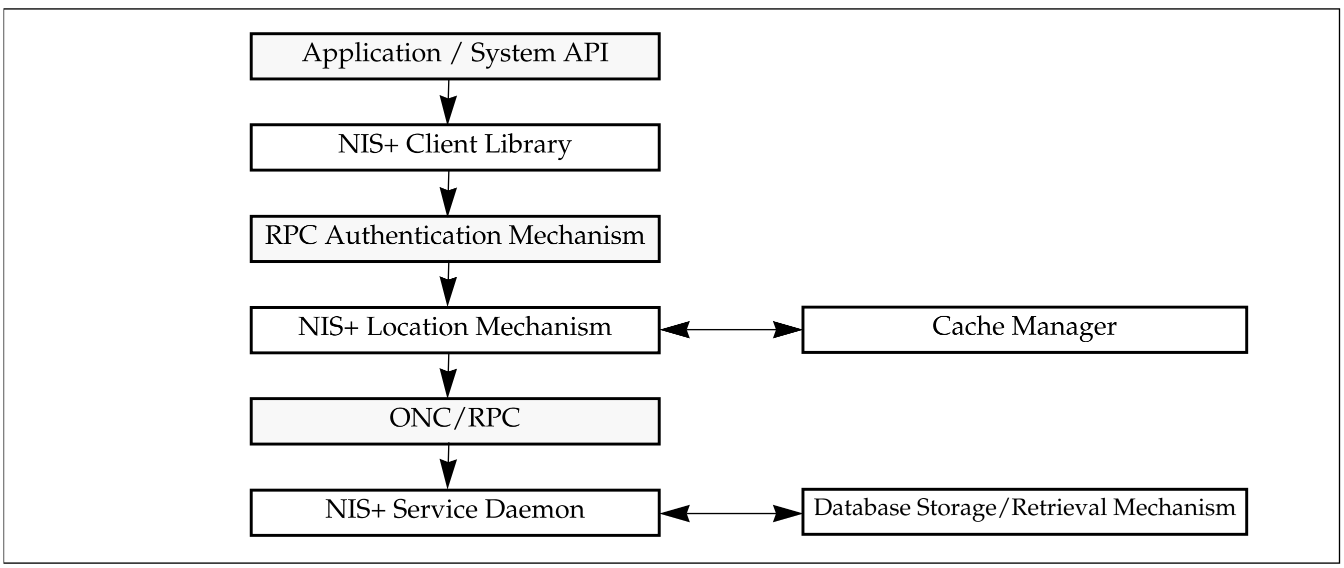

Figure 8 The Components of the NIS+ Service

NIS+ Client Library

The NIS+ client library interface provides the application interface to the NIS+ service. Generally, applications have their own naming interface that meets their particular requirements. For example the operating system defines a naming interface `gethostbyname( )`, which takes a host name and returns a structure containing addressing information about that machine. This additional level of indirection allows different naming services to be used in different implementations of the interface. The NIS+ client library is designed to provide a set of capabilities upon which custom naming interfaces may be built.

The two primary operations that are provided are the lookup and list operations. The lookup operation locates objects in the namespace and the list operation searches tables. Both provide the service binding and name resolution functions. Additionally, they provide services specific to their operation.

The lookup routine implements the symbolic link facility. When a lookup request is made and the object returned is a link, the function checks to see if a flag was passed that specified links were to be followed. If it was, the function restarts the lookup using the name stored within the link.

Similarly, the list function implements the table search paths. When a search of an NIS+ table returns no entries, and the user has specified that the table's path is to be followed, this function begins searching each table in the path until a match is found or the path is exhausted.

The client library also provides the service location mechanism. All NIS+ clients have a file in their file system that contains the name, address, and authentication information for at least one trusted machine. Requests for service go to that machine or machines first when the client boots up. When the desired name is not within the local domain, the machine serving the domain is queried for a server that does serve the desired domain. This query will be answered by either the name of a server who serves that domain or a server that is closer to that domain. The location function will cache this information with a local daemon named the NIS+ *cache manager*. If the returned information is not the desired domain, the domain returned is queried for the name of a machine serving the desired domain. In this way, all nodes in the namespace can be located.

