NetReg: An Automated DHCP Registration System

Peter Valian and Todd K. Watson – Southwestern University

ABSTRACT

NetReg is an automated system that requires an unknown DHCP client to register their hardware before gaining full network access. Through a simple web interface, the client is prompted for their user identification. Powerful scripts then retrieve the client's network fingerprint and store it along with the user's information in a database. The database provides administrators with real-time information for troubleshooting and auditing their networks. The entire system was developed utilizing unmodified, open source servers and in-house developed programs.

Introduction

At Southwestern University, we want to provide an "open" network that does not limit student access and gives them an easy and instant connection. In the past, we have used a simple Dynamic Host Configuration Protocol (DHCP) [5] server to assign IP addresses to any client connecting to our residential network (ResNet). This system worked well; however, it did not provide system administrators adequate information about clients connected to the network. Tracing a network address to an end-user proved to be a tedious and time-consuming task due to the lack of a unified data source containing information about what hardware belongs to which user. We wanted to improve and automate this system while maintaining client-side simplicity.

A few commercial products exist which will perform various types of network registration (such as Cisco's *Network Registrar* and Lucent's *QIP*)[1]; however, these products are often expensive and provide functionality not necessarily required for a simple registration system. None of these products fit our needs 100%, so it didn't make sense to spend a lot of money on them. Some universities have created registration systems which provide the functionality we were looking for, but they lacked the simplicity of using off-the-shelf servers without any modifications. This was mostly a result of DHCP servers not having needed features (such as supporting multiple address pools per subnet) when they developed their registration systems. Time and programming staff is limited at our small liberal arts school, so we needed something that would work well without having to put in a lot of development work.

The release of version 3 of the Internet Software Consortium (ISC) DHCP server[2] brought a free server supporting multiple address pools per subnet. This was what we needed to create our system based on readily available open-source servers without modifications.

[1]These seem to be the two most commonly used commercial systems at least in the university arena.

[2]At the time of publication, the current release was 3.0b1pl0. We have been running version 3 since the initial Alpha version, and it has been extremely stable.

The Registration System

The first step in designing our system was to outline what we wanted as an end product. We needed to have a single source of information linking a Media Access Control (MAC) address or IP address to a specific user ID. Each user on our campus network is responsible for his/her computer and university computer accounts as specified in our Acceptable Computer usage policy. We wanted an "electronic signature" stating that the user has read and understands the usage policy before gaining access to the network. We also needed a means of denying access to the network without proper credentials. In addition to functionality, we wanted a nice front-end administration interface that would not require the knowledge of command-line parameters or file names. The interface also needed to be cross-platform so Unix, PC, and Macintosh support people could administer the registration data.

The administration interface needed to provide basic search capabilities such as finding the user ID that registered a particular MAC address or finding out what MAC address is using a certain IP address. Also, NetReg needed to be powerful enough to cross-reference this information for queries such as, "Tell me what is the last user to use this IP address."

The Platform

Due to the way we decided to use DNS, we knew we needed to have a dedicated machine for NetReg. In the interest of our limited budget (and because we're partial to penguins), we decided to use RedHat Linux on a retired 200MHz Pentium. Since our residential network is so small (roughly 750 nodes), a more powerful machine wasn't necessary[3]. A basic installation

[3]Actually the system runs at such a low load average that it should be scalable to thousands of nodes without problem.

of the operating system was needed – no kernel hacks or windowing systems – just basic networking with a static IP address. On our Cisco router, we specified the IP of the NetReg machine as the **ip helper-address** in each subnet declaration of our residential network.

The Servers

Three servers are needed to make NetReg work: DHCP, DNS/BIND and HTTP. Again, our goal was to use reliable and low-cost software. We were very pleased that we managed to use completely open-source, well-known, heavily tested and hence very reliable servers. RedHat Linux 6.0 comes with BIND 8 and Apache 1.34. We chose to upgrade to the most recent version of Apache[4], but used the BIND server included with RedHat 6.0. Alternatively we could have used a simple Perl (or similar) program to act as a DNS server instead of the full DNS/BIND package, but since it is bundled with Linux, is well-documented, and does what we need, we didn't look any further. The ISC DHCP server is open-source; however, they have recently begun offering support contracts for those wanting commercial support. Similar contracts are available for RedHat (which support the servers shipped with it).

