

SMB3 and NFSv4

A view from above

Tom Talpey
SambaXP 2025
Göttingen

Definitions

- “NFSv4”: NFSv4.1/4.2
 - NFSv4.1 is widely implemented, though not universal
 - NFSv4.2 adds pNFS
 - NFSv3 still in use, but not a focus
 - NFSv4.0 is flawed, even less a focus
- “SMB3”: SMB3.1.1 with optional features
 - The benchmark version, due to Microsoft’s commitment
 - Widely implemented, and is simply the default
- At a sufficiently high level, they both provide a solution to the same problem: **sharing**

My history with the protocols

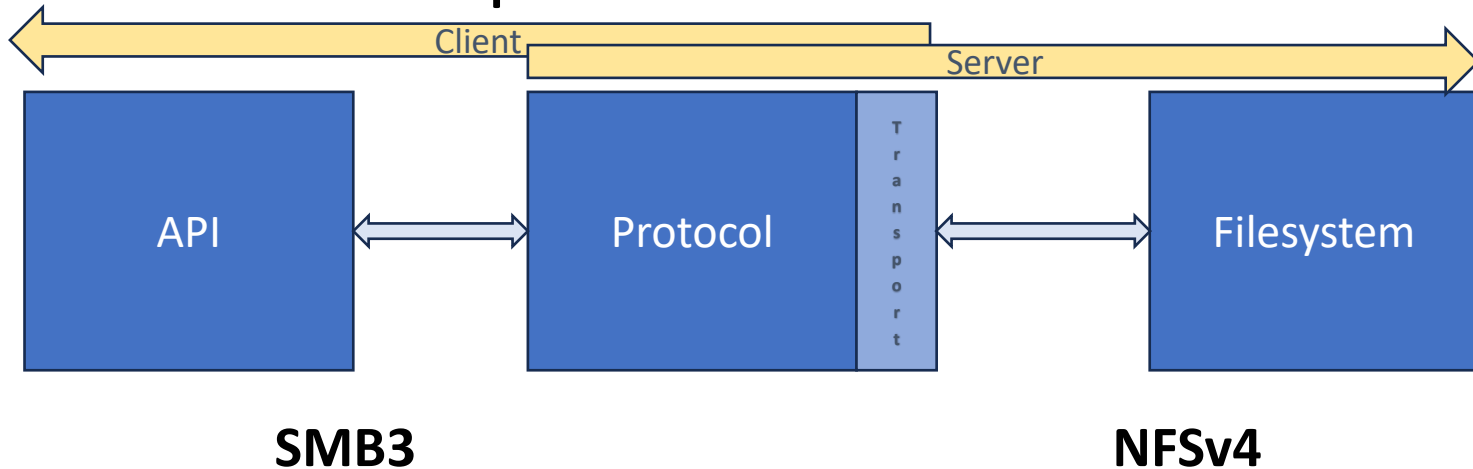
NFSv4

- My first NFS implementation of an NFSv2 X11 font loader for our startup's X-terminal in 1989
- Was a member of the NFSv3 spec author team in the very early 1990's
- Coauthored the NFS/RDMA protocol and implemented it on Linux 2.4
- Authored the NFSv4.1 Session and also the RDMA binding

SMB3

- Joined the SMB party in 2009, late in my career, to implement Microsoft's commitment to document the protocols
- The documentation greatly improved both the protocol and Windows, and enabled significant innovation which became ~~SMB2.2~~ SMB3.0
- Coauthored the SMBDirect protocol and implemented it on Windows Server 2012

