NFS: Genesis

Tom Lyon
@aka_pugs@mastodon.social

Zeitgeist 1983

4.2BSD finally available!

FIle sharing - obviously desirable, but non-trivial

Disks - expensive!

Systems – frequent down time (backups, esp)

Network – flaky shared media

UNIX explosion – for *new* vendors but not the installed base

Share files – or just disks?

Workstation vs Disk Cost

1982:

Sun-1: $8,900

With Ethernet, Memory, UNIX $13,900

Disk (SMD - 84MB): $13,900

1985:

Sun-2/50-2: $8,900

Sun-3/75M-4: $15,900

Disk(SCSI - 71MB): $5,900

Network Disk Protocol

Bill Croft 1983 - SunOS 1.1 (4/84)

Proof for viable diskless operation

Huge cost savings & feature parity with Apollo

Example of benefits of statelessness and idempotency

PITA to administer - none of today’s SAN tools

Beliefs/Fears/Constraints

Disks are/will be expensive!

Prior network file systems presume local disk! (Not Apollo)

VAXen and mainframes own the disks!

“Servers” are a concept not yet proven

Need a protocol – can’t even assume C language on the server

“Distributed UNIX” isn’t UNIX

Terminals <<$ Workstations <<$ Minicomputers

Jan. 03, 1984 - Bill Joy Perigee

“Design of the Sun Network File System”

“Sun Network File Protocol Design Considerations”

“Sun UNIX Modifications to use the Sun Network File Service”

Stakes in the ground:

- Heterogeneity, Statelessness, Datagrams, Idempotency
- File handles, Vnodes, Block-based for caching & VM
- *No* file use after unlink

RPC/XDR

Bob Lyon, 2/2/84

“Sun RPC Architecture”

Bob Lyon, 8/20/84

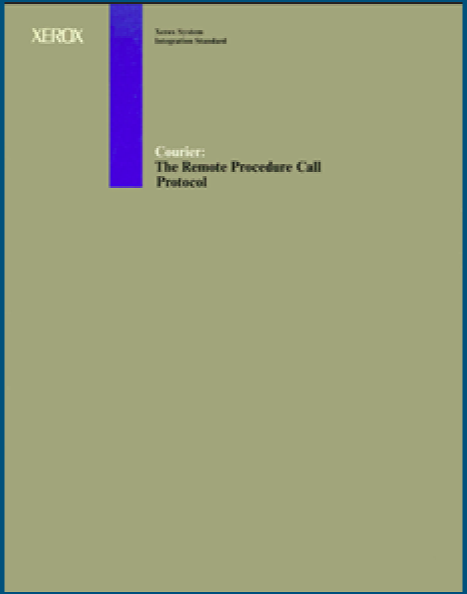
“Sun RPC Protocol Specification Version 2 aka “Son of Courier”

Xerox XSIS-038112 12/81

“Courier: the Remote Procedure Call Protocol”

Andrew D. Birrell & Bruce Jay Nelson, Xerox CSL 83-7 12/83

“Implementing Remote Procedure Calls”



NFS Architecture Offsite

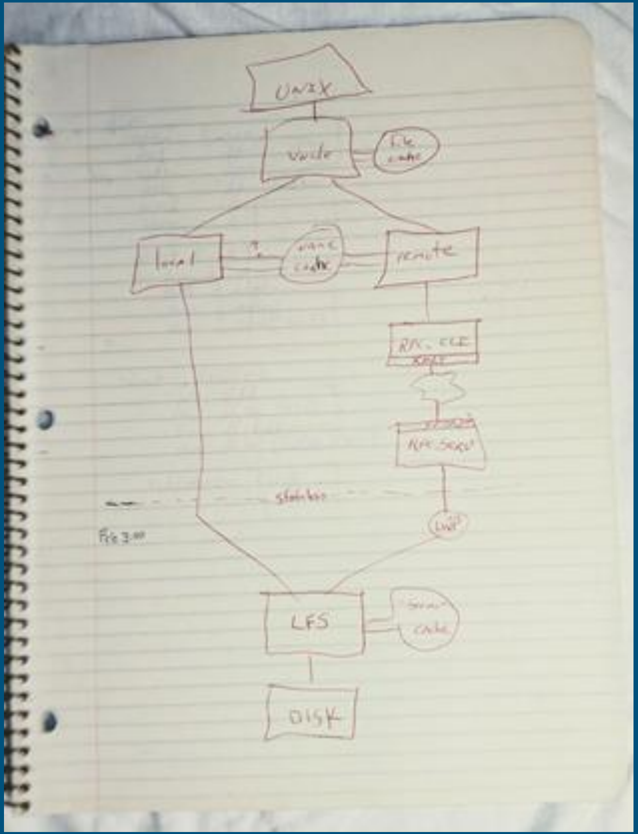
May or June 1984

Use after unlink! The “¾” solution - .nfsXXXXX files

Off to the races!

