For users that have to build multiple versions of a software package on each of several architectures, we generally suggest that they create separate object directories to store the architecture specific files in. This can be done fairly painlessly with either makefiles or with liberal use of symbolic links to make a common set of sources appear in several directories.

every operating system upgrade. Some programs are also extremely difficult to port.

A related problem is how to figure out what kind of system that you are running on. It is all well and good to be able to use the `sun3` command to tell whether you are on a sun3 or not, but it is difficult to find out what type of system you are on unless you enumerate all of the possibilities and run the appropriate commands (which may not even exist). System V and its look alikes provide `uname`, but not all versions of `uname` provide the same information. Furthermore, you may be interested in distinctions that the vendor does not make - for instance, tt uname will not tell you whether a machine is a diskless client, or who is responsible for it.

Programs like `configure` attempt to make guesses based on known programs like `uname` and known peculiarities of given systems. Many sites also have local systems for maintaining the type information that is useful to them. OSU CIS maintains a file in /etc that specifies host type information; SRI ITAD maintains extra parts of the host information in their name server records. Such systems are relatively easy to devise, but they form strong limits on the portability of software between sites. Furthermore, they tend to evolve and shift over time, resulting in situations like that at SRI ITAD where most of the programs pull host type information from the name server, but most of the staff pull host type information out of comment fields in the NIS map for ethers. The two databases contain overlapping but non-identical fields, and are approximately equally correct, at wrong for 10 percent or more of the hosts. Finally, only locally written software is capable of using the information. Software written elsewhere will still have to rely on either asking the installer, or making educated guesses.

8 The Users

A heterogeneous environment can be extremely confusing to the users. Part of this lies in the differences between the systems: if you are used to an HP system, a Sun feels strange (and visa versa). These problems can be exacerbated if you support a more homogeneous heterogeneous system by giving everyone a single home directory that they use on all of the different platforms. Now the users have to take care to use the right binaries on each system. Rather than maintain a single `/bin` directory, sites like this tend to advocate separate bin directories for each architecture that a user deals with (for instance, `/bin.sun3`, `/bin.sun4`, `/bin.pyr` and so on). It is easy to set up one's dot files to accommodate this, but the methods that we are familiar with rely on site-specific mechanisms to provide a consistent means for the users to find their host type when they login. `Tcsh` provides a built-in variable, `HOSTTYPE`, which solves the problem nicely if you happen to be running tcsh on all your hosts; if you aren't, it merely moves the problem to one of figuring out whether or not you have HOSTTYPE available to you.

There are also the well known problems with byte ordering on dump tapes. Robust versions of `restore` exist, and will automatically handle byte and quad swapping to allow you to read most dump tapes. However, you can still run into problems when trying to read tapes written with `dump` on systems from other vendors, especially when the `restore` program you are using does not support byte or quad swapping. Byte ordering and quad swapping aside, the `dump` format is not guaranteed to be portable, so you may not be able to restore files created by another vendor's dump even if the byte ordering is the same.

There are other complications as well. Not all vendors ship `dump`. Irix gives you `BRU`, which is better than `tar` but which is not generally available on other systems unless you buy it. HP supplies a version of `dump` that does not support their file system extensions (access control lists and context dependent files). They do supply a program called `fbackup`, but it is not available for other platforms (so you cannot restore those backups onto other systems).

7 Differences in Programs

Another source of anomalies is in how programs behave on various systems. Some of this is expected - for example, System V `ps` takes different arguments than the BSD `ps`. The differences are not always as simple as knowing whether a system is based on system V or BSD UNIX. How do you use find to search an entire file system without crossing into other file systems? On a Sun, you use `find path -xdev`. On an HP, you would use `find path -hidden -mountstop`. Some versions do not even have options like `-xdev` or `-depth`.

Differences like these make it difficult to write shell scripts that will run correctly on more than one platform. Another example is the `at` command. On some systems (Pyramid OSx) `at` assumes that commands given to it through stdin are to be executed with `/bin/sh`. HP's `at` uses /bin/sh by default, but also allows you to specify it through the SHELL environment variable. Sun's work the same way, but also allow you to use an option on the command line to override the normal behavior. It would seem safe to use Bourne shell scripts with `at`, but you would also need to ensure that SHELL is set correctly as well. Details like these are not always obvious and further increase the difficulty in creating portable scripts.

