autoMAC: A Tool for Automating Network Moves, Adds, and Changes

Christopher J. Tengi – Princeton University
James M. Roberts – Tufts University
Joseph R. Crouthamel, Chris M. Miller, and Christopher M. Sanchez – Princeton University

ABSTRACT

It is often difficult and time-consuming to manage computer 'moves, adds, and changes' that take place in a switched, subnetted environment. It is even more difficult when the accepted network policy requires that a computer be configured and always connected to the network on a specific Virtual LAN (VLAN) based on its usage. An on-going problem is to keep all of the network host information up-to-date, and to ensure that hosts always land on the correct subnet when they are plugged into switch ports. Utilizing some freely-available tools, as well as some home-grown software, we have built a system that automates a number of the tasks associated with moves, adds, and changes.

Where We Were

The Department of Computer Science at Princeton University has a routed IP network of 1500+ hosts, with a VLAN assigned to each subnet. IP addresses are assigned either from Princeton's address space or from any of a number of private IP address blocks, as defined in RFC 1918.[1] Hosts with private IP addresses are subject to Network Address Translation (NAT) by a CheckPoint firewall before any of their network traffic is sent out to the rest of the campus or the Internet. All intra-departmental routing is handled by a Foundry FastIron 1500 ethernet switch.

The department has a role-based network model, meaning that a host's place on the network is determined by its function and who uses it. Each host has a statically-assigned IP address tied to its ethernet MAC address. The network model specifies how hosts should be mapped to IP subnets, and how subnets correspond to VLANs. A faculty member's office machine belongs on the faculty-office VLAN. A host used by a member of the administrative staff belongs on the admin-staff VLAN. A graduate student's office machine belongs on the grad-office VLAN, while a host used in that graduate student's research belongs on a VLAN specific to the project. Figure 1 shows VLAN assignments for some of the roles a host might fit into.

[1] Address Allocation for Private Internets

Note that the main purpose of our network model is to segregate hosts by usage. In general, there are no restrictions on traffic between IP subnets. However, as some of the subnets are in private IP address space, access to hosts on those subnets is restricted from outside of the CS department.

As mentioned above, routing within the CS department is handled by a Foundry FastIron 1500 ethernet switch, which is at the core of our network. A number of other switches, from various vendors, are attached to it. They are configured as either "infrastructure" switches or "user" switches. The infrastructure switches are used for connections to servers and other network devices, such as printers, IEEE 802.11b [7] access points and network cameras. Infrastructure switch ports are in physically secure areas, preventing users from connecting to them. The user switches are used for network connections to end-user hosts. Most of the time spent dealing with ports is devoted to user switch ports.

There are a number of routine tasks involved in managing this network, including: host registration, maintaining compliance with the network model, and management of physical network connectivity. Prior to the implementation of autoMAC, these tasks required a lot of time and intervention by the Computer Science technical staff.

Class	Membership	Subnet	VLAN
Printers	printers	192.168.xx.0/24	19xx
Devices	cameras, env. monitors	192.168.yy.0/24	19yy
Faculty	faculty office hosts	128.112.aa.0/24	aa
Grads	graduate student office hosts	128.112.bb.0/23	bb
Graphics	graphics lab hosts	192.168.zz.0/24	19zz
PlanetLab	PlanetLab nodes	128.112.ccc.64/26	ccc1

Figure 1: Example network model host roles.

Host registration was accomplished using our inventory/host database system that was built around a Berkeley DB file, processed by a number of perl scripts. Hosts were entered into the system by a script that called 'vi,' allowing edits to be made on a template host entry. The same script allowed changes to be made to existing entries. Since the data entry environment used a simple text editor, it was difficult to assure accuracy and consistency of the entries. Once the required changes had been made to the DB file, DNS, DHCP, and NIS configuration files were generated and installed by using the 'make' command. Note that all of the preceding steps were carried out by a member of the technical staff.

In the process of entering or changing a host entry, it was necessary to examine the supplied information and make a decision about which VLAN the host should be assigned to. One problem with this method is that, given the same information, different technical staff members sometimes made different decisions, resulting in hosts with the same roles being assigned to different VLANs. Consequently, some hosts ended up where they didn't belong.

