Configuring a CUPS Print Server

Printing by numbers

Sharing a printer among multiple users can help save money and resources. Connect your printers to a CUPS server to give access to Apple or even Microsoft clients.

BY TIM SCHÜRMANN

Outmoded computers that have been replaced by newer desktops make ideal print servers on a network. A print server not only allows multiple users to share a printer, but a dedicated print PC also offloads some of the work from your own desktop, allowing it to proceed with its own tasks at top speed. On some home networks, different users will send output to different printers. While dad is busy printing out a mail merge document on a laser printer, the kids might be printing the pics from the last party on their inkjet. This scenario is not far removed from the printing environments in most enterprises, where the Common Unix Printing System (CUPS), gives Linux, Microsoft and Apple clients access to any printer on the network.

First Steps

Two steps are required to run CUPS on a network. First of all, you need to set up the computers to which your printers are attached. These machines can be normal desktops or dedicated servers that do not need a GUI. Today's distros will typically install CUPS as part of the original setup, or provide configuration tools that you can call after completing the setup. The latter category typically sets up CUPS to launch when you boot the system. If your distribution does not have CUPS, or if you want to update the existing version, you can download the source code from the CUPS website [1].

Prior to CUPS, distributions used to work with less powerful BSD or LPRng print systems. Both of these are difficult to integrate in a networked environment. If you have an older distribution, you should consider replacing the older printing system with CUPS. The CUPS documentation provides more information on this subject [2].

It is not necessary to modify your applications to use CUPS. Your programs will either interface with CUPS natively (this is the case for KDE applications), or they will use the so-called System V, or Berkley, printing commands. These are the *lp* and *lpr* command-line tools, which are called by the applications. CUPS provides compatible versions of these programs. These simple commands are useful if your need to send a file to a (remote) using only the command line. The CUPS Software Users Manual [2] provides details on these commands.

Browser-Based Configuration

In contrast to the older LPRng system, CUPS is perfectly adapted to life on the network, as it is based on the Internet Printing Protocol (**IPP**, [3]). IPP is an extension of the Hyptertext Transport Protocol (HTTP), which is used for transferring Web pages across an Internet. This also explains why IPP uses a similar paradigm to data communications across an Internet. A computer, known as a client in CUPS-speak, sends printing data to a CUPS server. The daemon running in the background on the server, *cupsd*, accepts this data, and performs some additional processing, before sending the data to the printer for output. In other words, *cupsd* is the core of the CUPS system.

You can access the daemon in your Web browser, by typing *http://localhost:631*. Simply replace *localhost* in this URL with the hostname of your CUPS computer. *localhost* is a placeholder for the local machine. CUPS does not permit external access in the default configuration, so you will need to launch the Web browser locally on the system you are configuring. If you want to permit external access, you will need to edit the */etc/cups/cupsd.conf* configuration file. Look for the sections starting with *<Location /...>* and ending with *</Location>* (see Figure 1).

The location tags contain details on how CUPS should control access to the various sections. *<Location />* refers to

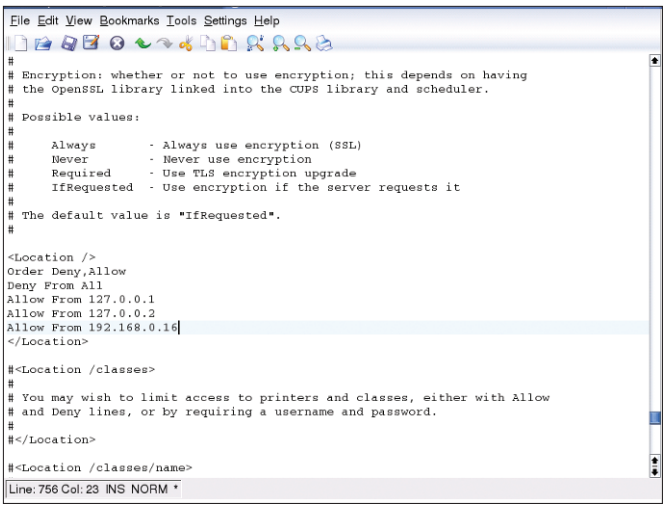

Figure 1: The Allow entry, permits the computer with the IP address 192.168.0.16 to access the Web interface remotely.

