USENIX Association

Proceedings of the 17th Large Installation Systems Administration Conference

San Diego, CA, USA
October 26–31, 2003

Splat: A Network Switch/Port Configuration Management Tool

Cary Abrahamson, Michael Blodgett, Adam Kunen, Nathan Mueller, and David Parter –
University of Wisconsin – Madison

ABSTRACT

We present the design and implementation of Splat, a tool for managing network edge switch ports and port-to-host configurations. We discuss the need for such a tool as part of a major network upgrade, and our discovery that most existing tools ignore this area or approach it from a core network point of view.

Important design considerations include the current procedures and habits of our system administration team, using open source components, and the tool's incremental development and deployment. In particular, the requirement for accurate configuration information in a database proves to be an effective means of enforcing correct procedures.

Splat enables us to administer an increasingly complex network by providing a simple interface for routine tasks as well as better diagnostics, reporting, and monitoring. Its reliable configuration management enhances the security and performance of our network.

Introduction

Modern network switches continue to become more complex to configure and manage as functionality is moved from the network core to edge devices. As a result, administrators of large Local Area Networks (LANs) face significant challenges, particularly if they utilize features such as Virtual LANs (VLANs) or per host Quality of Service. Most existing network management systems do not focus on the edge devices of LANs. Instead, they are designed for network specialists administrating the core devices of Wide Area Networks (WANs). This has left many desktop and server administrators who are responsible for moves, adds, and changes involving edge devices to manually configure and maintain them because they lack adequate tools.

There are several well-known problems with manually configuring devices. First, the amount of time the network specialist must devote to reconfiguring switch ports for each workstation or server change is significant. This can also lead to bottlenecks as other administrators wait for each network reconfiguration to be completed. A large volume of manual changes also increases the probability of error with the resultant increase in downtime and troubleshooting.

In this paper, we discuss the design and implementation of Splat – a switch port configuration management tool.[1] Unlike most existing network management tools, Splat focuses on the configuration management of edge devices, applying a traditional system administration configuration management approach to this area.

Configuration Management

In recent years, large scale infrastructure configuration management has been a topic of interest to many system administrators [3, 4, 30, 15, 6]. Organizations are now deploying comprehensive configuration management systems for many areas of their infrastructure such as workstations, servers, and account management. In the traditional network management community, the emphasis has been on provisioning, bandwidth, core devices, inter-device links, and fault management [9, 24, 29, 23, 16].

Those who address the network management of LANs have tried to take a similar approach. As Kevin Dooley states in *Designing Large-Scale LANs* [10]:

> However, remember the physical tracking side of configuration management, especially if you deal with the configurations of Layer 2 devices such as hubs and switches. If network managers have accurate information about physical locations, MAC addresses, and cabling for end devices such as user workstations, then they can easily handle hardware moves, adds, and changes. In most organizations, business requirements force network administration to respond quickly and efficiently to requests for end-user moves and service changes. However, the cabling and hardware records are usually out-of-date, so every small move requires a technician to visit the site and carefully document the equipment and cabling. This process is expensive and slow.

Having out-of-date records is a problem, but not one that is unique to network management. System administrators deal with the same issue in designing and deploying configuration management systems for workstations and servers. As Dooley points out, if the data is not current it cannot be used. If it is not being used then overworked system administrators have no incentive to update the data. This chicken-and-egg situation is the bane of every system administrator trying to introduce system configuration management to a site.

[1] During design discussions, we needed a name for the system. The placeholder name "Splat" stuck.

The best way to break this cycle is to integrate a single definitive data source into operational tools. When accurate data is necessary for the operation of a site it reverses the situation. This poses a twofold challenge for the system designer: 1) Data updates must be as easy and painless as possible for the staff. 2) The benefits of the new system must be significant.

Dooley continues:

> Unfortunately, no software can solve this problem; it is primarily a procedural issue. Technicians making changes have to keep the records up-to-date, and the cabling and patch panels have to be periodically audited to ensure accuracy of the records.

We agree that this area of network management is primarily a procedural one. However, such procedures are best implemented using software tools that validate input data and ensure integrity through traditional system administration practices.