DHCP

The ISC's DHCP server version 3 was a dream come true. We consider this package to be most crucial to making NetReg possible because of its multiple pools per subnet capability. The first step was to define two IP address pools per subnet for each of our residential network subnets (see Figure 1).

```
subnet 161.13.1.0 netmask 255.255.255.0 {
  option routers 161.13.1.1;
  # Unknown clients get this pool.
  pool {
    option domain-name-servers 161.13.1.25;
    max-lease-time 120;
    default-lease-time 120;
    range 161.13.1.100 161.13.1.150;
    allow unknown clients;
  }
  # Known clients get this pool.
  pool {
    option domain-name-servers 161.13.1.2;
    max-lease-time 28800;
    default-lease-time 28800;
    range 161.13.1.151 161.13.1.250;
    deny unknown clients;
  }
}
```

Figure 1: Defining Multiple Pools per Subnet in dhcpd.conf

One pool would be small and only allow unknown clients to get a DHCP lease for a short amount of time (two or three minutes), and the other pool would be larger and grant leases to known clients for longer periods of time (24 hours). Known clients are those which have their MAC address defined in the server's configuration file (more on how a client becomes "known" will be explained in following sections). Unknown clients will receive a temporary IP address and the NetReg machine IP as their DNS server. Known clients will receive an IP address and the true DNS server address. A variety of configurations can exist based on the network topology. The best method is to use temporary addresses which are not globally routable.

DNS/BIND

The NetReg machine also runs as a "fake root" DNS server. Essentially, the DNS server is told not to communicate with any real DNS servers, but to simply resolve all queries against its own database file instead. Its database file has a very simple, one-line wildcard configuration telling the server to resolve every address to the NetReg IP address. In doing this, clients that received the "unknown client" configuration from the DHCP server are using the restricted "fake root" DNS server to resolve all their DNS queries. Unknown clients are effectively locked out from accessing any machine except the NetReg machine[5]. Use of this will become clearer in following sections.

HTTP

One more server ties everything together – the HTTP server. For this server we used the ever-popular Apache HTTP daemon. A basic installation was needed with little special configuration. The main index HTML page is a simple form with an input for a user ID and password (see Figure 2). The only modification made to the configuration was to redirect errors to the main index URL. To do this, we simply added the following lines in Apache's httpd.conffile:

```
ErrorDocument 404 /
ErrorDocument 403 /
```

This will effectively redirect any page for any requested site to the registration page.

The Process

And now the answers to all your questions[6]. There is a graphical flowchart of the process we are about to describe in Figure 3. When a client connects to the network via DHCP, the router will forward the "DHCP Request" packet[7] to the DHCP server on the NetReg machine. The DHCP server on NetReg will determine if the client is "known" or "unknown" and grant it the appropriate IP and DNS server information. If the client is unknown, the DHCP server

[4]We upgraded to Apache 1.3.6, however, since then version 1.3.9 has been released.

[5]The users could technically work around this by using an IP address instead of a URL, but it becomes much of a hassle to browse the web like that!

[6]OK, maybe not **all** of your questions, but at least those pertaining to NetReg.

[7]DHCP is a broadcast protocol which is squelched at the gateway interface of most segmented networks. Most routers can be configured to forward boot requests to one or more IP addresses.

assigns the "fake root" DNS server which resolves everything to itself. So now, if an unknown client opens a web browser and attempts to access any URL (e.g., http://www.usenix.org/), the "fake root" DNS server will resolve the server name to the IP of itself. The HTTP server on NetReg will serve that request with its index HTML file, which is the registration page for unknown clients. Now, if an unknown client requests a particular file or directory (e.g., http://www.usenix.org/events/lisa99/) the server part of the URL will resolve to the NetReg machine, but it will still want a particular page or directory which will most likely not exist. This is where the error redirects come in. The 404 (Not Found) and 403 (Forbidden) errors get redirected to the main index HTML – which is, of course, the registration page!

On the registration page (see Figure 2), the user is with the full text of our Acceptable Network Usage Policy. The page explains that by clicking the "Accept" button they signify that they have read and understand the policy. If they do not wish to accept the policy, they cannot gain access to the network from their machine.