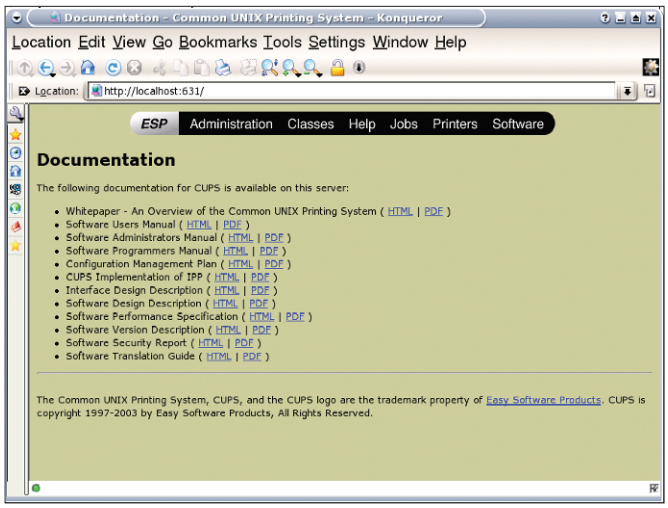

Figure 2: The welcome page of the CUPS daemon's Web interface. This assumes correct access privilege configuration.

the main menu, and *<Location /admin>* to the *Administration* item. Adding *Allow From 192.168.0.16* allows the computer with the IP address 192.168.0.16 to access the interface. For security reasons it is not recommended to allow cleartext passwords to cross the wire. The CUPS FAQ [4] provides an overview of access privileges.

The Web interface allows users to set up and manage CUPS (see Figure 2). Users can also view the jobs in the print queues. The *lpstat* command provides the same functionality. If your distribution has a configuration program for setting up printers, it is preferable to use that program. For example, Suse Linux has the YaST tool. There is a command line based alternative, *lpadmin*, which has a few additional options compared with the approaches mentioned so far [2]. If you do not like any of these methods, you can still opt for one of the many third party configuration programs. Newer KDE versions have a component for this task. You will find it in the Control Center under *Hardware|Printers* (see Figure 3).

We will be using the Web interface in the following examples. The entry for *Printers* in the interface gives you access to a list of print queues (see Figure 4).

Each queue has a name and individual settings. You can assign multiple queues to a printer or interface. The interface that the documents in a print queue will be sent to is defined by a Uniform Resource Identifier (URI). The notation looks similar to a Web page name, for example *parallel:/dev/lp0* for the first parallel port, or *usb:/dev/usb/lp0* for the first USB printer. The *lpinfo -v* command provides a list of available interfaces. We will be returning to the subject of URIs later on.

To create a new print queue, select *Administration* in your browser, and click on *Add Printer*. Then enter the name of the print queue in the *Name* field. The maximum length of the name is 127 characters. The screens that follow prompt you to select the interface to which your printer is attached, the printer model, and finally the appropriate GhostScript filter.

The following steps assume that CUPS has been set up correctly on any computer to which a printer is attached (servers). You can print a test page in the browser window by clicking on *Printers|Print Test Page*.

Cups Revealed

Printing is a simple thing, when you come to think of it. A user selects *Print* in the menu and the program sends the data to the printer, which then creates a pile of paper with the results. In fact, printing is not as trivial as it sounds.

The first problem is the fact that the computer can freeze while the print job is being processed. Most printers simply do not have enough memory to load a whole document. To avoid this problem, the print system uses a separate program, called a spooler, that provides caching in the form of the so-called print queue. The spooler monitors the printer and passes on the next job, as soon as the printer is ready for it. On a CUPS system, *cupsd*, the CUPS daemon (also known as a scheduler) takes care of this. The daemon is typically launched when a machine boots, and hangs around in the background waiting for new jobs.