Port configuration changes on the switches required an administrator to connect to the appropriate switch and to change its settings manually. If needed, a patch cable would be installed to connect the switch port to an office wall box. When a user moved a machine from one location to another, it was very likely that either a change in the switch configuration or change on the patch panel (or both) would be required.

The process of registering a host and getting it connected to the network, or moving it to a different VLAN because its role in the network model changed, frequently involved multiple email exchanges between the requesting user and the technical staff. Often, the initial request would be something like ''Please register my new computer on the network with the name mongo.'' As this was not *nearly* enough information to process the request, we would send a reply asking for the rest of the information we needed. It might take several rounds of email replies to get all of the information needed to register the host and to activate a wall box jack.

Including the time spent awaiting additional information from the user, it could be hours from when the initial request was received before a new host could be used on the network. The only semi-automated task was the generation and installation of the configuration files.

Another time consuming task was isolating a host that was the source of a network problem. It was necessary to locate which switch port it was using and to reconfigure the port manually. Some local command-line tools were available to assist in identifying the problem host, but it was still necessary to connect to the appropriate switch and reconfigure the port in order to move the host to a different subnet for isolation or testing.

While packages are available to manage host registration (such as NetReg [11, 10]) and switch configuration (such as Splat [2]), no freely available package ties everything together and automates all of the tasks our staff is required to do in order to manage how hosts connect to the network.

What We Did

The system we envisioned and then implemented allows users to register new hosts using a web interface, with minimal intervention by the technical staff. The request is processed automatically by a daemon, and the new host is typically available for use on the network within 10-15 minutes. The host can be connected to any ''public'' ethernet jack and will automatically be placed on the proper VLAN. Public ethernet jacks include those in open spaces as well as departmental offices and labs. In general, any ethernet jack that is accessible to people outside of the technical staff is considered public.

Access to the web form is restricted to members of the CS department by requiring that they enter their departmental username and password. The form presents the user with questions to determine the proper class and VLAN of the new host. Infrastructure VLANs used by secured service hosts are not available on the switch ports the users can access. In addition, we are in the process of implementing an enhancement that will offer to each user in the department only their allowed user classes.

The idea of a web interface for host registration is not new. What makes autoMAC different is that it enables automated control of how a host is connected to the network *after* registration.

Automatic VLAN Assignment

A key feature of the autoMAC system is that a user can plug their host into any public ethernet jack attached to one of our user switches, and it will be automatically connected to the correct VLAN. This is accomplished using a RADIUS [6] server that ''authenticates'' the MAC address of the host attempting to connect to the network.

A number of web-based and IEEE 802.1X [9] authentication/authorization systems exist that require information to be manually entered or supplicant software to be installed on a host before it can access the network. These systems are well suited for authorizing user access to a network, but are not designed to work with unattended devices, such as printers or network cameras. In addition, requiring our users to install IEEE 802.1X supplicant software before they access the network is not practical, given a stream of frequent, temporary visitors. Ideally, for our purposes, it should be possible to connect any type of ethernet

device to any switch port and have it placed on the correct VLAN without any additional human interaction. Using a MAC address to identify a host, rather than a name and password to identify a user, makes this possible. Our users are still required to authenticate with their username and password to access departmental servers.

The RADIUS server configuration is built from data in our host database. For each MAC address, a "user" entry is generated, using the MAC address as both the username and password. The entries contain additional tags that specify to which VLAN a host belongs.

A host whose MAC address is not listed in the host database is considered unknown, and is therefore not listed in the RADIUS configuration file. When a user switch makes an authentication request with an unknown address, the RADIUS server responds in the negative. The switch then sets the port to which the host is connected to a registration VLAN, where the NetReg server presents our web-based host registration interface to the user when a web browser is started.

If a user moves a host from one wall box jack to another anywhere in the Computer Science building, the host stays on the correct VLAN since the switch port behind it is automatically configured using information provided by the RADIUS server. To move a host to a different VLAN, we simply need to change its IP address in the host database to an address on the other VLAN and rebuild the configuration files. The next time the host is connected to any switch port in the network, it is automatically placed on the new VLAN. If we need to force the issue, we can use tools we have developed to identify the switch port currently being used by the host, and turn the port off and back on. This forces the switch to re-authenticate the host the next time the host sends out any network traffic, causing it to be moved to the new VLAN.