In some cases, a sufficiently determined administrator can solve these problems by buying source from one vendor, or obtaining Berkeley or GNU project sources, and installing the same program everywhere. For programs like `rdist` and `lpd` which are not too deeply intertwined with the rest of the operating system, and which must interoperate in order to function usefully, this is a reasonably achievable solution. On the other hand, it is a maintenance nightmare, as you have to recompile and fix dozens of utilities for since otherwise users who are in 11 groups may have problems using the hosts that only support 8 groups. These problems will include mysterious failures of NFS access, as NFS servers that support only 8 groups are unable to validate users in more than 8 groups. This is primarily a problem between vendors, but may also be a problem between operating system releases; SunOS 3.5 supports only 8 groups, while 4.0 and above support up to 16. Users who need more than 8 groups may be forced to resort to having multiple accounts, or using passwords on groups (a procedure which, while theoretically possible, is clumsy and unpleasant at best).

5 Rdist

It is often useful to be able to distribute files from system to system using tools such as `rdist`. Some vendors supply `rdist`, which is convenient. The sources are also available for free from a number of locations, which means that you can install it on the hosts that do not come supplied with `rdist` by the vendor. But one can run into interesting problems regardless.

`Rdist` requires that it be installed with the same pathname on every host - if it is not, then an `rdist` running on one host will not be able to start an `rdist` process on remote hosts that use a different pathname. This is easily fixable once you know that the problem exists, although the cause of your problem may not be readily apparent when `rdist` fails.

When `rdist` establishes a connection to an `rdist` on a remote host, it also checks its version number and aborts the connection if the numbers do not match, to avoid protocol errors. Unfortunately, some vendors ship different versions of `rdist` which makes it impossible to use it as is for updating files. It is easy to work around this by retrieving the `rdist` sources from one of its locations and installing the same version on all of your systems.

6 Dump, Restore and Tape Drives

One of the authors has been working on a generic version of the BSD `dump` program that supports (among other features) end of tape (EOT) handling. One of the interesting results of this was discovering that the tape devices on our various systems handle EOT in myriad ways. Some Sun tape devices (1/2 inch and 8mm) usually return 0 on the write that hits EOT. Sun 1/4 inch tapes return an error, though bugs in the driver lead to indeterminate results (sometimes the write succeeds, but error is returned, sometimes it returns an error, and the write failed. Sometimes it writes a partial block and returns a partial count without failing. This makes it hard enough to support EOT handling on Suns, but consider that other vendors handle EOT in other ways. For example, the 1/2 inch tape devices on Pyramids return ENOSPC (No space left on device) at EOT.

against the host's NIS map, and not against the DNS as well. This means that whether you will be able to resolve hostnames that are not in the local hosts NIS map will depend on whether your client is bound to a Sun `ypserv` or not.

The fourth option solves this problem by forcing all hostname lookups to go through DNS, avoiding NIS entirely. This is a real solution to the problem, and Sun does provide the means for accomplishing this (through the `shlib.etc` package that can optionally be installed on your system from the distribution CD).

4.2 NIS Compatibility Problems

As another example of NIS compatibility problems, consider shadow passwords. Suns allow you to use a shadow password file and to propagate that information as a NIS map. Some other vendors support shadow password files in other ways, or not at all. On top of the complication of maintaining password information consistent between the set of hosts that support NIS and those that do not, we now have the added complications of dealing with the various ways of supporting shadow passwords, so that we might end up with some set of hosts that all get their normal password information through NIS but the shadow password information from different sources. Additional complexity is incurred when you try to integrate password aging into the picture.

An interesting compatibility problem from the users' point of view is that Sun uses the `passwd` command to modify passwords in NIS or on the local machine, as appropriate. Other vendors that provide NIS follow Sun's previous (and annoying) behavior of using `yppasswd` to change NIS passwords, and `passwd` to change local passwords.

4.3 Unifying Data For NIS

In order for NFS to work correctly, the information about users and groups must be consistent throughout the network. This is primarily an administrator's configuration problem, but it presents new challenges in a multi-vendor environment. For instance, group 1 is daemon by default on Suns, but is "reserved for 'other"' under SGI's Irix. Creating a consistent group file is not that hard in itself, but you also need to make sure that files retain the correct group associations when you change the group file. If you change the kmem group from group ID 5 to 10, you also have to ensure that the `/dev/kmem` device and any programs that are setgid kmem are also changed accordingly. Since most vendors consider group ids under 100 reserved, they feel free to make use of them however they choose.

The addition of multiple vendors to a network can increase the number of groups that your users need to belong to. You have to take care then not to exceed the minimum of the group membership limits on your hosts, This scheme allows every machine to look the same, without concealing from users what machine their home directories are on. Other sites prefer to make directories visible to users as /home/username, regardless of where the files live. This can be done with a similar naming scheme, and clever use of symbolic links and/or and automounter.

