The Group Administration Shell and the GASH Network Computing Environment

Jonathan Abbey – The University of Texas at Austin

ABSTRACT

Managing large scale UNIX networks so that users can use resources on multiple systems is traditionally performed using NIS, NFS, and DNS. Proper use of these tools requires exacting maintenance of multiple databases under a single point of control. This is acceptable for a few dozen machines, but with hundreds of machines spread across several administrative groups, some mechanism is needed to share administrative control over the master administrative files.

The Computer Science Division of the Applied Research Laboratories, The University of Texas at Austin (ARL:UT) has developed the Group Administration Shell (GASH) system, which gives different administrators permission to administer subsets of the laboratory NIS and DNS files. In addition to distributing administrative control in a secure fashion, GASH makes a large number of system administration tasks very simple, permitting technically untrained personnel to safely perform complex system administration tasks. The Network Computing Environment (NCE) built around GASH and standard UNIX tools such as NIS, DNS, and automounter provides a flexible and reliable way of completely abstracting the user's environment from the physical configuration of the network.

Problem Statement

The proliferation of computer networking has outstripped the sophistication of the common mechanisms used to allow UNIX systems to share resources. It is not uncommon today for hundreds of UNIX systems to be linked together in a local area network. NIS and NFS [1] can allow those systems to share administrative information and file systems, but with such a large number of systems, centralized administration of the NIS databases becomes impractical. Local work-group administrators must either have root privileges on the NIS master server, with all the security risks which that entails, or they must submit their NIS change requests to a centralized administration group that is responsible for the maintenance of the NIS databases. Neither of these solutions is acceptable in an environment in which security and response time are critical.

There are other tasks that must be performed by a centralized administrative body which present an unacceptable bottleneck in administering systems in a shared network environment. For example, when a new system is added to the network environment, one or more IP addresses must be assigned. Detailed records of all system network information must be kept for network troubleshooting. There must be a simple and reliable way of keeping track of which systems and users have permission to access what NFS filesystems and network resources. Network filesystems must be mounted using standard naming conventions throughout the network computing environment. E-Mail delivery must be coordinated to systems that cannot access a centralized UNIX mail spool filesystem.

All of these factors make large-scale systems administration extremely challenging. Even if it were acceptable to have single-point centralized administration of the NIS and DNS [2] databases, keeping the databases correct and consistent is nontrivial and time consuming. If a large number of part-time account administrators are to be able to safely modify the NIS databases, complex tasks such as renaming users or systems must be made simple. Name and numeric id conflicts within and between databases must be prevented.

The GASH Network Computing Environment Solution

ARL:UT has gone a long way towards solving the problem of large scale distributed heterogeneous systems administration. By combining a set of carefully conceived administration conventions with a complex control and administration shell, NIS and DNS administration tasks can be divided among many part-time administrators. GASH performs a large number of complex tasks automatically, and constrains administrators to editing a restricted subset of the NIS and DNS maps, according to their administration privileges.

GASH controls a single, comprehensive NIS domain. All UNIX systems in the GASH NIS domain are part of the GASH NCE. In addition to the basic user and group NIS maps, GASH maintains NIS maps to control the delivery of e-mail within the laboratory, individual and group e-mail aliases, automounter [3], and netgroups.

GASH maintains DNS tables as well as NIS maps, and is therefore involved in the administration of all systems in the laboratory, whether or not they participate in the NCE NIS regime. GASH keeps Ethernet and Internet records for all systems in the laboratory. All IP addresses in the laboratory are assigned by GASH, and all changes to IP addresses resulting from the physical movement of systems within the laboratory are handled by GASH.

The GASH NCE has been in conceptual development at ARL:UT for close to three years, and 40,000 lines of C code have been written over the past 18 months to implement the GASH control program. In operation for the last 9 months, GASH currently contains system information for 984 network devices (2011 total network interfaces) and NIS information for 614 users in 166 account groups. More than 45 individuals have been granted GASH administration privileges. These individuals range in expertise from seasoned systems managers to administrative and technical staff with little UNIX experience.

