SAMBA Opening Windows to a Wider World

Implementing SMB2 in Samba

Jeremy Allison
Samba Team/Google Open Source Programs Office
jra@samba.org
jra@google.com

What is SMB2 ?

- Microsoft's replacement for SMB/CIFS.
- Ships in Vista, Windows7 and Windows 2008.
- Considerably simpler than SMB/CIFS.
  - Only 19 requests vs. ~100 for CIFS.
- Much better support for asynchronous request/responses and multiple concurrent requests.
- Larger read/write sizes for more modern networks/machines.

What is Samba ?

- Large, old codebase (started in 1991).
  - Getting larger and older..
- Only fully-featured implementation of SMB/CIFS outside of Microsoft.
  - Implements AD domains, NT domains, printing, registry, performance counter and other things most other implementations don't care about :-).
- Open Source/Free Software under GPLv3 license.

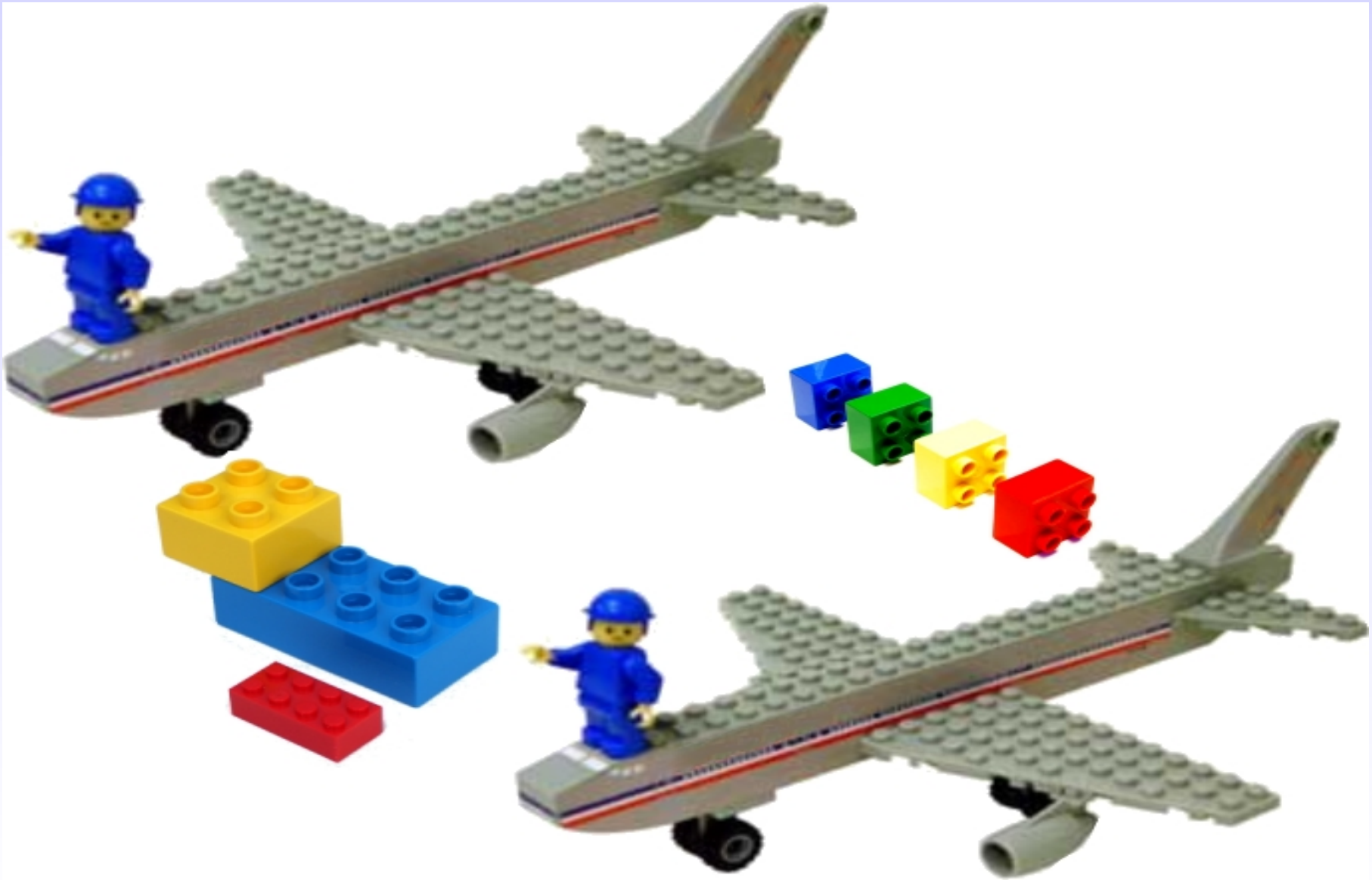
How to move to SMB2 ?