Background

The University of Wisconsin's Computer Sciences Department network consists of approximately 1500 computers using 2000 IP addresses on 50 routed subnets. All system administration tasks (including desktop support, software installation, account management, and network management) are handled by one group. This group has nine full-time staff and 12 part-time undergraduate students. There is some specialization in both the full-time staff and the student staff. But, everyone is expected to be able to perform some common tasks. For example, all of the student staff are expected to do routine tasks such as moving workstations from one office to another, which often involves IP address changes.

We previously had a small number of routers and a large number of relatively simple (and ''mostly unmanaged'') 100base-T ethernet switches with no VLANs. In that situation, it was reasonable to not use any network configuration tools since the only devices that needed ongoing configuration were the routers and firewalls, each of which was a special case. Router and firewall configurations were manually preserved using RCS [28].

Motivation and Requirements Identification

We began a major network infrastructure upgrade in the summer of 2002. All routers and the ''mostly unmanaged'' ethernet switches were to be replaced with new routers and switches.

The new network consists of three core routers and edge switches in the data center and edge switches in Intermediate Distribution Frames (IDFs)[2] throughout the building. Multiple routed networks enable traffic separation and functional grouping of computers. Each edge switch has redundant uplinks to the core routers. IEEE 802.1Q [17] VLANs are used to implement the separate routed networks on one set of equipment.

The previous network had only three managed devices; the new network would have over 50 devices that required active management. It became clear that we needed a switch configuration management tool.

Requirements

A switch configuration management tool must satisfy a number of requirements to be useful to our staff:

1. It must enable administrators with little networking experience to perform routine tasks without having to log into edge switches and make manual configuration changes. These tasks include adding, removing, and swapping workstations in offices throughout the department and making hardware changes such as replacing ethernet cards.
2. The tool should provide live port status and statistics from switches without requiring administrators to log into devices manually.
3. It should report reliable, up-to-date information on which host is connected to which switch port for troubleshooting and general auditing. Previously this would have been done by sending a staff person to an IDF and computer location to manually take an inventory of the switch ports, patch panels, and data jacks.
4. The tool should track network device configuration changes using a version control system such as CVS [7].
5. It must allow for different policies and configurations on different VLANs.
6. It must be possible to configure any VLAN in our network on any edge port of any switch.
7. The tool should be easy to integrate with our existing site configuration management infrastructure, and it must be scalable and extendable for additional features as our network continues to evolve.
8. Use of the tool should be easily integrated with existing staff procedures and habits, causing as little disruption as possible.

Additional Features

Although not strictly required, some additional features would enhance the functionality of the tool:

1. Allowing only a specified MAC address on a given switch port would enhance the security of the network.
2. The tool should automatically generate configuration files for other network tools.

Alternatives Considered

System administrators have a propensity for reinventing the wheel. Before creating a new system, we

[2] Network jargon for a wiring closet. More specifically, in a structured wiring system, the rack where end devices are cross-connected into the rest of the network.

looked for existing solutions. A number of proprietary and open source options were investigated.

As previously noted, most network management systems do not support the type of edge-device port configuration management we were looking for. Systems such as HP OpenView, IBM Tivoli NetView, and GxSNMP are best suited for management of WANs or campus core networks, not edge device ports. OpenNMS is primarily a fault and performance management system. We found that the LANdb project (''The Network Management Database'') [18] addresses similar areas as Splat, but it appears that development has been idle since July 2000.

We did not find any systems that met our requirements, so we decided to develop Splat.

Tool Design and Implementation

Splat is a command line tool, written in Perl [25], and a related set of Perl modules. It interfaces with a PostgreSQL [26] network configuration database using the Perl Database Interface module (DBI) [8]. It was implemented without any special PostgreSQL features and it would be easy to use a different relational database system in its place.

Splat interacts with switches using the RANCID [27] configuration tool. It has been tested and used on Linux and Solaris systems. Splat has not been tested on Windows.

Splat generates the appropriate switch configuration commands from templates and uses RANCID to apply those commands to the switches. RANCID then retrieves the complete configuration from the switch and stores it in a CVS repository (see Figure 1). This enables administrators to view and retrieve previous configuration versions.

Network Configuration Database

Managing site configuration information with a relational database has been a common strategy of system administrators for some time [12, 13, 14]. Splat applies this strategy in managing the edge devices of a network. Its network configuration database contains both static information (such as data jack locations) and dynamic information (such as current hostname/data-jack assignments). The details of the database design are covered in the next section.