Implementation Details

This system was not built from scratch. Rather, it was constructed by modifying our existing tools, and combining them with a number of open source packages available on the Internet. Specifically, we used FreeRADIUS [5] and NetReg, in addition to our existing host database and new code written to work with our web server and the above packages.

As mentioned earlier, our host database system is built around a Berkeley DB file. The vi-based data entry interface allows far too much latitude in what may be entered for a given field, making field validation difficult. However, the system is implemented with perl scripts that use a common library of functions to manipulate the data in the DB file. Having this library makes it fairly easy to implement new functionality such as bulk or daemon-based data entry.

The development of autoMAC began in July 2003, when we decided to implement a FreeRADIUS server to control access to our IEEE 802.11b wireless infrastructure. The Cisco access points (APs) we were using could be configured to query a RADIUS server using a MAC address for the username and password. The AP would then either allow or deny association of the client machine based on the response of the RADIUS server. While we realize that this is not strong security, it did, along with turning off broadcast of the SSID, prevent casual association with our wireless infrastructure.

While we were working on the FreeRADIUS server, we checked to see what other pieces of our network infrastructure might be able to make use of a RADIUS server. We discovered that the Foundry switches could utilize a RADIUS server to authenticate users, and configure their ports to specific VLANs. If this functionality could be extended to use MAC addresses for usernames and passwords, it would give us a *very* useful tool.

We spoke to a number of ethernet switch vendors to express our interest in MAC-based "authentication" for their switches, and received positive responses from several of them. As our user switches are from Foundry, we encouraged them to add this feature to their FastIron line, which they did. We have subsequently learned that several other switch vendors also now offer this feature, or will be offering it soon.

One of our goals for this project was to simplify the task of host registration for both our user base and the systems staff. To this end, we investigated two web-based host registration systems: Southwestern University's NetReg [11] and Carnegie Mellon University's NetReg/NetMon [10]. Both of these systems had features that we liked, but they also did much more than we needed. In the end, we built our own NetReg, utilizing some of the code from Southwestern's system which they graciously released under the GNU GPL [3].

Our NetReg implementation consists of a Linux system running the Internet Systems Consortium's BIND (DNS) and DHCP software, configured according to the instructions provided with Southwestern's NetReg system. The Linux system is also running the Apache Web Server, configured to use SSL and serve our custom registration page. The machine is connected to a dedicated registration VLAN as well as one of our production infrastructure VLANs. The infrastructure connection allows communication with our existing host database system. The machine uses an 'ipchains' firewall to limit vulnerability to attack from the registration VLAN and has routing disabled to prevent traffic from leaving the VLAN.

Tying Things Together

Once all of the parts of the puzzle had been identified, we needed to piece them together into a usable, cohesive system. We had a host database that associated

MAC addresses with IP addresses. We wanted a web-based system to add entries to the database. We had switches that could set ports to VLANs based on data from a RADIUS server, and we had a RADIUS server that could send the data. Now we needed to write some code to tie the components together.

Entering data in a text editor, based on email received from users, was an error-prone process. It was also a job we were looking to get out of. A web-based interface with field validation seemed to be a very good idea for a replacement.

The front-end we implemented allows users and support staff to easily enter all of the information required to register a new host and submit the request. Users of this interface are required to authenticate themselves with the same username and password used for connecting to our Solaris and Linux systems, as well as our e-mail server. We use SSL encryption to prevent user credentials from traveling over the network in the clear. All of the web pages are generated using PHP [4] scripts.

Figure 2 shows the host registration form. Extensive field validation is done to prevent duplicate entries and to ensure compliance with our network model. When the user submits the form, the request is sent to a new-host daemon that does the actual host registration and configuration file generation. The request is sent as a specially-formatted email message on behalf of the user filling in the form. The daemon parses the message body and invokes a perl script to update the host database DB file. It then runs a 'make' to generate and install all of our configuration files.