- Samba works as a separate daemon process (smbd) per connected client.
- Options were to write all SMB2 code as a new server:
  - Advantage is clean slate design.
  - Disadvantage is re-implementation of lots of subtle working SMB1 code that still applies to SMB2 semantics.
  - Separate server is still an option to explore, as changing to SMB2 is a gateway on initial client packet.

A Historical Precedent

- Samba has undergone a similar change in the past.
  - DOS/Windows 3.1/Windows 95 use old SMB semantics on the wire (SMBopenX and friends).
  - Windows NT uses NTCreateX with a different set of share parameters.
- Samba used to implement the DOS API, and map the Windows NT semantics onto it.
  - Was changed to implement DOS semantics on top of Windows NT.
  - Birth of the Samba VFS.

Refactoring in a nutshell..



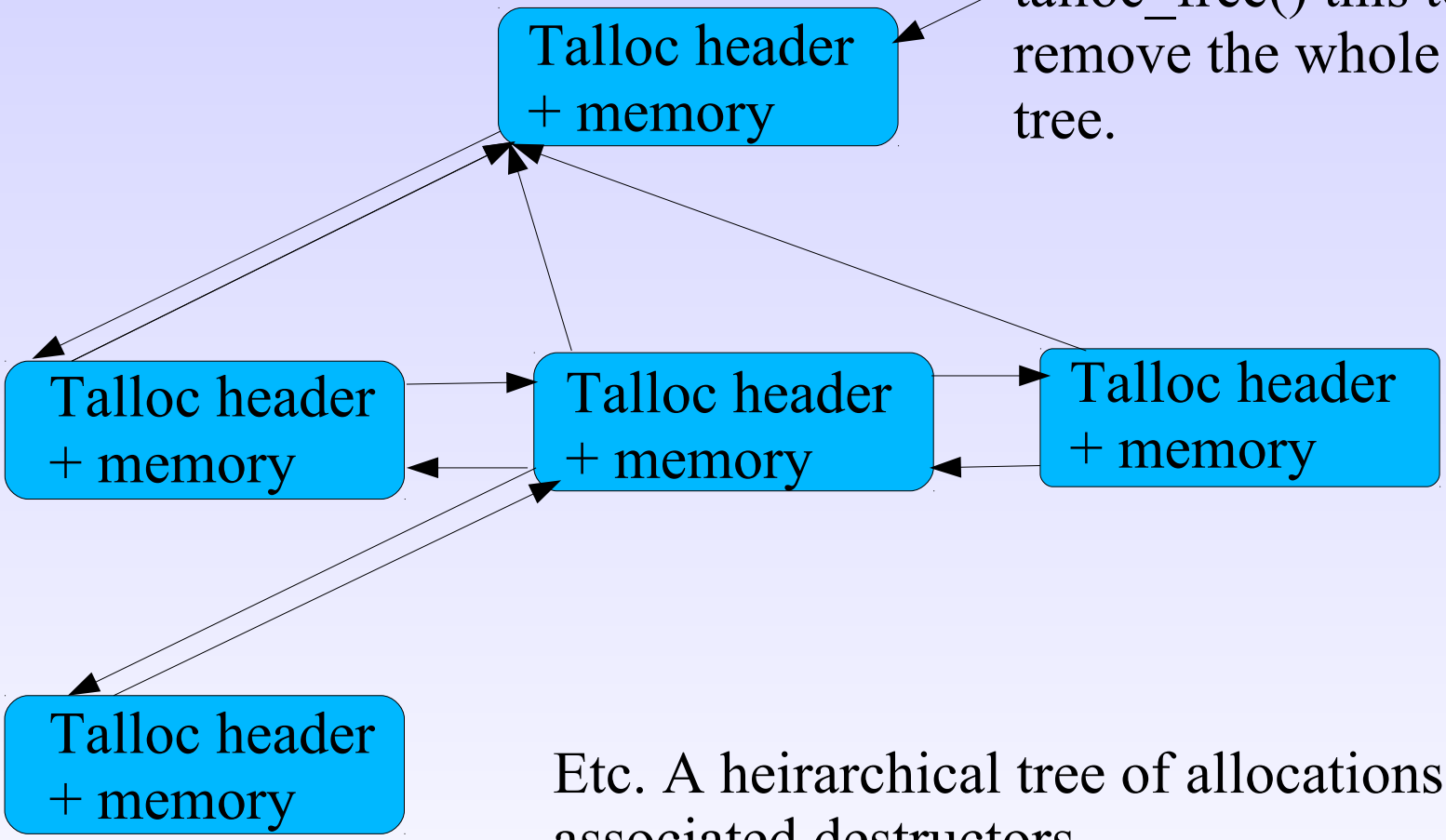
Samba VFS

- Provides an internal API that represents the Windows file API.
  - NTCreateFile() etc.
- Other associated API's make it easy to build Windows file sharing semantics underneath a packet parsing layer.
  - Not useful as an external API as too much knowledge of Samba internals needed.
  - But close to what we need to implement Windows semantics under an SMB2 parser.

The Four “T”'s

- Samba is built on top of four libraries:
- TALLOC (tridge):
  - Hierarchical memory allocator.
  - C++ style destructors.
- TDB (tridge, rusty):
  - Fast, multi-reader/writer key/value pair database.
- TEVENT/TEVENT_REQ (tridge, Volker, Metze):
  - Async event library.
  - TSOCKET (Metze).
    - Helper library for tevent_req.

talloc()



Etc. A heirarchical tree of allocations, with associated destructors.

“Trivial” DataBase (tdb).

- Was so called because it once took less than 1000 lines of code.
  - Once transactions were added this was no longer the case.
  - mmap()'ed shared memory area arbitrated by fcntl() locks.
  - Very scalable for multi-readers/writers.
  - Dynamically expands as entries added (munmap(), ftruncate(), mmap()).
  - Depends on coherent buffer cache between pread/pwrite/mmap.