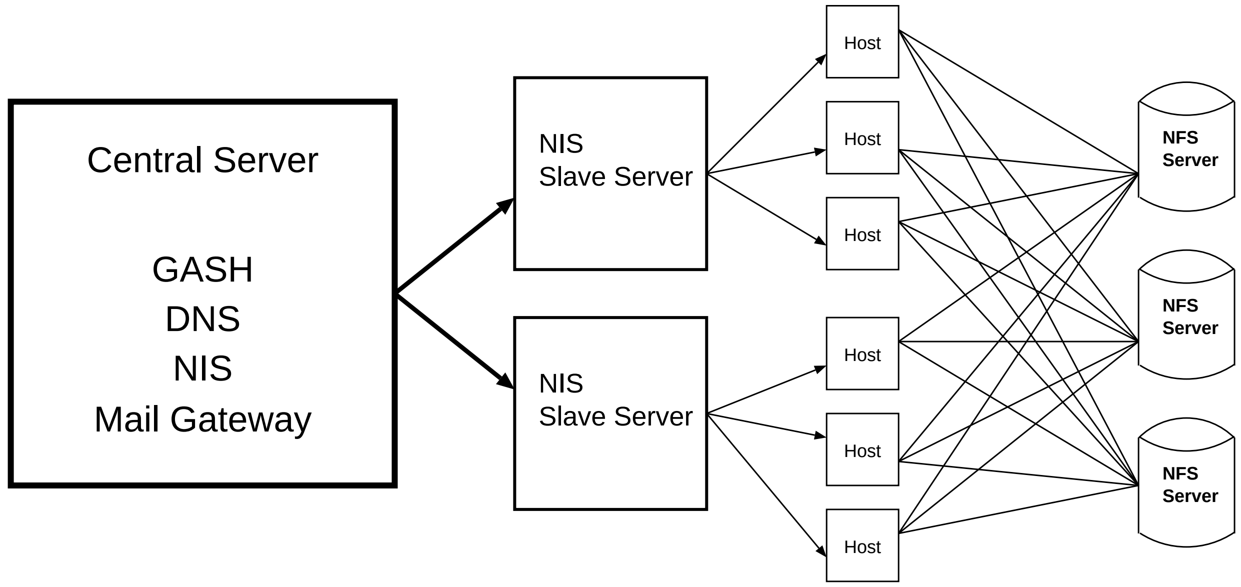

Figure 1: GASH NCE Physical Structure

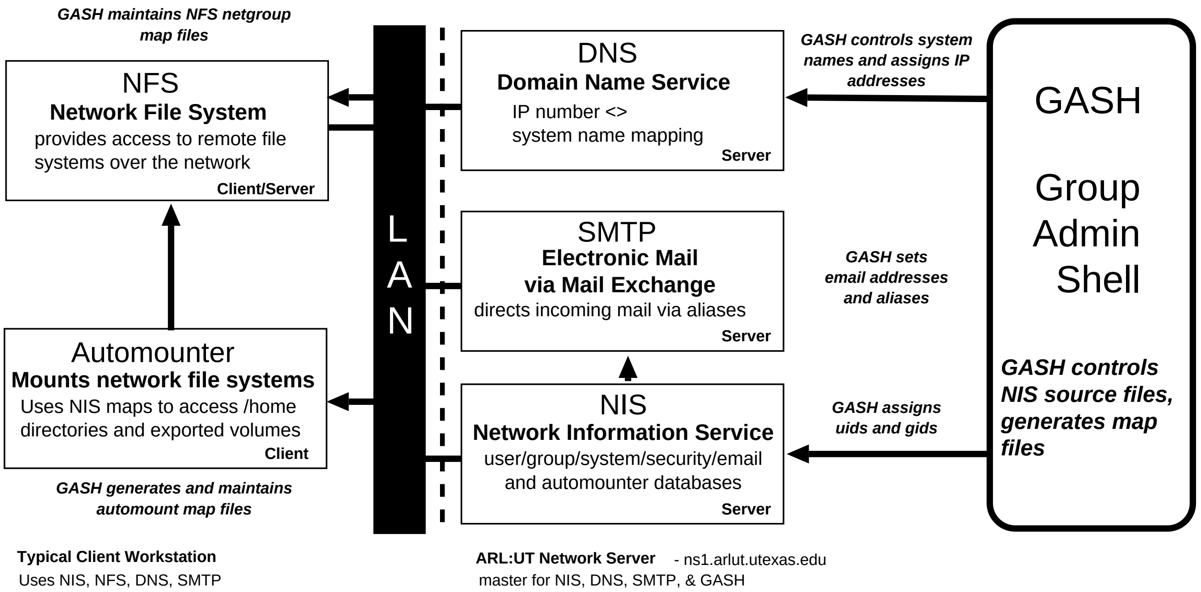

Figure 2: GASH NCE Logical Structure

NCE Network Structure

The NCE consists of a central server that maintains NIS and DNS tables, multiple NIS and DNS slave servers, and a number of NCE hosts. Administrators run the GASH program on the central server to modify the NIS and DNS tables. Changes to the NIS and DNS tables are propagated to the slave servers, which in turn feed information to the NCE hosts. For reliability and performance, NIS slave servers are maintained on each subnet of our network.

The NCE supports replicated filesystem definitions for read-only NFS resources. Critical volumes (software archives and the like) may be redundantly served. NCE clients will access whichever file server is most convenient to them, according to the sophistication of their NFS client software.

Figure 1 illustrates the basic physical structure of the GASH Network Computing Environment just described. **Figure 2** is an overall logical diagram of the NCE, detailing the ways in which GASH controls the NCE and its interrelated parts.

GASH NCE Filesystem Conventions

The adoption of a number of lab-wide conventions has been critical to fully realize the benefits of the GASH NCE. The GASH NCE defines a uniform naming method for NFS filesystems that hides the actual physical location of the filesystem. All NCE access to NFS service goes through a level of naming indirection that makes it possible to physically reconfigure NFS servers without disrupting access to logical volumes. Automounter is used for all NFS client access in order to insulate NCE clients from the vagaries of filesystem availability and to provide naming indirection. Automounter is configured on NCE clients using two primary automounter NIS maps, **auto.vol** and **auto.home.default**. GASH uses these NIS maps to centrally manage home directories and logical volume names for all NCE clients. A system without the ability to use automounter may directly participate in the NCE, but it must be locally configured to work with the NCE conventions. Any NCE client that does not use automounter will need to be manually reconfigured whenever any changes are made to network file servers.

NCE clients never refer to network filesystems by their machine name and path, but rather by a logical volume name assigned by GASH. At any time, a GASH administrator with appropriate privileges can change the association between a volume name and NFS filesystem. Such changes can be made to the network file servers without disrupting users that depend on access to particular volumes. All filesystems are automounted in `/v` using the **auto.vol** NIS automounter control map. For example, the volume `csdhome` may always be accessed through `/v/csdhome` by any NCE client, regardless of the physical location of the volume's filesystem.

