Undocumented CIFS

Jeremy Allison : Samba Team

But isn't CIFS documented ?

- SNIA document does a basic job.
  - Enough to get an implementor started.
  - Roger Binns (VisionFS author) quote: “Anyone following the CIFS spec has a beautiful server no clients will interoperate with”.
- However it is very incomplete.
  - Doesn't cover older Windows clients
  - Doesn't cover OS/2 clients, or newer clients like Linux or MacOS X
  - Doesn't have the details needed to correctly write a server for Windows clients.

What about the “licensed” documentation ?

- Can't accurately comment as Microsoft licensing terms prohibit Free Software from participating.
- NetApp presentation implied the “Microsoft” CIFS document is similar to the SNIA one.
- Licensees have to help Microsoft debug the document they paid for !
- Still many details missing.

Why is CIFS so hard to document ?

- Partly Microsoft intransigence
  - "Private protocol" easier to extend arbitrarily and preserve barriers to interoperability.
- Partly historical
  - CIFS covers many old clients no longer in wide use or testing, few resources are allocated to documenting these.
- Partly due to client to server homogeneity.
  - CIFS became "Windows NT kernel on the wire" semantics. Windows kernel features exposed.

How can Samba help document CIFS ?

- tridge had the epiphany several years ago.
- **Don't follow any published specification.**
  - Create client test suite and tools to test any theory about the "correct" behavior of CIFS servers.
  - Take the "latest" Microsoft server (currently Windows 2003 Service Pack 1) as the "standard".
  - Samba4 client suite and torture tester is the result.