Ideally, the network configuration database is part of a larger configuration management system, including a parts inventory of workstations components (for example, ethernet cards) and other information such as operating system and security policies. We import inventory data from our existing inventory database.

Sites without an inventory database need to enter ethernet card and hostname information into the Splat database. Splat includes scripts to initialize other information in the database (such as IDFs and jacknumbers) according to the naming conventions of the site.

Tables and Data Relationships

The major data tables and relationships are shown in Figure 2. The primary abstraction in Splat is the relationship between a particular host network interface and a particular data jack – ''where the host meets the network.'' It is entered in the *hosts* table in the Splat database. The computer's network adapter is represented as an inventory part number and interface number (to support multi-port network cards). The data jack is represented as an (IDF, jacknumber) pair. At our site, data jacks are numbered per IDF. Other naming schemes will work as long as the (IDF, jacknumber) pair is unique.

In the IDF, each switch port is patched to a data jack. This is represented in the *port_mapping* table. The *jacks* table reflects the building installed wire between the IDF patch panel and office data jack. If enough switch ports are available, the use of VLANs and trunking reduces the need for re-patching between switch ports and the IDF patch panel. To generate a *generic* port configuration script, Splat consults the *mac_address* and *vlans* database tables. Splat also needs the IP address, which could be stored in the configuration database or retrieved by a Domain Name System query on the hostname.

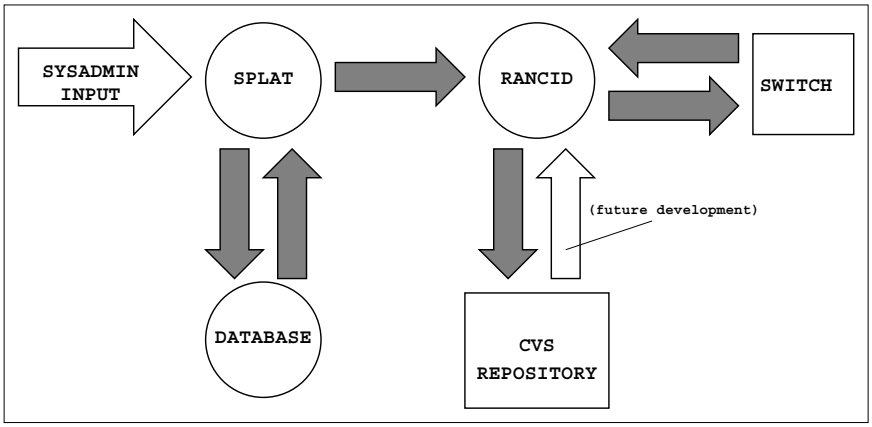

Figure 1: Splat processes and related components.

Additional database tables hold IDF data (names and locations), switch data (names, locations, and interface inventory), and ‘‘glue’’ to interface to our existing inventory database.

MAC Address Locking

On most of our production networks, the generated configuration includes commands to ‘‘lock’’ the switch port to the MAC address of the attached workstation. This serves as an integrity check on the configuration and also makes it more difficult for the casual ‘‘bandwidth borrower’’ to unplug a supported workstation and use the data jack for their laptop.

Not all switches feature the ability to lock a port to a specific MAC address. For those that do, the commands to configure MAC-locking vary from vendor to vendor. Splat can handle this by using different configuration templates for each vendor to generate command scripts to run on the appropriate switch.

Once the port is configured for only one MAC address, the switch handles violations depending on the switch and the specific configuration options used. In our case, the Cisco switches are configured to *restrict*, which means that packets from other MAC addresses are dropped, and SNMP traps and syslog messages are generated about the event (monitoring SNMP traps and syslog messages is outside the scope of Splat).

Generating a MAC-lock configuration uses the *mac_address* table, as shown in the middle and bottom of Figure 2.

Port Policies

Uplink ports use a different configuration template than edge-device ports. The port configuration does not use MAC address locking or assign a VLAN, but instead sets the port to *trunk* mode and DOT1Q encapsulation. Other port policies are *broken, reserved,* and *generic* (available for use).