2.8 Automounting

3 Keeping Clocks Consistent

The proper operation of NFS requires that the clocks on the clients and the servers be within a few seconds of each other, since otherwise you can run into strange cache consistency problems. Unfortunately, the vendors do not always supply tools for maintaining clock consistency in a networked environment. Even when they do, they do not always provide the same tools. Some supply `rdate`, some supply programs that use the Time Synchronization Protocol from 4.3 BSD, some supply both, some supply neither.

The easiest solution to this problem is to retrieve the sources for `xntpd` and install that on all of your hosts. `xntpd` is a UNIX implementation of the Network Time Protocol (RFC 1119).

4 NIS

4.1 NIS Host Lookup Problems

One has four options for host name/address lookups on Suns. You can use the /etc/hosts file alone (if you are not running NIS), you can use NIS and use a hosts map that is common to your entire NIS domain, you can use the NIS hosts map and back it up with Domain Name Service (DNS) lookups when needed, or you can create your own copy of libc.so that uses only DNS for host name/address lookups. If your site is connected to the Internet, then you pretty much need to use the third or fourth options.

The third option works as follows. If the hosts NIS map was created with `makedbm -b` then Sun's `ypserv` will use DNS lookups to try to resolve NIS requests that were not found in the hosts NIS map. In other words, if a request for `foo.bar` is made, and that host is not found in the hosts NIS map, DNS will be used to try to resolve that hostname and get information about it. This provides a convenient mechanism to make use of the DNS without having to change much about your systems, and it works well in a homogeneous environment where all of the `ypserv` processes operate in that fashion.

It fails in some heterogeneous environments where the `ypserv` processes do not necessarily act in the same manner. For example, if a Sun NIS client happens to bind to an HP `ypserv`, then host name lookups are only resolved others) would allow this sort of group access. This sort of group-write scenario comes up most often in system administration tools, which frequently must run as root (to do backups on one server, for example) and still retain the ability to write a group-writeable file or directory to update a log or database. Disallowing group-write access for NFS requests from root makes it much more difficult to write generic sysadmin tools like backup systems.

The usual work-around for this sort of problem is to allow root access to any of the hosts that would need to run those sorts of programs. You can only specify root access for 10 hosts on Sun NFS servers, so large sites still tend to run into problems with this. The work-around that has been suggested by at least one vendor, is simply to export everything with root access, because "everybody else does it that way."

Furthermore, it is extremely easy for system administrators to confuse the possible types of granted trust, because root access to NFS-mounted file systems is granted on a per-file-system basis, completely separately from root rlogin access or tape drive access such as is granted by `/.rhosts`. The usual problem arises when `tar -cf - foo | (cd bar; tar -xf -)` does not work, while `tar -cf - foo | rsh bar-host "(cd bar; tar -xf -)"` does work. Since the operations are effectively equivalent, people expect them to require the same permissions, but the first requires NFS export with root, while the second requires `/.rhosts` entries.

2.7 Mount Points and Naming

It is extremely easy to set up NFS so that it's hard to find files. So easy, in fact, that I have had vendors assure me that it is impossible to use NFS and have files appear in the same places on all the machines. (This appeared to be a result of assuming that one of Sun's default automounter configurations was the only possible NFS configuration.) Since install scripts usually want to put all software in /usr, and put home directory space someplace with a fixed name that does not depend on the machine name, following instructions results in 17 machines with /home partitions, 17 more with /export/home partitions, a few more with /user partitions, and heaven only knows what else. If you have 17 machines with /home partitions, you obviously can't export them all so that they mount under their original names. Furthermore, if you have some machines with /home, some with /export/home, and some with /user, you will be lucky if you can remember where things are supposed to be on any given machine at any given moment.

Throwing out the default naming scheme, and settling on one that names partitions by host or by function will usually reduce hassles. For instance, one of our sites names user partitions "/n/machinem", where m is the 0 for the first partition on the machine, 1 for the second, and so on; the other site uses "/machine/letter" where "letter" is "a" for the first partition on the machine, "b" for the second, and so on. Both schemes work relatively well (although working on both systems at the same time induces near-terminal confusion).

are running in; several separate vendors have added named pipes to the file systems; Hewlett Packard has added access control lists.

All of these extra features may show up through NFS mounts, and have to be interpreted or ignored by the NFS client side. Some of them fail relatively harmlessly, like conditional symbolic links, which point nowhere on non-Pyramid NFS clients. Some of them fail catastrophically, like named pipes, which cause several Encore operating systems to crash on any attempt to access them. They all tend to make NFS mounts less transparent.

