Samba Printing

By Jeremy Allison

Development Team

email: jra@samba.org

Windows network printing.... "What Lies Beneath"

- Windows developed from a single user system, which means a printing API (application programming interface) was developed first.
  - A network protocol (several in fact) was grafted on afterwards to provide remote printing using this API.
- UNIX printing started with what is essentially a protocol (programs writing text data into a pipe) and has not developed much beyond that.
- Windows printing undoubtedly is easier for users and programmers to use. However, many horrors lie beneath.........

How Windows Printing works (but which one ?)

- As with most things in SMB/CIFS, there are threee different ways to print within Windows.
- Original simple print spooling path, no concept of drivers.
- Windows 9x point-and-print path, uses "RAP" calls to send jobs, receive info.
- Windows NT/2000/XP and future uses DCE/RPC (distributed computing environment/remote proceedure calls) building a <u>very</u> complex printing system.

The original DOS network printing system

- Very similar to the original DOS printing mechanism of opening a PRN: or LPT: device and sending data down the handle.
- Remoted over the SMB protocol as the SMBsplopen call.
- Rudimentary support for monitoring printer queue status.
  - Query call allows data on job id's, job status (printing/held/error) time of submission to be returned.
  - This is very similar to the output of lpq on UNIX
- Note: no concept of drivers in the OS - each application has their own printer drivers.

The Windows 3.x printing system

- This was the first Windows to have the concept of application independent drivers.
- However, drivers had to be installed on each client - the concept of drivers being automatically downloaded to clients was not added until Windows 95 (correct me if I'm wrong :-).
- Created the "virtual printer" API that is the basis for all Windows applications today. Note it's an API not a protocol.
- Created a (Windows specific) page description langage allowing a metafile to be sent to a server.

Windows Printing Abstractions.

- OpenPrinter() creates a handle that represents a printer state.
- The current state of a printer in a Win32 client is represented by a large blob of data.
  - Called a "DEVICE MODE" or DEVMODE.
  - Changes to this view of the printer by the SetPrinterData() call are things like "portrait" and "landscape". Held locally on the client.
  - Contains driver specific private fields.
  - Specifies whether this driver emits RAW data (printer PDL data) or EMF (enhanced meta-file) data (network printer spooler will convert from Windows format to printer PDL).

The Windows 9x print system. A step forward

- Windows 9x has the capability to have the client automatically download the driver files for a connected printer at setup time.
  - This allows an administrator to determine at install time what driver is used for which printer on the client.
  - Windows 9x clients use RAP calls (Remote Proceedure calls tunnelled inside SMBtrans commands on the IPC$ share) to get a block of information about the driver to download.
  - This includes the pathnames (including share) to download, and information about the printer (name, location, comment strings etc).

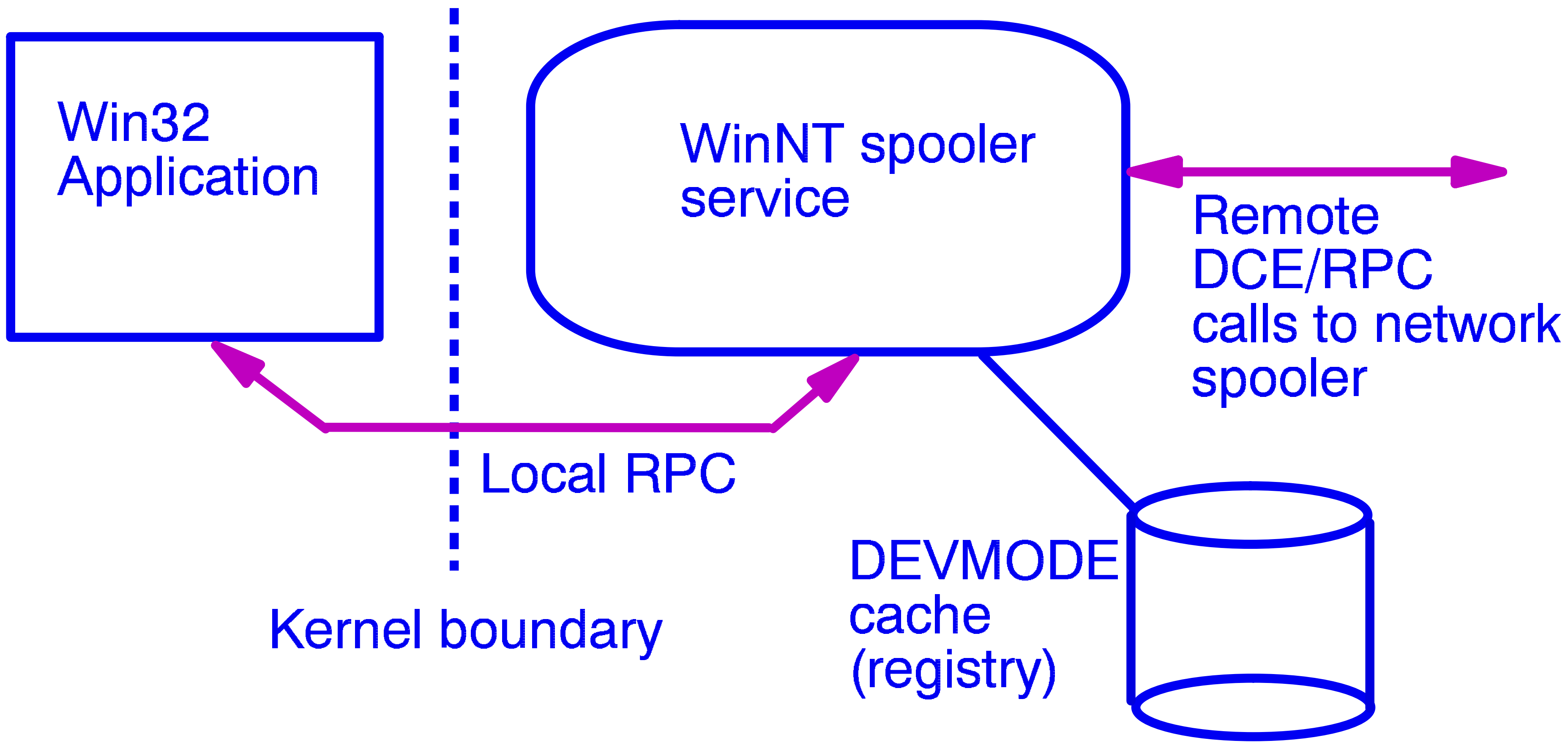
Windows 9x printing (continued)

