Design and Implementation of a Failsafe Print System

Giray Pultar – Coubros Consulting LLC

ABSTRACT

This paper describes a printing system designed for an environment with several hundred printers, such that there is no single point of failure, and the print service continues to be available to new print jobs, even if any part of the print system fails.

The paper also describes the following ideas implemented in this system:

- use of a single queue name (myprinter) for almost all print jobs from the client
- use of dynamically created print queues for each user (user-<username>),
- use of dynamically created print queues to manage low cost desktop printers, that are directly attached to a users display device (local-<xterm>),
- integration of the print service with the Zephyr notification system to send print progress messages, error messages, as well as route print jobs based on user location as reported by the notification mechanism.
- ability to route print jobs from legacy systems to all printers in the system, without having to define all the printers on the legacy system. (VM, VMS)

The system was implemented using LPRng [1], taking advantage of its job routing, control file rewriting and dynamic printcap features.

Overview

This paper describes a printing system designed for the research and development division of a pharmaceutical company. Because of its interaction with the FDA (Food and Drug Administration of the United States government), this division produces a large amount of documentation that needs to be printed on short notice. Therefore, printing is one of the top items on the list of ''mission critical'' systems.

Some of the features of the print system as implemented are:

- Banner pages on colored paper
- Pooling of printers under one queue
- Translation/mapping of usernames from other systems (VM/VMS/NT)
- Graphical user interface for queue management
- Pop-up notices when jobs are queued, being printed, and completed.
- Printing to inkjet printers connected to X terminals.
- printer defaults per user

The Old System

Like most environments, there was an existing print system in place. As much as possible, the team tried to treat the project as a ''design from scratch'', but in many instances the existing system affected the decisions made in designing the new system.

There were many disadvantages to having an existing system in place: Many users wanted the capabilities of the existing system replicated in the new system, even if these ''capabilities'' were quirks of the old system. Some users wanted ''all'' the features of the old system to be available in the new system, even though some of the features were not needed anymore. The technically advanced users had their own ideas about how the system could be fixed; and wanted their (in most cases stop gap) fixes implemented.

On the other hand, there were some advantages of having an existing system: Users were aware of their specific needs that were not being met with the current system; and were able to articulate their requirements for a new system well. Some of the well conceived but poorly implemented features could be re-implemented in the new system so that they would work. Also, the old system could be used as a test lab, or a rapid prototyping lab, to demonstrate some new ideas. It would then be easier to see the users' reactions to these new ideas that would be fully implemented in the new system.

It is therefore difficult to characterize whether the old system was an asset or liability in designing the new system.

Here are some quotes (some made up, others as best as I can recall) from users about the old printing system:

- ''I got this printer on my desk 3 weeks ago. When are you going to set it up?''
- ''Oh, I sent this job to the printer this morning. I hope I can get it before I go home.''
- ''I always print twice. One of them never comes out anyway.''
- ''Why can't I print from application A to printer B?''
- ''How was I supposed to know that he was printing 2000 pages. If I had known, I would have used the printer next to it.''

- "There are 5 printers in this room; and not a single one is printing my document!"
- "I hate those 'your print job is printed' messages. Can't I turn them off?"
- "The Payroll application can't print on my desktop printer. So every time I print something out, I have to run like mad to the printer room."