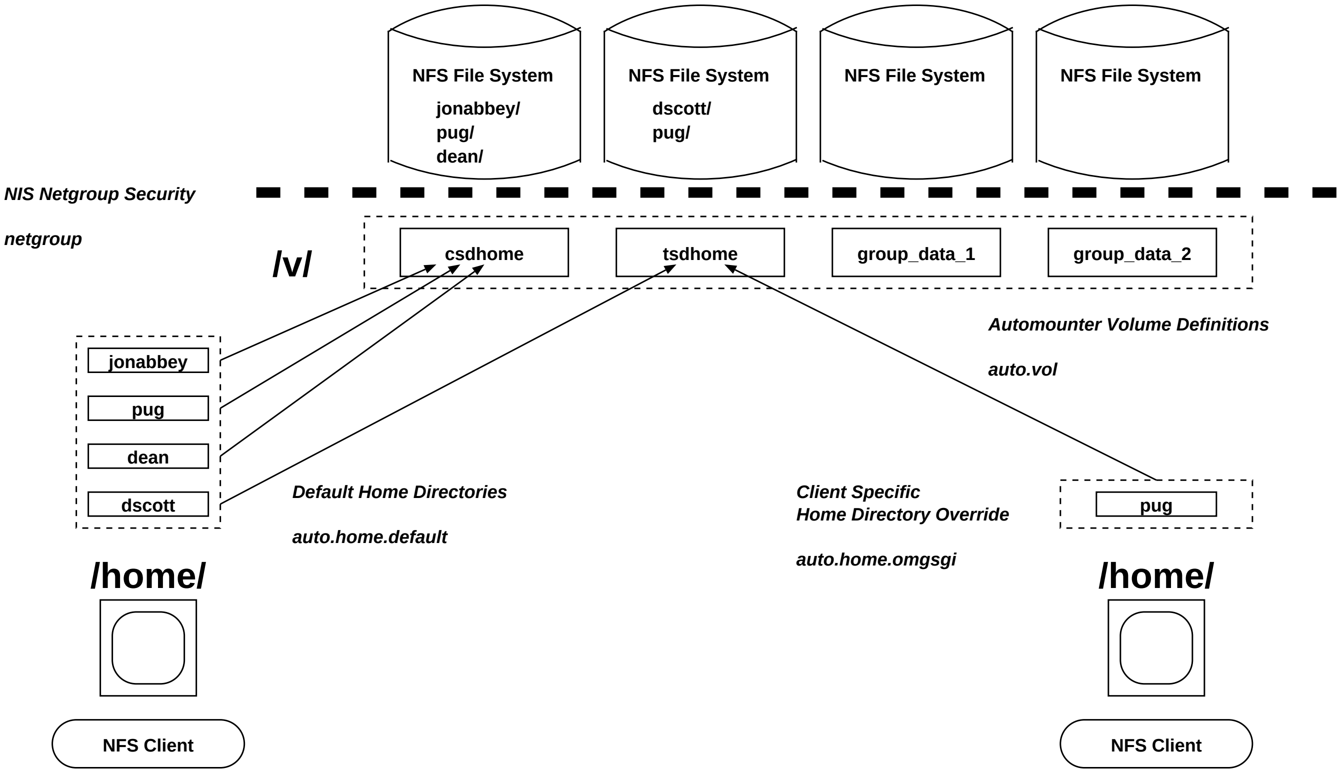

Figure 3: GASH Filesystem Management

All users' home directories are accessed through `/home/`*`username`* on all machines participating in the NCE. The GASH editor maintains automounter control files that map a user's home directory to a defined GASH volume. GASH administrators may create special control files to define alternate home directories for specific users on particular machines. A user may always access files on a particular volume by using the `/v` mechanism. If a user's default home directory is located on the volume `csdhome`, that user may always access that default home directory through `/v/csdhome/`*`username`*, even if he is logged into a machine using an automounter control file that specifies a different volume for his account's home directory.

NIS netgroups are used to control access to NFS filesystems. Each NFS file server in the NCE is configured so that permission to access exported filesystems is specified in terms of GASH managed NIS system netgroups. By using NIS netgroups in this fashion along with the automounter NIS maps, it becomes possible to perform all network access administration through GASH once an NFS server is properly configured. **Figure 3** illustrates the way in which the GASH controlled automounter and NIS netgroups control the NFS network environment.

GASH Client and NIS Conventions

To fully work with the GASH NCE, all host systems must be configured to use the **passwd**, **group**, and **mail.aliases** NIS maps in addition to the automounter maps. In order to use the **passwd** and **group** NIS maps, the `/etc/passwd` and `/etc/group` files must be modified to specify NIS chaining with the '+' syntax [1, p.32]. The **mail.aliases** NIS map will be used automatically by a properly configured sendmail [4]. Users' `.cshrc` and `.login` files should be configured to work with the NCE environment, and the `/etc/shells` file should include all shells defined in the GASH **shell_paths** control file.

All GASH NCE uid's and gid's are at or above a fixed boundary (1000 at ARL:UT). All local `/etc/passwd` and `/etc/group` entries should have uid's and gid's below the boundary to avoid conflicts with NCE accounts.

In the GASH NCE, all UNIX account groups are six characters in length. The first three characters are a unique GASH administrator code indicating the administrator that created the group. The last three characters are unique to the group, and generally are mnemonic in nature. Netgroups and auxiliary automounter control files contain similar administrator codes. These administrator codes allow GASH to restrict editing access to these objects and provide an easy way to determine quickly which administrators are associated with a group, netgroup, or automounter control file.

Other NCE Components

One of the goals of the NCE is to provide users with a consistent working environment throughout the laboratory. Supporting a single home directory throughout the NCE goes a long way towards this goal, but does not solve some important problems.

The first problem is that users' `.cshrc` and `.login` files are usually not portable across different kinds of UNIX systems. This problem was dealt with by splitting the traditional home `.login` and `.cshrc` files into two parts, one system dependent and one system independent. System dependent initialization is performed by `/etc/.login` and `/etc/.cshrc`. These files set the default PATH, MANPATH, and LD_LIBRARY_PATH environment variables, as well as any other system specific environment or shell variables. An NCE user's home `.login` and `.cshrc` files reads the system dependent `/etc` files on startup. The home `.login` and `.cshrc` files contain sections that allow customizations targeted to specific types of systems, along with a generic customization section.

The second problem is that the difficulties of managing application software are magnified in a large network environment. The location of software should be transparent to users, and software packages should be easy to update or remove. To handle these issues, a sophisticated software management suite called **opt_depot** was developed. **opt_depot** was inspired by Carnegie Mellon's Depot software [5], and allows easy maintenance of a single central package archive for freely distributable and site licensed software. System administrators in the laboratory can configure their workstations to access these packages from the central package archive or from a local copy without any recompiling or re-configuring. The central package archive, if used, is accessed using the volume indirection conventions of the GASH NCE, giving us the flexibility to relocate or replicate the archive as needed. The **opt_depot** software integrates all packages into the `/opt` directory structure, regardless of package location.

In addition to handling NIS and DNS, the central server acts as the laboratory's primary mail server and gateway. The server exports a mail spool volume, `/v/mail`, that is mounted by NCE hosts. Any mail entering or leaving the laboratory is processed by the server in accordance with the NIS **mail.aliases** map maintained by GASH. Mail leaving the laboratory can have its headers rewritten to reflect a canonical laboratory e-mail address for the sending user. Mail entering the laboratory may be redirected to machines that do not take advantage of the central mail server's mail spool.

GASH Command Tool

Much of the work in developing the GASH NCE was spent in the construction of the GASH command shell. GASH is a single-user, text-based interactive editing shell and DNS/NIS manager that provides task oriented (rather than file oriented) editing and browsing facilities. The GASH command shell is very configurable, and has many convenience and security features.

GASH Command Shell Commands

GASH contains 32 primary commands, grouped into eight sets of four commands. Each set includes a command to create, modify, inactivate, and list the appropriate data type. For instance, the user set includes `cua`, `mua`, `iua`, and `lua` to create, modify, inactivate, and list users. Sets exist for **`users`**, **`groups`**, **`netgroups`**, **`systems`**, **`e-mail aliases`**, **`volumes`**, **`automounter control maps`**, and **`administrators`**.

GASH commands are designed to allow the administrator to make changes to everything that relates to the subject at hand. While editing a user account, for instance, a GASH administrator may cause changes to be made to several groups, netgroups, and e-mail aliases. This tight binding between the GASH databases will be discussed in depth later in the paper. In general, the GASH administrator does not need to know anything more than what needs to be accomplished. GASH commands accept defaults, and the user can terminate any command all desired changes have been made.

The **create** commands will typically do everything necessary to get an object of the appropriate type ready for use. For instance, creating a user account with the `cua` command will prompt the administrator for a list of groups and netgroups to which the user account should be added, the volume on which to place the user's home directory, and the user's default e-mail address.

The **list** commands typically only show data items that the administrator has permission to edit. The exception to this is the list volume command (`lvm`), which shows all volumes defined in the NCE. Inactivated items may or may not be visible, depending on the particular command. The **list** commands accept command line parameters, which the **list** commands will try to intelligently match against the portion of each database entry that the user might wish to search.