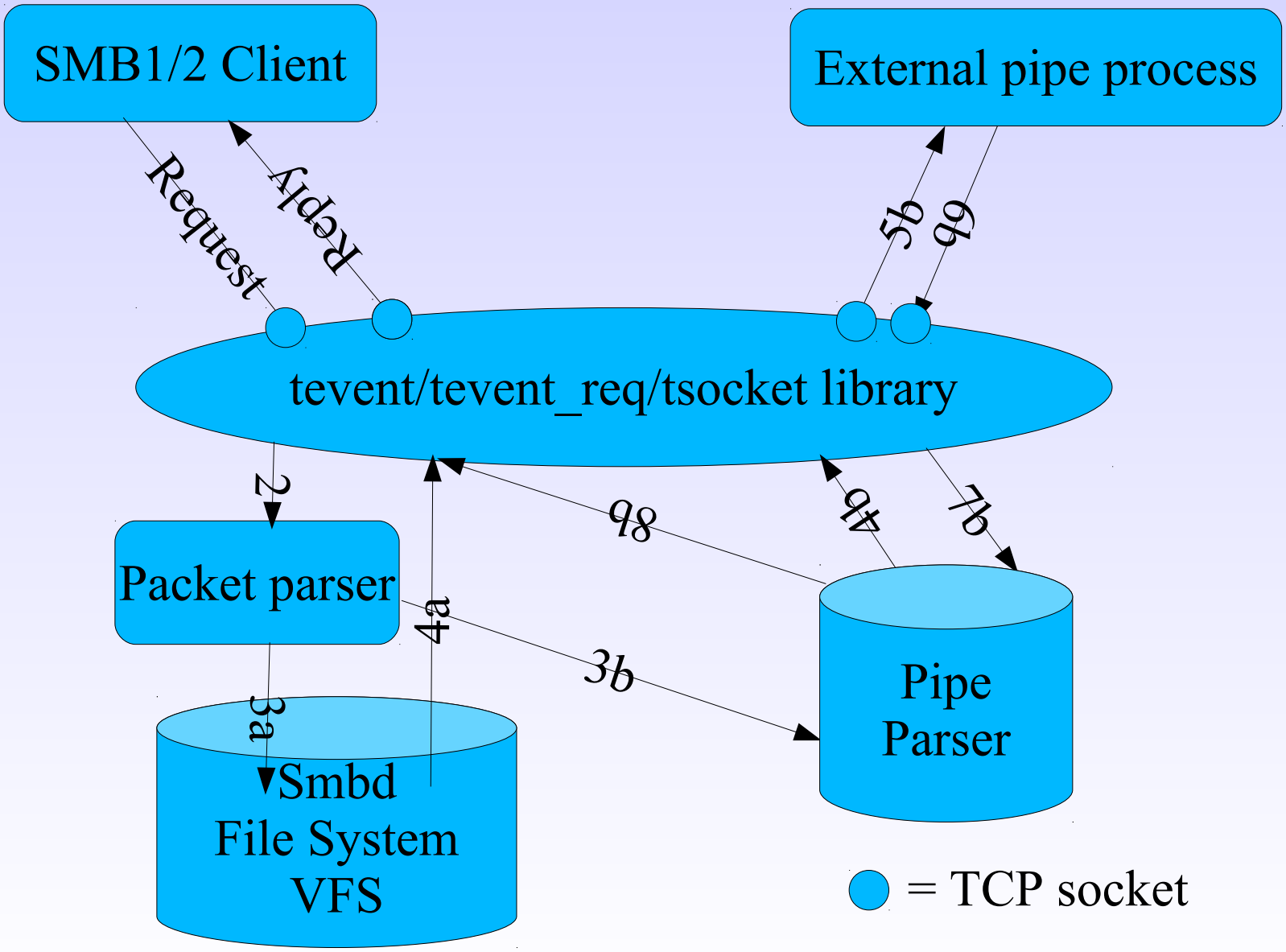
tevent/tevent_req/tsocket

- Similar to libevent/libev
  - In our defense, at the time tevent was created libevent did not handle POSIX realtime signals at all.
  - tevent heavily uses talloc(), making it much easier to integrate into the Samba event loop.
- Set all sockets into asynchronous mode, then just get callbacks when data is ready to read/write.

tevent/tevent_req/tsocket (continued)

- tevent_req is a pattern around tevent, making handling of asynchronous requests completely boilerplate.
  - Allows read and write requests to be asynchronously passed to named pipe services.
  - Finally named pipe servers can be moved out of process – reduce smbd bloat !
    - Simo has a prototype of spoolssd out of process.

The central role of tevent



tevent timeout functions

- Signals and time-based events functions are handled as callbacks within tevent.
  - Allows modular handling of all smbd housekeeping functions.
  - Can cause interesting bugs :-)
    - Thanks to Microsoft's Nick Meier for some iozone help with SMB2 in Samba.

