The following paper was originally published in the
Proceedings of the Twelfth Systems Administration Conference (LISA '98)
Boston, Massachusetts, December 6-11, 1998

Building an Enterprise Printing System

Ben Woodard, Cisco Systems

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Building An Enterprise Printing System

Ben Woodard – Cisco Systems

ABSTRACT

Cisco Systems has chosen to internally develop an enterprise wide print system that provides access to more than 2000 printers for both Unix and PCs. The requirements for this print system were that it had to be very cheap to construct, highly scalable, easily maintained by a very small staff, fault tolerant, and mission critical reliable. In other words, management essentially wanted everything for practically nothing. To meet our objectives we built our print system out of interchangeable low cost running PCs Linux, LPD and Samba as well as other standard Unix applications. The low cost of PC hardware and the lack of licensing fees for Linux allowed us to deploy the print system vary widely without having to go through all the managerial justifications necessary to authorize larger scale purchases. By making each print server interchangeable we achieved scalability as well as a certain degree of fault tolerance. The flexibility of running a Unix like operating system such as Linux as opposed to another more restrictive operating system allowed us to develop a worldwide printing application that can be managed very easily by only two or three people. And finally the robustness of Linux made it possible for us to use our print system in mission critical environments such as manufacturing production floors.

This paper discusses the process by which the print system was implemented and the wisdom learned in the process. It covers topics such as how to gain and maintain control of the printing process, why it is necessary and how to keep printers a completely network managed device, how we learned to deal with large numbers of server, the advantages and problems we ran into as the number of servers grew, and the many advantages and few disadvantages of basing the system entirely on free software. It also highlights some of the major processes that we automated and the success we had devolving power first to the the local technical support people and then ultimately to the users. Finally, it discusses many of the problems that we are running into now that the print system is a few years old and the steps that we are taking to ensure that we do not become victims of our our own success and that we do not have the whole system collapse due to data rot.

Since the real key to managing thousands of printers effectively is figuring out how to save time, the real world experience we gained and the time saving tips we discovered while learning how to manage thousands of printers should be valuable even to sysadmins that have only a few printers to manage.

Introduction

Managing just one printer can be difficult and just a few dozen could be enough to keep a full time sysadmin perpetually busy. When the problem is scaled 1000 or 2000 times to the size of the enterprise for a medium sized multinational corporation, the problem has the potential to be an unmanageable nightmare. We have heard vendors pitching printing software use figures such as 70% of the time spent by a network administrator is spent resolving printer related issues. This paper discusses how we were able to build a greater than 2000 printer print system for Unix workstations and servers as well as PCs that is managed by a staff of only two full time sysadmin. Since the key to managing thousands of printers effectively is figuring out how to save time, the time saving tips we discovered while learning how to manage thousands of printers should be applicable to sysadmins that have only a few printers to manage.

Getting Control ...

... of Unix

When my colleague, Damian,[1] arrived on the job there was no printing system to speak of. There were a couple of print servers but hardly anyone used them. The job of the person in charge of printing was basically to keep all the Unix servers' printcap files up to date. This was a monotonous job and he quickly tired of it. The first thing that he did was generate a canonical list of all the printers that he knew about and then sift through this list to find the ones that actually worked. This winnowing process can be greatly sped up today since most printers support protocols such as SNMP. Armed with the authoritative list of all the

[1]My colleague Damian Ivereigh actually began work on the print system in June of 1995. I joined the project around October of 1996. To contact Damian you can send email to damian@cisco.com.

printers, he made a script to distribute this to all of the Unix servers. In talking to several people, this beginning is where their print system ends. This in itself became a time consuming and cumbersome task but at least all of the Unix servers could print to all of the printers.

Keeping up to date with all the hosts that needed the printcap file quickly became a hassle and the repetition was boring him and so he decided to fix this problem once and for all. The key observation was that there were basically two printcap files. The one for the print server which sent the jobs to the printers and the ones that went on the Unix print clients which sent the print jobs to the print servers. The printcap file on the Unix servers that were not print servers was very repetitious with entry after entry redirecting the print jobs to the print server. The non print servers really didn't need to know anything about any printer except for its queue name and the name of the print server. Damian came up with a way to get out of all of this repetitious work. He got the source for LPR and modified it so that it would read a file called "/etc/lpr.conf" instead of "/etc/printcap". In this file, he put all the things that were common in every printcap entry, the name of the print server, and the root of the spool directory. Then he modified the routine in the LPR source code that found the printcap entry so that it would create the spool directory if necessary and generate a simple printcap entry that would redirect the print job to the print server. This way every time any of the LPD utilities looked up a printer, it would find an entry that simply directed the job to the print server.