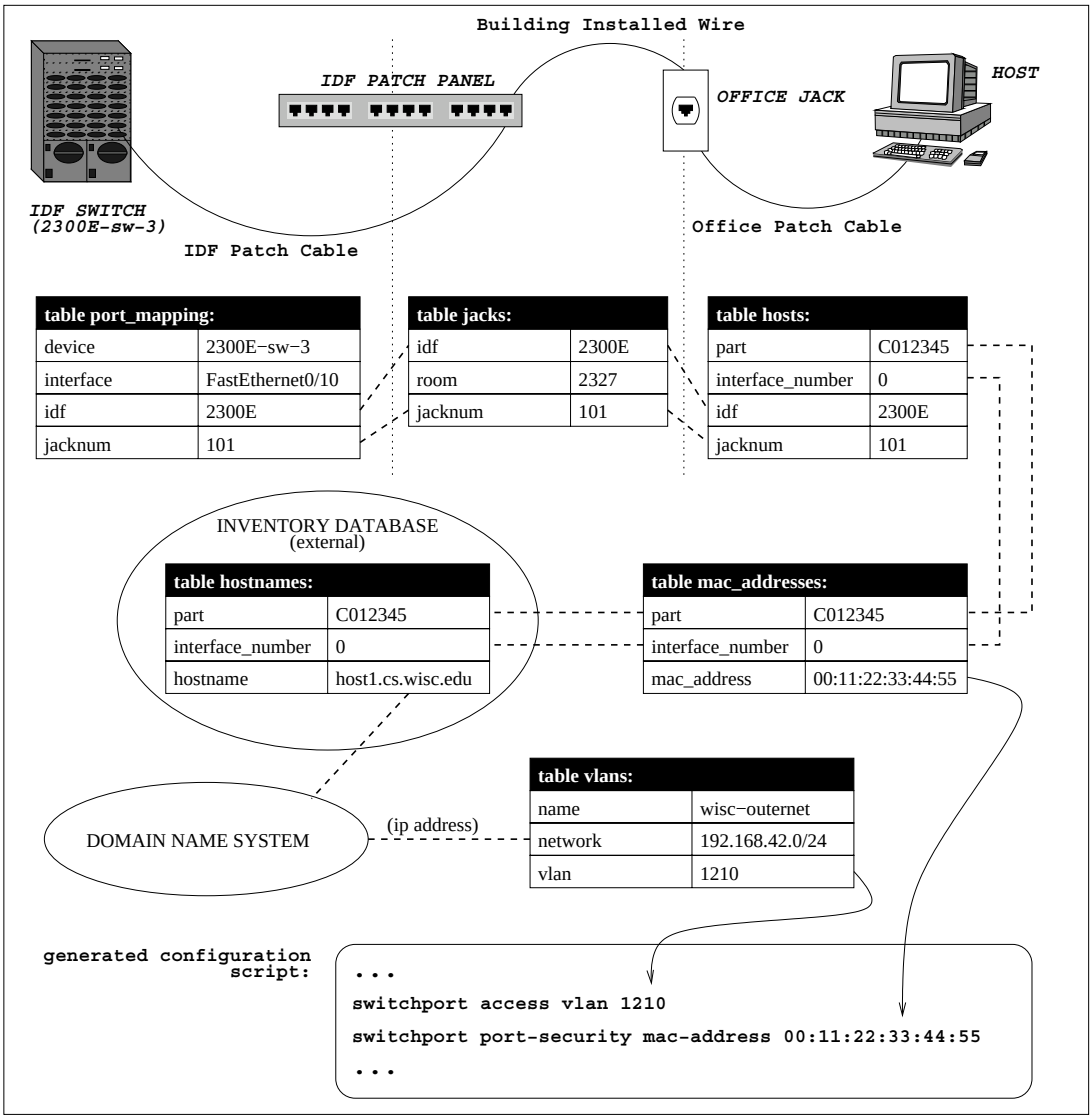

Figure 2: Data relationships.

Per-switch-interface (port) policies are in the *switch_interfaces* table (not shown in Figure 2).

VLAN Table

We use a *vlans* table instead of an IP address to VLAN scheme (such as using a ''subnet'' number) to eliminate any assumptions about IP network address allocation and avoid conflicts with reserved VLAN numbers. This is shown at the bottom of Figure 2.