2.5 A Maze of Twisty Little Parameters, All Different

Most implementations of `mount` that work with NFS allow you to adjust NFS parameters such as the maximum read and write buffer sizes, timeout period, or number of retransmissions. We have found that in heterogeneous environments it is often necessary to adjust the read and write buffer sizes for NFS mounts to get the best possible NFS performance. The reason for this is that when a host sends an 8 kilobyte NFS packet out on the Ethernet, it is actually fragmented into 6 Ethernet packets. On some hosts (such as Suns), those packets are sent in quick succession, and can overwhelm the receive buffers on the Ethernet interfaces on other hosts. This results in lost fragments, which means that the NFS packet cannot be rebuilt, and the sending end will time out and retransmit the NFS packet. The timeouts can degrade NFS performance by an amazing amount in some cases. We have found that decreasing the read or write buffer sizes to one or two kilobytes can noticeably improve NFS performance (since it reduces the chances of overrunning a receive buffer).

The problem here is that to do the job right, you have to tune the NFS mount attributes for every type of NFS client and server pair that you have. If you have 13 NFS clients (Sun 3s running SunOS 4.1, Sun 4s running SunOS 4.1.1, HP 9000/3XXs running HP/UX 7.0 and 8.0, Pyramid 98XEs and 9825s running OSx 4.4c, etc.) and 5 types of NFS servers, then there are 65 client-server pairs whose NFS mount attributes you need to optimize.

2.6 Root and Group Access Problems

Most NFS users are quite familiar with the way that NFS servers usually disallow root access on file systems from NFS clients. This is fairly consistent on most systems now, although some still do not provide a means of restricting root access to only specific hosts (its either all or none).

Sun's NFS server also effectively removes any group access rights that the client might have if the user ID for the request is 0 (root). This means that a program running as user root and group backup cannot write a file that is owned by group backup, and writeable only by group backup. The NFS servers from some other vendors (HP, Pyramid, Encore and BBN, among a block; thus, `du` determines block size at compile time, not at run time. NFS mounts between Suns and Pyramids, for instance, cause `du` to be off by a factor of 4, since one `du` believes that all blocks are 512 bytes, and the other that all blocks are 2 kilobytes. This causes users who receive disk space warnings either to be elated, because you think they have a quarter the space they do, or furious, since you are claiming they use 4 times as much space as they think they do.

2.3 Soft Mounts versus Hard Mounts

NFS provides two styles of mounting things; in a hard mount, all operations are retried until they succeed, whereas in a soft mount, operations that fail return errors and do not retry. Soft mounts are often preferred, because they reduce the extent to which machines hang when other machines go down. Unfortunately, because soft mounts give up when there are problems, and return errors which many programs do not check for, file systems that are soft-mounted with write permission tend to experience interesting forms of corruption. Writes are often buffered until file closure, which means that an NFS failure may cause a close to return errors. Many programs do not check the results of close, and fail to take appropriate measures to safeguard data before exiting, which can result in truncated or mangled files. In other situations, apparently successful writes replace whole blocks of files with nulls.

The meaning of a soft mount also changes from vendor to vendor. Pyramid, in some versions of OSx, assumes that you simply do not care at all about soft-mounted file systems, and fails to propagate error messages (for instance, "Write failed") at all. This can and does lead to horrible surprises, as people discover that when they saved their file and exited the editor, it was silently truncated to 0 length. This makes soft mounts an even riskier proposition on heterogeneous networks than on Sun-only networks, where you usually know when your files are being eaten. The rule of thumb is that soft mounts are only advisable for file systems being mounted read-only; unfortunately, vendor manuals usually fail to mention that writeable soft mounts may result in data corruption.

2.4 Extending File System Semantics

More and more vendors have taken to using "unused" permission bits to add new features to the file system. Sun, for instance, uses the setgid bit on directories as an indicator for how to set group ownership on objects created in the directory. Hewlett-Packard uses the setuid bit to determine whether or not directories should be visible. There are other, odder file system extensions: Pyramid OSx extends symbolic links to provide conditional symbolic links, which have different targets depending on which OSx universe you unauthorized host, no matter what your opinion of its trustability is. Thus, if you turn on secure RPC you may be unable to mount filesystems on machines that you would like to trust. In this case, the least common denominator is NFS without secure RPC, and the choice is between increased security and increased homogeneity.