This was a great boon. It made it so that once he installed all his modified print clients he could basically forget about the existence of the machine. It would have access to every printer that the print servers knew about. Another innovation was to wrap this up with a very quick and easy installation procedure. He made all of the print software into a shar file. So all that a sysadmin had to do was simply run this shar file and the script would uudecode the tar file that had the modified binaries for that architecture, put them in place, and restart the print services on that machine. Reducing the entire task of getting and keeping printing running on your machine down to the simple process of fpt'ing a file and running it as root almost instantly won the unanimous support of all the other sysadmins. It made it so easy to go along with the system, that almost no management leverage was needed to get several groups of sysadmins to install this print client. In an organization such as ours where we have a relatively large number of sysadmin arranged into several almost autonomous groups, getting such widespread acceptance was no small feat.

Every once in a great while we are asked to come up with a new port of our print software, there was one patch release to fix a bug, and every so often a new vendor patch will overwrite the software but other than that, the Unix LPR client has needed almost no maintenance. All the sysadmins know that part of process of setting up a new machine or upgrading the OS on a machine is to download and run our print software installer.

... of PC's

About this time, early in 1996, PCs were starting to replace the Macs at Cisco. It quickly became apparent that there was a lot of duplication between the team that was keeping the printers up to date on the NT servers and the person that kept the printers running for the Unix servers. The NT folk were already fairly busy trying to just keep their systems up and running and didn't want to bother with printing and so Damian fired up Samba on the backup print server to provide PC printing.

The PC population was even easier to convince to use the print servers than the Unix folk. They really didn't have much choice, it was generally technically beyond the majority them to install programs such as JetAdmin, or WLprSpool and print directly to the printer.

Drivers were a perpetual problem until we bought Jeremy Alison, one of the Samba developers, enough beer that he came over and got the Plug and Print automatic driver download for Windows 95 working for us. Now when a Windows 95 client double clicks on a printer on the print server for the first time it silently downloads the correct driver to their PC. This has many benefits. First of all, it reduces support calls by allowing us to make sure that the client is using the most up to date, most bug free driver that we can find. It eliminates the driver/device mismatch problems that can complicate printing. It also minimizes the amount of time that our internal technical support staff (or sometimes us) has to spend on the phone with a user to get a printer set up. We currently can't pull the same trick with NT servers because Samba doesn't implement the NT RPCs yet but we have found a place in the registry where NT looks to find the print drivers. This allows us to place all the needed drivers on the NT workstation's disk and have it load the correct driver from its disk much like it would have done had it downloaded it off of the print server. The samba developers assure us they are working on a plug and print implementation that works with NT and that it will be done soon.

Taking control of the PC printing as well as the Unix printing is quite a time saver for the organization. Instead of two groups trying to maintain an accurate list of the printers, you only have one and when you fix a printing problem with one, most frequently, you have also fixed the problem with the other. In other organizations, we have heard of many cases where the PC's have one printer and the Unix machines have another printer and each printer is fairly underutilized. We have also heard of cases where the PC support guy fixes a printer for the PC's

and in so doing breaks the printer for the Unix machines. Adding the appropriate entry to the smb.conf file was just another step in setting a printer up.

... of Printers

Another step in our complete take over of the print infrastructure was taking complete control over the printers. The first thing that we took control of was the printer standards. There were no IS enforced standards for printers at the time when Damian started. This led to some very unfortunate purchases in the early years. We have pretty much weeded all of the bad printers out but it took quite a while and quite a bit of coercion. The key thing for a printer in a large enterprise environment is that it must have a **great** network interface. You have to keep in mind that the network is your only connection to the printer and if you have trouble communicating with the printer or you can't do all jobs you will ever need to do or find out all the information that you might ever need to find out from the network interface, then you are stuck. Attention to detail is critically important when evaluating printers; it is rather amazing how very simple seemingly harmless semantic changes can cause problems in an enterprise. Another thing to keep in mind is that even simple tasks can become quite time consuming when they are multiplied across more than 1000 printers. Therefore everything must be scriptable or you cannot automate the job. In looking through many printers' network interfaces with a fine toothed comb, we have found that there are only a couple that really can stand up to the demands of enterprise printing: HP, Lexmark, and Tektronix, though Xerox is starting to come along.