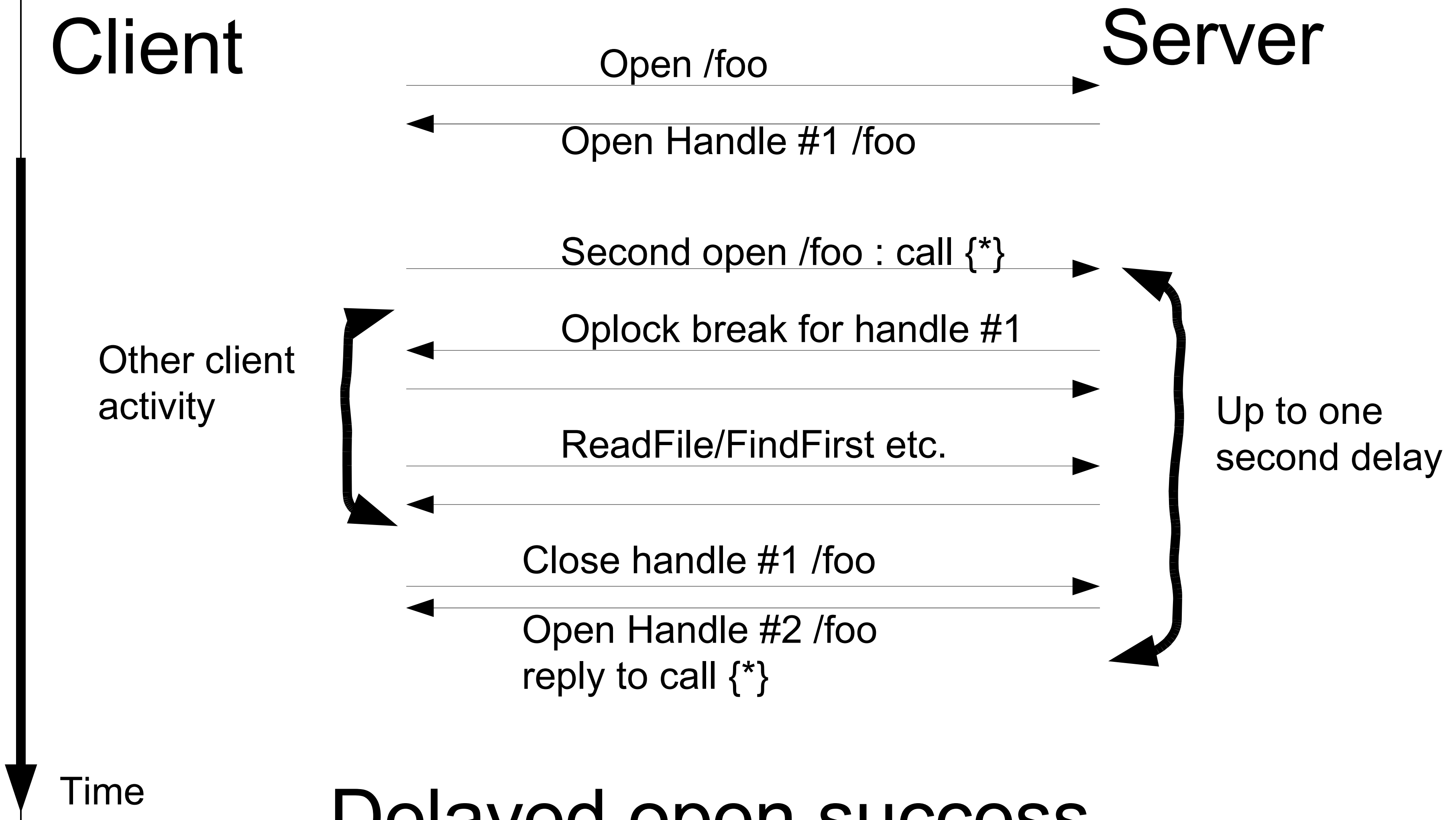
Samba contribution

Presenting some of the results of the torture tester, as implemented in Samba3 and Samba4.

“Ornery” open modes

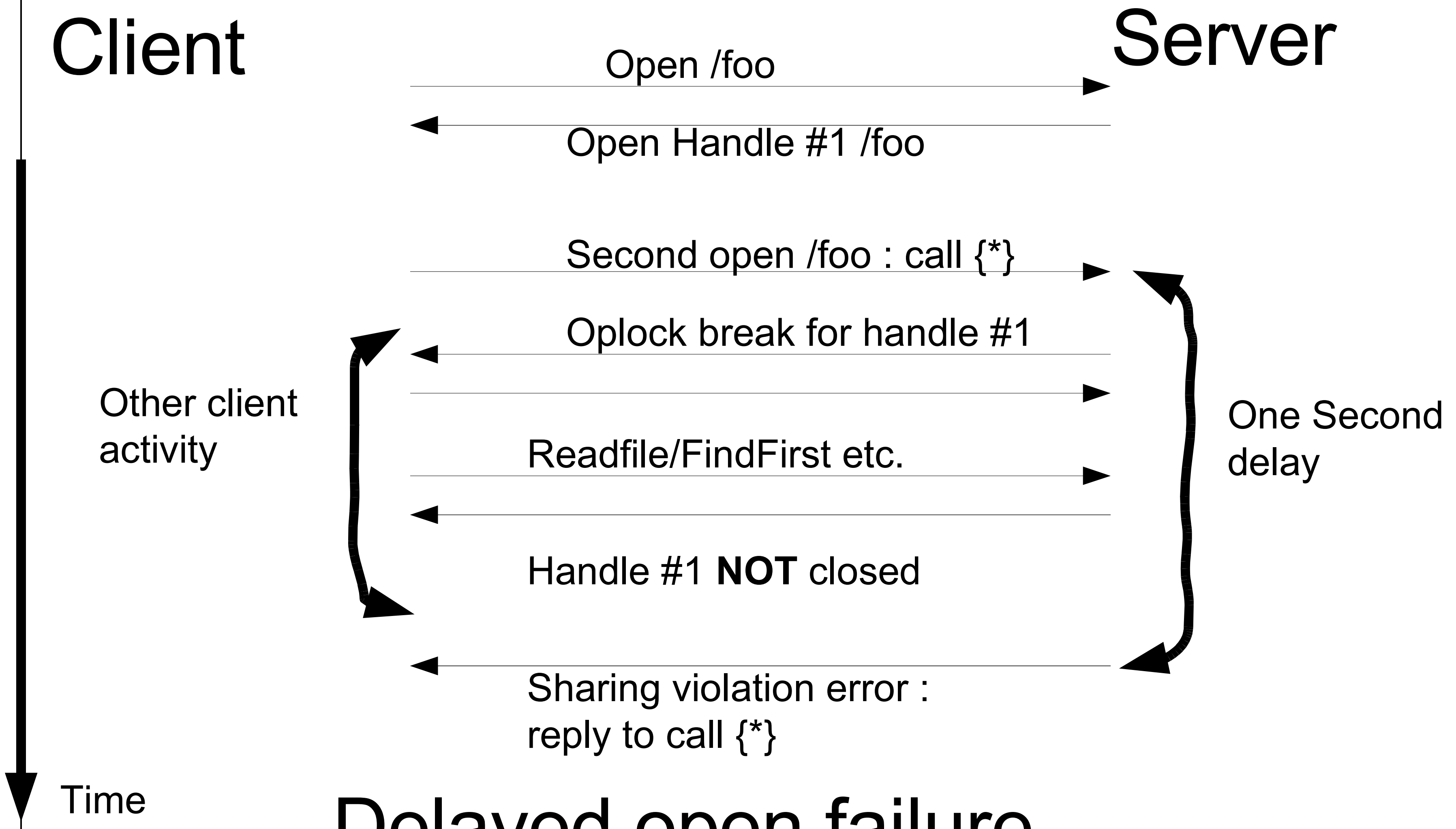
- If a second open on a file would cause a “sharing violation” error message, the server must delay processing the second open by approximately one second to allow the client to close the currently open handle.
  - Used *extensively* by Microsoft Office.
  - Time out is not exact, seems to be around one second but this is almost certainly Windows kernel dependent. Longest seen was 2 seconds.