The client library also provides the interfaces to add, remove, or modify NIS+ objects and entries in NIS+ tables. Various routines for manipulating NIS+ names, routines for returning the NIS+ names associated with the current process, and various object handling functions, round out the package.

NIS+ Cache Manager

The NIS+ location mechanism is predicated on the fact that a machine that serves a domain is itself a client of a domain above the one it serves. When a location request comes from a client for a domain, the service checks to see if it serves the requested domain. If it does not serve the domain, then it checks to see if the desired domain is below the domain it serves in the namespace. If the desired domain is below its domain, it locates the directory object in its domain that is an ancestor of the desired domain and returns that to the client. If the desired domain is located above the server’s domain, it returns the directory object for its domain, which is always above the one it serves. Eventually this mechanism will find its way to the root of the namespace and begin to work its way down again.

Once the desired directory object is found, the service returns it to the client. Prior to returning the directory object, it “signs” it with a cryptographic checksum, using a key that only a root process on the client machine will know. When the client library gets the directory object, it passes it to the cache manager which verifies the signature and adds it to its shared memory cache of directories. Future client requests will find the directory object in the cache manager’s cache and avoid the time-consuming location step.

NIS+ Service Daemon

Once the directory object for a domain is located, the client library will attempt to bind to each of the servers within it until a binding is successful. Update operations will attempt to bind only to the master server. Given a binding, the client library then makes an RPC call to the NIS+ service. The service uses a virtual memory database to store its information. It receives the request, checks the access rights, and then responds with the requested information.

The database used is linked to the service at runtime using the shared library facilities of the SunOS system. The use of shared libraries allows the database to be replaced with specialized versions for different applications.

An In-depth Look at an NIS+ Operation

A diagram showing the execution of an NIS+ lookup is provided in Figure 9. An NIS+ operation begins with a call to an application or system library interface (1) that uses NIS+ in its implementation. Most often, this operation will be a search of an NIS+ table. In our example, the `gethostbyname` function call searches the *hosts* table. The `gethostbyname` function then calls the NIS+ client library in Step 2. The client library looks in the cache manager's shared memory location cache for the name and address of an appropriate machine. If there are no appropriate bindings, the client library will locate one using Steps 2.1 and 2.2 to call the `nis_finddirectory` function in search of a server. When a machine is located, the client library issues an RPC request, Step 3, to talk to the service. The service consults its database and returns the results in Step 4. The client library takes the results and passes them up to the application interface in Step 5, and the `gethostbyname` call converts the data in the entry object into a `struct hostent` for the calling program.