How It Works

When the user clicks "Accept", the form data are sent to a Perl CGI script. The first thing the script does is authenticate the user against our POP server via the **Mail::POP3Client** Perl module [3] (all users on our network use this server for their mail, so if they have an e-mail account, they are permitted to use our network). Once the user is authenticated, the script grabs the user's IP address from the environment

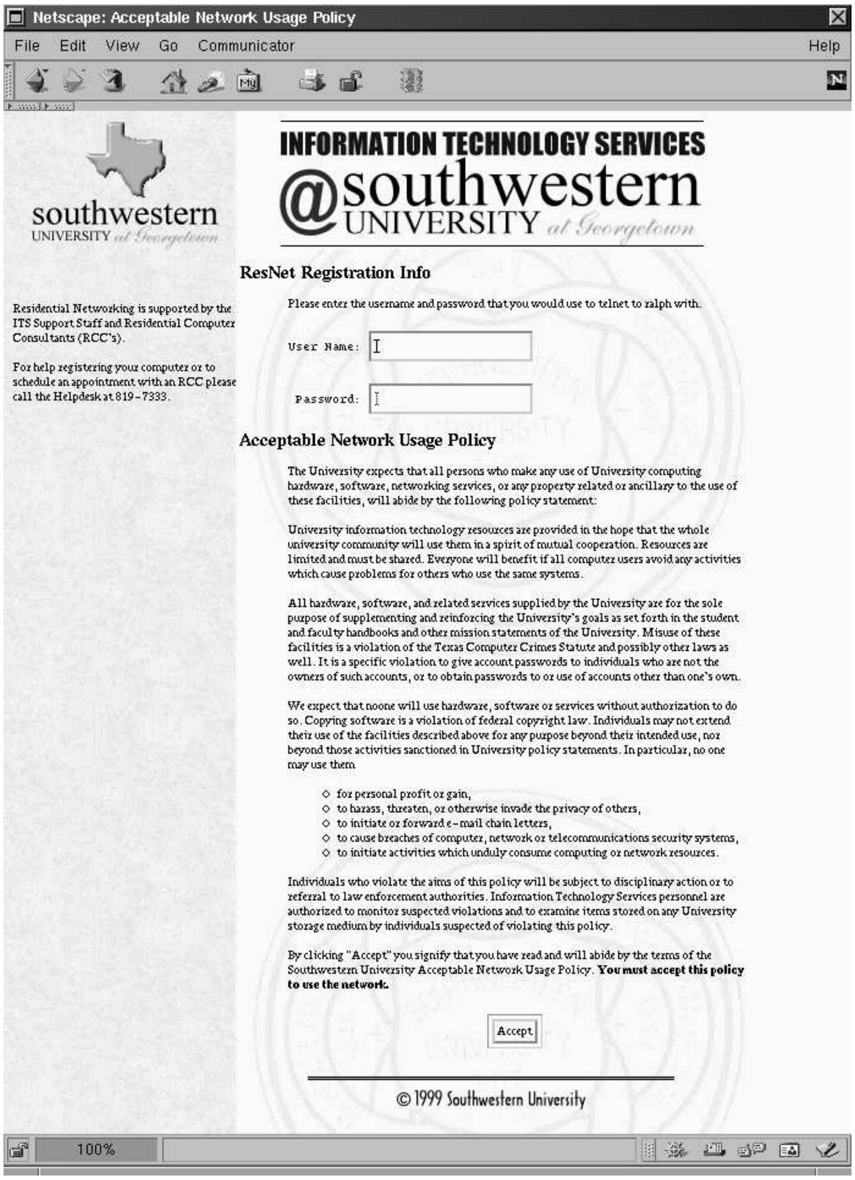

Figure 2: The NetReg Registration Page.

variables sent along with the CGI data. The user's IP address is searched for in the DHCP server leases database in order to obtain the user's MAC address. The user's MAC address is then added to the DHCP server configuration file along with other commented information about that MAC address, such as user ID, browser type used to register, time of registration and which subnet the user was connected to at time of registration.

Every minute, a script running from cron checks if there are any new registrations since the last time the DHCP server restarted. If there are, the cron script kills the server and starts it again so it reads the newly written configuration file[8]. During this time, the user is asked to reboot their system. Not only does this provide enough time for the DHCP server to reread its configuration, but also helps some clients with poor implementations of DHCP. We have found that sometimes a client will not change its DNS server to the real DNS server when it goes to renew its lease. Other times, we have found clients using both the fake and real DNS servers in a round-robin fashion. We opted to tell users to reboot rather than explaining how to release and renew a DHCP lease for each platform, while it gives the delay needed by cron[9] to reload the new dhcpd.conffile. Once registered, our DHCP server configuration will allow users full access to our campus intranet as well as the Internet. Users may also

[8]The DHCP server does not currently support HUP but may support on-the-fly configuration reloads in the future.