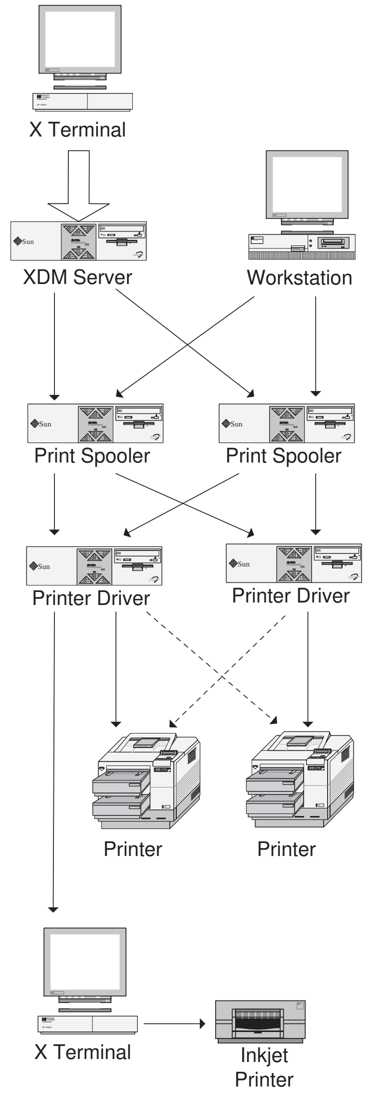

Figure 1: High level Physical design.

The Old Architecture

The old system consisted of two hosts that were dedicated to printing. All servers and workstations were configured with queues that would forward print jobs to server A. Server A would then talk to all the printers and try to send the jobs. Server B was supposed to be a backup for server A. Server B's configuration had not been maintained for a long time: it is unlikely that it could print anything without significant reconfiguration. Moreover, one would have to reconfigure all the workstations and servers to send jobs to server B instead of server A, in case it failed.

Administrators' Problems

From an administrators point, this system was designed to solve the following problems that were being faced:

- How can one manage (create, remove) hundreds of printer queues, if every user wants to have a printer on their desktop?
- How does one set up printer lists in each application, so that users can print to any one of the hundreds of printers available?
- How does one get print jobs from non-Unix systems to print to their printers, with cover pages with Unix login names, and proper accounting, without too much configuration on either the Unix or non-Unix side?
- How does one distribute printer configuration (aka /etc/printcap) information to all their workstations/XDM servers?
- Can one build a print system so that if any of the print servers go down, all printing service can continue?

The Design

The print system consists of two sets of identically configured host(s): The Print Spoolers (PS) and the Printer Drivers (PD).

The purpose of the print spoolers is to receive print jobs from clients, make any user name translations (for non-Unix systems), and/or formatting adjustments (such as landscape, 12 cpi, etc.) and send it to the appropriate PD server.

The purpose of the printer drivers is to communicate with printers and manage the pooling of printers.

A print job from a client (such as a user's workstation) goes through the following steps:

1. client contacts a PS server and sends the job
2. PS server makes any adjustments necessary to the print job
3. PS server sends jobs to PD server.
4. PD server puts job in a pool queue, and waits until a printer in the pool is available.
5. PD server sends jobs to the printer.

This flow is also covered in the section "Life of a Print Job".

Redundancy

As shown in Figure 1, there are several redundant paths a print job can take from a client to the printer. The redundancy of the system is achieved via the mechanisms below.

Client Contacting a PS Server

The clients are configured to send all print jobs to host 'printhost'. This hostname, using DNS, resolves to a round robin list of addresses of all the PS servers.

The client will try all the addresses in turn, until it succeeds in sending the print job.

If a PS server is down, the client will try the next one on the address list. If all the PS servers are down, the lpr client will reject the job at the client rather than queueing the job.

PS Server Contacting PD Server

The queues on PS servers are configured such that each queue is assigned to a deterministic list of PD servers. The PS servers go through the list until they can send the job.

If a PD server is down, the PS server will try the next PD server listed for that queue.

PD Server Choosing a Printer

The printers are arranged in pools. Users will typically submit their jobs to one of these pools. The PD server picks the next available printer in a pool to send the job.

If a printer is down, the PD server (after scheduling one print job that will never complete), will stop sending jobs to that printer as it is not available, and use other printers in the pool.

Failsafe Requirement

The requirement that was stated as: ”If a component of the print system fails, the system should continue to function for any NEW jobs submitted to the system. The system will also try to complete as many as the existing print jobs as it can.”

The implications of this requirement is discussed in the discussion section.

Detailed Configurations

Client Configuration

The clients that are using this system can be classified in two broad categories:

- hosts with LPRng client software
- hosts with LPD server software

The hosts with LPRng client software are hosts typically under our administrative control. There are two items required on these hosts to get printing working.