In evaluating a printer's network interface don't be misled by the printer vendor's cool GUI program. Unfortunately, the current trend with the printer vendors is to provide a very dumbed down graphical user interface that allows you to do ''everything that you will ever need to do'' with the printer. The truth is that graphical user interfaces are practically useless for more than a handful of printers. Intrinsic in these GUIs the developers have made a bunch of insidious assumptions about how you want to view and work with the data. You are frequently stuck when you try to do something out of the ordinary. The biggest problem is that they are not script-able. This is why we have resorted to using some of the more low level protocols such as SNMP for dealing with printers. In fact most of those fancy, command line and GUI tools are just wrappers on simple SNMP calls. When evaluating a printer, we first look at what the GUI tools can do, and make note of any interesting new features. Then we throw them away and never look at them again because they just can't be used in an environment as large as ours. Even the command line tools such as hpnpadmin don't generate the kind of output that lends itself easy to processing in a shell script. We were forced to write many of our own tools. Hopefully by the time that you read this, many of these tools that we have developed in house will be available on the Internet.[2] The next aspect of a printer's life that you want to control is its bootup sequence. There are quite a large number of reasons that you want to control the bootup sequence of the printer. The first reason is that you need to know what IP address the printer. The second reason is it allows you to do some simple configuration by using a bootp configuration file. Both HP and Lexmarks have a bootp configuration file. The details differ but they allow a few simple configuration parameters such as setting the community name and access lists. Access lists are remarkably useful items; they allow you to limit access to the printer to only the print server. This makes it so that you have a single point of queuing. This is a major benefit to technical support staff because if someone says that they are having trouble with a print job stuck in the print queue for thus and such printer, they know that the job is stuck in the one and only print queue for that printer. They don't have to search for which queue it is in. It also gives them one place to troubleshoot printer problems and one place to delete problem print jobs. This is a great time saver for them as well as us.

Another reason that you want to control the boot process is it gives you a chance to do some post boot configuration. Both HP and Lexmark TFTP their bootp configuration file. By using tcp wrappers and having the tftp of the boot file logged and having a program continuously looking at this file you can tell when a printer is booted up. You simply wait some appropriate amount of time and then use whatever tools you have to configure the printers fully. You can also use a SNMP trap to figure out when the printer is booted up. We basically use our SNMP tools to disable unused protocols, check the firmware version etc. The value of this post boot configuration should not be overlooked. It allows you to bring the configuration of all of the printers into harmony and therefor accumulate within the configuration of the printer all the wisdom that you have learned by configuring all the other printers of the same type. One of the things that I have learned managing a large environment is that NVRAM is volatile. You cannot count on the fact that you configured this printer once when it was first set up and so all the appropriate settings are still set.

When we first started the print system many of the printers were configured by either the front panel or telnet which caused problems when they were moved or the IP address range for the local LAN was changed. Someone had to physically go to the printer and change it. When you have a printer configured with a static IP address, if anything changes such as the subnet it is on or the IP address range of the network, the printer quickly ceases being a network managed device. Some of the newer printers grabbed

[2] http://pasta.penguincomputing.com/pub/prtools

DHCP IP addresses which also proved problematic. There are several problems with using DHCP for printers. The first is that the print system is not informed when a printer changes its subnet or its IP address changes. The second problem is that the print system is not notified when the printer boots up and so you do cannot provide a boot configuration file nor can you do any post boot configuration. HP is currently addressing this problem with the latest version of firmware in the JetDirect EIO cards. These new cards send out a multicast packet when they boot up which can be monitored by the print system to initiate all sorts of post boot configuration.

We resorted to using bootp to configure our printers. With the current technology, bootp is probably the best method to configure a printer. Adding the printer to the bootptab became just another step in the process of setting a printer up. The main advantage of bootp is that the printer always remains a network managed device. The moment a printer ceases to be network managed and you or someone else has to go out and visit a printer, you have lost a great deal of time. The second benefit of running printers off of bootp is that you can specify a configuration file for them to tftp down after they boot up. Using TFTP as a trigger allows you do all sorts of post boot configuration of the printer. There are two main down sides to using BOOTP. The first is if a printer moves subnets, then you must modify the bootptab. The second is that you have to know what devices you are going to BOOTP. We overcame both of these limitations by modifying the bootpd. What we did was made it so that when a device such as a printer sent out a BOOTP request, we checked to make sure that it was coming from the network we expected it to come from. If it did not we quickly rewrote its bootptab entry so that it had an appropriate IP address and informed the other aspects of the print system that needed to know about this change. This allowed us to get out of the moves process which we greatly despised because it was always interrupting our weekends. The second modification we made BOOTP seem even more like DHCP. If we get a BOOTP request from a device we had never heard of before, we allocate an address for it and set it up in the print system with a dummy name. That way when a user calls us asking us to set up a printer, it is largely already done, all we have to do is rename the printer. This is an incredible time saver.[3]