The **inactivate** commands may remove an object immediately (systems, e-mail aliases, administrators, volumes), or may cause an object to be removed at a later time (users, groups, netgroups). Objects to be removed at a later time are immediately made unusable, and entries describing the actions to be taken to ultimately remove the objects are made in the GASH **pending_actions** file.

This delayed removal is necessary for many kinds of database entries. Users and groups must not be removed immediately, because doing so will cause confusion in the NCE file systems. By delaying final removal, GASH provides a consistent environment for local system administrators to do clean-up, and prevents new users or groups from being granted the newly available uid or gid too soon.

When an object is removed from the GASH databases, all references to it are cleanly removed. An object may generally not be inactivated or removed if this will cause any GASH database to become inconsistent. Administrators must have editing authority in order to inactivate an object. See the sections on the individual GASH databases and the **pending_actions** control file below for more information on object inactivation.

The **modify** commands allow administrators with editing authority to change a logical object. As with the **create** commands, modifying a single logical object may cause changes to all databases that reference the object. Modifying a user with the `mua` command, for instance, will prompt the user to specify groups and netgroups from which the user should be added or deleted as part of the editing process. Whenever GASH prompts for a modification to a list of objects, the administrator may enter a list of objects to be added or removed from the list. If an object that the administrator entered was on the list, it is removed. If the object was not on the list, it is added.

GASH Security and Administrator Privileges

The GASH command shell includes several features to preserve NCE security. GASH uses its own password file to control access to the command shell. The machine that GASH is running on only allows logins from users who have been granted GASH privileges. GASH has a ten minute time out feature, and will re-prompt for the administrator's password if it is left idle for more than five minutes. GASH will not display an administrator's privileges unless the administrator's password is correctly re-entered. All incorrect password attempts are logged, and mail is sent to the system administrators overseeing GASH.

GASH supports a **supergash** account that can take action on behalf of any group administrator. The password to this account is guarded at least as closely as the system root password. At ARL:UT, considerably fewer people know the GASH **supergash** password than know the Computer Center's root password.

Each GASH administrator is assigned a range of uid's and gid's for which the administrator has editing authority. In addition to uid and gid range permissions, each administrator has a unique three letter identification code and a set of privilege masks to control access to groups, netgroups, automounter control files, and e-mail aliases not attached to a user account. The use of privilege masks allows a group of administrators to have overlapping privileges. An administrator might have an identification code of `omj`; another administrator, with an identification code of `omg`, might have an automounter control file mask of `om*`, allowing full access to automounter control files generated by either administrator.

GASH administrators do not necessarily need to have access to each of the eight GASH command sets. Permission to use each of the sets can be individually granted or denied to administrators.

Logging / E-Mail System

GASH has a sophisticated system for tracking modifications to the NCE databases. Each individual database modification may result in an event being generated. Each event is logged to a **gash_log** file, and may be logged to the system **syslog** file.

Many events result in e-mail being sent to the GASH administrator or administrators responsible, as well as any users or system managers who should be notified of the event. The **mail_control** control file contains a list of event codes, along with a list of administrators or users that should be notified whenever an event of that type occurs.

Pending Actions / Nightly Processing

Many system administration actions require an initial action followed by a clean-up action some time later. Removing a user, for instance, requires that the user's password and login shell be changed, but the user's GASH records need to be maintained until such time as all files associated with the user's uid can be removed. GASH has a facility for writing notes to itself for later action. These notes are written into the **pending_actions** control file. Whenever GASH makes a note in the **pending_actions** file of an action to be taken in the future, it also makes notes to send mail to the users affected by the action. Users will typically receive mail once a week until the action occurs, reminding them of the action to be performed.

Each night, GASH runs in an automated mode that performs any scheduled actions from the **pending_actions** file and does cross-database consistency checks. GASH compiles a nightly report that includes any inconsistencies detected, along with a list of any pending actions performed. This report is sent to a list of administrators specified in the **mail_control** file.

GASH Databases and Control Files

GASH manages a minimum of seven administration databases. These databases are listed in **Table 1**. The NIS and DNS tables generated from the GASH databases are listed in **Table 2**.

Filename	Purpose
user_info	Users
aliases_info	E-Mail
auto.vol	Volumes
auto.home.*	Home Directories
group_info	Groups
hosts_info	Systems
netgroup	NIS Netgroups

Table 1: GASH Databases

Map	GASH Source File
NIS auto.vol	auto.vol
NIS auto.home.*	auto.home.*
NIS ethers	hosts_info
NIS group	group_info
NIS hosts	hosts_info
NIS mail.aliases	aliases_info
NIS netgroup	netgroup
NIS passwd	user_info
DNS named tables	hosts_info

Table 2: NIS and DNS Tables Generated from GASH Files

The GASH databases are as similar as possible to the standard `/etc` files and NIS source files upon which they are modeled. The **netgroup** database is actually identical to the standard NIS **netgroup** source file. They are all line-oriented text files, and can generally be edited by hand with little trouble. The exception to this is the **hosts_info** file, which is quite complex, and formatted quite unlike any standard NIS or DNS source file.

GASH includes several perl scripts that are run by a makefile at the end of each GASH session in which databases have been changed. These scripts take the several GASH databases and transform them into the standard files expected by NIS and DNS. The makefile then takes whatever action (pushing NIS maps, updating DNS named tables) is appropriate to propagate the new information into the NCE.

One of the primary responsibilities of the GASH program (and the primary source of its complexity) is to maintain constraints and relationships within and among these databases. These constraints may be as simple as making sure that a user's account name in the **user_info** database is unique and less than 8 characters, or as complex as making sure that all relevant e-mail addresses in the **aliases_info** database are updated when a system in the **hosts_info** database is renamed. One of the best ways to get a good idea for how GASH works is to review in detail the databases that it manages. The next section will discuss the GASH databases and the relationships that GASH maintains between them.

Figure 4, below, shows the seven primary GASH databases along with the GASH administrator control file. The lines connecting the databases indicate cross-database references. The arrows indicate the direction in which automatic changes can propagate through the databases. Not all connections are shown.

The user_info GASH Database

The **user_info** database is one of the primary GASH databases. It contains an entry for each user granted a UNIX account in the NCE. Each entry is similar to that found in the traditional `/etc/passwd` file, with user account name, password, user id, default group id, a structured GCOS field providing user identification, the user's home directory (`/home/`*`username`*), and the user's login shell. In addition, the **user_info** database contains the social security number for each user. This social security number is used by ARL:UT as a unique identifier to insure that NCE accounts are granted only to valid personnel, and that users are not given duplicate accounts.