1. The lpr binary from the LPRng package
2. The /etc/lpd_client.config file. The main purpose of this file is to indicate where to contact to send print jobs. (e.g., printhost.domain.com). It is unlikely that this file will change. Thus, it is unlikely that the configuration will ever have to be updated on the clients.

Most of the clients not under our administrative control could communicate using the LPD protocol.

```
# $Id: lpd_client.conf,v 1.3 1998/05/05 16:52:19 giray Exp $
default_remote_host=printhost.domain.com.
use_identifier
use_queuename
originate_port 2000 3000
check_for_nonprintable@
```

Listing 1: Sample /etc/lpd_client.conf file.

```
# $Id: printcap.pl,v 1.28 1.3 1998/09/05 13:51:16 giray Exp $
# PS servers
spoolhost:This is a print spooler host:::server:PS
otherspooler:This is another print spooler host:::server:PS
# PD servers
badhost:This is a printer driver server:::server:PD
goodhost:This is another printer driver server:::server:PD
# A pool of printers (aka floor server), floor_0 with the two printers
# rdprint3 and rdprint4, using the PD server badhost, and if badhost
# is unavailable then goodhost
floor_0:Basement Printers:visible:badhost,goodhost:floor:rdprint3,rdprint4
# A HP 4si queue for the printer with hostname rdprint3.domain.com.
# Note that this printer is not directly visible by users, but
# is part of the floor_0 pool.
rdprint3::::hp4si:rdprint3.domain.com
# A remote printer: using LPR/LPD with rp=pr11xcs1 and
# rm=mvs.cmis.domain.com. Send print jobs to this printer using LPD.
MVS_pr11xcs1:Remote pr11xcs1 on MVS:::remote:pr11xcs1@mvs.cmis.domain.com.
```

Listing 2: Sample printer.conf file.

We would typically request the addition of one queue on their hosts pointing to the PS servers.

For example, one would set up a queue on VM that is configured to forward jobs to a queue called ‘‘fromvm’’ on printhost.domain.com. (the print spoolers).

The operation of these queues from legacy systems is covered in the ‘‘Legacy Printing’’ section.

PS and PD Server Configuration

The PS and PD server are set up as typical LPD servers [3], with one major exception: Instead of a static printcap file; the printer configuration information is generated dynamically.

LPRng has the ability to use a script to generate printcap entries instead of using the traditional /etc/printcap file.

For this purpose, we have developed a perl script called /etc/printcap.pl that generates this information using a config file of our creation called printer.conf.

```
floor_0|Basement Printers
:sd=/usr/spool/lpd/floors/floor_0
:as=|/usr/local/lib/print/queued
:bq=floor_0@badhost,goodhost
```

Listing 3: Sample output from /etc/printcap.pl on a PS server.

User Specific Queues

The user queues are queues of the form user-<username>. These queues only exist on PS servers, and are used to route jobs from a user to their selected printers. The main purpose is to make it easy for applications to print: they can print to user-<username>, and the print job will go to the user’s selected printer.

This mechanism uses the dynamic printcap feature in LPRng: LPRng will execute the dynamic printcap script /etc/printcap.pl (described earlier) when looking up the printcap entry for user-<username>.

The script, will then look up the user’s preferred printer, and return a printcap entry that routes the job to the user’s preferred printer. The script, if necessary, create a spooling directory for this user. Therefore, when a new user tries to print for the first time, their queues is automatically set up; and no administrator action is necessary.

If the user has selected ‘‘My Desktop Printer’’ as their preferred printer; then this script will look up the user’s location via Zephyr, and route the job to the appropriate local queue (local-<hostname>), described below.

Local Queues

The local queues are queues of the form local-<hostname>. These queues exist both on PS and PD servers, and are used to route jobs from a user to the printer connected to the X terminal that they are logged in on. All the X terminals in this environment are equipped with parallel ports, and several users have connected inkjet printers, mainly for privacy concerns [6, 7].

The main purpose of local queues is to make it easy to route jobs to a terminal that a user is logged in on. Print jobs can be routed to local-<hostname>, and they will go the printer connected to that host.

Similar to user queues, the spooling directories for local queues are created dynamically. As a result, whenever a new X terminal is added, or a printer is attached to an existing X terminal; a queue is automatically created when the first job is queued.