There are differences in the details of RPC implementation as well, which may cause difficulties even with apparently straightforward RPC calls. For instance, the maximum allowable size of an RPC response varies, but is not a queryable parameter. People writing RPC programs would be well advised to be conservative.

2 Network File System - NFS

The Network File System (NFS) is one of the prime tools for making large networks homogeneous. A reasonably clever naming scheme and possibly an automounter can make every machine appear to have the same files as every other machine. Unfortunately, there are both technical and logistical problems involved in achieving this effect. The technical problems have two causes. First, NFS is designed as a stateless protocol, and there are a number of operations on the file system that require state information. Compensating for this requires adding in new services to go along with NFS. Second, while most versions of UNIX are capable of hiding the facts about their network implementations and their filesystems from their users, many of them achieve this effect by coding assumptions into programs. These assumptions are often invalidated by NFS mounts.

2.1 Failures of Statelessness

The two famous places where the design of NFS conflicts with the uses of file systems are quotas and file locking. Sun provides `rpc.quotad` and `rpc.lockd` as work-arounds to allow the use of hard quotas and file locks. `rpc.quotad` actually works - for Suns - but few other vendors are capable of using it. `rpc.lockd`, on the other hand, is infamous for being flaky and bug ridden, even when every machine involved is a Sun. Neither of these problems are particularly amenable to work-arounds. Using quotas under NFS is particularly dangerous, since the most frequent failure mode is that quotas are silently enforced; writes fail once you are over quota, but no error messages are produced.

2.2 The Invisible Becomes Visible

Many programs need to know what size a block is. For instance, `du` attempts to provide the size of a file in kilobytes, while reading the size of the file in blocks. Unfortunately, on many machines `stat` does not return the size of running different software platforms. There are many software tools that make it easier to maintain such an environment, including NFS, NIS and rdist. These serve to present a more homogeneous view of the heterogeneous systems. For example, NFS allows one to mount file systems back and forth, without regard for where they came from or where they are being mounted. These are great, but differences between the systems often make these tools less useful than they might be.

There are differences in 2 regards: bugs or misfeatures that are present on one system but not others, and intentional features that are present on one system but not the others. This forces system administrators to choose between taking a "least common denominator" view of the net (forsaking some features to accommodate all of the systems), and reintroducing heterogeneity into a psuedo-homogeneous world. Sometimes even giving up features is difficult simply because it can be hard to identify just what the least common denominator is. Sometimes it is not clear that there is a viable least common denominator, as witnessed by the frequently quoted statement by the IEEE 1003.7 committee that the only standard administration tool they could find was `sync`, a least common denominator well below the threshold of usefulness.

This paper describes some of the tools that can be used in a heterogeneous environment and the problems that one can encounter in using them.

1 Remote Procedure Call

RPC (and XDR) forms one of the main bases of communications in heterogeneous Unix systems today. RPC is used in many packages, such as NFS and NIS. Initially RPC supported 2 forms of authentication - "none" and "unix". None really means no authentication at all (as the name implies), and the Unix style of authentication is little better than no authentication at all. It is fairly easy to "spoof" NFS or NIS packets and pretend to be a particular user (to read or write their files) or to provide fake NIS service if one has root access to a UNIX system, or any access to a PC on your LAN.

Consequently, Sun introduced a third form of authentication that uses DES to pass more secure authentication information with each RPC interchange. Proper use of secure RPC makes it much more difficult to spoof RPC transactions.

For example, secure RPC can be used with NFS. There is an option to mount(8) that allows one to specify that secure RPC should be used in all NFS transactions for a particular file system. You can also export a file system with exportfs(8) so that it can only be used from systems that use secure RPC. If you export a file system with secure RPC enabled, then it is more difficult to access that file system from an unauthorized host than it would normally be, and this is the crux of the problem. Any host that is not capable of the same secure RPC that the server knows is automatically an

Suns and Others; Accommodating Heterogeneity in a Sun Network

Steven M. Romig
Computer and Information Science Dept.
The Ohio State University
Columbus, Ohio 43210
614-292-8018
romig@cis.ohio-state.edu

Elizabeth D. Zwicky
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
415-859-3290
zwicky@erg.sri.com

July 4, 1993

Abstract

Sun provides lots of tools for administering machines in a network environment. Many of these tools are also available from other vendors. Unfortunately, adding machines of other architectures to a Sun network, even when those machines apparently come with all the right tools (NFS, NIS, rdist) is not as easy as this might suggest. This paper discusses some of the common problems with heterogeneous setups, and some solutions.

Many sites consist of heterogeneous environments in the sense that they consist of computers of several different architectures, from multiple vendors,