Within the **user_info** database, GASH guarantees that each user's name and id will be unique and will conform to the length and character set restrictions of standard UNIX. GASH assigns a free uid in the creating administrator's uid range when a new user is created. GASH always sets the home directory field to `/home/`*`username`*, in accordance with NCE convention. The automounter control maps (**auto.home.default** and any specialized control maps) determine where `/home/`*`username`* is ultimately located.

There are many relationships that GASH maintains between the **user_info** database and the other GASH databases. A user's default group id is guaranteed to refer to a valid group in the **group_info** database containing the user. The user is guaranteed to have an entry in the **aliases_info** database, and an entry in the **auto.home.default** database, in addition to at least one entry in the **group_info** database. In addition, the user's name is guaranteed not to conflict with any alias names registered in the **aliases_info** database, and is guaranteed to be changed in all databases managed by GASH if the user is renamed.

A user account may be inactivated, in which case the user's password will be set to '*', and their shell will be set to `/bin/false`. When a user is inactivated, a record is placed in the **pending_actions** file which will cause the user to be removed at a future time. Inactivated users have their e-mail address in the **aliases_info** database retargeted to the administrator who inactivated them. Inactivated users may be reactivated by a GASH administrator using the `mua` command. A reactivated user account must have a new password, login shell, and (optionally) e-mail address assigned.

An active user account may instead be transferred to the control of a GASH administrator. When this happens, the user's GCOS information and social security number are changed to that of the GASH administrator. If the user account is registered as a GASH administrator account in the **admin_info** control file, inactivating or transferring the user will remove GASH administration privileges from the user account.

The aliases_info GASH Database

The **aliases_info** database is the master e-mail addressing database for the NCE, associating simple names with fully qualified e-mail addresses. These names may correspond to users in the **user_info** database, external users (individuals with an NCE address who do not have UNIX accounts and may in fact be receiving mail outside of the laboratory), or group e-mail accounts. Each of these three types of aliases have specific constraints associated with them that are maintained by GASH. Modification

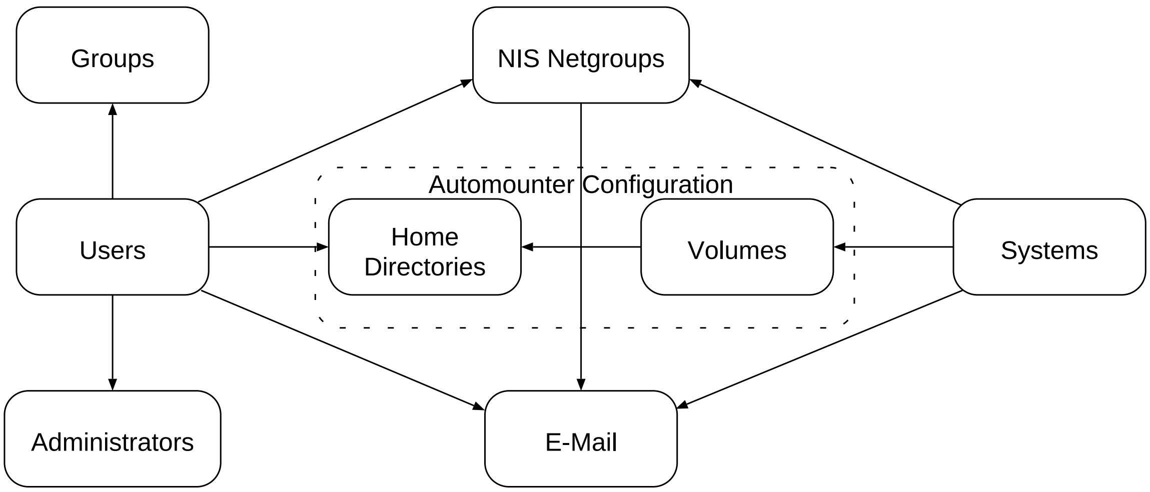

Figure 4: Logical Relationships Between GASH Databases.

privileges for NCE user aliases are controlled by uid range. Modification privileges for external user and group aliases are controlled by a three letter code indicating the identity of the administrator that created the alias.

User and external user alias records are identical, with the exception that external user alias records have an additional field to hold the administrator code. Each user and external user record consists of the canonical user name (identical to the NCE user name in the case of NCE user aliases), a set of alternate names, and the e-mail address or addresses to which the user's e-mail should be forwarded. One of the alternate names may be chosen to be the ''signature'' name, which will be placed in the ''From:'' header on all e-mail from the user which leaves the NCE.

In the following example (**Figure 5**), NCE user broccol has jonabbey as an alternate name. Mail addressed to this user in care of the NCE (e.g., jonabbey@arlut. utexas.edu or broccol@arlut. utexas.edu) will be redirected to broccol@ arlut.utexas.edu. Mail messages from this user passing through the NCE mail server to destinations outside of the NCE may have their headers re-written, when using an appropriate version of sendmail [6] on the NCE mail gateway. If this happens, the user's account name is replaced with the signature alias (the first entry in the alternate names field, in this case jonabbey). Mail from broccol@ csdsun1.arlut.utexas.edu will be received as having come from jonabbey@arlut.utexas.edu.

```
broccol:jonabbey,
    broccol:broccol@arlut.utexas.edu
```

Figure 5: Example User Alias Record

Group e-mail alias records are essentially external user alias records without the alternate names field. The GASH program draws a distinction mainly for administrator convenience. All e-mail alias records may have multiple e-mail targets and may be used to implement e-mail groups. All e-mail alias records may have simple names in their target field, provided the simple names are registered elsewhere in the **aliases_info** database. GASH converts any alternate (non-canonical user or external user) names referenced in a target field to the canonical name to simplify tracking target alias changes.

As with all GASH databases, a number of sophisticated constraints and relationships are maintained that affect the **aliases_info** database. Within the **aliases_info** database, all names and alternate names are guaranteed to be unique, and to cohere with the contents of the **user_info** database. By default, whenever a user is renamed, the user's old name becomes an alternate name for the user's new name, preventing the possibility of dropped e-mail during the transition.

Simple name e-mail targets are guaranteed to exist elsewhere in the **aliases_info** database. E-Mail alias loops may exist, but GASH does check to make sure that an alias does not target itself. If an e-mail alias is removed, all aliases that targeted that alias will have the alias removed from their target fields. Likewise, any time a user is renamed, all aliases that target that user will be updated with the new name.

Absolute e-mail targets (i.e., e-mail targets that include an '@') that refer to machines in the **hosts_info** database will be kept synchronized with any major changes (such as system renaming or deletion) in the **hosts_info** database.

GASH automatically maintains group e-mail aliases for all user netgroups in the **netgroup** database for administrative convenience. These aliases may not be edited directly; any changes to the corresponding user netgroup will be reflected in the alias.