The Most Commonly Used Queue: myprinter

This queue routes its jobs to user-<username> based on the user who submitted the print job. This is accomplished via the routing filter mechanism provided by LPRng.

When a job is sent to ‘‘myprinter’’, LPRng will invoke the routing filter which picks up the username from the control file, and routes the job to user-<username>

DNS Configuration

The system uses multiple address records in DNS to achieve redundancy at the PS level.

An entry for ‘‘printhost’’ is required. This entry will have the addresses of all the PS servers in the system.

X Terminal Configuration

All X Terminals in the environment are configured to accept connections on a specific port, and pass all data received on this port to the parallel port. Using this mechanism, users can connect an inkjet printer to their X terminal.

Support for Other Systems and Legacy Printing

Printing via the LPD protocol:

To allow non-LPRng systems to send jobs to this system, we establish one queue on the legacy system, and one queue on this system for each legacy system that is going to send jobs to this system. This queue is then run through a control file rewrite filter and a routing filter, before being forwarded to the ‘‘myprinter’’ queue.

The control file rewrite filter rewrites the username field in the control file; by doing a lookup of the users username on the foreign system and translating it into their UNIX username. The translation is completely automated, except for the case when the username cannot be found in the mapping table.

There are several reasons for this setup:

- Only one queue on the other system is needed
- The job can be rerouted to any printer
- Using username mapping, proper accounting, and print progress messages are possible.

The major disadvantage, however, is that the username mapping needs to be maintained. It is hoped that in the future, this can be stored in an LDAP style directory.

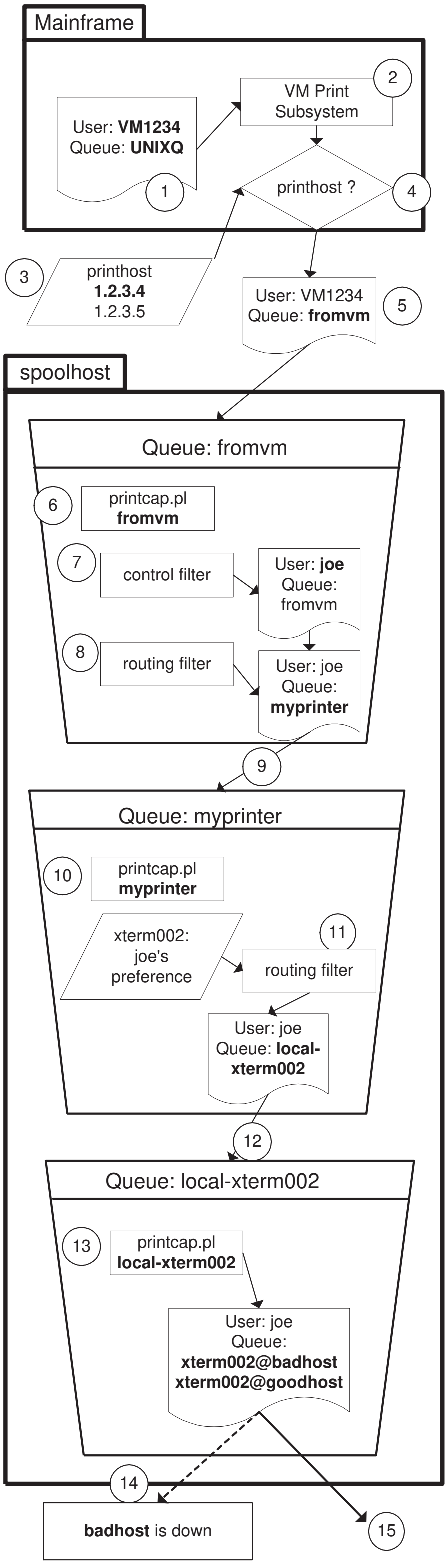

Figure 2a: Logical print job flow.

Printing Via "Port" Functionality in Terminals

Historically, people would use terminals (or dumb terminals) to login to multiuser hosts (e.g., VAX/VMS). Typically, these terminals would also have a printer port that users could attach a printer to.

