# Scalable, Remote Administration of Windows NT

Michail Gomberg, Craig Stacey & Janet Sayre
*Mathematics and Computer Science Division, Argonne National Laboratory*

## Abstract

In the UNIX community there is an overwhelming perception that NT is impossible to manage remotely and that NT administration doesn't scale. This was essentially true with earlier versions of the operating system. Even today, out of the box, NT is difficult to manage remotely. Many tools, however, now make remote management of NT not only possible, but under some circumstances very easy. In this paper we discuss how we at Argonne's Mathematics and Computer Science Division manage all our NT machines remotely from a single console, with minimum locally installed software overhead.

We also present NetReg, which is a locally developed tool for scalable registry management. NetReg allows us to apply a registry change to a specified set of machines. It is a command line utility that can be run in either interactive or batch mode and is written in Perl for Win32, taking heavy advantage of the Win32::TieRegistry module.

## 1. Overview of MCS Environment

The computing environment in the Mathematics and Computer Science Division of Argonne National Laboratory consists of nearly a thousand computers, including supercomputers, servers, workstations, desktop machines and laptops. Our NT infrastructure consists of 10 production NT servers, 8 Samba servers, 8 experimental NT cluster servers, approximately 70 NT workstations, and approximately 30 machines running Win95/98, most of which are laptops.

The systems group that manages these machines consists of 8 full time administrators, 3 of whom are at least partially responsible for the Windows environment. As with all other systems administrators, we have plenty to do and our users' expectations are high. Our environment continues to grow in complexity and scale; thus we are constantly searching for better and more scalable techniques for managing our collection of hardware and software. To that end, we have installed commercial packages such as HP OpenView, and have resorted to writing our own software when other packages did not fit our working model. Fortunately for us, the techniques for scalable management of machines running variants of UNIX are fairly well understood. We spent a considerable amount of time trying to find similar techniques that could be applied in a Windows NT environment. We did this partly as an academic exercise, but primarily because our NT environment is large enough that we could save significant time and effort if many mundane management tasks were handled remotely.

## 2. UNIX vs. NT Remote Management

UNIX machines come with many tools that make remote management possible. Many of the tools such as telnet/rsh daemons and rdist are provided by the vendor and are considered to be part of minimum standard installation. Remote management for UNIX machines, in many cases, involves obtaining a shell via rsh or telnet, editing configuration files that control the operation of the machine, and possibly restarting daemons. In some cases low level console support exists, allowing one to manage aspects of the hardware, such as rebooting, via a serial line. These methods alone allow us to manage all of our UNIX servers from a single console. We can connect to any machine, work with a familiar environment, and change any aspect of the machine configuration as long as we know what file to edit.

As our NT environment grew, we became very concerned about remote access issues. We could imagine running an environment in which we would have to walk to the machine room simply to add a user, or run from server to server trying to figure out why some service is failing. We also were concerned about the lack of command line administration tools. Our ability to manage a large number of machines effectively depends on the ability to automate tasks. We usually do this by scripting complex or repetitive operations, but we didn't see a way to accomplish this under NT.

As our experience grew, we realized that NT actually had many of the remote access primitives built in, but the model for remote access on NT was markedly different from UNIX. Instead of requiring a remote shell on the machine, we could manage aspects of the network using C/C++ code. A simple example of this is the NetUserAdd() system call, which creates a new user either on a target machine or in a domain. We discovered that NT has many of such system calls for managing the network, accessing the registry remotely, shutting down remote systems, etc. These calls generally either connect to a remote machine first, or name the target host in the first parameter of the call. The calls either perform the action or return some opaque handle. Additional calls then can be made to perform more operations using the handle as a parameter. The actual underlying mechanisms for the connection tend to vary. Some are implemented via RPC, others via NetBIOS functions. However, in most cases the underlying mechanism is never revealed in the documentation.

These functions provided us with the ability to do some remote management, but this method doesn't scale well. We didn't have the time to write management suites in C, and those of us who are familiar with C didn't have the necessary experience writing Windows code. Finally, even if we did invest time writing this code, it would not have helped us on the UNIX side. Our UNIX management code is written almost exclusively in Perl and uses a different set of primitives for remote communication. Fortunately, Microsoft was building some of this functionality into many of its own management tools. However, the ability to manage machines remotely was neither well-documented nor emphasized, probably because Microsoft was not considering enterprise issues at the time.