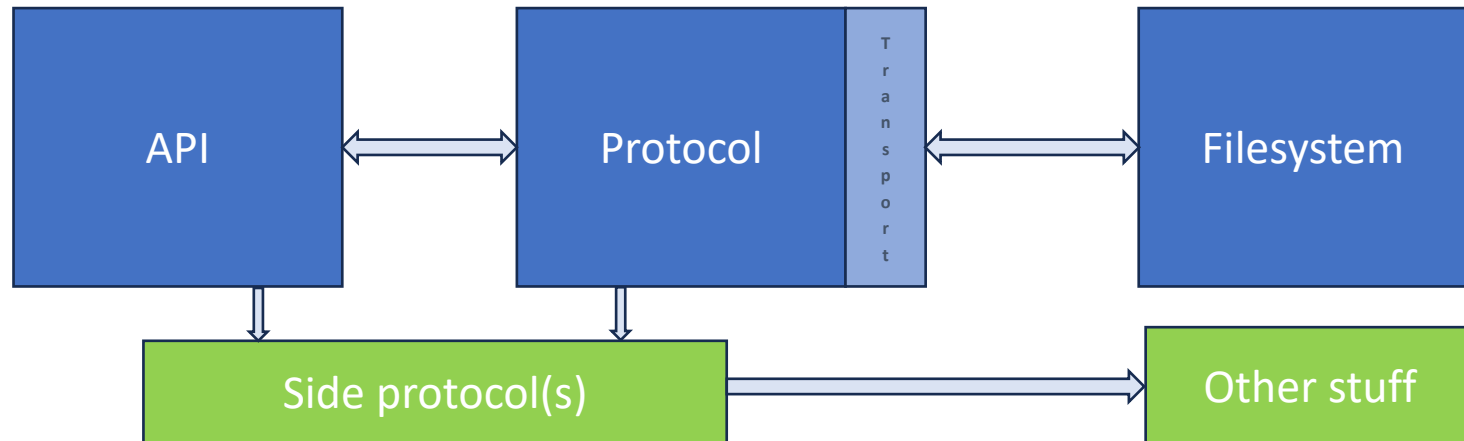
An oversimplified view



- API: Windows, Posix, IPC/RPC
- Protocol: SMB3
- Transport: TCP, RDMA, QUIC
- Filesystem: Variable, with NTFS “built-in”

- API: Posix
- Protocol: NFSv4
- Transport: TCP, RDMA
- Filesystem: Posix

To which add



SMB3 side protocols

- DFS, clustering, etc
- Management ecosystem

NFSv4 side protocols

- Mostly integrated
- NFSv4.2: pNFS layouts

What's Special about SMB3

- Not a filesystem, but typically deployed as one
- An authenticated, recoverable session for issuing requests to peer servers
- Flow-controlled synchronous or asynchronous (cancelable) processing
- Native integrity and/or encryption, **per-user** and per-session
 - Not per-machine and therefore shared
- Many-to-many transport connections for these requests
 - N_1 connections per session, including zero
 - N_2 sessions per connection
 - Trunking, resilience ($N_1 > 1$) or recovery (when N_1 drops to zero)
 - Shared (maximal N_2) or nonshared (minimal N_2)
 - Arbitrary connection types, including RDMA
- Extensible by design
 - Fsctl's, including file-less
 - Negotiate contexts (top-level capabilities)
 - Tree Connect contexts (per-share capabilities)
 - Create contexts (per-handle capabilities)
 - Transforms (per-message encryption, compression, etc)
 - Ok, and dialects – but don't go there please

What's Different From NFS?

- NFS is inflexibly Posix, all the way down
 - No RPC pipes, ACLs are futile, ...
- NFS is hard to extend, by design
 - Doesn't have 5 of the 6 previous SMB3 slide's bullets
 - There are no remote ioctls, even
 - Overspecified (IMO)
 - Many requirements, few behaviors
 - Changing it requires IETF process
 - Extensions may involve new minor version (Big Job)
 - pNFS (layouts) maybe an exception
- SMB3 has better RDMA support
 - I should know, since I wrote 'em both? 😊

Defining a Protocol

- Protocols have natural, non-obvious boundaries
 - Which need to be decided first, and not overloaded
- Example, SMB2 at right
 - The APIs aren't there
 - The Filesystems aren't there
 - The applications and app requirements aren't there

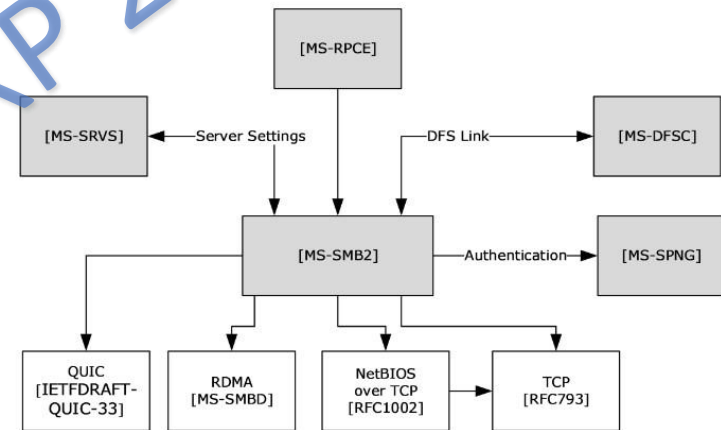


Figure 2: Relationship to other protocols

High-level semantic differences

SMB3 supports...