Participants:

Bill Joy, Dave Goldberg, Bob Lyon, Tom Lyon,
Joe Moran, Rusty Sandberg, Steve Kleiman



SunOS Release History

Sun UNIX - 1982 - Unisoft V7 UNIX, no networking

SunOS 0.4 Beta - Aug 1983 - 4.1cBSD, networking!

SunOS 1.0 - Nov 1983 - 4.2BSD Beta

SunOS 1.1 - Apr 1984 - 4.2BSD final and ND

SunOS 2.0 - May 1985 (long beta) - NFS! But also ND

SunOS 3.0 - Feb 1986 - Sun-3 HW and System-V compat libraries

SunOS 4.0 - Dec 1988 - VM rewrite, ND Eliminated – diskless NFS, Automounter

Statelessness

Error recovery is 100x harder in state-ful protocols

Like guaranteed vs best-effort delivery

“Just Retry” is so much easier

Network was flaky – single fat yellow coax

Servers – frequent downtime (backups, etc)

Servers must not depend on clients

Open Systems

Sun was committed to Ethernet and TCP/IP from day one.

Ethernet and TCP/IP dominated because of a community committed to openness and interoperability.

It was natural (at least for us engineers) to push NFS the same way.

The earliest partners were nascent large system vendors - Convex, Gould, Pyramid

Connectathon/Uniforum Feb. 1986

16 vendors

5 operating systems

PC-NFS 1.0 - June 1986 - MSDOS/PCDOS 3.x

Merges Not Made

Microsoft/SMB

Approached by MS early on, but - printers?