The huge variety of printer models is another issue. Each printer uses different control sequences, that is, it speaks a different language. An application wanting to send a document to the printer needs to format the data in a way that the printer understands. Unfortunately, to do this, the application would need to speak the same language as all of today's printers. As this is unrealistic, filters translate between the spooler and the printer. These tools translate the incoming data into the printer language before the job is printed. In other words, applications output data in a standard format and pass the output on to the spooler. Linux uses the PostScript format. Adobe [6] developed PostScript as a special programming language for graphics. PostScript capable printers can interpret this kind of data directly and do not need a separate filter.

Open Source programmers invented GhostScript, a program that allows applications to output the PostScript format on non-PostScript printers. GhostScript uses a printer driver to convert PostScript documents to the target language for the printer. When faced with a printer that can't interpret PostScript, the CUPS spooler simply calls GhostScript, which converts the data as needed.

Back-ends are the last link in the chain. A back-end represents a port such as a USB or parallel port. Back-end definitions allow us to add new interface types, which do not currently exist.

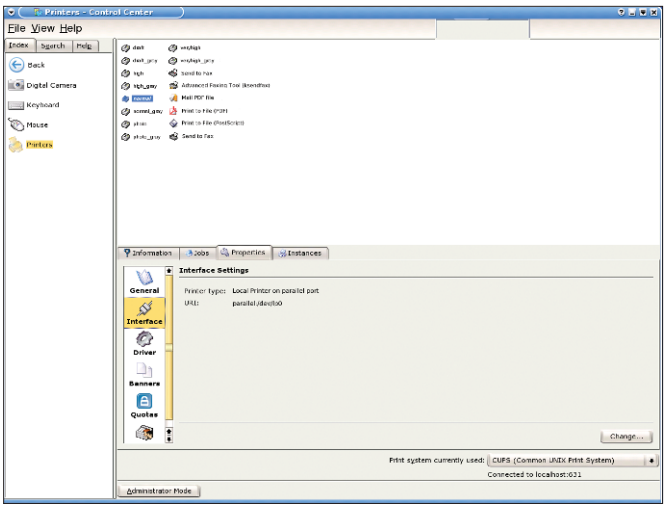

Figure 3: The CUPS configuration item in the KDE Control Center. Click the empty space to display more useful features, such as a function for re-starting the CUPS daemon.

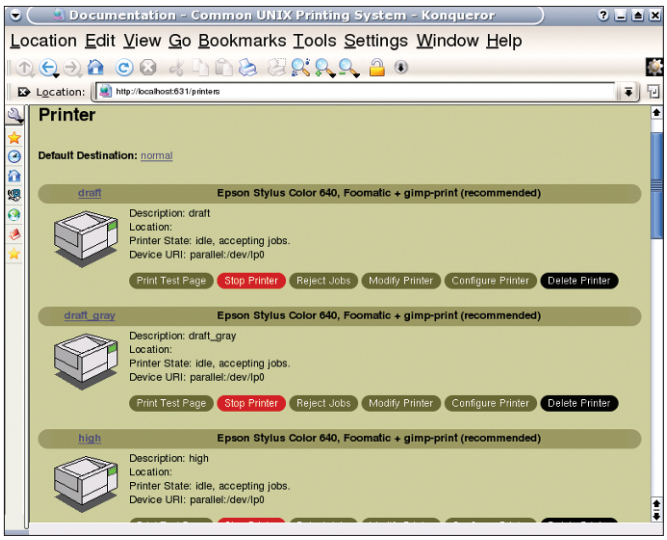

Figure 4: The "Printers" entry lists the print queues that CUPS recognizes, including their URIs (Device URI).

Client Setup

The second step is to configure the computers that will be sharing your printer (clients). There are several possible approaches, the common denominator being the fact that the CUPS daemon, *cupsd*, needs to be running as a background task. Some distributions only launch the daemon automatically on booting a machine if a printer has been set up on the machine. YaST on Suse Linux 9 has an option to set up the server at a later stage. To do so, select *Change|Advanced* and then *CUPS Server*. If you have a distro where the configuration program does not allow this, write a CUPS init script, and add this to runlevels 3 and 5 in your */etc/init.d/* directory.