[9]Cron will only execute scripts once every 60 seconds at most.

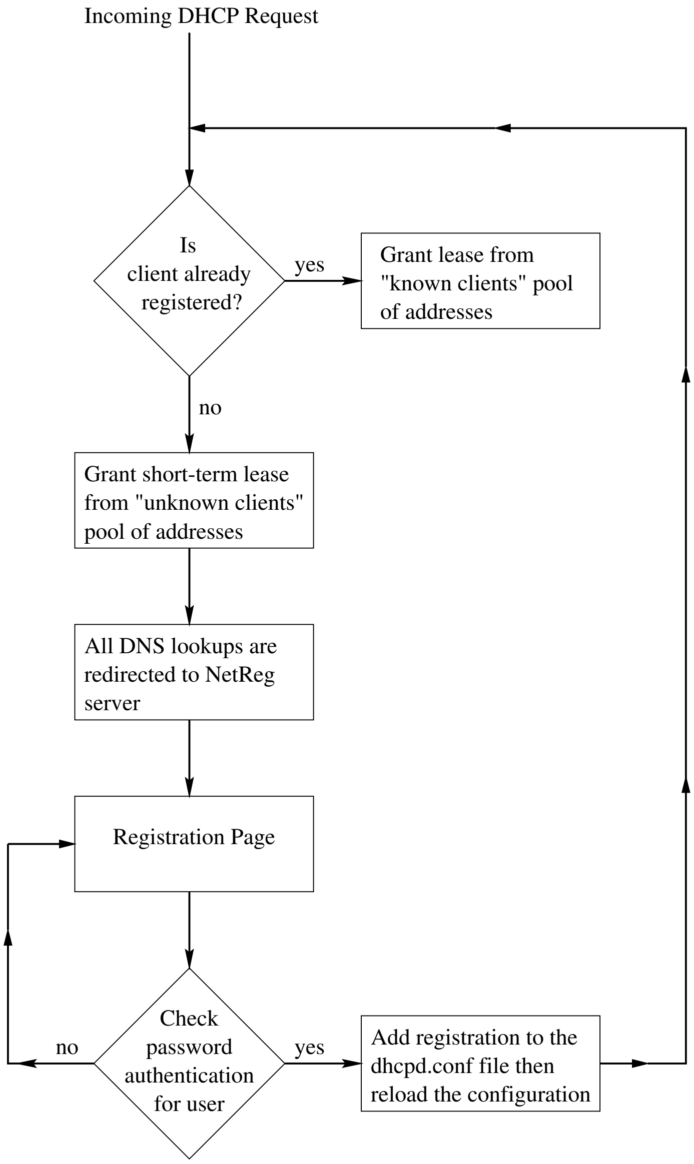

Figure 3: The NetReg Process

roam to any student lab or residential subnet and have the same network privileges as they would from their room without changing any network setting on their machine and without having to re-register.

Administrative Interface

NetReg is a great registration system, though the real usefulness comes in the administrative interface, which allows a network administrator to view, search, and manage the collected data. A single Perl CGI program provides an initial view of all subnets (see Figure 4) with a graphical representation of the number of clients registered on each subnet. From there, the admin can then view the individual registrations for a particular subnet (see Figure 5), perform a search for an indivual MAC address, user ID, IP address, or check the current status of the servers running on the NetReg system.

Although you can allow the administrative interface to be accessed through the same HTTP server as the registration page, we opted to set up a separate server to do this. For this server we used the SSL patched [6] version of Apache 1.3.6 using OpenSSL [7].