VLAN Policies

Not all of our VLANs use the same configuration policy. For example, the ''laptop'' network is designated for laptop use. It is available in certain locations for transient laptops. The VLAN policy for the laptop network does not specify ''MAC-locking''.[3] Also, on general-purpose networks, data jacks that are not currently *attached* to a computer are disabled in the configuration. Laptop network public data jacks are enabled, although they are not *attached* to a specific computer.

VLAN policy assignments are in the *vlan_policy* table (not shown in Figure 2).

RANCID

Splat utilizes RANCID to securely communicate with network devices, execute device commands, and maintain device configurations with CVS. Although RANCID is the ''Really Awesome New Cisco conflg Differ,'' it actually supports a number of well known network devices in addition to Cisco switches running IOS [5]. As of RANCID version 2.2.2, this includes Bay routers, Juniper routers, Cisco Catalyst switches, Foundry switches, Redback NASs, ADC EZT3 muxes, MRTd, Alteon switches, and HP ProCurve switches. This flexibility is important as we do not know what devices we will need to support in the future.

To run commands on a device, Splat first generates a device configuration script from a template. The commands are then run on the switch via RANCID's *clogin* utility. Clogin is an Expect [19] script that automates the process of logging into devices using the facilities available on the device such as SSH [31].

For each device configuration change, RANCID makes a revision entry in CVS. It is important to note that we do not use CVS for classic revision control. While it is possible to load a previous configuration revision directly to a switch, the Splat database would not be synchronized. Instead, we use CVS revisions for logging changes and as a backup for disaster recovery if the switch needs to be reinstalled or completely reconfigured.

Using Splat

The Splat command syntax is described in Appendix A. Output from these commands are shown in Appendix B. The two most common commands are *attach* and *detach* to update which computer is connected to a particular data jack.

Desktop and Server Administrators

Splat enables desktop and server administrators to make network configuration changes that are needed when doing routine tasks such as moving or reconfiguring a computer without intervention from network specialists. When attaching a computer to a data jack, the system administrator does not need to know the MAC addresses or VLAN number, just the hostname and data jack number.

When replacing an ethernet card in a computer, the system administrator only needs to update the inventory database. Splat will see the new MAC address the next time the computer is *attached* to a data jack. The same applies when a computer changes hostname, is moved from one room to another, or is replaced by a new computer.

Information commands available to the system administrator can aid in preliminary diagnosis of host specific network problems.

Network Specialists

Compared to desktop and server administrators, network specialists often have a different view of the network and have different needs from a network management system.

Long-term and short-term planning tasks are served by the variety of information Splat makes available. For example, it is easy for network specialists to check the inventory of available switch ports and data jacks. When a research group decides to add additional computers to their desks, it is easy to check the availability of data jacks in their offices. When planning the budget, it is easy to get a site-wide inventory of available switch ports.

When troubleshooting network problems (short of total connectivity failure) it is helpful to have information about a computer's switch port connection, the switch port status and statistics, and the switch global status and statistics. The same holds true for all of the switches in the data path between the problem computer and the ''other'' end. Splat can present all of this information based on only one unique element of the configuration (typically either a hostname, data jack, or switch port). So, the network specialist does not have to untangle all of the different aspects of the configuration for that element.

We make extensive use of Cricket [1] for gathering and presenting network performance and utilization data, and NetSaint [22] for status/fault monitoring. Using the Splat database, we have automated the generation of configuration files for both[4]

[3] Laptop network authentication and authorization are not part of Splat. We use authipf [2].

[4] Note: Nagios [21] is the successor to NetSaint, and we will generate Nagios configuration files when we switch from NetSaint.

Deployment and Experiences

Splat was developed on a test network composed of a subset of the switch setup that was to be used department-wide. It was deployed one network at a time as existing switches were replaced. As each of the first few networks were converted, we assessed the process and made appropriate adjustments to the code.

There was some hesitation by the staff when they started using Splat. During the transitional phase, it was unclear which networks were under Splat configuration, and which were to be done "the old way." This was a communication problem, not a Splat problem.

Networks were converted to Splat by a handful of student staff (including, of course, the students who specialize in the network and had been involved in planning and designing Splat). They soon became very familiar with Splat, and were willing to work around any bugs they found (which were usually fixed within hours). The vast majority of the time the only command being used was *attach*, as workstations moved from the old (unmanaged) network to the new network with Splat.