- Windows
- Posix (Linux)
 - Only to Samba/ksmbd
- IPC/RPC
 - Authenticated and protected transport for side protocols and anything else

NFSv4 supports...

- Posix (Linux)
 - “all the way down”

Filesystem differences

SMB3

- Windows-native (NTFS, ReFS, CSV, etc)
- Non-Windows
 - “not supported” results
- IPC/RPC

NFSv4

- Unix-native Posix
- Re-export (ick)

Identity, auth and security

SMB3

- Windows native (SIDs)
- Per-user tokens
- NTLM, Active Directory
 - Provided by Windows
- Matched to native filesystems

NFSv4

- AUTH_SYS
 - Traditional numeric uid
 - Which will never die
- AUTH_TLS (new)
 - TLS with machine key
 - Handy, but basic
- AUTH_KRB
 - Kerberos 5i, 5p, etc
 - External infra required

Transport

SMB3

- TCP
 - Including RFC1001/1002
- RDMA
 - Via SMB Direct
- QUIC
 - Firewall-friendly and encrypted by default
 - (but not much else)
- Rich multichannel
 - Any and all types at once

NFSv4

- TCP
- RDMA
 - Via RPC/RDMA
 - Good, but somewhat bound by legacy XDR
- Multiconnect
 - Trunked, single type
 - not true multichannel

Documents

SMB3

- The Microsoft docs
 - Excellent, but quirky
- Tested and maintained
 - Microsoft-sponsored processes
 - Open source test suites
- Broadly implemented
 - Successfully!

NFSv4

- IETF RFCs
 - Weighty, and highly normative
 - Too much so, perhaps!
 - Slow to change, by design
 - Updated via full replacement
- Informally tested
 - Interop events
 - pyNFS testing client

Pet peeves

SMB3

- Name perception
 - “SMB” == “Windows”
- Maybe a little bit too extensible
 - Leads to limited interop
- Fsctl’s limited to 64KB in, 64KB out, inline

NFSv4

- No truly asynchronous ops
 - NFS4ERR_DELAY
 - “It’s too hard, try again”
 - Returned from everything, including OP_SEQUENCE
 - Aka NFSv3 EJUKEBOX
 - Reply cache to protect non-idempotent ops
- Posix semantics wire -into the protocol
 - Caching
- Not readily extensible

So, is there a conclusion?

SMB3

- SMB3 is the richer protocol
- SMB3 runs on more total platforms
- SMB3 is (much) more extensible
- But SMB3 Posix extensions are not enough

NFSv4

- NFSv4 is the most faithful to Posix (Linux)
- NFSv4 runs everywhere Linux does
- NFSv4 itself won't change much

It's all about meeting the needs of applications!

My opinion, part 1

- NFSv4 is a stable and trusted solution for Linux
 - It's mature and will change very little
 - That's a good thing
- SMB3 is flexible and extensible
 - It presents more opportunity for growth
 - It can readily express diverse client needs via the protocol

My opinion, part 2

- The biggest and best thing for SMB3 is for Windows SMB service to support the SMB3 Posix Extensions
 - Samba and ksmbd already do
 - The Linux client already does
 - WSL already implemented the backend
 - This would increase the SMB reach, overnight
- The second biggest thing is to expand the scope of Linux SMB3 applicability
 - By better supporting new application needs
 - Exotic workloads (e.g. HPC striding, filesystem optimizations, ...)?
 - Minimally, with new infolevels and fsctls
 - Ultimately with new SMB3 extensions
 - This would take time, applications change slowly

My opinion, part 3

- It's not silly to consider SMB3 as a pNFS layout
- Or for SMB3 extensions to refer to NFS
- Or for SMB3 to tunnel other traffic
- But this is crazy talk, just get the basics right

Thank you!

Questions/discussion?