Automate Everything

With the rapid growth of Cisco (and therefore our workload) coupled with the complete lack of growth of our staff, we were forced to automate everything. Other than expanding responsibilities, with no corresponding growth of resources, there are many good reasons to automate every aspect of system management. The first reason is that in many places you have the same information. Making sure that all these areas are kept up to date with each other can be difficult and the problems associated with having one piece of information out of sync with another can be difficult to track down. As a rule you want to put information one place and then have the computer copy it to all the places it needs to be. A good example is the printer's IP address. Several parts of the print system need to know the printer's IP address. For example the filter scripts need to know the printer's IP address so that they know where to send the print data, the BOOTP daemon needs to know what IP address to give the printer when it boots up and the several scripts that we use to help administer the print system need to know the printer's IP address so that it can do all the SNMP queries to find out the status of the printer. You don't want to have to maintain the IP address of the printer in three places. So what we initially did was put all this information in a flat file and then wrote programs to read this flat file and turn it into the needed configuration files such as /etc/printcap, /etc/bootptab, and /etc/smb.conf. Now we have progressed beyond that and we store the printer information in a little home grown directory service. We have also modified certain critical programs such as bootpd and lpd so that they read directly from the directory service.

I have already discussed some of the first tasks we automated: setting up the spool directories, building of the ''/etc/printcap'', the ''/etc/bootptab'', and the ''/etc/smb.conf''. The next big time saver was the modifications to bootpd which automatically set up or moved printers.

DNS and Oracle

A couple other chores that we automated were keeping DNS and the Oracle databases up to date with our printer database. Although both of these programs were great time savers, they both took three or four times as long as expected to get into production. Doing a post mortem on the project and why it took so long to implement, took us quite a while but led to a useful piece of insight. It turns out that the problem was that both of these programs reached into someone else's database and updated their records. In so doing, they pushed back the frontier of what is managed by the print team. The added complexity of dealing with someone else's system made these task more challenging and introduced some support cross functional support challenges. I have run into cases where a script that ran fine a couple of days ago quit working all of the sudden because some interface or some subtle semantic has changed. For this reason, we don't rely on any of these programs for mission critical functions. This allows us a couple of days to figure out what is going wrong with the script and fix it before it becomes a problem. We also plan accordingly when we begin a project that will involve working with another group.

[3] So much so that it even surprised us.

The Script-able Ultimatum

When evaluating printers the approach we took was that every task we would need to do on a regular basis needed to be automatable. If something required a step that we couldn't write into a script then we couldn't apply it. There were several features on certain printers which could only be set up through telnet or through some web interface, we deemed these features unusable. If they were in critical operations such as booting, then we deemed the printer unusable. There are quite a few operations that the vendors will tell you don't need to be script-able because "you will only have to do them once a year" like upgrading the firmware. In practice however, these operations come up more frequently than you would imagine. For example, say there is a nasty bug in the firmware of some printer, you have a fix for it but the printers that have been manufactured but not delivered yet might still have the old firmware. So for the next two months every printer that you set up needs a firmware upgrade. In Cisco, that translates to more than 90 printers. Repeating a simple operation 90 times can be really time consuming.

Automation and its effects on you

Managing a system this large is a demanding job both technically and psychologically. We discovered that we as the administrators of the print system are part of the print system. Our psychological state really affects the smooth operation of the system as a whole. We are the vision, the creativity, the wisdom and intelligence behind the print system and We are not as creative when we are feeling burned out. Managing a large print system can really quickly burn out any sysadmin if they are not careful. The thing that gets to you is the fact that you are dealing with literally hundreds or thousands of devices that have similar problems. The repetition can easily grind you down. That is one of the reasons that automating as many tasks as possible is so critically important. In most other companies that we have talked to this is really a problem. Printing is a job dumped onto the lowest man in the pecking order. As soon as that person can, he will pass responsibility onto another person. Consequently, there is no consistent vision or direction to the system and the wisdom gained by one generation is not passed onto the next. To maintain a reliable print system, it is vitally necessary to have that consistency of vision behind it. Print administration will kill your spirit very quickly if you do not automate it.

One problem that we had was the feeling that we were on a treadmill. Even with our reasonably successful automation of many processes sometimes we feel like things never change and never get better, like we are still plagued by the same problems that we had a year ago. Automation in itself feels like a never ending task that seems to have no noticeable benefit. What seems to happen is that the moment that we get one task automated, we have more time to address some issues that were in the background. They end up being a can of worms and our time is once again completely used up. Print services can be maddening that way. The only thing that seems to help is trying really hard to maintain perspective. Knowing where we are, where we have been and where we are going. Every so often we take a look back at what we have accomplished and find that we are doing much more in less time. For example at Cisco, it used to be one man's job to take care of 180 printers, and two print servers for 30 some odd Unix servers. Now it is two man's jobs to take care of 2000 printers, close to 100 print servers, for more than 100 Unix servers, a few hundred Unix workstations and close to ten thousand PCs. Every little thing that we automate adds up to more time savings and helps push you back burnout just a little bit farther.