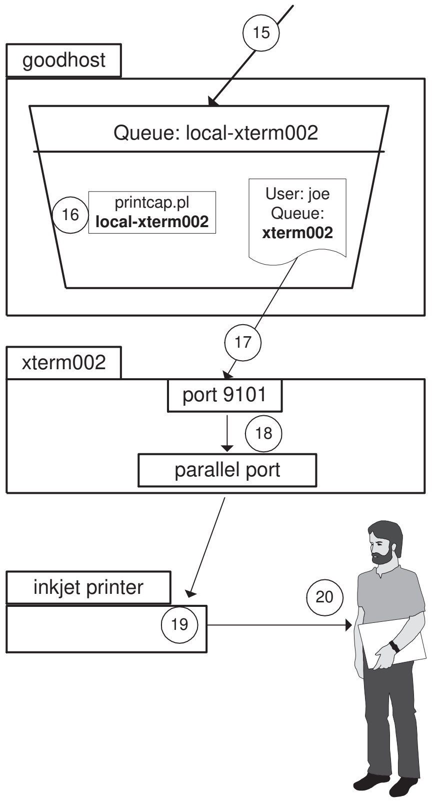

Figure 2b: Logical print job flow.

Instead of having to define all printers connected terminals on a VMS system, they designed a way to print to "any" printer attached to a terminal. The way this works is that the host (the VAX in this case) sends a special sequence of characters that tells the terminal to switch to "printer mode" [4]. Once in "printer mode" the terminal will send all the characters it receives to the printer instead of displaying them on the screen. This continues until the host sends another sequence that tells the terminal to switch back to terminal mode and start displaying the characters.

During a VAX terminal session all characters coming from the VAX are screened using a filter program, that watches for the special sequences that switch into and out of "printer mode".

Once the "printer mode" is switched off, this filter then calls lpr to print all the characters collected.

The Life of a Print Job

This section will describe, in detail, all the processes a sample print job goes through when being printed in order to demonstreate how the different parts of the system work together to get the job printed.

The sample print job is one that originates in the VM system and is destined to a printer connected to an X terminal. In addition, it is assumed that one of the PD servers is down.

- User on VM
- Print Job
- Print Subsystem on VM
- printhost via LPD protocol
- DNS multiple A record
- to spoolhost (PS server) on queue from-VM
- username translation filter
- print job with Unix login
- routing filter
- print job with new destination
- myprinter queue
- routing filter
- print job with user-<username>
- user-<username> queue
- routing filter (look up from settings for desktop printing, look up from Zephyr for location)
- new print job with local-<xterm>
- local-<xterm> queue -
- badhost server local-xterm queue fails
- goodhost server local-xterm queue succeeds
- interface filter
- Xterminal
- parallel port.