There are some nice features that are built in to the web-admin interface. For example, if we search for a MAC address we will then be able to query the system for the user ID that registered the MAC address. From there we can see all registered MAC addresses for that user ID, or we can perform a query

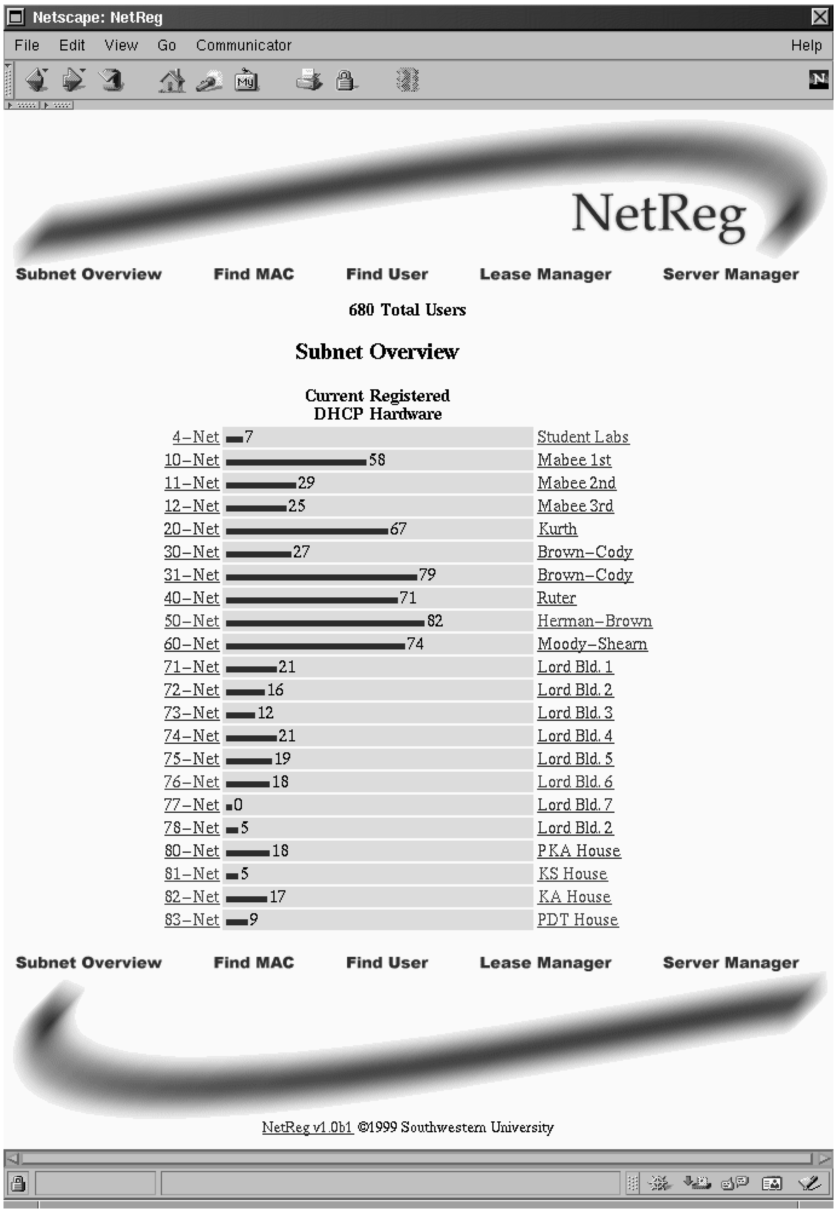

Figure 4: The Main NetReg Admin Page.

for the identity of that user ID against our internal directory server. A pleasant surprise came out of the "Find MAC" routine. Actually we can search for any string, for instance we can search for the string "PPC" or "Linux" to find useful statistics on non-Intel or Linux systems that are registered. Simple scripts can be written to get these data, but a quick-look can be done with the search function, which is very useful.

Finally, we have the ability to **delete** a registration entry via the web interface. By doing this, when the DHCP client requests a renewal of its current lease it will be denied a renewal and offered a new temporary lease from the "unknown" address pool and forced into the registration process again. If the case arises where an ethernet card changes ownership, the computer with the card will be allowed access to the network until we force another registration. This can be done when we flush the registrations each semester or if we discover the computer is roaming for an extended period from its original location[10]. Since many students relocate each semester, we plan to force them to re-register each semester[11].

These features put the power of managing the registrations into the hands of the people who need it, without having them editing configuration files by

[10]We have a simple script that provides a list of computers roaming from the network segment they registered on.

[11]We may relax this policy to just require them to re-register at the beginning of Summer and Fall since that is when the majority of students change rooms.