Power to the People

One of the greatest time savers that we put together was providing a web interface to the print system. We initially rolled this out to the internal technical support people and once we were sure that it worked acceptably, we rolled it out to the rest of our users. The web page provides users with the capability to start and stop print queues and delete print jobs. At first there was some apprehension to allowing every user to do this. In the more than two years that it has been available, we have yet to hear one report of someone misusing the privilege. One thing about the web interface to the print queue is that it is accessible to everybody and it allows people to solve their most common problems by themselves rather than calling us. We have found that this feature alone really eases the load on us and makes our customers much happier.

Dealing With More Servers

Originally we had two print servers, a primary and a backup. The primary was the one that did all the print spooling and the backup had been pressed into service for the Windows clients. This was a very simple system and worked fine for headquarters. Then our company decided to open a facility all the way across the country in Research Triangle Park, North Carolina (RTP). Since one of the purposes for this new site was disaster recovery and because some production servers there were going to need to print to printers at headquarters and vice versa, we were given a requirement that all the print servers need to be able to print to all the printers. This one decision more than anything else took us down the road that led to an enterprise print system rather than just maintaining local printers. Allowing all the server's in the world to print to all the printers without unnecessarily burdening the WAN introduces a couple new twists to the printing architecture.

First of all there was the matter of the printcap file. We had to make the scripts that made the printcap file know where the printer was in relation to the print

server whose printcap file was being generated. This introduced two new pieces of information into the print system. The first was the concept of a print server having a ‘‘location.’’ The second was that a printer had a ‘‘location.’’ For convenience we grouped the printers together into locations such as San Jose and RTP and then assigned a print server to a location. All the other print servers that were not spooling that particular location code still created the spool directories and had bootp records but in the printcap file they made the printer a remote printer. That way if a print job landed on any print server that was not spooling that particular printer, it would be redirected to the print server that was serving that printer. This effectively turned all the print servers into print job routers.

We didn’t realize this at the time but this also allowed us to scale the print system almost infinitely where needed. By keeping the size of the location codes reasonably small and then assigning quite a few of them to a print server. We have been able to very easily add capacity where needed. When we need more capacity, we simply add a new server and give it some of the location codes. We currently have twenty four print servers at our biggest site, headquarters. This scalability is one of the true successes of our print system. Cisco has been doubling its size almost every year for several years now and the print system has been able to keep up with this growth without having to spend a lot of money. In fact, we have been using desktop PCs running Linux as our print servers and so anytime we need to add capacity or place a print server at a new site, it is simply another $1000 server. $1000 is well within the purchase authority of even the lowest level manager and so it is comparatively easy to get approval. One of the problems that we ran into while adding capacity in this way was as the number of print server grew past six to ten, twelve, or twenty four, the job of dividing the locations amongst the print servers became more difficult. We eventually ended up writing a program to do it for us because the problem became so intractable. To make it easy for people to find printers and to make it so that we could easily divide the load amongst many servers we made the locations pretty fine grained, one floor of one building. As Cisco grew and the number of locations grew appropriately allocating these to servers became increasingly time consuming.

Another feature that dividing the print system into locations did for us was it allowed us to provide a more reliable service. If a print server was down for whatever reason, we simply moved the location codes to the rest of the servers. We found that it was difficult remembering which locations went where once the server came back up and we also found that our typical outage, a reboot, took less time than moving the location codes to a different machine. We greatly simplified this by giving each location a primary and a backup print server. That way if we marked a print server as down, the print jobs would go through to the backup print server and we wouldn’t have to move the locations. To make things even more reliable, if both the primary and the backup print server is down things are spooled from our "last chance server". To date, we have never had to use the last chance server. If we know that a print server is going to be down for more than a few minutes, we re-balance the load amongst the remaining print servers by running the same script that we use to fairly distribute the load amongst the servers. This resets the primary and the backup print server for each location without considering the print servers that are marked as down. Since we put these features in place, we have had essentially no down time. It has also made it so that we really don’t have to come in on weekends anymore to do system maintenance. For instance, we have done things like one by one moved all the print servers from one server room to another server room during the working day without causing any user noticeable down time. The only downside to this is we are not collecting anywhere as much on-call pay as we once did.