## 3. Remote Management Issues

We must consider what types of things we need to manage. The following is a non-exhaustive list of the major issues.

- <u>User account/security management</u>. This task consists of adding and deleting users and groups from the global or local user list, and changing user access privileges to specific resources, such as files or machines.

- <u>Service configuration and management</u>. Many of the services that we run need to be configured for our specific environment. Because of open standards, many of the services function in similar ways on NT and UNIX. For example, web daemons, DHCP servers, and DNS servers all need to be configured continually in a growing environment.

- <u>Exception notification and logging/security audit</u>. Both hardware and software generate exceptions and notification messages that warn of failure or other problems. Messages that show user access violations (repeated failed logins) and messages that warn of impending hardware failure (repeated failed writes to disk) are of particular interest.

- <u>TCP/IP stack and client configuration management</u>. A task we recently faced was changing the netmasks on most of our machines as we flattened our routed network to a switched environment. Although these tasks are not performed often, they usually have to be performed on many hosts at the same time.

- <u>Shutdown/reboot</u>. There are many situations in which we need to reboot machines remotely. This is especially true with NT. The netmask example above required a reboot for the change to take effect. We also have needed to shutdown all machines for planned power outages.

- <u>Software configuration</u>. Some software packages are run by a specific group of users. For example, our administrative group runs a custom application for ordering equipment and supplies. These packages sometimes need global configuration changes, such as moving a database server to a different machine.

- <u>Software distribution</u>. Software distribution is mostly concerned with keeping the software versions on each individual host up to date. A common misconception has this problem limited to Windows NT because software is often shared in a UNIX environment. In reality, the problem is simply a larger one on NT because almost all software resides locally. UNIX also has local software, but it is often limited to vendor patches or small confined packages that have limited scope such as a local version of a shell executable.

- <u>Troubleshooting/debugging</u>. When software or hardware fails, the cause of failure is not always immediately evident. On the UNIX side, we often log on, look at running processes, kill resource hogs, or, in the case of hardware failure, we often can reconfigure the OS to provide a reduced level of service so that the effect of the outage is minimized. NT generally requires us to perform similar tasks, although we often find that our software tool set, while functional, is more limited in coverage.

- Automated tasks. Automated tasks perform required daily maintenance or periodic routine jobs. An example we employ on our NT servers is a script, run nightly, to copy IIS log files to our UNIX servers where another script is run to calculate usage statistics.

## 4. Scalable Remote Management

The problem with remote management on UNIX or NT is that, in its simplest form, it's not really scalable. Using telnet or rsh alone is no better than sitting at the console of the machine. The ability to manage all such machines from a single console, one machine at a time, is no more convenient. Tools that provide good, simple abstractions for the environment and allow entire groups of machines or services to be managed as one allow us to manage that environment in a scalable way. A simple example using such tools is a file containing a list of all the hosts, together with a script that reads the file, issuing a remote command (rsh) to each host. In this example, the file represents the entire environment, and rsh provides a simple remote management protocol. Another example of a simple but widely used tool in UNIX is rdist. Rdist allows a single file, such as /etc/passwd or the configuration file for TCP wrappers, to be distributed to multiple hosts.

Some management scalability on UNIX machines is achieved via file sharing through NFS. The simplest example of this is an NFS mounted /usr/local, which allows software to be installed and configured only once for all the client machines. A more complicated example in our environment is our global Samba configuration file, which is merged with local configuration files on the servers, allowing us to manage our Samba servers as a group.

The aforementioned management techniques suffer from another problem. They are not really aware of the environment as a whole. All require explicit knowledge of which files need to be edited, and may require multiple versions of a single file to be distributed to different classes of machines. Thus, they don't significantly shield the administrator from the complexity of the environment.

Slightly better approaches are used in our "Config" system, written by Remy Evard, and in Mark Burgess's cfEngine. [1][2] Both of these tools are still somewhat file- centric (our Config system currently defines over 100 files). The strength of these tools is that they allow system management based on abstract classes such as OS type, machine function, or other arbitrary classifications. These tools describe the system as a whole and manage machines by making them comply to the desired standard. CfEngine is also different in its lack of dependency on built-in UNIX communication primitives. While the Config system relies on rsh and NFS, cfEngine uses its own daemons and communications protocols.