The auto.vol GASH Database

The **auto.vol** database contains a list of all NFS volumes defined in the NCE. Each record consists of a volume name and a list of one or more NFS filesystems (host and path) to which the volume name refers. Multiple NFS filesystem entries are typically used to list multiple interfaces on a single machine, but may also be used for replicated filesystems.

Volume names are guaranteed to be unique within the **auto.vol** database. GASH maintains relationships between the **auto.vol** database and the **hosts_info** and **auto.home** databases. Volumes are guaranteed to refer to systems registered in the **hosts_info** database. Systems may not be removed from the **hosts_info** database if there are any volumes that solely reference the system to be removed. An administrator must remove the volume definition before removing the system or systems from the **hosts_info** database. Likewise, a volume may not be removed from the **auto.vol** database if there are any entries in any of the **auto.home** databases that reference the volume to be removed.

In the GASH NCE, the **auto.vol** database is used to directly control the automounted `/v` directory. All volumes in the NCE may be accessed through `/v/`*`volumename`* on all systems with appropriate access privileges.

Volumes may only be created, modified, or destroyed by administrators who are listed as managers of one or more of a volume's component systems.

The auto.home GASH Databases

Just as the **auto.vol** database is used to control the automounted `/v` directory in the NCE, the **auto.home** databases are used to control `/home`. Each **auto.home** database consists of a sequence of entries binding an NCE user with a volume from **auto.vol**. Each user may appear only once in each **auto.home** database. Volumes are guaranteed to exist in the **auto.vol** database.

There is one special **auto.home** database, **auto.home.default**. All NCE users are guaranteed to have an entry in the **auto.home.default** database. Under normal conditions, a user will have an entry in **auto.home.default** only, and all machines that the user uses in the NCE will mount `/home/`*`username`* from the volume specified in **auto.home.default**. Occasionally, however, a user will need to have a different home directory on a particular machine or set of machines. In this case, the user will have an entry in an auxiliary **auto.home** database, and the machines in question will be configured to search through the auxiliary database before **auto.home.default**.

Auxiliary **auto.home** databases are given names of the form **auto.home.*identifier***, where ***identifier*** is composed of an administrator code and a unique identifier. Administrators with appropriate administrator codes may add, change, or delete any entry in an auxiliary **auto.home** database. Administrators who do not have administrator code access to an auxiliary **auto.home** database may still add, change, or delete any entry referring to an NCE user under their uid range. All administrators have access to the **auto.home.default** database on this basis.

When the **auto.home** databases are turned into NIS maps, the volume name in each entry is replaced with the current definition of the volume. If any volumes are ever modified, GASH will regenerate all the **auto.home** NIS maps with the new filesystem information. In this way, the definition of volumes may be changed at any time and the change will be reflected almost immediately throughout the NCE.

The group_info GASH Database

The **group_info** database is essentially the standard UNIX `/etc/groups` file with a pair of extra fields added. The first additional field is an ARL:UT accounting code. The second additional field contains a description of the project or administrative group associated with the account group. The password field is not used for active groups in the NCE. GASH assigns a password of `ZzZz` to active groups, and `**Inactivated**` to inactivated groups.

Within the **group_info** database, GASH guarantees that group names are unique and conform to the length and character set restrictions of standard UNIX. All gid's are guaranteed unique; new groups are created with a free gid from the creating administrator's gid range. Between GASH databases, GASH guarantees that all users referenced in the **group_info** database actually exist, and will not allow an administrator to remove a user from his default group without either changing the user's default group or inactivating the user.

Editing access for groups is restricted both by administrator gid range and by administrator control code. Administrators with permission to edit a particular user account may perform actions that will result in changes to all groups referencing that user, regardless of individual group permissions.

The hosts_info GASH Database

The **hosts_info** database keeps a complete list of all systems connected to the laboratory network using IP addressing. For each system, the **hosts_info** database records the system's main name, any specific interface names or CNAME aliases, the room that the machine is located in, a list of authorized system managers for the machine, and all Internet and (optionally) Ethernet addresses associated with the machine. In addition, information is kept on the type of system, the manufacturer and model, and the operating system being run.

GASH guarantees that all system and host names are unique, and that all IP addresses are unique. GASH uses the **networks_by_room** and the **internet_assign** control files to assign IP addresses that correspond to the type of system and to the networks that connect to the locale (room) containing the system. GASH guarantees that a system will not be removed if a volume is defined on that system. If a system is removed or renamed, appropriate action is taken in the **aliases_info** database, the **netgroup**s database, and the **auto.vol** database. If a system manager's user account is removed or renamed, this is likewise reflected in the **hosts_info** database.

GASH supports moving a system from one room to another, and will carry over all Internet addresses that are valid in the new location. GASH supports multiple interfaces on systems, and can add or remove interfaces at any time. GASH has special support for SLIP users; if a system is of type SLIP, the first manager listed in the system manager field will be considered to be the name of the NCE user being granted SLIP privileges. The makefile that runs when GASH exits editing database files will use this information to build SLIP control files.

Permission to modify host entries is restricted to those administrators who are listed in the system manager field of a system. Only those administrators may remove or modify a system, or volumes that depend on such a system. Any valid GASH administrator with system management privileges may add new systems.

The netgroup GASH Database

The **netgroup** database contains NCE-wide definitions for user and system NIS netgroups. Each record contains a netgroup name along with a list of netgroup entries that belong to the netgroup. Each netgroup has a name of the form `<3 character` `code><identifier>{-u|-s}`, with user netgroups ending in `-u` and system netgroups ending in `-s`.

Standard NIS netgroup entries are tripartite, containing a field for a user name, a system name, and an NIS domain name. Groups of these entries, along with references to other netgroups, make up a standard NIS netgroup. Netgroups are used to grant or deny permission to access a resource to a group of entities. In practice, there is no occasion to grant or deny access to a group that includes both users and systems [1, p. 167].

In GASH, netgroups are simplified; system netgroups can only contain system netgroup entries and references to other system netgroups, and user netgroups likewise. A GASH netgroup can thus be thought of as just a list of user names or system names. The NIS domain name field always contains the name of the NCE NIS domain name. This allows NFS servers to verify that client access requests are coming from systems within the NCE.

GASH guarantees that all users and systems referenced in GASH netgroups actually exist in the **user_info** and **hosts_info** databases. Any user or system renaming or deletion results in the appropriate entries in the **netgroup** database being modified or deleted, as appropriate. GASH guarantees as well that any references to other netgroups are valid, and will remove netgroup to netgroup references when a netgroup is removed. GASH maintains group e-mail aliases (without the trailing `-u`) for all user netgroups in the **netgroup** database as an administrative convenience.

GASH restricts access to netgroups to those administrators who have an appropriate netgroup code mask in the **admin_info** control file. Systems and users may always be removed or renamed by an administrator with appropriate permissions, and the corresponding changes will be made throughout the **netgroup** database, regardless of individual netgroup permissions.