Having print servers at multiple sites created one significant problem. We were using programs to read flat files and generate all the print server configuration files. When we only had one site, and one print server we were able to edit these files on this machine and then we would build the configuration for that print server right there. As we got more print servers we had to make sure that this file was up to date on all print servers. Initially we were using rcp to copy the file from the source to the outlying print servers. As the number of servers increased, it became a bit of a problem keeping track of which print servers had an up to date set of printer information files from which to build its configuration files. There was also the problem of multiple users. Since it was a simple file, we had to serialize access to this file so that changes were not lost when two people accidentally edited the file at the same time. We found that the local sysadmin had better knowledge of the local printers than we did back at headquarters and so it was helpful for them to make the changes that pertained to their sites. With the initial system of a couple of flat files, if the network was down between the sites, they couldn’t make any changes to their print environment. All in all this was unacceptable. So we created what we thought was a database of the print information. (It eventually turned out to be more of a directory service like DNS rather than a database like Oracle in the end.) We wanted it to take care of the propagation of the printer information so that we wouldn’t be tied to rcp and we wanted it to keep track of which servers had up to date information. We also wanted it to update a particular value in a record rather than locking the whole table. That way we didn’t have to worry about serializing access to the table. We could allow people to update particular fields in a record and make that an atomic operation. We also wanted it to allow local people to update their print environment even if the network was down

and when the network was reconnected we wanted all the changes that they made to propagate around the world.

Damian who wrote this program called it SDDB for simple distributed data base. As it turned out this is quite a misnomer. First of all it is not simple; it is distributed, but it really isn't a database. Name not withstanding this program is at the heart of our print system. We provide a front end for people to update records and all the programs (except for Samba) have been modified to read directly form the SDDB directory service rather than reading from their native files. This allows changes that we make to be propagated more quickly and also allows us to react more quickly to server failures or other interruptions in service.

With the rapid proliferation of servers to handle capacity and at remote sites, we ran into several problems keeping them up to date. The first problem we addressed was keeping our custom print software up to date on all the print servers worldwide. Although, it is not really an ideal solution, we use rdist to push out the binaries to the machines. We have not addressed the problems associated with servers that don't have the data pushed to them properly. As there are getting to be more servers, this is becoming a more and more of a problem. rdist just isn't designed to keep large numbers of hosts in sync with each other. Ultimately, we will probably move away from a push model and go to a pull model based upon RedHat Software's RPM. That way when the machines boot and every night, they go and get their configuration. If they happen to be down, when they next boot they will get their correct configuration. Or if the net is down when we happen to do an rdist to them, then instead of missing that update, they will simply get their correct configuration the next night.

Another problem we addressed was how to load up lots of machines quickly. When you are deploying two or three new servers per week, the time taken to load up a new machine is significant. What we did was created a NFS boot floppy that boots up the machine and then partitions the hard disk and loads all the initial software. We sort of use a basic RedHat 4.2 configuration. After the disk is partitioned and formatted, we lay down a skeleton directory tree and then just start dropping rpm's onto the disk. Once again this was an interim solution that hasn't been replaced yet.

One of the big problems that we need to address is although we have a way to make a brand new machine with all the patches applied, and we have a way to keep all our custom software up to date, we have yet to make a system where we can quickly and easily apply updates to all the print servers. At the moment, when in our opinion a machine gets too far out of date, we have someone put a floppy into it and then we dd the new install NFS boot floppy onto it and reboot the machine. It comes up and reloads itself using the exact process we use to create a new machine. Once it is up and we are satisfied that it updated itself properly, we ''zap'' the floppy and put a new boot sector on it so that the system will boot to the kernel on the hard drive. Some time later someone comes along and removes the floppy for us. This can be a really time consuming task and we can only really do one machine at a time.

We currently are working on a solution to address all three problems as well as solve a few other problems we haven't yet addressed. We want a system that will create a new server from scratch over the net, will keep the custom software up to date, keep the vendor supplied patches up to date, allow us to have variations from our standard configuration, allow us to partially roll out test versions of our software and finally, allow us to boot off a special recovery partition of the disk which will allow us to rebuild the disk in the event of a bad crash that leave the main partitions unbootable. The way that we plan to tackle this problem is to have a kind of target configuration for each print server. Then when a server boots up, it will have a current configuration. If those do not match it will go through a series of steps that will turn one state into another state. This process will probably be done by using RPM controlled by make files. Hopefully, this will allow us to maintain even more servers with less effort.

Printer management

Since we have been running the print system for more than two years, we have accumulated quite a bit of data rot. For example, we have an automatic process that moves a printer within the print system if it comes up on a different network. We had a large site that moved and once everybody was out of the building there were six printers left. When we looked more carefully, we discovered that these printers had been decommissioned long ago but no process had been put into place to notify us. We still don't have such a process. Instead of trying to come up with a process to deal with this, a simpler solution is to leverage the work of another team who's specialty is monitoring servers and routers and other pieces of IS infrastructure and have them monitor the printers. Then if a printer drops off the network for more than a few days we can call the printer's contact and find out if the printer has been decommissioned or if it is broken or something like that. We can also use the the printer monitoring facility to find out many other pieces of information about the printers such as how well utilized our printers are. Are we over utilizing or under utilizing our printers? We also will be able to find out if certain models of printer are not standing up to the rigors of use.