The scalability of our UNIX management approach is achieved through the availability of a highly diverse tool set. We have many traditional tools that allow us to manage single machines and a new generation of tools to manage the whole environment. The common themes for these tools are a simple, built-in communication method and a building-block approach to create more complex behaviors.

## 5. Tools for Remote NT Management

Today we can manage most of our NT infrastructure from a single console. We still make infrequent trips to the machine room, but most of those are necessary because of hardware issues and not day to day management. The most important tools at our disposal are the Ataman telnet/rsh daemon, ActiveState's port of Perl, and the NT Resource Kit. These tools allow us to manage accounts and services, make registry changes, distribute software, and reboot machines. Other tools included with NT and the NT Resource Kit allow us to view event logs, monitor process state and resource utilization, capture and filter network traffic, and remotely connect to the console when required.

The Ataman telnet service and Perl are a huge advance for the NT world because they provide a familiar management environment. We recently used the following script to shutdown all of our NT workstations in preparation for a planned power outage. Interestingly, this script was triggered using rsh from a UNIX workstation. Also note that portions of the NT Resource Kit are copied into a DFS share so they are accessible to all of our NT machines via a UNC path. This is a similar concept to /usr/local in our UNIX environment.

```
use Win32::NetAdmin;

Win32::NetAdmin::GetServers(undef, "MCS", SV_TYPE_NT, \%all_server_ref);

Win32::NetAdmin::GetServers(undef, "MCS",
     SV_TYPE_SERVER_NT | SV_TYPE_DOMAIN_CTRL | SV_TYPE_DOMAIN_BAKCTRL,
     \%nt_server_ref);

# get rid of servers and only leave workstations
foreach $server (keys(%nt_server_ref)){
    print "$server\n";
    delete $all_server_ref{$server};
}

foreach $server (keys(%all_server_ref)){
    $result = `\\\\mcsnt\\DFS\\soft\\adm\\packages\\ntreskit` .
            `\\shutdown \\\\$_ /T:60 /Y /C`;
    print $result;
}
```

Another common task is changing a registry value on all machines. The following example copies AT&T Research UK's VNC settings (including the password) from the host machine to all other machines on the network. This script can be run on any of our machines.

```
use Sys::Hostname;
use Win32::TieRegistry;
use Win32::NetAdmin;
$Registry->Delimiter("/");

Win32::NetAdmin::GetServers(undef, "MCS", SV_TYPE_NT, \%all_server_ref);

my($hostname) = uc(hostname());
$hostname =~ /^(\w+)\./;
$hostname = $1;

my($src_key) = $Registry->Connect($hostname,
       "Users/.DEFAULT/Software/ORL/WinVNC3");

foreach $server (keys(%all_server_ref)){

      next if( $server eq $hostname );
      my($rem_dst_key) = $Registry->Connect($server,
            "Users/.DEFAULT/Software/ORL/WinVNC3");

      if(!$rem_dst_key){
            $rem_dst_key = $Registry->Connect($server,
                                    "Users/.DEFAULT/Software/ORL");
            next if(!$rem_dst_key); # host doesn't have VNC at all
            $rem_dst_key->CreateKey("WinVNC3");
            $rem_dst_key = $Registry->Connect($server,
                              "Users/.DEFAULT/Software/ORL/WinVNC3");
        }

      my(@value_names) = $src_key->ValueNames;

      my($value, $value_string, $value_type);
      foreach $value (@value_names){
            ($value_string, $value_type) = $src_key->GetValue($value);
            $rem_dst_key->SetValue($value, $value_string, $value_type);
      }
```

```
	undef $rem_dst_key;
}
```

Conceivably, both of those tasks can be accomplished on our network by hand. However, very large or distributed organizations could certainly benefit from this more scalable approach. The above script 65 seconds to complete on 126 hosts on our network, which includes bandwidth ranging from switched 100 Base-T to 128kb/s ISDN. Expanding this to a network of 5,000 hosts, the script would take approximately 43 minutes to run, significantly less time than it would take to visit every machine in person.