RFC 2868[2] defines RADIUS attributes to be used for "compulsory tunneling in dial-up networks." These attributes are leveraged by IEEE 802.1X (in Annex D) and RFC 3580,[3] which specify additional tunnel attribute information that can be used to assign VLANs to hosts being authenticated by a RADIUS server. In a VLAN environment, these attributes are returned as

[2]RADIUS Attributes for Tunnel Protocol Support
[3]IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines

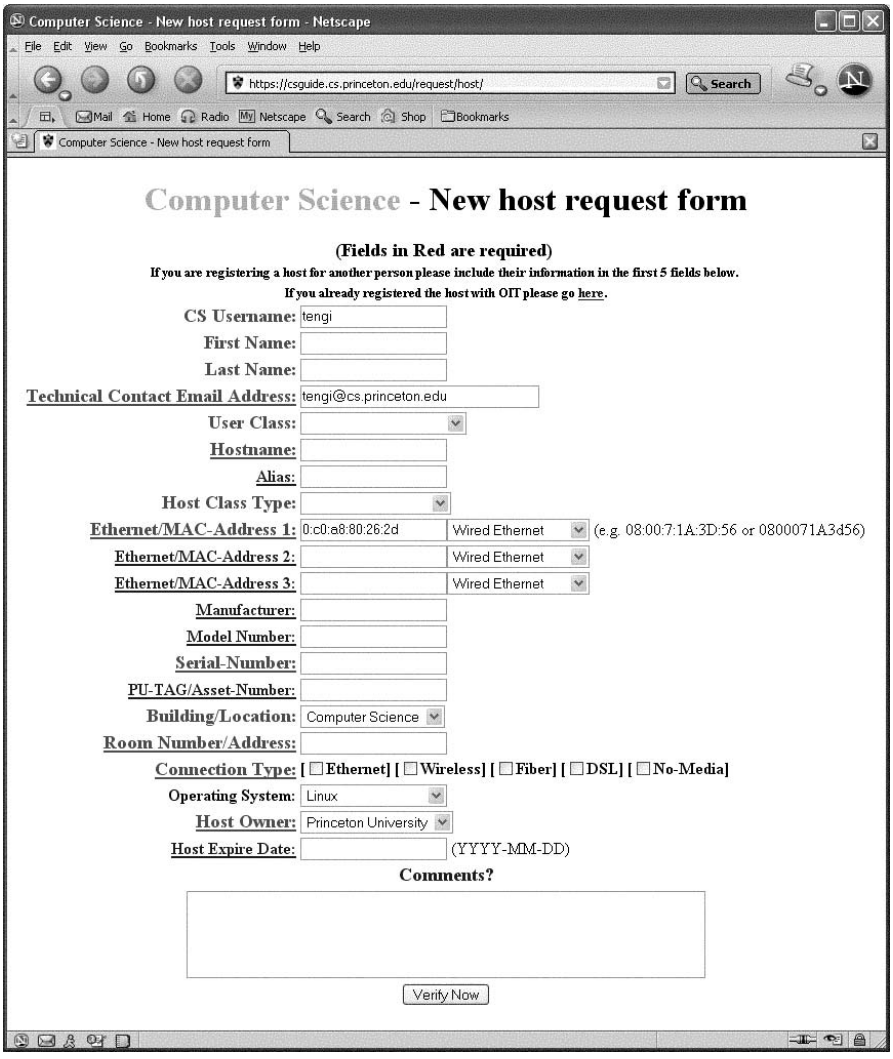

Figure 2: Host request form with default information filled in.

part of the 'Access-Accept' message from the RADIUS server when access is granted to a host, and can be used by the switch to configure a port's VLAN.

We generate a 'users' file for the RADIUS server that contains entries for every host registered in our database. There is a VLAN-based look-aside table used in this process to determine whether or not tunnel attributes should be added to the entry. Figure 3 shows an example RADIUS user entry with tunnel attributes while Figure 4 shows one without.

One of the VLANs that is not in the look-aside table is the one we use for 802.11b wireless access. Therefore, a host registered for wireless access would be listed as in Figure 4, without tunnel attributes. This is the type of entry we started with when we first implemented the RADIUS server for use with our 802.11b Access Points.