Another kind of data rot we have is keeping the contacts and the locations of printers up to date. We try to set the location and contact when a printer is first set up but ever since we have automated the move process to the point where a printer can be moved

without intervention by us, we have found that the locations and the contact information for the printer is not being kept up to date. We have the beginnings of a system to keep the printer's contact name and location up to date. If someone prints to a printer that doesn't have the location and contact set, we will send email out to the user asking them for the information. Eventually, we find someone who gets fed up with getting email from us and decides to tell us where the printer is and who to contact regarding it. However, we need to check this information every so often and we have yet to put in place code that double checks the location and contact every so often.

Yet another kind of data rot that we are experiencing is with regards to names. It would be nice if a printer had one canonical name, however printers seem to accumulate names. These alternate names appear for various reasons. The first place is when a printer is renamed, we keep the old name around for a while to maintain backward compatibility. This problem was exacerbated by the fact that we used to name printers with a bit of their location in their name. For example, a printer in building K on the second floor might be called k2-opsteam. The problem is that with the rate that Cisco has been growing, groups are always being moved from building to building to make room for expansion. So every time a printer moves it would pick up another alias. We have no automated procedure to trim down the number of aliases a printer has. To try to curtail the explosion of names, we have stopped encoding the printer's location in the printer's name. That way a printer named opsteam doesn't have to change its name every time it moves.

One kind of data rot that we have taken very seriously is getting people to use our print system rather than the old NT server print services. When we take over a site that used to have an NT server in it, we redirect all the NT print queues to the print server. That way the local technical support does not have make changes to everyone at that site. Then when the users print through the NT server, we send them an email telling them to change their printer configuration to point to the Linux print server rather than NT server. This eases the transition between the two services and makes it much easier for the local support people. They don't have to change all their clients' configuration; the clients will get email telling them how to do it.

There are certain places within the company on the fringes of IS support or in new acquisitions where we don't have an accurate list of all the printers. We need to put in place a process where we scan the network for unknown printers so that we can keep our list of available printers accurate and up to date. Right now this is a very time and network intensive process because you pretty much have to probe every address on the network. HP is addressing this problem by having their printers respond to multicast requests but it will be a couple of years before this feature is widely deployed.

Open Source Software – Beyond religion

There is a lot of religion around the Open Software movement both sides have their reasons for sticking to their point of view. We brought free software and many of the ideas associated with it, into Cisco and in the beginning it was a bit of a struggle. When we started using Linux for a mission critical function such as printing was pretty much unheard of within a large company. Now it is fairly commonplace. We have learned many things by running free software these past few years.

Flexibility

The claims of the free software folk about ''free'' referring to freedom rather than price is really true. For us this has been the key to the print system. Because we had the source, we were able to modify the code to better meet our needs. To the free software people this sounds obvious but in the corporate world where fully supported is considered the norm, this idea was considered radical. In our case the benefits were immense. The one thing that we learned was in a rather large environment like Cisco the advantages of scale work differently than in a small environment. It probably isn't worthwhile for a company to customize an application exactly to their needs. However in a large corporation such as Cisco, the benefits of customization are great enough that it is economically feasible to modify an application to better suit the needs of the company.

The Final Authority

The downside to this flexibility was that because we were running custom applications, we were the final point of escalation. It is a humbling and sometimes scary thought that if a problem comes up you are the one that has to solve it. You are the final authority on the matter. It requires the corporation to have more faith in its own people. By running custom software, the company is in effect saying that their technical people are competent enough to handle anything that comes up. In many ways, this situation is not that different than that of a commercial piece of software. Somewhere within a commercial software company's organization, there is some engineer who is ultimately responsible for resolving problems with a piece of code. Code is not fixed or modified by corporations. Code is modified by people. The only difference is who that person works for.

The situation has only come up a couple of times. It is scary knowing that no matter what you must find the bug and fix it. I remember having to find a file descriptor leak in LPD. It took me a couple days to actually find it and I felt the weight of the world those two days. It was very sobering. On the other hand one of my coworkers installed a piece of software from the OS vendor. On the dev machine it

installed cleanly with no questions asked. However on the production box, after the files were put into place the installation asked to reboot the machine. There the sysadmin was, sitting in front of the package installation software with a little X-Windows message box saying, ''Hit OK to reboot the machine.'' This just happened to be during quarter end processing and he certainly didn't want to reboot the machine right then. Without the source to the application or the installation script within the package, he didn't know if he could exit the software installer without rebooting the machine. He didn't even know for sure if the machine was in a stable state to keep on running. To say the least he was rather concerned. He called the vendor and their support people, who had never looked at the application source didn't know what would happen either. When they tried to install the package on their test machine it didn't ask to reboot the machine. It took them several hours but they were finally able to replicate the situation and they did find out that if he closed the application, his machine would have rebooted. It took them even longer to figure out that the machine was still stable and how to get out of the package installer without having the server reboot. My co-worker was on tenterhooks during this whole time, and there was nothing he could do to fix the situation. He has to sit there and wait by the phone for the vendor's support people to call him back. I realized then and there, that I couldn't have handled that situation as well as he did. It would have been hard for me to allow the fate of my responsibility to be determined by someone that I didn't know and didn't necessarily trust. At least when I had to find the bug I trusted my own skill. I was afraid but I was in control.