While the UNIX management model is file-centric, the NT management model is almost entirely registry-centric. Although the registry is a binary file, logically it is arranged as a hierarchical forest much like a POSIX directory structure. This similarity allows some of the techniques that we learned from our UNIX experience to be applied to NT management. For example, we can extend the Config system or cfEngine (if ported) to manage a set of registry keys. Unfortunately, Microsoft has not completely committed to using the registry as a central and sole store of configuration information. A notable exception is IIS, which uses its own binary format file as well as the registry for configuration data. One positive direction that Microsoft seems to be taking is the ADSI style management interface. This technology relies on OLE objects addressed in the form of PROVIDER://<Host>/<object1>/<object2>/<etc...>. Each returned object manages a particular aspect of a service or a machine. Because the objects are OLE-based, they are language neutral and can easily be used from Perl. Since the objects can specify remote hosts, they can be used as building blocks for higher level scalable tools.

There are, however, several problems with this approach. Since ADSI objects rely on OLE, the objects must be known in advance by the client workstation that is doing the management. This means that for true scalability, all the known objects have to be distributed to all the machines. Since these objects are usually a part of a much larger package, such distribution is impractical and may break license agreements. Another problem is that not all management functions are covered, so changes may still have to be made through the registry or through configuration files. The final, and possibly most difficult, problem is that because ADSI is not available on UNIX platforms, we can't use our NT developed tools for integrated enterprise management. We hope to avoid much of the duplication of effort that goes into managing these two environments.

Many of the other tools we use for remote management come with NT. For example, we use the User Manager and Server Manager tools to manage accounts, services, and shares. We use Regedt32 for remote, single-machine registry edits. We often use the Event Viewer as a first step in our trouble shooting, and we use the Performance Monitor as both a data collection tool and a first line of defense against problems. If necessary, we use VNC from AT&T Research UK to connect to the console remotely and perform operations that can only be done from the console.

## 6. The NetReg Tool

The motivation for NetReg came from writing many Perl scripts to manage different portions of the registry. We realized that we needed a tool that was easily scriptable by someone familiar with the registry but not necessarily with Perl. We wanted a tool that could be used from the command line (in a telnet window), did not depend on regedit, and could be used in interactive mode to make individual changes. We first looked at the NT Resource Kit, which contains at least 14 different tools to edit the registry. Each tool uses a slightly different syntax, and all work in slightly different ways. Only some can be used to manage remote registries, and only a few support scripting. We wanted to have a single tool that combined all these functions, had a simple unified syntax, and could build on the power of Perl.

NetReg is built around the Perl TieRegistry module written by Tye McQueen. The TieRegistry module is extremely powerful and easy to use. It lets you manipulate the registry via an object or via tied hashes. The "chaining" feature allows intermediate registry objects to be created, which then can then be used for subsequent connections to sub-objects. TieRegistry also allows remote connections, can manipulate any type of registry value, and can cache lookups for enhanced performance.

In its current incarnation, NetReg is in many ways a proof of concept. Both the syntax and the execution model will change in further releases. The current syntax consists of the following elements.

```
load [net <domain>|sms|file|machine_list] –pick –regexp
search [keys|values|valuename|all] –r –pick –regexp <path> <search value>
copy [keys|values|valuename|all] [<identifier>|<src_path>] <dst_path>
list [keys|values|valuenames|all] [<path>|<identifier>]
select –regexp –pick <identifier>
edit <identifier> <perl_regexp_replacement>
reboot <identifier>
bookmarks [load|save|list] –pick –regexp
bm <bookmark_name> <path>
```

As stated before, NetReg can be run in interactive mode or in batch mode. The interactive mode is currently command line driven but will be extended in future versions to support a TCL/TK GUI as well as a command line.

Most NetReg commands return an implicit result. The succeeding operations use and augment the result. For example, a load operation creates a list of machines; a search operation uses the current machine list and will return a list of machines and keys that match the search criteria; an edit operation simply uses the current machine and key list to perform a global edit operation. The following example illustrates the usage. This three-line sample performs the same VNC registry copy as the previous Perl script.

```
$vnc = LMachine/Users/.Default/Software/ORL/WinVNC3/*
$all = load net MCS
copy values $vnc $all/$vnc
```

A very useful feature of NetReg is the ability to use bookmarks. Bookmarks can either be a simple alias for a complex key name or can consist of a complicated chain of registry keys and values. The following examples demonstrate this.

```
bm vnc LMachine/Users/.Default/Software/ORL/WinVNC3
bm services “LMachine/System/CurrentControlSet/Services”
bm netcards <services>/NetBT/Adapters/*
bm ipaddr  <services>/{<netcards> =~ /\/(\w)$/}/Parameters/Tcpip/IPAddress
```