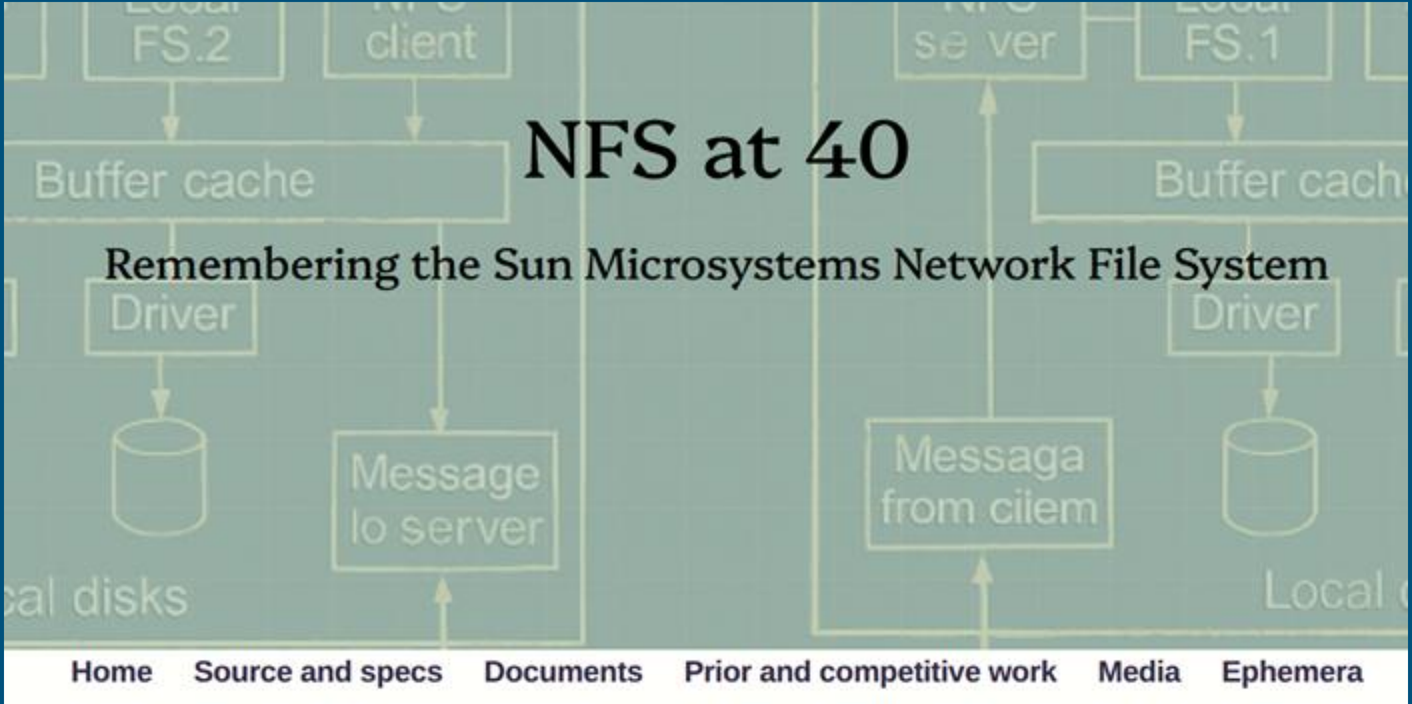
TOPS/Macintosh

Centram Systems West/TOPS/Sitka acquired by Sun

Lots of work ~1988 to define merge, thankfully dropped

Sun + Apple = Snapple

1996: serious merger talks incl. network services architecture



https://nfs40.online

NFS Evolution

“Protocols live forever.”

Brian Pawlowski
beepee@gmail.com

Lessons from the Journey

Separate Protocol from Implementation (don’t put the local API on the wire)

- Connectathon → Bakeathon

Standards Matter - but must evolve

- NFS is an open standard, proprietary protocols come and go

Evolution, Not Revolution

- Each version solved specific problems, backward compatibility maintained

Timing is Everything

- pNFS arrived just as AI/ML needed it

Simplicity Wins (Ubiquity is a terrible advantage

- NFS is native everywhere that is important)
- Built into OS = no special software, Standard Ethernet = no special hardware

NFS Distributed File System

crash recovery · CMU · NLM · DS/MDS · portmapper · flexfiles · UNIX semantics · delegation · reply cache · Y2K38 bug · RPC

NFS · delegations · UC Berkeley · automounter · sessions · pNFS · Bakeathon · seqid · stateless · idempotent

SPEC SFS · grace period · MTU discovery · ~~NetWare~~ · caching · Connectathon · LADDIS · nfsd · NFSv3 · nhfsstone

RFC · ~~DCE/DFS~~ · Prestoserve · ~~Sprite FS~~ · UDP · Sun Microsystems · XDR · ~~VINES~~ · mountd · VFS

soft mount · ACLs · GSS-API · IETF · ~~AFS~~ · TCP · NSM · CTO · clientid · RPCSEC · stale NFS · COMPOUND

hard mount · DEC · NFSv2 · ~~Apollo Domain~~ · Kerberos · nfsd · ~~POSIX~~ · NFSv4 · uidmapping · readdir cookies · BSD

opaque handles · knfsd · Athena · ND · time skew · SunOS · ac timeout · DRC · writeback coalescing · lock reclaim

Core NFS · Protocols · Benchmarks · Organizations · Versions · Features · Security · Defunct/Dead · Research/OS · Quirky Bits · NFSv4 Features · Testing Events

Broken Standards ❌ · Obscure Bits

Zeitgeist 2025

Linux finally available! 😉

File sharing - obviously desirable, but non-trivial

Disks - expensive! (can you say SSDs?)

Systems – frequent down time (backups, esp)

Network – flaky shared media

UNIX implosion

Share files – or just disks?

A Lot Happened Between 1994 and 2025

- For good or ill: World Wide Web (but Wikipedia!)
- Friends lost/gained: ~~Sun, SGI, DEC, Yahoo, AOL, Netscape, Apple,~~ VMware, Google, Amazon, nVidia, Paypal, Facebook, Netflix, OpenAI, Apple, Spotify
- Is that a computer in your pocket? Mobile revolution
- Looking very cloudy: AWS, Azure, Google, Oracle…
- Intel’s Pyrrhic victory with the x86 architecture (a story still writing itself)
- Music styles come and go, but it’s always coming via streaming
- Wi-Fi (“I don’t know what the network of the future will be, I just know it will be called Ethernet.”)