"Ornery" Open modes



Delayed open success

“Ornery” Open modes



Delayed open failure

“Odd” oplocks

- The CIFS spec. says that any second open on a file already opened with an exclusive oplock granted will cause an oplock break.
  - This is untrue.
- Doing NTCreateX with “attribute only” flags
  - FILE_READ_ATTRIBUTES| FILE_WRITE_ATTRIBUTES|SYNCHRONIZE

does not cause an oplock break.

“Odd” oplocks (continued)

- The CIFS spec. says that a second open on an already open file cannot be granted an oplock.
  - It seems that a level 2 (read-only) oplock can be granted on a file already open with no oplock !
  - A write on either of the open handles causes a “break to none” packet on the level 2 open handle, but not the no oplock handle.

“Shifty” share mode checking

- The CIFS spec. says that any second open on a file with a share mode conflicting with an existing share mode causes a sharing violation.
- Doing NTCreateX with an access mask <u>NOT</u> containing any of the following :
  - FILE_WRITE_DATA|FILE_APPEND_DATA|FILE_READ_DATA|FILE_EXECUTE|DELETE_ACCESS

does not cause a sharing violation.

What explains these strange semantics ?

- This only makes sense if you consider how the Windows kernel handles CIFS file opens.
  - Requests for “attributes” only seem to be handled at a “meta-data only” layer above the file open code.
  - Requests for “real opens” go into the file handle layer which causes the oplock break and share mode processing to take place.
  - Example of Windows internal semantics leaking into the protocol.

“Wrong” write times

- Windows file timestamps contain a “last write time” timestamp.
  - The CIFS spec. says this is to be updated to the current time whenever a file is written to.
- CIFS contains calls to set the “last write time” timestamp.
  - But what is the timestamp set to when a write subsequently happens ?
  - CIFS spec implies the write overwrites the explicit “set timestamp” call.