Using Splat requires that the inventory database be correct and current. As can be expected, a number of data entry errors were found when Splat either refused to make a connection (no network part in the inventory) or configured the wrong MAC address (wrong part in the inventory). Once the staff became aware of the problem, these data errors were fairly easy to fix on a case-by-case basis.

The requirement that the inventory database be current also caused some issues, as staff who were not in the habit of updating the inventory in a timely manner were forced to make the update before being able to attach a computer to the network.

We experienced a few interesting events throughout the deployment. For example, during a rack cleanup in the data center, it was necessary to move the database server that included the Splat database from one network switch to another. It was immediately apparent that *detaching* the Splat database server was a mistake, as it was no longer possible to run the Splat commands necessary to re-*attach* it to the network.

Also, we had been aware for some time that a few rogue users had been occasionally disconnecting workstations from the network in order to use the data jacks for other computers (most likely laptops). We did not have adequate tools to catch them and it was not considered a major problem. But, it served as a catalyst for the MAC-locking feature. As soon as we deployed Splat with MAC-locking, a few MAC-lock violation alerts were generated, but no one ever complained or even asked about the change in functionality.

Additional Splat Tools and Future Development

We are continuing the development of Splat and related tools. An existing related tool is splat2cricket which generates configuration files for monitoring switches with Cricket. We are in the process of creating DNS [20] and DHCP [11] scripts that will automatically generate DNS and DHCP configuration files from the Splat network configuration database.

Needs we expect to address in the near future include multi-level access control, configuration of our core routers (in addition to edge switches), and better support for reinstalling a complete switch configuration.

Multi-level Access Control

Multi-level access control will be necessary in our department. At least one of our research projects needs the flexibility to frequently reconfigure several test networks without intervention from the system administration staff. Sites with a different system/network administration organizational model, or sharing administration of a network between different organizations, would also need it.

One approach is to control access based on the set (network device, VLAN, host). Splat, as currently implemented, directly accesses the database and configuration files. Database and file system access controls limit access to the system administration staff. Multi-level access control will require finer-grained controls than are provided by either, so it will have to be implemented in Splat.

Most likely, this will require a client-server implementation, with the server enforcing fine-grained access control. This may be the basis for web or other graphical user interfaces.

Core Routers

Our core routers are currently configured manually using RCS [11] to preserve configuration revisions. We would like to adapt Splat to manage these devices as well. This would require adding a new *router* port type to the *switch_interfaces* table and the code and templates to support it.

Reconfiguration

Currently, we can manually retrieve a switch configuration from the CVS repository when it is necessary to completely reconfigure a switch (for example, when a switch is replaced). We would like to add automated configuration recovery and initial switch configuration for new switches.

Availability

Splat is available for download from http://www.cs.wisc.edu/csl/projects/splat .

Conclusion

We presented a method for managing network edge switch ports and port-to-host configurations in the form of a relatively simple tool. Most network configuration management tools do not cover this area. Instead of relying on manual procedures, a better

way to approach this problem is by applying traditional configuration management techniques. In particular, the requirement for accurate configuration information in a database proves to be an effective means of enforcing correct procedures.

Considering the existing procedures of a site in the tool's design, and providing a simple interface, makes it possible for administrators of varying network experience to perform routine tasks. This tool enhances our ability to monitor and troubleshoot problems as our network continues to evolve.

Acknowledgements

This work would not have been possible without the help and support of the Computer Systems Lab staff.

Author Information

All of the authors work for the Computer Systems Lab at the University of Wisconsin Computer Sciences Department.

Cary Abrahamson is a System Administrator at the Computer Systems Lab. His professional interests include network management and security. You can reach him at cary@cs.wisc.edu .

David Parter is a Senior Systems Administrator and Associate Director of the Computer Systems Lab. He also serves on the SAGE Executive Committee, and was program chair for LISA '99. His professional interests include network systems, security, configuration management, and System Administration education. You can reach David at dparter@cs.wisc.edu .

Michael Blodgett is an undergraduate student and System Administrator with the student staff. Michael can be reached at mblodget@cs.wisc.edu .

Adam Kunen is an undergraduate student and System Administrator with the student staff. Adam can be reached at ajkunen@cs.wisc.edu .

Nathan Mueller is a System Administrator with the student staff who recently graduated with a degree in Computer Science from the University of Wisconsin. You can reach him at nate@cs.wisc.edu.