Introducing the Printer

You need to enter the details of a printer on the client to allow the client to find that printer on the network. To do so, launch the Web interface as described previously. Select the *Internet Printing Protocol* (IPP) as the (*Device*). Then enter the URI for the remote printer, for example: *ipp://myserver/printers/myprinter*. Replace *myserver* with the name. or IP address, of your print server, and *myprinter* with the name of your print queue. The configuration programs provided by most distributions have similar options (Figures 5a and 5b show Suse Linux). Repeat this procedure for each external print queue. Admittedly, this approach is quite convoluted, if you have a large number of printers.

It makes sense to configure CUPS to use a remote print server for every job, to avoid running the CUPS daemon on the clients. This means modifying the configuration file, */etc/cups/client.conf*. Look for the line that starts with *ServerName*. You may need to remove the number sign (#). Type a space, and then the name, or preferably the IP address, of the server that will be processing your print jobs after the *ServerName* keyword (see Figure 6).

The third approach is probably the most convenient. Every CUPS daemon periodically broadcasts its configuration across the network. Clients that receive the signal can immediately start using the queues. The advantage of this approach is the extremely simple config-

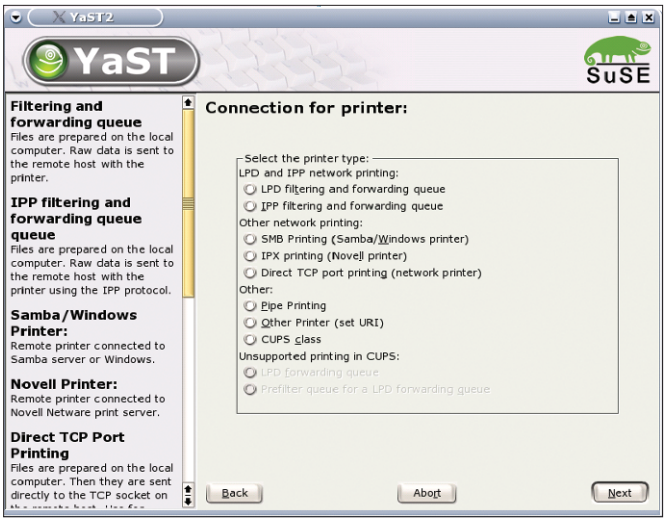

Figure 5a: Setting up remote printers on Suse Linux 8.2. First, tell YaST that you want to access a remote printer that uses IPP ...

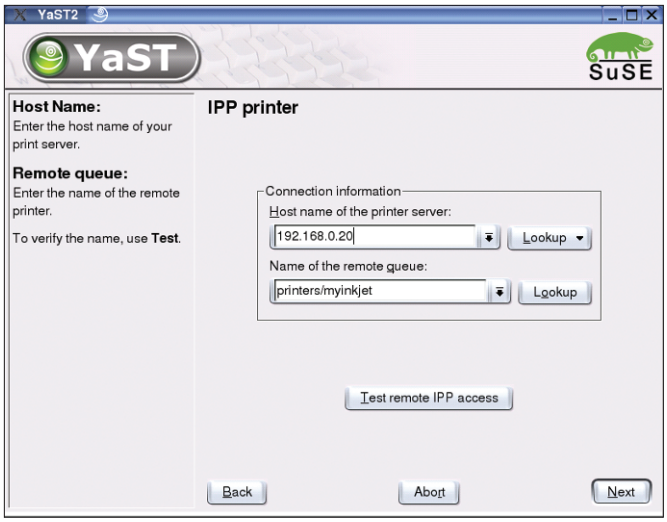

Figure 5b: ... then enter the hostname and the remainder of the URI to complete the configuration.

uration. You simply set up the new printer on the machine to which it is attached to allow any machine on the network to access that printer. Of course, broadcasting is not the most secure approach. Any computer that can receive the signal can access the printer. At worst, this would allow the whole Internet to print to your printer. Browsing is enabled in the old CUPS package, newer versions do not broadcast by default.