When a host is connected to one of our user switch ports, the switch sends an Access-Request message to the RADIUS server with the host's MAC address as the username and password. The RADIUS server returns either an Access-Accept or an Access-Reject message to the switch, based on whether or not the MAC address is known. If an Access-Reject is returned, the switch sets the port to the registration VLAN so that the user may register the machine using NetReg. The switch will also set the port to the registration VLAN if an Access-Accept is received without any VLAN information included. On the other hand, if the RADIUS server recognizes the address and has VLAN information for that address, it returns an Access-Accept message with the VLAN information which the switch then uses to set the port to the correct VLAN.

When a host is connected to the registration VLAN, its DHCP request is answered by the NetReg server. The reply specifies the NetReg server itself as the router for the host to use. Therefore, any subsequent IP traffic from the host will be sent to the NetReg server. The only other devices with IP addresses on this VLAN are other hosts awaiting registration, so it is unlikely that any network traffic will leak out from this VLAN to the rest of the campus or the Internet.

The DHCP reply also specifies the NetReg server as the DNS server to use. The DNS daemon code is configured to return the NetReg server's IP address for any lookup received. This means that hosts on the registration VLAN will be directed to the NetReg server for any web sites they may try to contact.

The Apache HTTP daemon is configured to present a login page for any unknown URL. This login page tells the user that they are using an unregistered host, and prompts them for their Computer Science username and password so that they may register the machine. This limits requests to members of the CS department. If someone is visiting the department for a few days and wants to use their laptop on the network, it needs to be registered by the member of the department they are visiting.

Upon successful authentication, the user is redirected to the host registration form shown in Figure 2. Submitting the form sends the registration request to the new-host daemon and displays a page to the user, stating that the registration request is being processed and instructing the user to disconnect from the network port and reconnect in ten minutes. When the host is disconnected or rebooted, the RADIUS authentication process starts all over again. This effectively removes a host from the registration VLAN once it has been successfully registered.

The email address to which the registration information is mailed uses procmail [12] to validate the message and save it in a spool directory. The new-host daemon, mentioned earlier, processes messages from the spool directory and adds the new hosts to the host database. The daemon then runs the 'make' command to generate and install all of the required configuration files.

An Example Scenario

Here then is an example scenario, showing how all of the parts operate together. The assumptions are that a Computer Science user wants to register a new host on the network; that the host will try to get an IP address with DHCP; and that the host will try more than once to obtain an IP address. Switch ports are configured to be on an unused, isolated VLAN until the switch receives information from the RADIUS server. Note that the level of detail in the following list items will vary, to prevent us from getting bogged down in low level details.

- The user connects the new host to a public ethernet jack, which is connected to a port on one of our ''user'' ethernet switches, and turns on power to the machine. The machine attempts to obtain an IP address using DHCP.
- The ethernet switch blocks the DHCP request, extracts the host's MAC address from the request

```
00d0b7b600ee    Auth-Type := Local, User-Password == "00d0b7b600ee"
        Tunnel-Type = VLAN,
        Tunnel-Medium-type = 802,
        Tunnel-Private-group-ID = 702,
        Fall-Through = Yes
```

Figure 3: RADIUS user entry with tunnel attributes.

```
00c04f7f1006    Auth-Type := Local, User-Password == "00c04f7f1006"
        Fall-Through = Yes
```

Figure 4: RADIUS user entry without tunnel attributes.