References

[1] Allen, Jeff R., "Driving by the Rear-View Mirror: Managing a Network with Cricket," *First Conference on Network Administration (NETA '99)*, pp. 1-10, Santa Clara, CA, USENIX, April 7-10, 1999.

[2] Beck, Robert, "Dealing with Public Ethernet Jacks – Switches, Gateways, and Authentication," *13th Systems Administration Conference (LISA '99)*, pp. 149-154, USENIX, December, 1999.

[3] Burgess, Mark, "A Site Configuration Engine," *Computing Systems*, Volume 8, pp. 309-337, USENIX, Summer, 1995.

[4] Burgess, Mark, "Computer Immunology," *Twelfth Systems Administration Conference (LISA '98)*, p. 283, Boston, Massachusetts, USENIX, December 6-11, 1998.

[5] Cisco Systems, Inc., *Cisco IOS*, http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/index.htm .

[6] Cons, Lionel and Piotr Poznanski, "Pan: A High-Level Configuration Language," *Sixteenth Systems Administration Conference (LISA '02)*, pp. 83-98, USENIX,, November 2002.

[7] *The Concurrent Versions System*, http://ccvs.cvshome.org .

[8] Descartes, Alligator and Tim Bunce, *Programming the Perl DBI*, O'Reilly, February, 2000.

[9] Dooley, Kevin, *Designing Large-Scale LANs*, O'Reilly, January, 2002.

[10] Dooley, Kevin, *Designing Large-Scale LANs*, Chapter 9.1.1, p. 274, O'Reilly, January, 2002.

[11] Droms, Ralph, *Dynamic Host Configuration Protocol,* RFC 2131, March, 1997.

[12] Finke, Jon, "Automating Printing Configuration," *LISA VIII Conference Proceedings*, pp. 175-183, San Diego, CA, USENIX, September 19-23, 1994.

[13] Finke, Jon, "Institute White Pages as a System Administration Problem," *10th Systems Administration Conference (LISA '96)*, pp. 233-240, Chicago, IL, Usenix, September 29-October 4, 1996.

[14] Finke, Jon, "Automation of Site Configuration Management," *Eleventh Systems Administration Conference (LISA '97)*, p. 155, San Diego, CA, USENIX, October 26-31, 1997.

[15] Finke, Jon, "Monitoring Application Use with License Server Logs," *Eleventh Systems Administration Conference (LISA '97)*, p. 17, San Diego, CA, USENIX, October 26-31, 1997.

[16] *GxSNMP*, http://www.gxsnmp.org/ .

[17] "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks," *IEEE Standard 802.1Q-1988*, 1998.

[18] *LANdb: The Network Management Database*, http://landb.sourceforge.net/about.shtml .

[19] Libes, Don, "expect: Curing Those Uncontrollable Fits of Interaction," *USENIX Summer 1990 Conference Proceedings*, pp. 183-192, USENIX, 1990.

[20] Mockapetris, P., *Domain Names – Concepts and Facilities*, STD 13, RFC 1034, November, 1987.

[21] *Nagios*, http://www.nagios.org/ .

[22] NetSaint, http://www.netsaint.org .

[23] openNMS, http://www.opennms.org .

[24] *hp OpenView*, http://www.managementsoftware.hp.com/ .

[25] *Perl*, http://www.perl.com .

[26] *PostgreSQL*, http://www.postgresql.org .

[27] *RANCID – Really Awesome New Cisco confIg Differ*, http://www.shrubbery.net/rancid/ .

[28] Tichy, Walter F., ‘‘RCS – A System for Version Control,’’ *Software – Practice and Experience*, Vol. 15, Num. 7, pp. 637-654, 1985.

[29] IBM Tivoli NetView, http://www.tivoli.com/products/index/netview/ .

[30] Traugott, Steve and Joel Huddleston, ‘‘Bootstrapping an Infrastructure,’’ In *Twelfth Systems Administration Conference (LISA ’98)*, p. 181, Boston, MA, USENIX, December 6-11, 1998.

[31] Ylonen, Tatu, ‘‘SSH – Secure Login Connections Over the Internet,’’ *6th USENIX Security Symposium*, pages 37-42, USENIX, San Jose, CA, July 22-25 1996.