The Scale Challenge: 1994 vs 2025

1994 Reality:

- Typical file: 10KB-1MB documents
- Large dataset: 100MB database
- Network: 10 Mbps Ethernet
- Storage: **$1000/GB hard drives**
- NFSv2: 4GB file size limit seemed huge

2025 Reality:

- Typical file: 4K videos (100GB+), AI model checkpoints (10-500GB)
- Large dataset: Multi-petabyte data lakes
- Network: 100-400 Gbps standard, 800 Gbps emerging
- Storage: **$0.02/GB NVMe SSDs**
- NFSv4.2 with pNFS: Exabyte-scale deployments

- The 1,000,000x Challenge: Files grew 1000x, datasets grew 1,000,000x, but latency tolerance “stayed the same”

Timeline - 30 Years of Evolution

1995	RFC 1813	NFSv3
2000	RFC 3010	NFSv4.0 initial (we were this close)
2003	RFC 3530	NFSv4.0 stable (but…)
2010	RFC 5661	NFSv4.1 fixed minor versioning, pNFS
2010	RFC 5663/5664	Block/Object layouts
2015	RFC 7530	NFSv4.0 bug fix
2016	RFC 7862	NFSv4.2 Server-Side Copy, Application I/O Advise, Sparse Files, Labeled NFS.
2017	RFC 8154	SCSI Layout
2018	RFC 8435	FlexFiles layout
2020	RFC 8881	NFSv4.1 revised
2025	Active Development	FlexFiles v2, ACL/I18N clarification

NFS Version 2 from Sun Microsystems

Small Files: NFSv2 supported files to 4GB, UDP and 8KB block transfer

Ancillary Protocols: Locking, mounting was offloaded to separate, often problematic, protocols resulting in firewall and management complexity

Security: Weak security (AUTH_SYS only)

Performance Bottlenecks: Synchronous writes were required for data integrity, creating a significant performance ceiling (without solid state acceleration)

Weak Caching Semantics: Client-side caching was limited and relied on polling (chatty GETATTR calls)

NFSv3

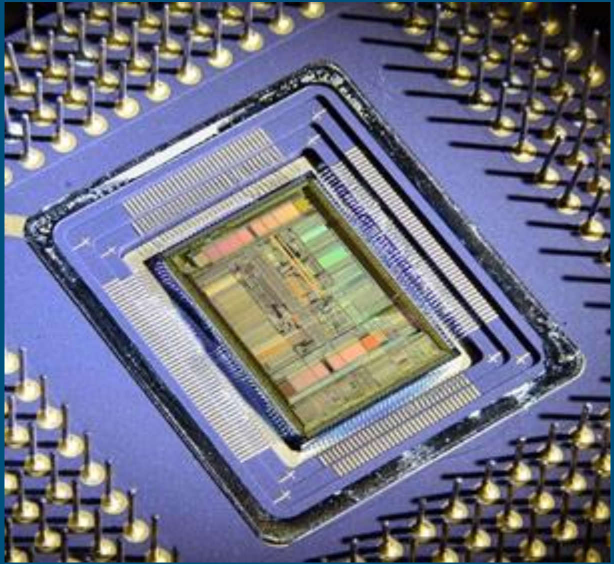
Alpha architecture (CPU wars) x NFS success drives DEC address 64 bit address (large file) support