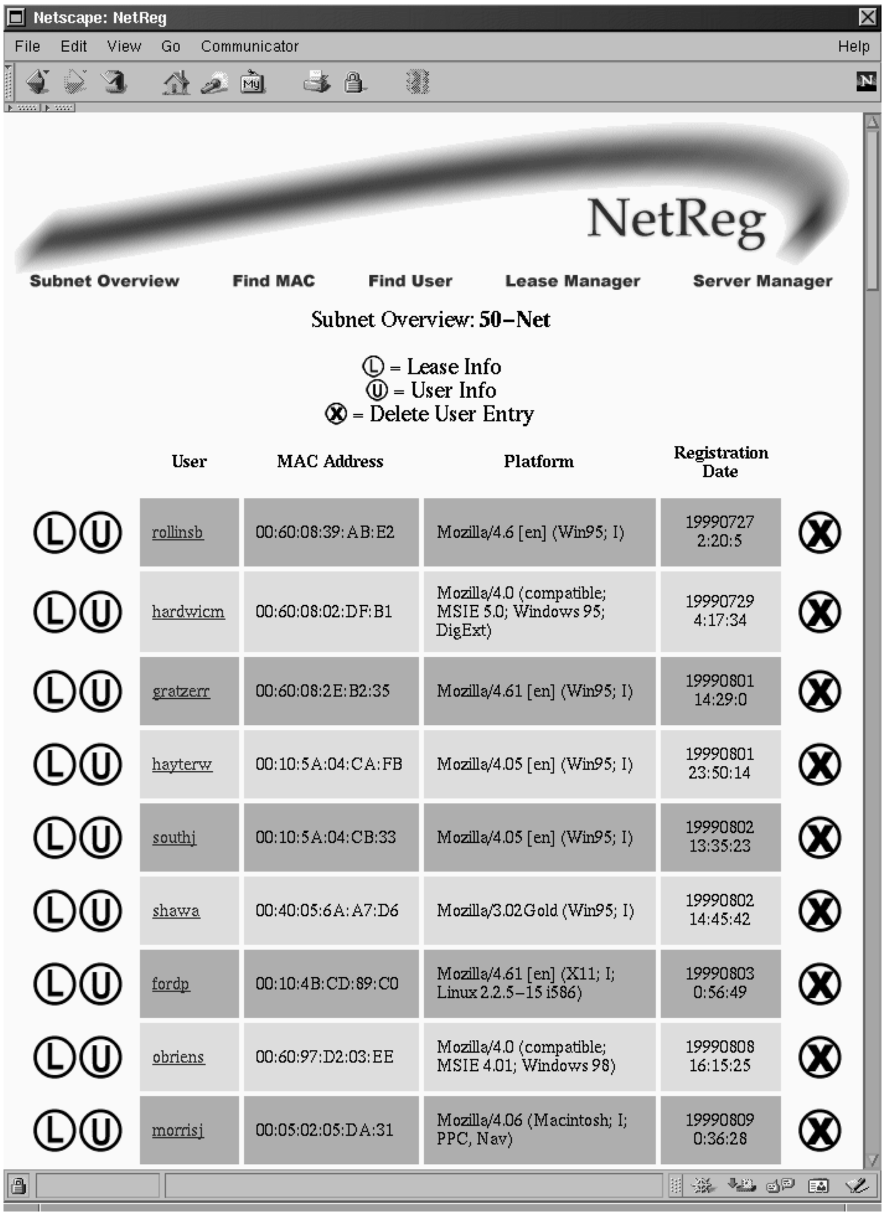

Figure 5: A View of Registrations on a Particular Subnet

hand on a system that they don't need administrative access to. Of course, due to the obvious security issues, entry to the admin site is protected by a two-layer[12] *htaccess* file [1].

Pitfalls and Lessons Learned

Even though the development of NetReg went very fast and was fairly uneventful, we did learn a few lessons which we should share. It is not a great idea to be setting up "root-level" authoritative DNS servers on a whim on your network, but there are some safety tips for doing so. We used the "Bogus Name Server" feature of BIND [2] to tell our legitimate campus DNS servers to not communicate with our NetReg DNS server.