“Wrong” write times (continued)

- The “correct” action is to ensure that a write time set via the “set write time” SMBtrans2 or other calls must remain as the “current” write time – even if a SMBWriteX call is subsequently made.
  - So write times are “sticky”. Once a write time is modified via an explicit call, further writes don't change the time.
  - But what happens with multiple handles open ?

“Wrong” write times (continued)

- When a “sticky” write time is set on one of multiple handles open on the same file, subsequent attribute reads of the write time return the “sticky” time.
  - However, once the handle on which the “set write time” call is done has been closed, then subsequent writes update the time – but not until all existing handles are closed.
  - These semantics in a protocol are insane, but Microsoft Office (Excel) depends on them.

What is happening ?

- Again, the internals of Microsoft's CIFS implementation have bled into the protocol on the wire.
  - Windows keeps open handles on a file in an in-memory cache. This explains the “set by one, seen by all semantics”.
  - Reference counting on the handles explains the last inconsistency. The pending change gets flushed onto disk once the “master” handle is closed, but the other handles still remember the “sticky write time” while they are open.

“Daft” directory listings

- The CIFS spec says nothing about a client directory listing other than that all the files in a directory must appear.
  - In reality the two entries dot and dot dot <u>MUST</u> appear as the first two entries.
  - If not, Windows NT 4.x clients will display bogus folders entitled dot or dot dot in Windows explorer.
  - It's a bug – but one every server vendor must be aware of.

“Daft” directory listings (continued)

- The CIFS spec. shows three ways of continuing a directory listing from a client.
  - Resume by “continue” from the end of the previous search.
  - Resume by filename.
  - Resume by “resume key” - a special value
- Windows servers suffer from different bugs in each of these that CIFS client writers must be aware of.

"Daft" directory listings (continued)

- Resume by continue bit :
  - Windows servers have interesting bugs with this option, they miss filenames (seemingly at random). Not safe to use in client code.
- Resume by filename :
  - Works against Windows servers, but only on NTFS exported drives.
- Resume by "resume key" :
  - Used by Windows servers exporting FAT filesystems.

“Daft” directory listings (continued)

- In order to cope with all varients of Windows servers :
  - Use the require resume key bit in the trans2 FindFirst/FindNext SMB. Don't set the “continue” bit
  - Send the returned resume key to continue from.
  - Also send the filename to resume from.
  - Seems to give reliable directory listings with all tested Windows servers.

"Dodgy" delete on close

- "Delete on close" seems to be a state attached to a file - but also settable at NTCreateX time.
  - If set when the last handle to the file is closed it causes the server to delete the file.
  - The state can be set and unset via the SMBtrans2 call SetFileInfo – info level SET_DISPOSITION_INFORMATION.
  - The file handle must have been opened with an access mask containing DELETE.

"Dodgy" delete on close

- Once the "Delete on close" state has been set on a file then no further opens are allowed.
  - The file is <u>still</u> seen in directory listings however.
  - The delete on close state is attached to the underlying file state – all handles opened on the file will see this state.
  - The file must have been opened with FILE_SHARE_DELETE to allow delete on close to be set.

“Secret” SMB signing

- SMB signing is partly documented, but details are missing.
  - Signing session numbers are associated with the SMB_MID (multiplex-id) number.
  - This means that SMBtrans2 calls split into primary and secondary requests and responses use the same session number for signing.
  - oplock break messages from server to client (asynchronous) are not signed and don't use a sequence number.

Locking for LuZ3r5

- SMB byte range locking acts as a restaurant “plate stack” from the same process.
- Locks can't overlap, or be split or merged (as they can in POSIX), but read locks can be “stacked” on top of a write lock.
  - So long as the same locking context is used.
- The locks remain as a stack and are removed off the bottom on unlock.

Crazy comments and questions ?

- email: jra@samba.org
- samba-technical@samba.org mailing list.