and uses the MAC address in an Access-Request message to the RADIUS server.
- Since this is a new host, the RADIUS server's configuration files do not list the MAC address, so it returns an Access-Reject message to the switch, which then configures the port to be on the registration VLAN.
- Because the switch blocked the host's first DHCP request, the DHCP client process on the host times out and sends another request.
- The NetReg server on the registration VLAN receives the DHCP request from the host and responds with a short-lived IP address on the registration subnet, as well as default router and DNS server addresses pointing to the NetReg server. The host uses this information to configure its network interface.
- The user starts a web browser on the host and attempts to connect to some web site. This causes the host to send a DNS query for the host named in the URL. We assume that the URL will be for 'HTTP' or 'HTTPS,' and that the host portion will contain a domain name, and not an IP address. The DNS daemon on the NetReg server resolves the domain name portion of the URL to its own IP address, and sends the address back to the host. If an IP address is used, the HTTP request will time out, as the request will not be forwarded off of the registration VLAN.
- The web browser on the host now makes a connection to the HTTP daemon on the NetReg server, requesting whatever page was specified in the URL. The HTTP daemon responds with an information page notifying the user that he is using an unregistered host. The page includes a link to the host registration web page.
- The user clicks on the link, and is prompted for their CS username and password by the HTTP daemon. If they enter valid credentials, the HTTP daemon sends the host registration form to the host. Otherwise, an ''access denied'' message is sent, and the user needs to try again.
- The user fills out the form, and clicks the ''Verify Now'' button. A CGI script on the NetReg server validates the entry data and, if all is well, sends a display-only page to the user with a ''Submit Now'' button at the bottom. Otherwise, it displays a partially completed registration form, with instructions on how to correct any errors.
- The user clicks on the ''Submit Now'' button, sending the form information to another CGI script which formats and sends an e-mail message, on behalf of the user, to the new-host daemon. The script also displays a page to the user, stating that the registration request has been sent and that the host should be rebooted in ten minutes.
- The new-host daemon parses the e-mail message, validates the information, and if all is well, selects an IP address and adds the host to the host database. The daemon then runs 'make' to generate and install the configuration files.
- After ten minutes, the user reboots his host. When the host reboots, it drops ethernet link, causing the switch to remove the port from the registration VLAN. The host then attempts to obtain an IP address using DHCP.
- The ethernet switch blocks the DHCP request, extracts the host's MAC address from the request and uses the MAC address in an Access-Request message to the RADIUS server.
- The RADIUS server now has the MAC addresses listed in its configuration files, along with VLAN information, so it sends an Access-Accept message, including the VLAN information, to the switch, which then configures the port to that VLAN.
- Because the switch blocked the host's first DHCP request, the DHCP client process on the host times out and sends another request.
- This time, the DHCP request is received by our production DHCP server, which responds with a reply containing the host's registered IP address, along with our normal DNS server and default router information.
- The user may now use his host normally on the CS network. All future reboots and DHCP requests from this host will result in the switch port to which it is connected being configured to the proper VLAN, as the MAC address is now known.

Security Considerations

While autoMAC limits user host connections to the appropriate VLAN (according to our role-based network model), we do not rely on the system to provide strong security. Intentional spoofing of MAC addresses is not addressed, as this possibility always exists.

Anti-spoofing ACLs on our core switch prevent hosts from pretending to originate from different subnets. As we mentioned earlier, users still have to authenticate with login name and password to access our departmental servers. Multiple attempts to register different MAC addresses from the same port within a short period of time could be logged and trigger a report.

If a host gets a new ethernet address due to a hardware change, the user will need to register the new ethernet address before he can use anything other than the registration VLAN. Ethernet card swaps between machines are most likely to occur within a user class (e.g., graduate student to graduate student, not graduate student to faculty), so the worst problem is one of host identification, not VLAN access. In the case of a graduate student swapping the ethernet cards of a research host and an office host, the machines will

end up on the wrong VLANs, and the student will no longer be able to use either machine effectively.

In the event that a host needs to be quarantined due to virus infection or other problems, it can easily be moved to an isolation VLAN. The only way for a user to circumvent that isolation would be to change the host's MAC address. If the new MAC address was unregistered, the host would be connected to the registration VLAN. If the MAC address of another registered host was used, the infected machine would be connected to the VLAN associated with the stolen MAC address. If the MAC address was already in use, the result would be two machines getting the same IP address from the DHCP server, and neither machine would work reliably. This would most likely trigger a complaint to the technical staff from the owner of the host whose MAC address had been stolen.

Future Enhancements

With our new system it is relatively easy to reconfigure hosts to be on different VLANs. As a result, we think it should not be too difficult to integrate this with other systems that scan automatically for active viruses and current patch levels before a registered host is allowed network access. In addition, infected or vulnerable hosts on the network can be quarantined, and transferred to isolated VLANs where the appropriate host patching tools are available.

According to the documentation, our Cisco 802.11b APs can also make use of the RADIUS tunnel tags to specify a VLAN for a wireless client. Adding tunnel tag support to the APs will allow us to implement a registration VLAN for wireless users as well as wired users. It will also allow us to segregate wireless traffic by host role, as we do for wired traffic. Currently, all wireless users are on the same VLAN.