In the example above, *vnc* and *services* are simply aliases for longer key paths. The bookmark *netcards* is an alias that may return multiple paths or values. The *ipaddr* bookmark is special. The curly brackets denote an executable query. Thus, *ipaddr* will first evaluate the *netcards* query, then apply a Perl regular expression to the result. For Perl novices, the regular expression selects the last word after the “/”, which in this case should be the name of the device driver which implements the NetBT service and, in most cases, the TCP/IP service as well. The result of the regular expression match then is inserted into the bookmark. Finally, the bookmark is evaluated as a whole to return the IP address assigned to the adapter. In machines with multiple adapters configured with TCP/IP, the same query would return all the IP addresses assigned to each card.

The bookmarks can be used in any command that accepts a path. To delimit the bookmarks from the rest of the path, they are enclosed by “<>”. The bookmarks’ usefulness depends on the user. While there will be some predefined common bookmarks in the distribution, a judicious use of additional bookmarks will help simplify registry navigation.

Since NetReg is still evolving, it is premature to discuss all its features in this paper. Additional documentation and examples will be available on-line in the distribution.

## 7. Other NT Remote Management Solutions

We took an informal look at other Argonne divisions and one educational site to discover how other groups were coping with the problems that we encountered.

Some sites find they are able to use NT efficiently without adding any outside tools. For example, Argonne's Office of the Chief Financial Officer has about 20 servers (for development, files, and applications) and approximately 300 workstations, all managed by the tools that come with NT. They have to visit the machines in person occasionally, but find that most remote management tasks they need to perform are available. They evaluated SMS, and plan to start using it when the new version becomes available, but have managed without adding to NT for quite a while. [7]

A little further along the continuum lies Concordia University. While they use no remote management tools for NT servers located on their main campus, they have added to NT's remote capability by using PCAnywhere and CarbonCopy to install and monitor NT at other locations. Unfortunately, a phone call to the person at the console is still the solution to some problems. For managing workstations, KiXtart is used for small changes. Major changes are made by cloning machines to identical states using GHOST. This works in that environment since these are lab PCs. Concordia does have concerns for the future, however, and hopes to solve the remote administration challenge more satisfactorily, since they are due to become the central support for a consortium of colleges spread across the country. [8]

Argonne's Electronics and Computing Technologies (ECT) division goes a little further still. They used SMS in the past, and plan to use it again when version 2 is released with the new version of BackOffice. They had a successful installation of Office 97 using SMS to push the install, but currently they must physically visit a machine to install software. They remotely manage servers by mounting administrative shares from the server to the workstation. Most tools (user manager, server manager, etc.) are designed to run remotely, and the administrators find they can do everything they need. They're reasonably happy with their existing solution, but have encountered some tasks that simply do not have an obvious remote solution, *e.g.*, installing FrontPage extensions remotely. ECT manages three main domains, with about 24 servers. Users have administrator privileges, even though SMS is used for installs, since installs run as the user.

One ECT project did turn out to be a solution to the remote install problem. Autolink. an Argonne developed tool, was designed to solve the problem of different programs requiring different versions of a .dll file, but it became a solution to the remote install problem. Managing the .dll's by keeping them on the server meant that a solution had to be found for installation on the remote clients. Although it is narrow in scope–only designed to handle a well-defined set of applications and requires user input to execute–Autolink's combination of Oracle and PowerBuilder allows entire groups of applications to be managed from a single location. [9]

An even heavier user of SMS is Argonne's Chemical Technology division. The systems administrators customize SMS heavily, writing scripts that run on the desktop to allow remote installation of anything they want. This is an extremely centralized environment, with workstation users denied administrative privileges on their machines. Despite the authoring efforts put into remote installation on workstations, server management in that division is accomplished by the tools that come with SQL manager, user manager, DCHP manager, etc. [10]

## 8. Unsolved Remote Management Tasks

We have encountered some management tasks that we simply cannot perform remotely because of current NT and hardware limitations. Some issues we simply decided to ignore, partially because our environment permitted it, and because the payoff for solving those issues at our site was small.

### *8.1 No locked down machines*

We run our machines using a trusted environment model. We operate under the assumption that if you have physical access to the machine, you have the ability to get root-level access on that machine. Because of this, the local administrator account gets no domain privileges. In fact, whenever a user is assigned to a particular workstation, we grant administrator access right away. We experimented with not doing this, but it proved to be more trouble for us as well as the users because they couldn't get their machines to do what they needed to when they needed it.