1. A user on VM decides to print, invokes the print command and generates a print job.
2. This job is processed via the VM Print subsystem, which will decide to send the job to the ''fromvm'' queue on ''printhost'' using the LPD protocol.
3. VM will perform a DNS lookup to resolve the name ''printhost''. VM will receive a number of addresses associated with this name, and pick one to send the job to.
4. Assume that VM picked the address for spoolhost.
5. VM will then send the job to the lpd daemon running on spoolhost destined for the queue ''fromvm''.
6. The lpd daemon on spoolhost will execute /etc/printcap.pl to get a printcap entry for ''fromvm''. This script will generate a printcap entry that lists that both a control file filter and a routing filter will be used. The lpd daemon on spoolhost will then receive the print job and store it in the spooler directory.
7. The lpd daemon on spoolhost will invoke the control file filter associated with this queue. This filter is the username translation filter, which will rewrite the username in the control file, and replace the user's VM username with their Unix login.
8. The lpd daemon on spoolhost will then invoke the routing filter associated with this queue. This filter is the routing filter, which returns ''myprinter''. The lpd daemon, will redirect this job to the ''myprinter'' queue.
9. The lpd daemon spoolhost will execute /etc/printcap.pl to get a printcap entry for ''myprinter''. This script will generate a printcap entry that lists that a routing filter will be used.
10. The lpd daemon will transfer the print job to the ''myprinter'' queue.
11. The lpd daemon on spoolhost will then invoke the routing filter: This filter will look up the users' preferred printer. Assume that the user has listed ''My Desktop Printer'' as their default printer. The routing filter will then use ''zlocate'' to locate the user, and find which X terminal they are logged into. Assume the user is logged in from xterm02. The filter will return local-xterm02.
12. The lpd daemon on spoolhost will execute /etc/printcap.pl to get the printcap entry for local-xterm02. /etc/printcap.pl will create a spooling directory for this queue if necessary. It will return a printcap entry that lists PD servers to try in a particular order: first badhost and then goodhost.
13. The lpd daemon on spoolhost will transfer the print job to ''local-xterm02''
14. The lpd daemon on spoolhost will try to connect to badhost. After a while, this connection will time out.
15. The lpd daemon on spoolhost will try the next PD in the list. It will try to connect to goodhost. This time the connection will succeed. The lpd daemon spoolhost will send the job to the ''local-xterm02'' queue on goodhost.
16. The lpd daemon on goodhost will execute /etc/printcap.pl to get a printcap entry for local-xterm02. printcap.pl will create the spooling directory if necessary. It will also return a printcap entry pointing at port 9101 of host xterm02.
17. The lpd daemon on goodhost will then open a connection to port 9101 of host xterm02 and send the job.
18. The Xterminal xterm02 is configured to send all input from port 9101 to its parallel port. It will send the print job to the parallel port.
19. The printer is connected to the parallel port and and will print the job.
20. The user will pick up their print job, hopefully smiling, and thinking ''Hey, this is great. I hit print, and seconds later, here's my print job!'' without being aware of all these processes that have gone on in the background.

Features and implementation

This section describes, the features introduced in the overview, and how they are implemented.

Banner Pages On Colored Paper

The banner page program includes the necessary PCL and Postscript code to switch to different paper tray for the cover page and then switch back. The printers are stocked with colored paper on one tray and white paper on other trays.

Pooling Of Printers

This feature is already implemented in LPRng with subserver queues.

Translation/mapping Of Usernames From Other Systems (VM/VMS/NT)

The translation of usernames from other systems is done using the control file filter feature in LPRng.

Once a job is received, LPRng will execute a filter. This filter programs looks for the username field in the control file, extracts the username, then looks up this username in a simple directory; and finally rewrites the control file with the new username.

Graphical User Interface For Queue Management And Preferences

A simple Tk frontend to LPRng's lpq and lpc was developed. Users can look up the jobs in queue and cancel or move jobs.

This tool also allows the user to change their preferred printer, i.e., the printer that user-<username> will print to.

Pop-up Notices When Jobs Are Queued, Being Printed, And Completed.

This is accomplished via the Zephyr package from MIT's Project Athena. Using Zephyr, it is possible to send pop-up messages to users as their print jobs progress through the printing system. The accounting filters in LPRng are set up to send these Zephyr messages.

Printing To Inkjet Printers Connected To X Terminals.

Printing to inkjet printers connected to X terminals is done via the local-<Xterm> queues mechanism described above.

When the user has selected "My Desktop Printer" as their default destination, their user-<username> queue will route jobs to local-<xterm>, by querying Zephyr to locate the xterminal that the user is logged in on.

For example, for a user "joe" working on xterm02, their queue user-joe would forward jobs to the local queue would be local-xterm02.

The PD server for X terminals will send the job to port 9101 on the host specified: If a user prints a job to local-xterm02, the PD server would contact xterm02 on port 9101 and send the job.

It is possible to print to the workstations using this mechanism as well. All the workstations have a daemon running on port 9101 /usr/bin/localprintd (started through inetd), that will send the jobs to the workstation's parallel port.

Printer Defaults Per User

Another simple Tk application was developed that allows the user to edit their printer preferences. Using this application, they are able to select their preferred printer, orientation and see their username mappings for print jobs received from other systems.

These preferences are stored in a simple flat file, which is used by the PS servers.

Modifications to LPRng

The following modifications, merged into LPRng distribution, were made to LPRng.