We hoped to not have to require the client to reboot for the new lease to be obtained. In actuality, we were able to get the clients to obtain their new lease without rebooting using the network control panels on the various OS's. This did not appeal to us since the instructions for this process varied greatly based on the platform, so we opted to use the simple process of rebooting. There was another reason too. NetReg checks for new registrations with cron each minute so there is a possibility that a client who registers at one second past the minute will have to wait as much as 59 seconds until their registration entry appears (or they will obtain another temporary "unknown client" lease). At first we planned to have the actual registration CGI program just restart the DHCP server with the new registered client in the configuration file, but this created security issues we did not want to address. The problem is that the CGI program runs as the non-privileged UID of the web server, whereas the DHCP server runs as 'root'. We therefore chose to just have cron check for the new registrations. Because cron has a lower limit of one minute on how often it will run, and because boot times are getting quicker and quicker, we may have to re-address this in the future. For now the solution is that if the client is rebooted before the server reloads[13], they will be taken back to the registration page which will tell them they are already registered and should reboot their computer (again).

Another lesson we learned was about how browsers cache pages. Since the browser's default homepage gets redirected to the NetReg registration page the first time a client boots on the network, it caches the registration page for that address. Once the machine is registered and reboots, the browser reloads the cached version of registration page for the default homepage, which is not correct. Browsers should respond properly by using the "reload" button, however, our tests showed they didn't always do so. We remedied this problem by including two META tags in the header of the HTML of the registration page:

```
<META HTTP-EQUIV="Expires"
                        Content="0">
<META HTTP-EQUIV="Pragma"
                   Content="no-cache">
```

These tags seem to fix the reload problem on Netscape Communicator and MS Internet Explorer[14].

One pitfall of our current configuration is that the NetReg system runs on a single system. This single point of failure could be disastrous for extremely large sites with very short lease times and high rate of registrations. We plan to eliminate the possibility of failure by implementing multiple servers when the ISC DHCP server supports multiple server configurations. Until then, we have our previous DHCP server configured for operation and only need to load a different configuration on our router to put it back into operation taking over for the NetReg system should a fatal error occur. Also, being a low-end Pentium system running a free OS, we can easily have a replacement up in a short time period.

Security

Since much of the intention of NetReg was to improve security (mostly the management of security-related data), we should mention a few security-related issues we have and have not addressed. One of the more recent hot discussion items in the security realm is "MAC Spoofing." This is a process where a user modifies the MAC address of their ethernet card. This type of change could lead to some interesting events, such as denial of registration for another machine with the same (but valid) MAC address, or denial of service attacks by being given a duplicate IP address by the DHCP server. Also, with this technique one could exhaust all available IP addresses from the server. These such instances could be quite troublesome for a network administrator to track down. These types of attacks could exist with nearly any DHCP system however. Unfortunately, there is currently no known method[15] for the DHCP server to identify a conflict in MAC addresses (since theoretically they are supposed to be unique). It is even questionable whether this is something that servers should be able to test as opposed to the network routers and switches.

On a similar note, what happens if someone decides to not follow our rules and alters their computer's network configuration? For instance, we know it is not difficult to just enter an IP address. A relatively computer literate individual can learn your network topology and addressing scheme without too much difficulty and time. How do you deal with those

[12] We use both **Limit** and **AuthConfig**.

[13] not extremely likely since they would have to have a fast boot time and have registered very close to the start of the minute

[14] We haven't had an opportunity to test all versions of browsers on all platforms. We just picked what was "common" in our environment.

[15] Known to the authors who have kept up with the DHCP server developments fairly well.

individuals? We have a fairly crude way of handling this. We wrote a script that will find ''rogue'' addresses by regularly comparing the IP address leases of registered MAC addresses with MAC addresses that appear in our router's ARP table. We do this with SNMP and Perl in our script which we appropriately named *findrogue*.

Another thing a ''belligerent'' person may try in order to bypass NetReg would be to enter a valid DNS server (instead of our ''fake-root'' server). This can be deterred by using non-routable (or ''internally routable'' only) IP addresses for the temporary leases. Configured this way, the router will discard any packets not destined for the NetReg server.

There are undoubtedly numerous ''tricks'' that one can come up with to wreak havoc on our system; however, we designed it with functionality in mind more than being a highly secure system. We have developed utilities such as *findrogue* to help us find any users attempting to bypass it, but it is certainly not a system that an extremely security-conscience organization would want to implement without better studying the security issues.

Wish List

When we decided to create NetReg, we had only a couple of months of non-dedicated time to put toward developing the system. We had a set of features that had to be included, but in our opinion there is much left to be done. We hope NetReg will gain features as we, along with other users, explore the potential of this system.