- Job control calls were added to do things like pause jobs/cancel jobs/purge print queue.
- The Windows 9x print monitor application is written to use these network calls underneath.
- On Win32 calls opening a printer handle, a local DEVMODE is created on the client and all changes are held there.
- Some Win32 printing calls have no effect on a Windows 9x machine (no way to remote them to a print server).

Windows NT Printing (the dream becomes a nightmare)

- Windows NT/2000/XP use a completely different method of printing.
- SMB calls open the IPC$ share, then open the \\SPOOLSS pipe beneath it.
- DCE/RPC calls are done to SPOOLSS pipes to open printers, create printers, add drivers.
  - Anything that can be done locally via the Win32 printing API's is remoted via this method.
  - Unfortunately the application Win32 calls are not directly remoted to a network print server. The spooler service is between. This is a very fragile piece of code (even for Windows).

Windows NT printing system



Windows NT/2000/XP Point and Print concept.

- Printers (representing queues to different printers) have data structures called DEVICEMODES attached.
- Standard capabilities are stored in the DEVICEMODE.
- Non-standard capabilities are created by the Win32 printer driver code and stored as key/value pairs associated with the printer - sent to the print server.
- Notification mechanism allows server to notify clients on capability changes and on printer status changes.

Windows NT Point and Print (continued)

- Translation can be done either on the client (RAW) print type, or by sending a metafile to the server (EMF).
- Metafile on the server depends on associated driver code being run on the server - Samba cannot do this.
  - Special per-printer DEVICEMODE sent when printer handle opened to enable remote translation (EMF).
- GUI representation of capabilities shared by system print dialog and by Win32 driver code.
- Printing (and printer administration) security done by associating Win32 ACLs with printer object.

Windows Point and Print (continued)

- All print communication done using DCE/RPC calls over SMB.
- Print path starts with printer handle being opened.
  - Print "Job" submitted into queue (job ID returned).
  - Data spooled into job.
  - On "close" then the print is started.
- Backchannel notification very poorly done (reverse SMB connection from server to client).
- Standard job commands (enumerate, delete) and queue commands (pause, resume, purge).

Why printing in Samba is so complex

- What this means is Samba must support all three different systems in order to compete in the Windows file and print market.
- But it gets worse.....
  - The DCE/RPC implementation of the remote calls in Windows is <u>unbelievably</u> complex. PDC emulation is easy compared to this.
  - The implementation is <u>APPALLING</u>. Send an incorrectly formatted packet back to the spooler, it crashes. Send data it isn't expecting - it crashes.
  - The implementers did not understand network protocols. At all.

Why printing in Samba is so complex (continued).