Multiple A Records

LPRng only used the first address returned by DNS. The code was modified to try all the addresses returned by the name query. (This is similar behavior to most telnet clients).

With this change, clients using the new LPRng lpr client would try contacting all the Print Spoolers, until it found one that worked.

Multiple "rm" Entries In Printcap

Traditionally, lpd servers could only have one remote machine (rm) entry for each remote queue. This was changed to add the ability to define multiple hosts to be tried in order.

This change was necessary to implement failover between PD servers. Each queue has a list of PD servers that it is hosted on. It was necessary that PS servers to try these hosts in order; for queue listing consistency.

Solutions To Administrators Problems

The section describes how this system solves each of the following administrators' problems.

How does one manage printer queues, if every user wants to have a printer on their desktop?

This is solved using the local-<xterm> capability. One can print to any users X terminal/workstation, by sending the job to local-<hostname>. Since the queues and spooling directories are created dynamically, the administrators do not have to get involved in managing these queues. Any user/department can purchase a cheap inkjet printer and connect it to their terminals.

How does one set up printer lists in each application, so that users can print to any one of the hundreds of printers available?

Each application is configured to only print to "myprinter". This queue is automatically routed to the users user-<username> queue. Instead of having the printer manage the list of applications, this functionality is moved into the printing system (more details in "Local Queues" section).

How does one get print jobs from non-Unix systems to printer to their printers, with cover pages with Unix login names, and proper accounting, without much configuration on the Unix side?

The print jobs from non-Unix systems are received on a special queue that does non-Unix username to Unix username mapping. The job can then be routed to ‘‘myprinter’’, and follow the usual print path.

There is only need for one queue on the non-Unix system, which will be used to send jobs to all possible printers on the Unix system. (More details in ‘‘Other systems and Legacy Printing’’.)

How does one distribute printer configuration (aka /etc/printcap) information to all their workstations/XDM servers?

The only printer configuration information stored on the clients is the hostname of the print spoolers, which is the same for all clients, and never changes.

The clients, regardless of queue name, always send the jobs to a print spooler. The solution is to move all the spooling and configuration information off of the clients, and put them on the ‘‘print spooler’’ hosts.

Can one build a print system so that if any of the print servers go down, all printing service can continue?

In this implementation, there are two PS servers, and two PD servers. In theory it is possible to have any arbitrary number.

If a PS server were to fail; clients would connect another PS server using the round robin multiple A record in DNS.

If a PD server were to fail; PD servers will use the next available server defined in the server list for that queue.

Therefore, if any server fails, the print service is still available for any new job. The only loss would be jobs that are in queue on the server that went down.

Discussion

Roll out

The roll out of the new print system was not without its problems.

By the time this project was started, the entire system administrator team had changed. Having noone know how the old system was set up in all its intricate details was a big hindrance during the roll out. It seems, no matter how hard the team tried, there was always, yet another undocumented feature, or an application that used the old system with hard-coded entry points. Parts of the old system that did not initally make sense to the team, seemed crystal clear once the same modes of failure were encountered in the new system (e.g., WordPerfect assuming that hosts were always single user, and see the print system fail in mysterious ways on multiuser XDM servers.)

Another problem was keeping up-to-date with the new versions of LPRng. During the development cycle of the project, there was a new version of LPRng almost every other week, and it contained changes that the project team had contributed or changes that were desperately needed. Of course, there were several occasions where LPRng had changed in a way that had not anticipated, especially in the bleeding edge areas that this system depended on most: control file filters and routing filters.

Failsafe

As stated earlier in this paper, the failsafe requirement of this project only applied to new jobs being submitted to the system.

It would have been a pretty difficult design challenge if this requirement also applied to existing jobs. After all, if a print job is currently on server A, and server A goes down; how does one recover the print job? One would have had to build a system that tracked print jobs on multiple servers simultaneously: not a simple feat!

One side effect of the system as implemented is that queue listings maybe inconsistent. The queue listing requests (lpq) go through the same failsafe mechanisms as the jobs. As a result, if one submits a job, and queries the queue immediately, he may not see his job: His job may be being processed on a different PS server than the one he is using to get the queue listing.