Availability

Some of the tools we have developed are available to the public, so that others may implement similar systems. The host database system we use will not be made available, as we are in the process of replacing it. The software and some configuration examples can be found at http://www.CS.Princeton.EDU/autoMAC/ .

RFC documents can be found on the Internet Engineering Task Force web site http://www.ietf.org/ .

Conclusions

In a network environment such as ours, where hosts from different VLANs can be plugged into any public ethernet jack, it can be difficult and time-consuming to ensure that the switch ports behind those wall jacks are always on the correct VLAN for a given host. Using a number of different tools and some switch firmware features, we have put together a system that uses the host MAC address to automatically configure switch ports properly.

Combining automatic port configuration with web-based automated host registration allows users to start using new hosts on the network with little or no intervention by the technical staff. This means that if a user brings a new host into the department after normal business hours, he won't have to wait until the next day to start using it. When paper deadlines are looming, and our students need to add a few machines to the network in order to complete their experiments (at 3 a.m.), near-instant host registration is a big win.

By implementing this system, we have moved from a mostly manual to a mostly automated method of handling network moves, adds, and changes. This has improved response time to our users and also provided support staff with time to work on other projects. Additionally, we have improved compliance with our network model, by automating the switch-port-to-VLAN mapping. This project would not have been possible without the availability of a number of Open Source Software projects.

Author Biographies

Chris Tengi is a System and Network Administrator in the Princeton University Computer Science Department. His professional interests include network architecture and management, and just about any new networking technology. He can be reached at tengi@CS.Princeton.EDU .

James M. Roberts is the Director of Computing for the School of Engineering at Tufts University. He can be reached at jmr@cs.tufts.edu .

Joseph R. Crouthamel, Chris M. Miller, and Christopher M. Sanchez are System and Network Administrators in the Princeton University Computer Science Department. They can be reached, respectively at jrc@CS.Princeton.EDU , cmmiller@CS.Princeton.EDU , and cmsanche@CS.Princeton.EDU .

References

[1] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)," *Network Working Group Request for Comments 3575*, Internet Engineering Task Force, *Update of [6],* July 2003.

[2] Abrahamson, Cary, Michael Blodgett, Adam Kunen, Nathan Mueller, and David Parter, "Splat: A Network Switch/Port Configuration Management Tool," *Proceedings of the 17th Large Installation Systems Administration Conference (LISA-03)*, pp. 247-256, USENIX Association, October 2003.

[3] Free Software Foundation, *GNU General Public License*, http://www.gnu.org/licenses/licenses.html .

[4] The PHP Group, *PHP scripting language*, http://www.php.net/ .

[5] The FreeRADIUS Project, *FreeRADIUS Server Project*, http://www.freeradius.org/ .

[6] Rigney, C., S. Willens, A. Rubens, and W. Simpson, ''Remote Authentication Dial In User Service (RADIUS),'' *Network Working Group Request for Comments 2865*, Internet Engineering Task Force, *Updated by [13] and [1]*, June 2000.

[7] IEEE Computer Society, ''Higher-Speed Physical Layer Extension in the 2.4 GHz Band,'' *Std 802.11B-1999*, IEEE, *Supplement to [8]*, 1999.

[8] IEEE Computer Society, ''Part 11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, *Std 802.11*, IEEE, 1999.

[9] IEEE Computer Society, ''Port-Based Network Access Control,'' *Std 802.1X-2001*, IEEE, 2001.

[10] Carnegie Mellon University, *NetReg/NetMon*, http://www.net.cmu.edu/netreg/ .

[11] Valian, Peter and Todd Watson, ''NetReg: An Automated DHCP Registration System,'' *Proceedings of the 13th Conference on Systems Administration (LISA-99)*, pp. 139-148, USENIX Association, November 1999.

[12] van den Berg, Stephen R., Philip Guenther, et al., *procmail Mail Processing Suite*, http://www.procmail.org/ .

[13] Zorn, G., D. Leifer, A. Rubens, J. Shriver, M. Holdrege, and I. Goyret, ''RADIUS Attributes for Tunnel Protocol Support,'' *Network Working Group Request for Comments 2868*, Internet Engineering Task Force, *Update of [6]*, June 2000.