Configuration tools allow you to define broadcast options. Again, this means using YaST on Suse Linux. The menu items you need are located in *Change/Advanced/CUPS Server Settings* (see Figure 7). To set up a server for broadcasting manually, modify the */etc/cups/cupsd.conf* configuration file. Lines that start with a number sign (#) are comments that CUPS will ignore. Each setting occupies a single line and consists of a keyword with a value. Refer to [2] for an overview of the values. The file provides short explanations and samples.

If you are configuring broadcasting, you need the lines that start with *Browse*. For example, the value for *BrowseInterval* defines the interval in seconds at which a server will broadcast its configuration data. The *BrowseAddress* entry is followed by the broadcast address. For example, *BrowseAddress 192.168.0.255* sends the settings to any computers on the local subnet *192.168.0*.

Polling is a useful alternative to browsing. In this case, the client retrieves its configuration from the server. In the client-side *cupsd.conf* file, set the value for the *BrowsePoll* keyword to the name or IP address of the server. You can have multiple *BrowsePoll* entries, allowing the client to query multiple servers. Enabling *BrowseRelay* tells the computer to relay any information it has learned by polling to all the computers on the subnet.

CUPS Reloaded

You need to restart the CUPS daemon after changing the *cupsd.conf* file. The command for doing this depends on your distribution. Users with Suse Linux can type */etc/init.d/cups restart*. If you used YaST to change your configuration, the YaST tool also takes care of restarting the daemon. The KDE *Print* dialog will have a list of printers from this point onward (see Figure 8).

If browsing does not seem to work, check your hostname configurations. You may need to add them to */etc/hosts* for broadcasts to work.

Integrating Apple and Microsoft Systems

You can integrate any operating system that supports IPP. This includes MacOS X version 10.2 or later. If you have MacOS, simply check *Printer Sharing* in the system settings below *Services*. This will display the printers as *Shared printers* in the print dialog. Older MacOS versions need the *netatalk* package. Add an entry for each printer to the *papd.conf* configuration file. Refer to [5] for more details. The important thing here is to ensure that you have the correct PPD file from */etc/cups/ppd* for each printer:

```
Description:MyPrinter@MyServer:\
:pr=|/usr/bin/lp -d MyPrinter:\
:op=deamon:\
:pd=/etc/cups/ppd/MyPrinter.ppd:
```

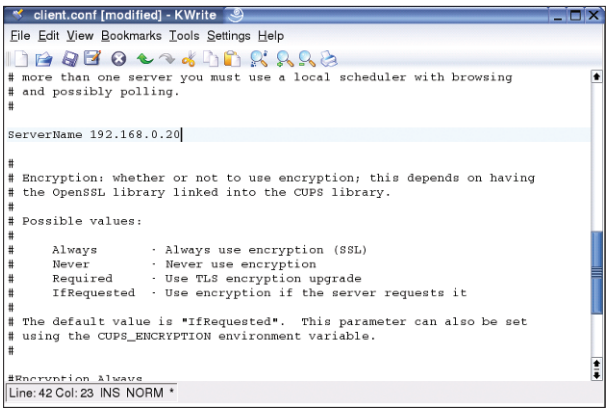

Figure 6: After enabling forwarding, only the computer at IP 192.168.0.20 will need a print daemon – any print jobs will be forwarded to this machine.

Windows 2000/XP also has native IPP support. Any other Windows versions need a working Samba version 2.0.6 or later. Add the following two lines to the *[Global]* section of the *smb.conf* configuration file: *printing = cups* and *printcap name = cups*. Current distributions should have this configuration by default. The Windows client needs a printer driver capable of generating PostScript output, such as the Apple LaserWriter.

If you prefer to use the original printer driver, you will need to set up a new print queue on your CUPS server. Select the *raw* device as the printer model. This tells CUPS to send incoming data directly to the output port. Note that computers without the original driver will not be able to produce useful results on your printer.

Samba 2.2 or later automatically exports the appropriate printer driver when a Windows client attempts to print. *cupsaddsmb*, as used in the following examples, can only handle the Adobe [6] PostScript drivers, or the drivers from its own homepage. The Adobe drivers are available as EXE files only. You will need to run a program such as Winzip to extract them on a Windows machine first. Store the driver files in the */usr/share/cups/drivers* directory. The file names must be capitalized.