Filename	Purpose
admin_info	Administrator Privileges
mail_control	Diagnostic Mail Targets
networks_by_room	Room Connectivity
pending_actions	Future Actions
shell_paths	Default Shell Choices
internet_assignment	IP range control

Table 3: Major GASH Control Files

GASH Control Files

In addition to the primary databases that GASH manages, there are a number of control files that GASH consults as it goes about its job. **Table 3** contains a list of major GASH control files, along with their general purpose.

admin_info

The **admin_info** database contains a list of NCE users who have been granted GASH administration privileges. **Table 4** shows the fields present in the **admin_info** database.

Field	Purpose
user name	NCE user name
password	Administration password
admin code	Unique administrator code
commands	Command set privileges
low uid	Lowest uid in authority range
high uid	Highest uid in authority range
low gid	Lowest gid in authority range
high gid	Highest gid in authority range
group mask	Group name privileges
user netgroup mask	User netgroup privileges
system netgroup mask	System netgroup privileges
email mask	Group, external user alias privileges
automounter mask	Automounter control files privileges
email address	Optional email address

Table 4: admin_info Fields

The first two fields in **Table 4**, user name and password, are self-explanatory. The third field is a unique three letter code that is embedded in any group, automounter database, netgroup, or e-mail alias created by the administrator. The commands field is a bit field granting or denying the administrator permission to execute the various GASH command sets. The next four fields grant the administrator privileges to edit a subset of the user uid and group gid ranges. The next five fields, the mask fields, are masks to match against the previously mentioned embedded administration codes. These masks may be up to three characters in length, including an optional trailing wild card. The last field, the optional e-mail address field, is used for those administrators who want e-mail resulting from their administrative actions to go to an address different from their personal address. This is often useful for automatically notifying a group of administrators of one's actions.

As with the GASH databases, the **admin_info** control file is automatically kept consistent within itself, and with the other GASH files. Renaming a user who is also a GASH administrator will make the appropriate changes in the **admin_info** file. Changes to the **hosts_info** database can effect changes in the **admin_info** e-mail addresses.

mail_control

The **mail_control** file contains a list of event codes that correspond to significant events that can occur during the execution of the GASH control program. Upon such an event, GASH consults the **mail_control** file to construct a list of e-mail addresses to which notification should be sent.

networks_by_room

The **networks_by_room** control file contains a mapping of room numbers to IP networks. It is used by the GASH systems commands to present a choice of valid IP networks when a system is created or modified, or to verify a proper IP address when a system is moved from room to room. This file must contain a record for a room before a system can be placed in that room by GASH.

pending_actions

The **pending_actions** control file contains a record of actions to be taken by GASH at some point in the future, one per line. Each pending action entry contains a time stamp for the action, a type field indicating the kind of action to be performed, an optional field indicating the object that the action is to be performed on, and a list of users to send mail to when the action is performed.

Events in the **pending_actions** file may lead to modifications in one or more GASH databases, or may cause mail to be sent to users or administrators reminding them of an action to occur in the future. Typically, any database modification event will be preceded by a set of mail events, one per week until the time of the event.

The **pending_actions** file is processed every night during automated GASH processing. Any events scheduled to occur are performed and removed from the **pending_actions** file. If an event could not be performed, the record for the action is left in the **pending_actions** file and mail is sent to the system administrators overseeing GASH.

shell_paths

The **shell_paths** control file contains a list of login shell choices to be presented to GASH administrators creating or modifying a user account. Administrators may enter arbitrary shells, but problems may occur with ftp if the shell chosen is not in a client's `/etc/shells` file. internet_assignment

The **internet_assignment** control file contains a list of system types recognized by GASH. This list will be presented to GASH administrators creating or modifying a system. Each system type is associated with an IP subrange corresponding to a portion of an IP class C network or class B subnet. This allows, for instance, all routers to be assigned IP addresses with a host number between 254 and 250. The **internet_assignment** file allows both ascending and descending ranges to be defined.

Limitations of the GASH Control Program

While GASH is extremely functional, it does have several more or less arbitrary limitations and restrictions as currently implemented. GASH will allow users and systems to be renamed, but it will not allow groups, netgroups, or volumes to be renamed. GASH has some support for augmenting automounter home maps with the NIS '+' syntax, but has no support within the editor for removing or re-ordering such augmentation lines. With the exception of the **admin_info** control file, all GASH control files must be set up and maintained by hand.

While it has not yet become a significant problem at ARL:UT, the single-user limitation could prove to be a significant bottleneck in very large installations. Various avenues to making GASH multi-user have been discussed, and in the future we may produce some kind of multi-user GASH.

GASH does not yet have any way of creating users' home directories on NCE volumes. Instead, when a user is added to a home volume, moved from home volume to home volume, or renamed, GASH will send mail to the administrators of the systems involved, describing the actions that need to be performed. Because remote human administrators are still involved at this point, some GASH actions do not have immediate turn-around.

There are many areas of GASH that would benefit from a richer user interface. The GASH commands accept command line parameters, but this facility is not very well developed. The list commands in particular would be well served by command line switches to specify search criteria. Many places in GASH will attempt to prompt the user with a reasonable list of alternatives at a prompt, but most do not. The linear quality of the current GASH interface makes most tasks very straightforward, but a graphical user interface would have significant advantages in a number of these areas. It would be convenient for GASH to allow an administrator to browse the GASH databases in a separate window or session while working with an editing command. It would also be nice to arrange for some kind of scripting language interface for GASH.

GASH NCE Administrative Restrictions

The GASH control program enforces a number of administrative restrictions that are designed to make the divisions of control and responsibility within the NCE immediately apparent. GASH requires that administrator uid and gid ranges be contiguous. This restriction makes transferring users or groups between administrative groups non-trivial, but makes it feasible to allow a group of GASH administrators to share responsibility for a set of users and groups. Group and netgroup names must start with a three letter administration code that can be used to immediately identify the department or group to which the group or netgroup belongs. GASH currently requires that group names be six characters in length.

GASH insists on assigning all uid's, gid's, and IP addresses. This guarantees that the id's and addresses are allocated in such a way as to insure that a previously used id or IP address will not be re-used until all other free id's or IP addresses in the appropriate range have been used. Unfortunately, this means that the initial construction of the GASH databases must largely be done by hand. Fortunately, the database consistency checks in GASH are extremely helpful in establishing a comprehensive and consistent NIS and DNS regime. In general, the lack of automated support for transitioning into the GASH environment is the weakest area of the GASH implementation as it currently stands.