- A multitude of client application Win32 calls are coalesced in the spooler service and end up as one or more calls on the wire.
- Many of the Win32 calls are simply remoted (easiest to figure out). Some are not - completely new structures and data types seen on the wire.
  - This has been the hardest interoperability problem the Samba Team has ever faced.
- Approximately 5-10 person years of work has gone into this code.

Why the Windows NT printing model is broken.

- In implementing the Windows NT printing model we came across several glaring design flaws.
- Driver binary dependence.
  - In order to correctly load a driver on a Windows network printer server it has to <u>EXECUTE</u> on the server that it is being served from.
  - This is a severe disadvantage if your server is a Windows MIPS or Alpha CPU and your clients are x86 (I guess that's why Windows doesn't run on anything but Intel. No, WinCE doesn't count :-).
- Driver versioning doesn't work.
  - Many drivers share file names. All drivers of the same type (user or kernel) are put into one directory.

More complaints about Windows printing....

- There is no protection from race conditions.
  - Two printer admins installing drivers at the same time could ruin each others day :-).
- Between Windows NT 4.x and Windows 2000 the drivers changed from running in kernel mode (NT4.x) to user mode (2000 and above).
  - As the old drivers need to work, Windows 2000 has to support both.
  - A new directory structure was added under the magic printing share to support this.
- The print subsystem looks like it was cobbled together by sophomore (1st year) CS students.