The first things we'll mention are those which should come as a result of development on the servers which we utilize.

1. Support for multiple DHCP servers for redundancy and load-balancing via the ''DHCP Failover Protocol'' [4]. This is a feature to be included in version 3.1 of ISC's server, but has not yet been implemented.
2. Dynamic DNS integrated with DHCP. This is also expected soon with the ISC server[16].
3. Better/Smarter DHCP clients are sure to come. ISC is working on implementing authentication into their DHCP clients and server. That, along with general improvements on properly releasing and renewing leases, will help people in all areas using DHCP.

The remaining items are things which we may possibly implement or hope that others will contribute.

1. We built NetReg's authentication methods around two common servers (POP or FTP), but we would like to add modules for other more secure and appropriate methods. Obvious alternatives would be to use a Network Authentication Server (NAS), such as Tacacs+ and Radius, or Kerberos.
2. As our campus network becomes fully switched to the desktop, it could be beneficial to have the NetReg server use SNMP to make dynamic VLAN assignments to ports on switches based on the profile of an authenticated user.

[16]Check the ISC web site (*http://www.isc.org/*) for the latest status of this server.

Summary

As stated, we developed NetReg for our specific needs; however, we knew many others were looking for a solution similar to ours. It is our hope that NetReg will help those who are looking for simple DHCP management. We have done our best to keep portability in mind during development. The code was written in a very modular format with hope that others will contribute better modules and keep NetReg in an ever-developing state. A handful of other universities have put NetReg into production on their network. We hope that it works well for them, and also hope others will find it useful as that was our intent on making it available to the public.

Availability

NetReg has been released under the GNU Public License (GPL) and is available at: http://www.southwestern.edu/ITS/netreg/ . Also, discussion and announcement mailing lists have been created. Information about subscribing to them is available at the above mentioned URL.

Acknowledgements

We would like to thank Bob Paver for both funding and allowing us to pursue this project. Thanks to Rich Graves for pointing out the obvious! Thanks to Pat, Rich, Sharon, and Traci for putting up with us and the phone calls! We would also like to thank Bob Horick for his listening ear and suggestions from the first ideas of this project.

Peter would like to thank his advisor (and friend) Dr. Walter M. Potter. . .thanks for always believing in me.

Todd would like to especially thank Stephanie, Nathan, and Jacob for accepting his absence (physically and mentally).

Author Information

Peter Valian <valianp@southwestern.edu> is currently an undergraduate at Southwestern University majoring in Computer Science with graduation expected May 2000. When he's not studying or sleeping, he gets to play with cool networking toys and will hopefully develop a script or two to simplify the administration of those toys. Peter's interests are network administration, network security and making Perl do cool things. You can learn more about Peter at http://www.southwestern.edu/~valianp/ .

Todd K. Watson <tkw@southwestern.edu> is the "Systems and Network Administrator" for Southwestern University since 1997. At Southwestern he has the privilege of working with some really talented people and faculty. Previously he was pursuing the life of a poor Astronomer. He also worked for a short period at a really cool place called the International Institute of Theoretical and Applied Physics located at Iowa State University. His interests are in Astronomy, programming, systems security, and Unix/Linux advocacy. He has successfully avoided running NT on any system that has any importance. His almost always out-dated homepage can be viewed at: http://www.southwestern.edu/~tkw/ .

Bibliography

[1] Ben Laurie & Peter Laurie. *Apache: The Definitive Guide*, Second Edition. Sebastopol, CA: O'Reilly & Associates, Inc. 1999.

[2] Paul Albitz & Cricket Liu. *DNS and BIND*, Third Edition. Sebastopol, CA: O'Reilly & Associates, Inc. 1998.

[3] Ellen Siever & David Futato. *Perl Module Reference*, Volume I. Sebastopol, CA: O'Reilly & Associates, Inc. 1997.

[4] Internet Engineering Task Force (IETF). *DHCP-Failover Protocol*, Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-dhc-failover-04.txt . expires Dec. 1999.

[5] Internet Engineering Task Force (IETF), *Dynamic Host Configuration Protocol*, Request for Comments (RFC) 2131, http://www.ietf.org/rfc/rfc2131.txt, 1997.

[6] *The Apache SSL Project*, http://www.apache-ssl.org/ .

[7] *The Open SSL Project*, http://www.openssl.org/ .