Observations on implementing the GASH NCE at ARL:UT

Before we developed and implemented the GASH NCE, computing resources at ARL:UT were poorly integrated. Several groups within the laboratory had independently developed NIS and DNS domains, and record keeping and account administration were fragmented throughout the laboratory. NFS file sharing was possible only within local groups, and software management effort was being duplicated several times over. Network records were difficult to keep up to date, and local network configuration changes were often not promptly reported to the central network administrators.

Within the groups, the complexity of maintaining the many administrative databases by hand was becoming unmanageable. Even in the ARL:UT Computer Center, user and group records were poorly synchronized between computers.

These problems do not exist today. The GASH NCE has provided significant, concrete advantages for all involved. A common NIS and NFS regime allows much easier access to data across the network than was previously available. The GASH program makes complex system administration tasks trivial, and giving this power to local group administrators makes turn-around time for administrative tasks much faster. Our records are complete and up to date, and remain so with minimal effort. Users can rely on having a common working environment on UNIX machines throughout the laboratory.

The transition to GASH was long and somewhat painful. We had to shuffle users and groups around, moving uid's and gid's into organized blocks. A great deal of data entry and data verification had to be performed. Our experience with the great mass of data entry required led directly to the automated cross-database consistency checks that are now part of GASH. In addition to data entry and conversion, the effort of developing and documenting GASH itself was not trivial. The gains in organizational clarity have been exceptional, and remain so even as the number of users and networked machines continues to increase rapidly.

What About NIS+?

NIS+ [7,8] is a drastically expanded and enhanced product compared to NIS. It provides significantly better security, and more intelligent database update propagation. NIS+ contains a mechanism for splitting NIS domains into hierarchical subdomains, a mechanism aimed at solving the same data-space partitioning problem that GASH was developed to address. We began developing GASH before NIS+ was announced, but it turns out that the mechanisms that NIS+ provides in this area do not adequately address our needs.

The main capability that GASH provides over standard NIS+ is the ability to split control over an NIS domain by uid, gid, and object name. NIS+ only allows domain splitting via subdomains, with different client systems belonging to different subdomains. It is not possible with the standard NIS+ mechanisms to distribute administrative control over a single machine. At ARL:UT, we have a SPARCcenter 2000 that is used by researchers from several different research groups. With GASH, the research groups are allowed to control their own users, groups, and e-mail aliases. With standard NIS+, such control could only be afforded to researchers with their own group of workstations.

Until such time as NIS+ client software is available on all common UNIX platforms, ARL:UT will need to support an NCE that supports standard NIS clients. This means that even when we go into NIS+ with GASH, we will not be able to make use of the hierarchical domain system provided by NIS+. Migration to NIS+ for the enhanced security and intelligent database propagation features is inevitable. NIS+'s hierarchical domains and intelligent update propagation may provide a useful substrate for a multi-user GASH (with one administrator per subdomain concurrently), and might make supporting multiple DNS domains more straightforward.

Anticipated Future Developments

We plan on enhancing our NIS support for e-mail by supporting a **mail.generics** NIS map, that will map NCE user names to their outgoing e-mail signature. We are also ultimately planning to make the overhaul of GASH required to integrate GASH with an NIS+ central server.

We are interested in developing GASH into a more fully automated tool, with RPC protocols for automating remote volume administration. In the more long term, we can see a significant benefit from reworking the basic architecture of GASH around an object oriented database system. A good deal of GASH's code is devoted to maintaining cross-database consistencies that could be handled automatically by an object oriented database system. A graphical user interface or batch / script interface would be significantly easier to implement with such a base architecture.

We currently run code during GASH's nightly processing which generates the CSO e-mail directory for the laboratory. We are interested in doing more work within the ARL:UT environment to connect the GASH databases to our Personnel Office's databases, so that user accounts can be validated against Personnel Office records.

Availability

Preliminary user documentation is available for World Wide Web access from URL http://www.arlut.utexas.edu/csd/gash_docs/gash.html. The GASH software itself is available through this web server. The **opt_depot** software is available for World Wide Web access from URL http://www.arlut.utexas.edu/csd/opt_depot/.

Credits

GASH is the product of long effort on the part of many people at ARL:UT, both within the Computer Science Division and within the GASH user community. GASH could not have happened without the support of the systems administration and user communities within ARL:UT.

The GASH NCE was primarily conceived and designed by Dan Scott. The initial design of the GASH control program was developed by Dan Scott and Dean Kennedy. Dean Kennedy was the original implementor of the GASH control program, and is responsible for much of the program's infrastructure and all of its complex systems support. The author designed and developed the e-mail alias support, implemented large portions of the final code, and acted as a clearing house for bugs and Things That Needed To Be Done. Pug Bainter is the mastermind behind the GASH - NIS interface code and the finer points of GASH's netgroup and system support. Pug is also responsible for having nailed down exactly what an NCE client needs to do to play ball with GASH. Kristi Hennard implemented the initial pending_actions processor and helped determine how GASH should behave so that job accounting would work out properly. Navin Manohar worked diligently to create WWW documentation for GASH. Alison Louise Brown and Jim Phillips worked long and hard to show us many new and interesting bugs that we were not aware of, and to manage the transition of our user and group databases into the brave new world of GASH. Virgil Moore, Pat Flinn, Pug Bainter, Joe Collins and several group administrators did the same for some 980+ network devices that are running around the laboratory. Dan Scott, Frank Douma, Jay Scott, and Elaine McSwain helped see GASH through over two years of development with advice, supervision, and patience.

Author Information

Jonathan Abbey is a system analyst in the Computer Science Division of Applied Research Laboratories, The University of Texas at Austin, where he has worked since September 1989. He graduated from The University of Texas at Austin with a B.S. in computer science in May 1993. His e-mail address is jonabbey@arlut.utexas.edu (a GASH generated alias!). General queries regarding the GASH software or the GASH NCE should be directed to gash-authors@arlut.utexas.edu.

References

[1] Stern, H. ''Managing NFS and NIS'', O'Reilly & Associates, Inc., 1991.

[2] Albitz, P., Liu, C. ''DNS and BIND'', O'Reilly & Associates, Inc., 1993.

[3] Callaghan, B., Lyon, T. ''The Automounter'', Sun Microsystems, Inc. White Paper.

[4] Allman, E. ''SENDMAIL Installation and Operation Guide Version 8.35 (For Sendmail Version 8.6)''.

[5] Wong, W., Colyer, W. ''Depot: A Tool for Managing Software Environments'', *Proc. LISA VI,* October, 1992.

[6] Allman, E. ''SENDMAIL Installation and Operation Guide Version 5.1.3++ (For Sendmail Version 5.67a+IDA-2.0)''.

[7] ''NIS to NIS+ Transition'', Sun Microsystems, Inc., December 1993.

[8] ''SunOS 5.3 Administering NIS+ and DNS'', Sun Microsystems, Inc., October 1993.