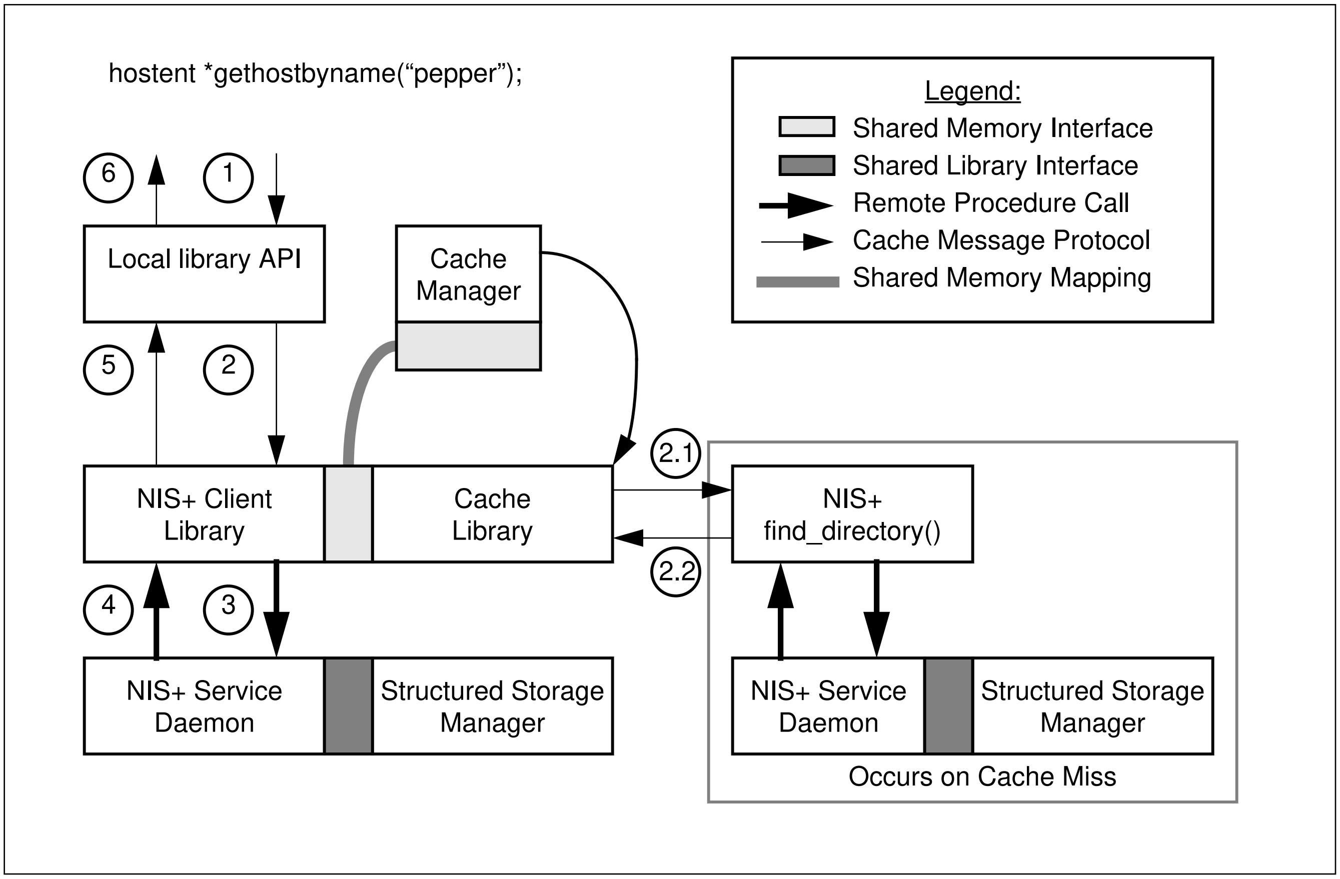

Figure 9 Execution of an NIS+ Lookup

Glossary of Terms

Throughout this document, terms are used that have specific meanings. These terms are collected here for easy reference.

API

The Application Programming Interface, or API, consists of a series of C routines that give application programs access to the naming service.

Binding

An NIS+ name, protocol information and valid credential for the local NIS+ server. This information is sufficient to contact the server and resolve names administered by it.

Credential

A piece of information that identifies the user or host providing it. If the information is encrypted in such a way that only its true owner, as opposed to an imposter, could encrypt it, then it is a Secure Credential.

Datum

The `dbm` structure that NIS uses: it consists of an opaque key and an opaque data array.

`dbm`

A collection of C language routines that implement a very simple database. The database is divided into *maps*.

DNS

The Domain Name Service: the Internet name service that is defined in RFC 1034 and RFC 1035.

Domain

A namespace administered by a single authority. The domain contains one or more servers that have its directory.

Global Namespace

The space of all possible names that NIS+ can name. This space starts at dot (.) and goes down.

Information Base

A generic term for a collection of information that is named.

Internet

A network whose participants are machines and users from more than one organization. Additionally, those organizations do not share any common management at any level.

Leaf Node

A leaf node or name has no other names below it in the namespace tree.

Namespace

A namespace is a collection of names administered by a single authority. This space can be composed of several branches of the Organizational Namespace.

Object

An NIS+ object is the basic unit in the namespace. This data structure consists of an administrative part that identifies ownership, access rights, and so forth, and a data part that contains the value of the object.

Object Value

The data portion of an NIS+ object.

Principal

Clients of the naming service, such as users and hosts, having credentials.

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

For more information, call 1 800 227-9227.

Printed in USA 9/91 91027-0 1.5K