Appendix A: Splat Syntax

As stated earlier, Splat is a command line tool. In order to meet the requirement of enabling administrators with little networking experience to make routine changes to the LAN, a simple syntax was employed. The basic operations are:

- Attaching a host to a jack.
- Detaching a host from a jack.
- Swapping two hosts.
- Getting information on a hostname, jacknumber, IDF, switch, or room.

Syntax Summary:

```
splat [ -a|--attach [ -b|--brief ] [ -n|--nolock ] hostname jacknumber [ vlan ] ]
      [ -c|--commands switch ]
      [ -d|--detach [ -b|--brief ] hostname ]
      [ -h|--help ]
      [ -s|--swap [ -b|--brief ] hostname1 hostname2 ]
      [ -i|--info [ -b|--brief ]  [ hostname | idf | jacknumber |  switch ]]
```

Appendix B: Examples

Host Information

When given the name of an attached host, splat -i prints the data jack connection information and available switch port statistics for the host. Switch port statistics vary depending on the switch type. Below is an example involving a Cisco 3500 switch.

```
$ splat -i host1.cs.wisc.edu
-------------------------------------------------------------------------
SPLAT NETWORK DATABASE                          Mon Jun 30 10:39:55 2003
-------------------------------------------------------------------------
 Hostname: host1.cs.wisc.edu
     Part: C013317                                 Bldg: CS
Interface: 0                                       Room: 2327
      MAC: 00:e0:18:71:91:0b                       Note: NA

IDF    Jack  Switch                                Portnumber
-------------------------------------------------------------------------
2300E  218   2300E-sw-3.cs.wisc.edu                FastEthernet0/42
-------------------------------------------------------------------------

Available Switch Statistics
-------------------------------------------------------------------------
FastEthernet0/42 is up, line protocol is up (connected)
  Hardware is Fast Ethernet, address is 000a.8aac.73aa (bia 000a.8aac.73aa)
  Description: host1.cs.wisc.edu via 2300E-218
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s
  input flow-control is off, output flow-control is off
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue :0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute ouxtput rate 0 bits/sec, 0 packets/sec
     1167225 packets input, 545465471 bytes, 0 no buffer
     Received 66 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog, 0 multicast, 0 pause input
     0 input packets with dribble condition detected
     1338665 packets output, 968728968 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 PAUSE output
     0 output buffer failures, 0 output buffers swapped out
```

Switch Information

2300E-sw-3.cs.wisc.edu is the hostname of an IDF edge switch. When given a switch name, the splat -i command shows information about the switch:

```
$ splat -i 2300E-sw-3.cs.wisc.edu
------------------------------------------------------------------------
SPLAT NETWORK DATABASE                          Mon Jun 30 10:42:09 2003
------------------------------------------------------------------------

Switch: 2300E-sw-3.cs.wisc.edu

Interface         Jacknum    Hostname
------------------------------------------------------------------------
FastEthernet0/1  2300E-201   chopin.cs.wisc.edu
FastEthernet0/2  2300E-210   tonic.cs.wisc.edu
FastEthernet0/3  2300E-211   lime.cs.wisc.edu
FastEthernet0/4  2300E-212   NA
FastEthernet0/5  2300E-213   vodka.cs.wisc.edu
FastEthernet0/6  2300E-214   ojuice.cs.wisc.edu
FastEthernet0/7  2300E-215   NA
. . .
```

Attach

The -a option is used to *attach* a computer to a data jack. The -b option requests brief output. Otherwise, each command and the resultant switch output (if any) is printed.

If the hostname is not a fully qualified domain name, an attempt is made to find one in the current domain using gethostbyname().

In this example, the host *host1.cs.wisc.edu* is moved to data jack *2300E-218*:

```
$ splat -a -b host1 2300E-218
Use fully qualified domain name host1.cs.wisc.edu? [n]/y : y

Updating the network database...
WARNING - Hostname host1.cs.wisc.edu is currently attached to jack 2300E-319.
OK to detach this connection? [n]/y : y
Host host1.cs.wisc.edu detached from jacknumber 2300E-319 in the splat database.

Updating the switch...
Host host1.cs.wisc.edu attached to jacknumber 2300E-218.

Running RANCID to collect switch config...
CVS commit for 2300E-sw-3.cs.wisc.edu config successful.
```