Then modify your Samba configuration to allow the fileserver to export printer drivers. The Samba documentation provides details on this. *cupsaddsmb -U root -a* will export your printers. This will put the drivers on your Samba server. Note that *cupsaddsmb* uses the root account to copy the files with the *smbclient* program. Ensure that Samba permits access to this account.

Samba has a new CUPS back-end called *smbspool* that allows Linux computers to access printers shared by Windows machines. Your distribution should set up the back-end correctly. If not, su to root, and enter the following command: *ln -s 'which smbspool' /usr/lib/cups/backend/smb* (backtick). This will allow you to set up the printer just like a remote printer on another CUPS server. Instead of an address like *ipp://…*, use a device such as *smb://workgroup/server/sharename*. If the printer is attached to a computer with NT, or a Windows 9x machine with passwords, you will also need to supply the password: *smb://user:password@workgroup/server/sharename*.

Expert Options

This article only deals with setting up CUPS on a network. The next step would be to set up access and security functions. For example, CUPS allows you to assign printer quotas and passwords. This would not typically make sense on your home network, unless you happen to have an expensive color laser, that is. Admins are well-advised to check out the documentation.

Classes can also provide useful options. A class is a set of printers. When a print job is sent to a class, CUPS prints the job on the first idle printer in the class. Again, the Web interface provides a convenient simple approach to setting up classes via the *Classes* menu entry.

The documentation provided by your distribution may contain some useful troubleshooting information; if not, check out the detailed online documentation for CUPS at [2]. The protocol files below */var/log/cups* can also provide useful hints. ■

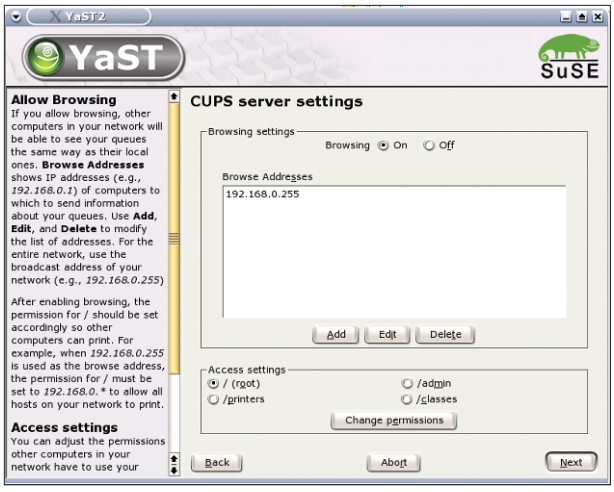

Figure 7: Enable browsing in Suse's YaST. In our example, details of the attached printers are broadcast to the 192.168.0 subnet.

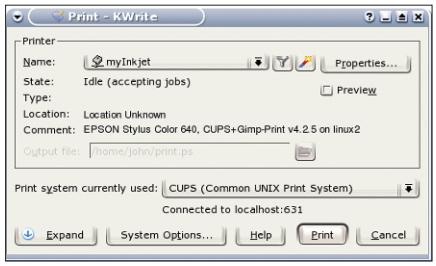

Figure 8: After completing the setup, the KDE print dialog should have a list of remote printers and their details.

Glossary IPP

The Internet Printing Protocol was created by the Printer Working Group. Although it is based on HTTP 1.1, it listens on port 631 (rather than port 80). In fact, the two protocols are so closely related that IPP implementations like cups can be accessed natively via HTTP. The printer URI would be *HTTP://servername:631/...* rather than *IPP://servername/...* in this case.

INFO

[1] CUPS project: *http://www.cups.org*
[2] CUPS documentation: *http://localhost:631*
[3] IPP: *http://www.pwg.org/ipp/*
[4] CUPS FAQ: *http://www.danka.de/printpro/faq.html*
[5] Netatalk: *http://netatalk.sourceforge.net*
[6] Adobe: *http://www.adobe.com*