We have found that allowing users to have administrator access has not been as problematic as we originally assumed. The users are warned not to keep any data on the machine's disk, and to expect that a full rebuild is a possibility if they mess up the machine to the point where we can't fix it. This has rarely been the case, but it has happened.

### *8.2 User-Installed Software*

In giving the users administrator access to their machines, the distribution of software has become even easier for us. For most packages, we have set up a "come 'n' get it" system, where all of the software we distribute to the users is available from a central place—— in our case, a web page.

We created a Windows Software Repository web page (http://www.mcs.anl.gov/windows). It's a manually updated web page that uses UNC hyperlinks to direct users to the setup programs for various packages. The packages are distributed from a DFS shared repository and the web page includes any helpful hints for installing the software, as well as license keys. We've also taken advantage of the automated installs we've created for some packages for use in the machine building scripts. We provide links to those for users who wish to have a default install on their Windows NT machine.

In order to allow users to take full advantage of this web page, independent of their browser or OS (Windows 9x on laptops vs. Windows NT on desktops), each network-resident distribution directory has a setup.bat, which calls the appropriate setup program. This ensures the setup program is launched with the correct path, as Netscape does not pass the path along when directly running a "file:" hyperlink. Using this setup.bat method also allows us a crude but easy installation tracking method, by sending an email using the freeware email sending program BLAT(http://www.interlog.com/~tcharron/blat.html) whenever the installation program is run. This is especially helpful in dealing with software that is not under a site license. Because we use UNC links to a DFS shared directory, we maintain security in that only users logged into the domain can access the software directory tree.

## 9. Remaining Problems

While programs like SMS remote control and AT&T Research's Virtual Network Computing help us get to the console of machines that are up, we still do not have quite the low-level console capability we have in UNIX. If our servers get stuck in the boot process before Windows actually starts, we are left in that awful position of physically having to be at the machine, which is not very practical on a weekend. While this will still happen with our UNIX machines from time to time, it is still far less frequent an event than in with PCs. Still, it makes a remote reboot a little more risky.

Our current procedure when an NT box can't boot is to boot it with ERD disks at the console. It would be incredibly useful if Microsoft implemented ERD-like functionality into NT and allowed the console to be transferred to a serial port.

## 10. Summary

Our experience with NT has shown us that, with the right mix of tools, it is possible to manage NT remotely, in some cases in a scalable way. NT has a different set of primitives than UNIX for remote communication. It looks like Microsoft is taking enterprise management issues much more seriously. We hope they will learn from the UNIX community how to handle large installations of machines and software.

We also hope that Microsoft will take more of a building block approach to providing management applications for NT. Experience has shown repeatedly that monolithic applications have too great a learning curve and are often too inflexible. We also would like to see Microsoft stick with a particular set of its own standards, such as using the registry or ADSI, so that we can more easily build tools to our own specifications.

## 11. Author and Project Information

Michail Gomberg is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. He was the lead technical architect for this project. His email address is gomberg@mcs.anl.gov.

Craig Stacey is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. His email address is stace@mcs.anl.gov.

Janet Sayre is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. Her email address is sayre@mcs.anl.gov.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

## 12. References

[1] R. Evard. An analysis of unix system configuration. Proceedings of the 11th Systems Administration conference (LISA), page 179, 1997

[2] Mark Burgess, Cfengine A Site Configuration Engine, USENIX Computing systems, Vol8, No. 3 1995

[3] Microsoft Windows NT Workstation 4.0 Resource Kit, Microsoft Corporation, Microsoft Press, 1996

[4] Comparing Microsoft Windows NT and UNIX Remote Management. Microsoft White Paper.

[5] Comparing Microsoft Windows NT and UNIX System Management. Microsoft White Paper.

[6] Microsoft Windows NT from a UNIX Point of View. Microsoft White Paper.

[7] Kay Burdi, Argonne National Laboratory-OCF, 5/26/99

[8] Rich Helmke, Concordia University, 5/26/99

[9] Rich Raffenetti, Argonne National Laboratory-ECT, 5/28/99

[10] Steve Gabelnick, Argonne National Laboratory-CMT, 5/27/99