Samba4 – A New Beginning

Andrew Tridgell
Samba Team
Major Features

• The basic goals of Samba4 are quite ambitious, but achievable:
  • protocol completeness
  • extreme testability
  • non-POSIX backends
  • fully asynchronous internals
  • flexible process models
Protocol Completeness

• CIFS/SMB is a huge protocol, but is not infinite.

• In previous versions of Samba we implemented new protocol elements “on demand”, only adding an element when we saw an application using it.

• In Samba4 the new attitude is “implement everything”
Old testing method

- The Samba project has previously developed testsuites of 3 main kinds:
  - ad-hoc tests for a range of specific conditions
  - full-coverage tests for a very small range of operations
  - randomised testing for a very small range of operations
- This approach did work to some extent, but suffered from some major drawbacks:
  - many parts of the protocol remained completely untested
  - many fields untested within the tested parts of the protocol
  - difficult to expand to be comprehensive
New approach: extreme testability

- The new testing system in Samba4 is based on a few basic components:
  - a comprehensive raw client library
  - individual tests covering every field of every call
  - a randomised dual-server tester with broad coverage
  - a "CIFS on CIFS" storage backend for the Samba4 server
- These components work together to provide a testing capability far beyond what could be achieved with our earlier testsuites
CIFS Plugfest
Raw Client Library

• The heart of the new testing system is a 'raw' comprehensive client library. Unlike our previous client library this allows easy generation of all SMBs, with control over all fields in each request.

• New features include:
  • async interfaces
  • oplock support
  • no 'smarts' - send exactly what is asked for

• Note that it takes a lot code to use the new interface compared to the old one. The old interface is still available as a wrapper.
C interface to raw library

Old interface:

```c
int fnum = cli_open(cli, "\\test.dat", O_RDWR, DENY_READ);
```

New Interface:

```c
NTSTATUS status;
union smb_open io;

io.generic.level = RAW_OPEN_OPENX;
io.openx.in.flags = OPENX_FLAGS_ADDITIONAL_INFO;
io.openx.in.open_mode = OPEN_MODE_ACCESS_RDWR;
io.openx.in.search_attrss = FILE_ATTRIBUTE_SYSTEM|FILE_ATTRIBUTE_HIDDEN;
io.openx.in.fileattrs = 0;
io.openx.in.write_time = 0;
io.openx.in.open_func = OPENX_OPEN_FUNC_OPEN;
io.openx.in.size = 0;
io.openx.in.timeout = 0;
io.openx.in.fname = "\\test.dat";

req = smb_raw_open_send(tree, &io);
status = smb_raw_open_recv(req, mem_ctx, &io);
```
CIFS Backend

• A new feature in Samba4 is the ability to define arbitrary storage backends at the 'raw' CIFS level

• A backend that has proved incredibly useful for testing is the 'CIFS' backend, that uses a remote CIFS server for all operations:
  • uses the raw client library for remote server access
  • ideal for testing core server infrastructure
  • combined with the individual tests and gentest it allows the server side CIFS parsing to be tested in isolation
gentest

- gentest is the 'big gun' CIFS test program that I have wanted to build for many years. Basic features include:
  - dual server, dual instance testing
  - randomised, broad coverage request generation
  - automatic backtracking for finding minimal request subset
  - can cover all fields of all requests
  - full async oplock testing
Dual Server Testing

- The basis of gentest is 'dual server testing', the same basic technique used in the 'locktest' program from earlier versions of Samba:
  - The test program establishes two connections to each of two servers
  - Random requests are then generated, with identical requests sent to the two servers
  - At each step gentest compares every field of every response between the two servers
  - When a response differs gentest uses backtracking to find the minimal subset of the requests sent so far that generates a difference in response
Request Generation

- Request generation is based on the concept of a 'generator' function for each request in CIFS
- The generator for a CIFS request calls into a library of 'field generators' that produce constrained random values for each type of field in the protocol.
- Field generators include things like gen_timeout(), gen_io_count(), gen_fnum(), gen_fname() etc
Field Generation

- The generators for individual fields are heavily biased towards interesting values, while allowing for arbitrary values in most cases:
  - `gen_fnum()` will most of the time generate an open file handle (if one exists), but will sometimes generate an invalid handle
  - Some fields (like IO counts) are tightly constrained to prevent filling of disks
  - Flags fields are heavily biased towards valid sets of flags, but have a small chance of generating arbitrary sets of bits
Backtracking

• When a difference is discovered between the two servers gentest goes into 'analyze' mode, using a backtracking technique to find the minimal subset of requests that produce a difference:
  • successively smaller chunks of the request streams are blocked out
  • If a difference is still reported when a chunk is blocked out then that chunk is not needed and can be discarded
  • reconnects to the servers and wipes all files at each pass
  • The final pattern of requests can be replayed for analysis with a network sniffer
Unix<->Unix Connectivity

- Samba is finally breaking away from its Windows-only roots and starting to look seriously at providing a good Unix to Unix filesystem.
- The Unix CIFS extensions are gaining acceptance by several vendors.
  - hard links, symlinks, devices
  - rename and unlink open files
- The new cifs-vfs Linux client is leading the way, and may eventually become a viable challenger to replace NFS
Process Models

- Samba3 only supported a “one client, one fork” process model
- In Samba4 the process model is pluggable, allowing the model to match the environment and backend
- Three process model modules are currently available:
  - 'single' - one process for all clients
  - 'standard' - the old Samba3 model
  - 'thread' - a pthread per client
Portability

• Samba is aggressively portable
• See build farm at http://build.samba.org/
Current Status

• The effort to build Samba4 has so far taken 2 people about 6 months
  • RAW client library done
  • test suite done
  • NTVFS layer done
  • CIFS backend done
  • TANK backend done

• To get this far we have dropped a great deal of fundamental functionality that users have come to expect from Samba. That needs to be replaced.
So, you want to help? Good!

- Get the code from the 'samba4' cvs module on samba.org
- Join the samba-technical IRC channel and mailing list
- Not for the faint of heart! This is not production code yet
- See http://samba.org/ftp/samba/slides/samba4_auug.pdf for a copy of these slides

Questions?