Supportability

The biggest fear of the corporation is that all the developers who understand the application and who have worked on it leave the company and they are left with a completely unsupported unsupportable application. The way that we found to address these fears is to take all of our work and release it to the Internet under the GPL. This has many benefits not the least of which is it benefits the rest of the Internet community.

In having worked with both commercial and free software, I must say that there is a difference. Most of the commercial software is distributed in binary format rather than source form and this difference strikes at the heart of one of the problems associated with custom applications developed in house. I call this problem the ''precious binary'' problem. With many commercial pieces of software, they are built in carefully controlled environments. The developers do whatever it takes to get the software to compile and run properly and then they have the one precious binary which they package up and distribute. However, it would take a great deal of time and effort to create that binary in another environment. This is one of the problems with software developed in house. Most of the time, the developers create the binary on their machine and then they distribute it out to their little portion of the world. A potential problem for a company could arise if the developer who created this precious binary leaves and no one can get his source to compile. By distributing it out to the Internet in source format, we the developers have to make it easy for someone else to create a binary. Since the compilation process is done many times with people who have different environments, the compilation process is well wrung out so that it can be repeated by people other than the principle developers.

Another problem that plagues programs developed in house is that source is all too easily lost. A developer leaves and the source trees are left stagnant for a while and eventually they get lost or deleted. If the source is handed out widely then the chances of it getting lost are much smaller.

Another problem for custom applications is that they all too frequently are not documented very well. The person who wrote it knows how to use it and never quite finds the time to write down the documentation. When you release something to the Internet, other people are going to begin to use it. They need the instructions and so the developer must take the time to write them down so that others can make use of their work. Also if the developer hopes to reap some of the rewards of Open Source development and have someone else contribute bug fixes to his program, then he will need to document his source to some degree or another. Also since you know that your peers will be looking at it, you try harder to make sure that the code is good and clean.

Another benefit of releasing code developed for in house applications is that other people will test it more throughly than you can. For example, one of the programs that I released did SNMP queries on printers. I thought that I had tested it very extensively but there was one set of SNMP that crashed one kind of printer.[4] I was one day away from releasing code that made heavy use of that function to all of Cisco when someone reported that problem to me. It saved me from a significant print services interruption.

A common argument against using custom software in a corporation is that if the developer leaves then the company has no hope of finding someone else who has used it. With off commercial software they can hire someone who has used that software before. If the software is released to the Internet, the company can potentially hire someone that has used the program before.

Another way that we convinced management that releasing software to Internet was in its best

[4]The command npadmin --languages crashes HP 5M and 5N printers with a 79. This turns out to be a bona fide bug in HP's 5M and 5N printers rather than a problem with npadmin. At the time of the writing of this paper HP has yet to provide a fix for this problem.

interest was that once the software is released it somehow becomes personal to the developer. I know that I will be working on the software I wrote for Cisco long after I leave Cisco. Even though it is technically owned by Cisco, it is still my baby. I would never do this for a piece of software that remains locked within a company.

Finally there is the PR value, Cisco likes to hedge its bets. In the early days, that meant that they supported Appletalk, TCP/IP, and IPX/SPX. Now it means that they don't care which wins, DSL or cable modems, they make equipment for both. While they have a strategic relationship with Microsoft, they are also ''Empowering the Internet Generation'' and so they want to be seen as a friend of Free Software. If Linux ever supplants NT they will be able to point a finger back at the print project and say that they have been a long time friend of Free Software.

As a lowly developer I do not know how much stock they put in any of these reasons. I do not know what was in the managers' minds when they accepted our proposal. However, all of the above elements were in our sales pitch to management.

Author Information

Ben Woodard is 27 years old and lives by the beach in Santa Cruz, CA with his wife Nina and his cat Folger (he's black and wakes you up in the morning). He a Linux fan who loves the wide open spaces provided by OpenSource software and he works as a ''System Administrator'' at Cisco Systems Inc. but does everything he possibly can including recoding applications to avoid doing any real system administration.

Conclusion

Managing a large organization's printers can be done with a rather small staff. It just requires quite a lot of ingenuity, the flexibility of Unix, and some sysadmins that really are truly lazy through and through and will do anything to get out of boring work.

Reach the author at <bwoodard@cisco.com>.