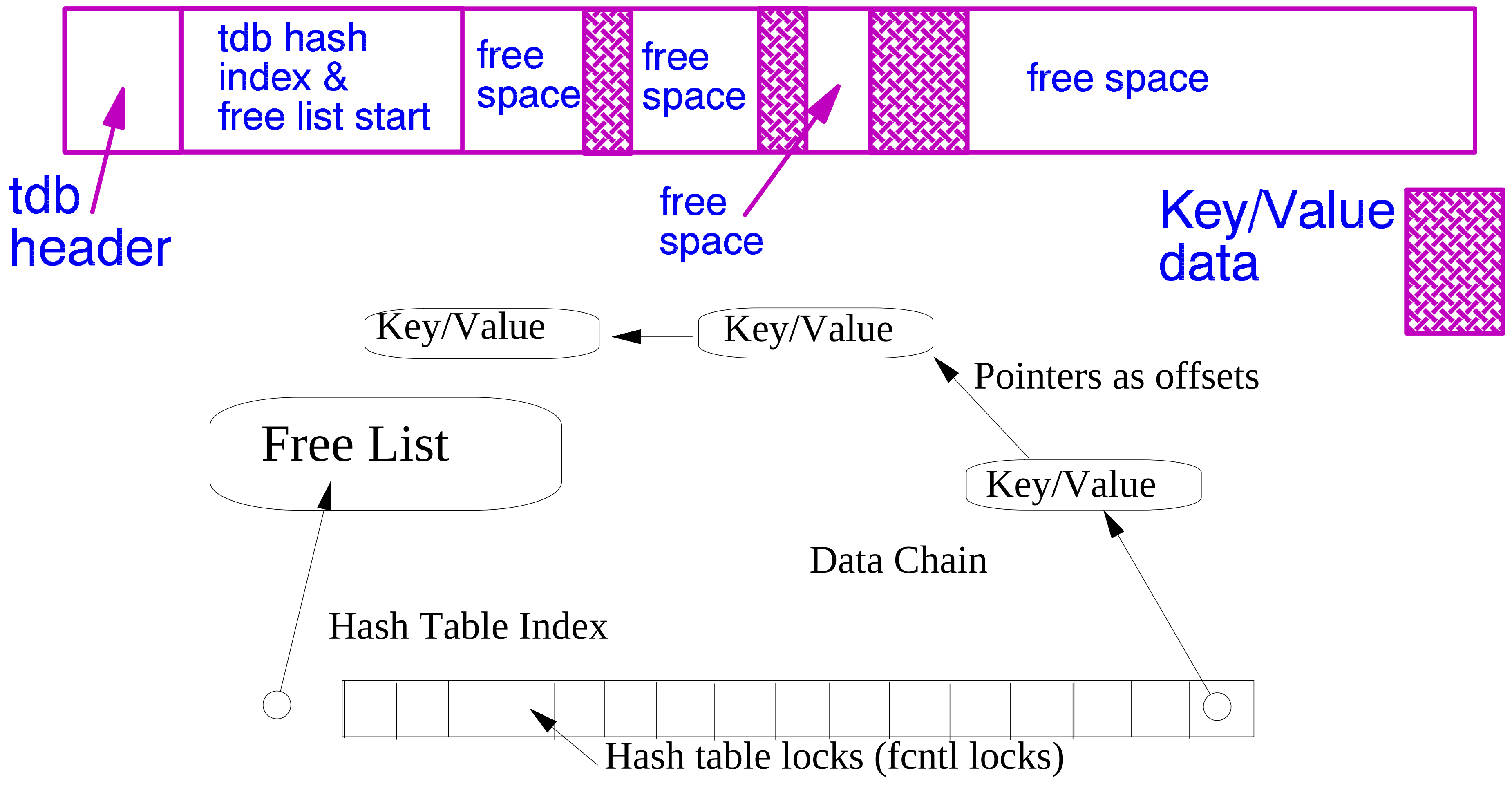
Samba Printer Code

- Implements Windows NT/2000/XP "point and print" interfaces.
  - Provides driver download.
  - Provides remote store for capability data.
  - Maps Windows "printers" onto UNIX print queues.
- Depends on drivers doing the data translation (rasterization/conversion to PS etc.) on the client (RAW mode - no EMF).
- Uses internal tdb database to store capabilities such as DEVICEMODE and key/value pair data.
  - Can fail with drivers that are expected to be run on the server.

An Aside about TDB

- Developed from our original shared memory code for distributed file open data between smbds.
  - Re-written mainly by tridge. Uses mmap and cleanly falls back to read/write.
  - Allows multi-simultaneous readers and writers to access data as key/value pairs.
  - Now used as the back-end store for most parts of Samba (the Samba registry :-).
  - Code is GPL - NOT LGPL !

TDB Internals



Samba Printer Code (continued)

- Printer tdb acts as a registry store for printer capabilities.
- Print queue tdb keeps track of Win32 submitted jobs. Associates Windows data with underlying UNIX spooling data (as returned from lpq).
- On job submission a job entry is created in the queue tdb.
- Data then spooled into a tmp file.
- On completion job submitted into UNIX print system using internal Samba vectored API.
- Polling used to report print status.

Reporting Print Queue Status

- Many Windows clients "poll" when a print monitor application is open.
- Running lpq for each query request would kill server performance.
    - The last queue status is cached with a timestamp.
    - Very complex code to ensure multiple writers can update/query simultaneously.
    - Full database traversals are the most costly.
- UNIX print jobs are mapped into a "smbprn" job name when reporting back to a Windows client.

Samba Interface to UNIX print system

- Kept as simple as possible. Consists of operations :
  - get_queue
  - pause_queue
  - resume_queue
  - job_delete
  - job_pause
  - job_resume
  - job_submit
- CUPS currently only real API user. Others map UNIX commands (lpq,lprm,lpc) under interface.

Samba Interface to Windows Printer manipulation

- To allow Samba based "print appliances" smbd calls external scripts on events like "add printer", "delete printer" etc.
  - These scripts allow UNIX print queues to be created/deleted etc.
- tdb based messaging system is used to notify smbd's of change events (such as "add job", "delete job").
  - These events are used to send notifications back to the client.
  - Event model not fully understood yet.

Win32 Printer capabilities in Samba

- DEVICEMODE stored per printer object in tdb.
- Security : ACLs stored per printer in tdb.
  - Accessing user checked against stored ACL before allowing desired access.
- Generic key/value access provided by GetPrinterData()/SetPrinterData()/EnumPrinterData() calls.
  - These can set arbitrary capabilities and enumerate the list
  - Treated as "blobs" of typed data and stored in the tdb.
- No mapping between UNIX capabilities and Win32.

Setting up a Samba Printer

- A driver needs to be bound to the client view of a printer.
- "Printer" administrator must bind a driver to a UNIX print queue.
  - Clients then transparently download and install this code.
  - Users don't need to know printer type or how it is configured.
- Driver takes care of GUI dialog capabilities. Changes are stored on Samba server and sent via notification to other clients.

Known Problems

- No way to ensure Win32 printer driver associated with a printer by the printer Admin is correct.
- No way to ensure capabilites set from Win32 match to capabilities set under UNIX (must trust print admin).
- Even with perfect UNIX API, transition to use extra features will be slow as Samba must compile on many older systems.
- Win32 printer status decoupled from UNIX printer status (must pass through narrow API) and error codes don't always match.

Futures

- The UNIX printer API must get fixed and become richer.
  - There are various groups attempting this.
  - IBM omnidriver group.
  - Open Source Printing group (headed by HP).
  - CUPS is currently the best choice for an API (IMHO).
- Postscript printers handled well under UNIX/Linux
  - Inkjet driver improvements sorely needed.
- Microsoft are moving the goalposts with the new UNIDRIVER in Windows XP and beyond....

Resources

- Main Samba Web site :
  - http://samba.org
- Newsgroup :
  - news:comp.protocols.smb
- Samba discussion list :
  - email: samba@samba.org
- Samba development list :
  - email: samba-technical@samba.org