Another interesting side effect is the disappearance of jobs when a PD server is resurrected. While a PD server is down, print jobs that would have been processed on it are sent to a different server. When the PD is back up, the system will automatically switch back to using the PD server for queue listings. However, there may be jobs queued on a different PD server waiting to be printed.

Weaknesses In Implementation

The dynamic printcap generation script, /etc/printcap.pl is currently implemented in perl. This means that a new copy of perl is spawned every time lpd needs some printcap information. Under heavy printing, this can put a nontrivial load on the system.

The PS servers utilize users’ preferences when processing print jobs. Currently these preferences are stored on a flat file on a NFS mounted filesystem. The design was to store these files locally on all PS servers, and when the user updated their preferences, update all copies.

Another weakness is in the implementation of Zephyr. Since the environment had no prior installation of Zephyr, it was installed specifically for this project. During the implementation Zephyr was used, without much regard to how else it could be used. Therefore, to use the current installation of Zephyr as a general purpose messaging mechanism may involve some rework in the print system implementation.

Comparison To Prior Work

In [8], the author uses a number of sections to cover printers, and their configurations. As an observant reader will note, this paper does not cover the configurations (and difficulties associated with) the printers themselves. One interesting feature covered in his paper; that can also be applied to this work in the future, is the assignment of a location to print servers. In the current implementation of this print system, all the PS servers are treated equally. In a future version, it is possible to assign locations to these servers, and have clients contact a print spooler close to their location. The author also has several sections on the importance of automation. In this system, the approach taken was to ''eliminate'' instead of automating a lot of the maintenance work. For example, instead of distributing printer information to all the clients, the system was designed such that the client configuration is not affected by changes to the print system; and have, in effect, eliminated the need to distribute this information.

In [9], the authors describe a print system that they have built using the LPRng package. In their paper, they describe a design that they rejected that is similar in nature to this system: ''in which a master front end machine distributes print jobs to an array of workers'' because they considered the front end machine a single point of failure. This system uses multiple front end machines to accomplish the distribution, and thus does not suffer from the single point of failure. Moreover, in this system, when a server failed all new jobs are automatically routed around the failure; as opposed to the CERN work, which in the authors words: ''In case of a failure of one of the servers, a reconfiguration of the naming service databases suffices to reallocate the queues served by the failed server onto active ones.'' Another comment the authors make in their ''Future Developments'' section is that they would like to implement user notification via Zephyr, which is already implemented in this system.

It is unfortunate that the work described in this paper had already been completed by the time [8] and [9] were published.

Availability

This system is based on the LPRng print system, whose source is available from the LPRng web site [2]

Due to the legal requirements of the company where this work was performed, the additional sources (such as /etc/printcap.pl) can not be distributed.

Future Work

There are several possibilities for future work in this area:

- As suggested in [8], implement the idea of assigning locations to the PS servers. This should make the system scale much better.
- Consider using lbnamed [5] to load balance to PS servers.

Author Information

Giray Pultar received a Bachelor of Science degree in Engineering and a Bachelor of Arts degree in Economics from Swarthmore College in 1994. He worked as a System Administrator at Motorola in the Cellular Infrastructure Group in Arlington Heights, IL until 1996; followed by a similar position at Abbott Laboratories in Abbott Park, IL until 1998. He is currently working as a consultant at John Hancock Funds in Boston, MA. He can be reached at <giray@coubros.com>

References

[1] *LPRng – An Enhanced Printer Spooler System*, Patrick Powell, and Justin Mason, LISA '96.

[2] http://www.astart.com/LPRng .

[3] Man pages: *lpr*, *lpd*.

[4] VT100 terminal documentation – printer mode escape sequences.

[5] *lbnamed, Load Balancing Name Server*.

[6] HP Envizex X terminal documentation.

[7] NCD X NCD 17/19 documentation.

[8] *Building An Enterprise Printing System*, Ben Woodard, LISA '98.

[9] *Large Scale Print Spool Service*, Ignacio Reguero, David Foster, and Ivan Deloose, LISA '98.