Implementing the SMB2 parser (design by metze)

- Tevent/tevent_req used to construct an array of io_vec structs containing the parts of an SMB2 packet.
  - iov[0] = NBT header for all requests.
  - Iov[1] = SMB2 header (fixed)
  - iov[2] = SMB2 header (variable)
  - iov[3] = (variable length) rest of SMB2 packet.
- Chained SMB2 requests create a tuple of io_vec structs for each incoming SMB2 request inside the NBT packet.

Implementing the SMB2 parser (continued)

- Every incoming request is matched with an outgoing io_vec tuple.
  - To deal with each packet in turn, simply increment the index into the io_vec array by 3.
- All incoming requests and outgoing replies are on a queue with attached state.
  - Allows multiple requests/responses “in flight” inside the server code.
    - Significant improvement on the SMB1 code design.

Asynchronous SMB2 requests

- Every SMB2 request has the ability to go “asynchronous”.
  - Server can send an interim response to inform the client “this may take some time”.
    - Internally this is handled by splitting out the delayed request into a separate entry on the request queue, and generating an interim response iovec tuple.
- Allows a later SMB2_CANCEL request to delete any unprocessed request.

SMB2 asynchronous IO

- Windows client redirector (finally) will do multiple outstanding reads/writes to reduce the effects of latency.
  - Samba SMB2 implementation uses POSIX aio to allow this (or Volker's fork VFS module).
  - On Linux, current glibc design has problems with this.
    - Patch available from speaker :-).
- sendfile()/recvfile() (zero copy) not yet implemented in current SMB2 code – planned for final 3.6.0.

Interfacing to the internal API

- The SMB1 API needed some updates to cope with SMB2.
  - The SMB1 : MID (multiplex ID) field is only 16 bits – was expanded to 64 bits for SMB2.
    - Unfortunately means share mode database format changes.
  - Many of the internal functions take a “struct smb_request”, which is SMB1 only.
    - Each SMB2 request fakes up a “struct smb_request” for the API.
      - Eventually this will get reversed.

Interfacing to the internal API (continued)

- Remarkably few other changes were needed to make the SMB1 oplock, locking, change notify and deferred open (share mode) code work with SMB2.
  - Only 23 instances of :
    - “if (using_smb2)” needed in the entire codebase.
- Full support for all named pipes (print spooler, LSA, SAMR, registry etc.) available

What is left to do ?

- Support for the Windows tools to get and set user quotas is the only missing SMB2.002x feature in the server code.
  - Credit “algorithm” may need work.
- Post 3.6.x we need to add SMB2.1 dialect features to the underlying file sharing engine.
  - “Lease” level on File oplocks.
  - Durable file open handles.
- Some of the SMB2.1 features may need extra kernel support.

SMB2.x features

- Adding durable file handles will mean additional state added into a persistent tdb – kept updated on every handle change.
  - Making this work across clustered Samba will be interesting.
- SMB2.1 leases
  - Allow handle caching across multiple opens by the same client.
    - Easier than fixing Microsoft Office code :-).
- Implementation will start with smbtorture tests

What is left to do (continued) ?

- Modify libsmbclient to allow SMB2 connections.
  - Allow Linux GNOME/KDE desktops to easily integrate with SMB2 servers.
  - Use smbclient as a test bed to get the right internal API's.
  - smbtorture4 already does some SMB2 tests written by Tim Prouty (tprouty).
- Samba 3.6.0 will ship with production-ready SMB2 support, but not on by default. The next release (4.0) should have this on by default.

An anecdote

- Early OEMs running v3-6-test code in advance of release have reported 2x the performance of the same applications running over SMB1.
  - This is before SENDFILE() fix added to the server.
  - Almost certainly due to improvements in Windows client redirector with SMB2.

Praise and thanks

- The ease of production-ready SMB2 support in the 3.6.0 server is a testament to the hard work done in design and coding by:
  - Andrew Tridgell (tridge)
  - Stefan Metzmacher (metze)
  - Volker Lendecke (vl)
  - Simo Source (idra)
  - Guenther Deschner (gd)

Questions and Comments ?

Email: jra@samba.org
jra@google.com