Chet Juszczak at DEC threatens fork of NFS

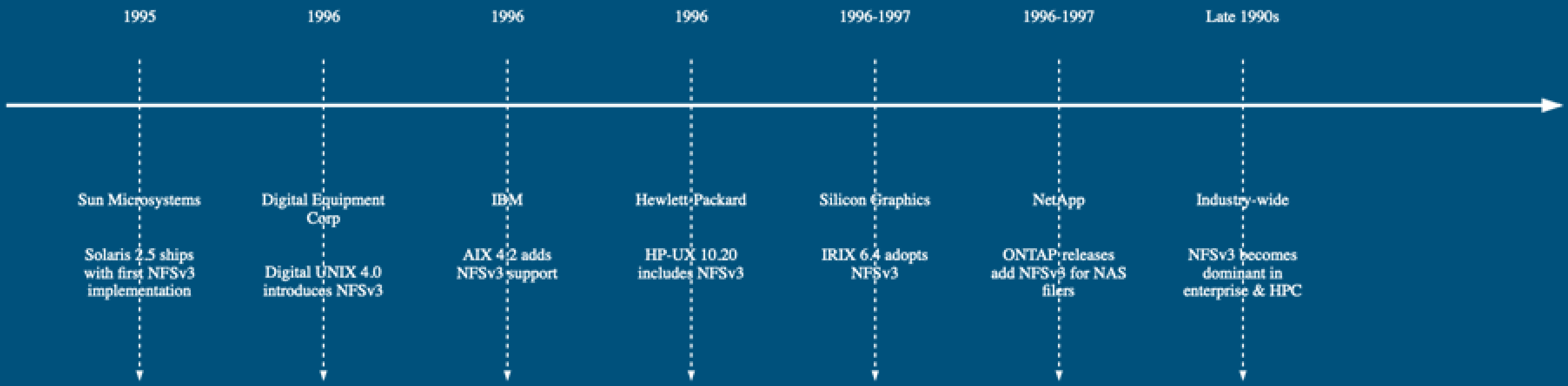
Companies collaborate at two week offsite in Massachusetts woods July 1992

UDP deprecated for TCP and 1+ MB transfer sizes

NFSv3 becomes ubiquitous (NFSv2 deprecated)



NFS Version 3 Adoption



Dead simple crash recovery - little server state (locking) to manage - widely implemented (dare we say ubiquitous)

NFS Version 3

~~**Small Files:** NFSv2 supported files to 4GB, UDP and 8KB block transfer~~

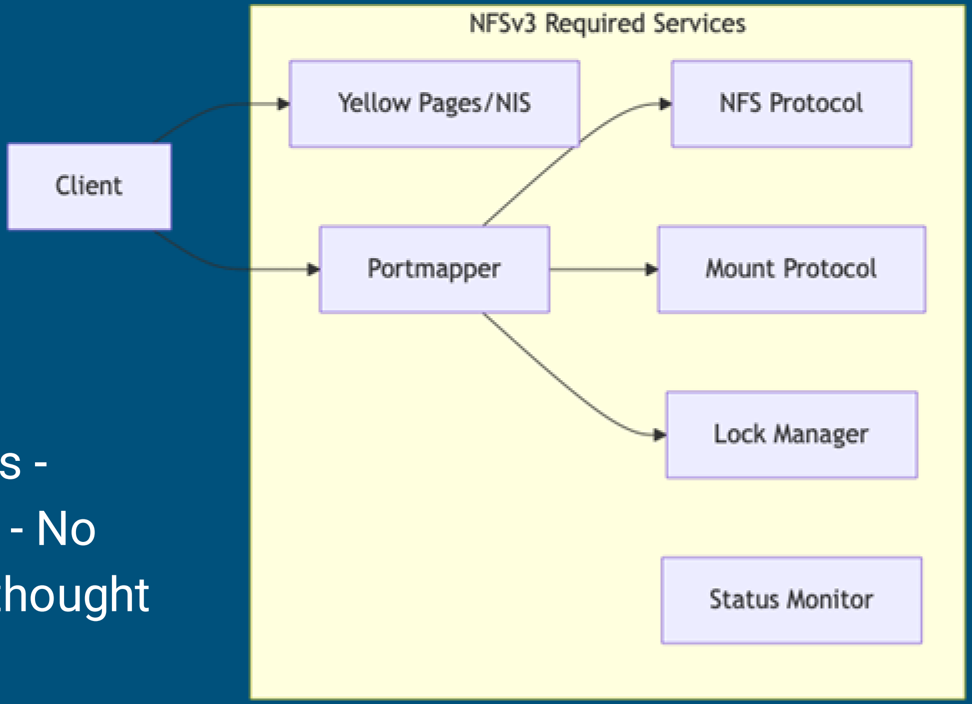
Ancillary Protocols: Locking, mounting was offloaded to separate, often problematic, protocols resulting in firewall and management complexity

Security: Weak security (AUTH_SYS only)

~~**Performance Bottlenecks:** Synchronous writes were required for data integrity, creating a significant performance ceiling (without solid state acceleration)~~

~~**Weak Caching Semantics:** Client-side caching was limited and relied on polling (chatty GETATTR calls)~~

The NFSv3 Ecosystem Complexity



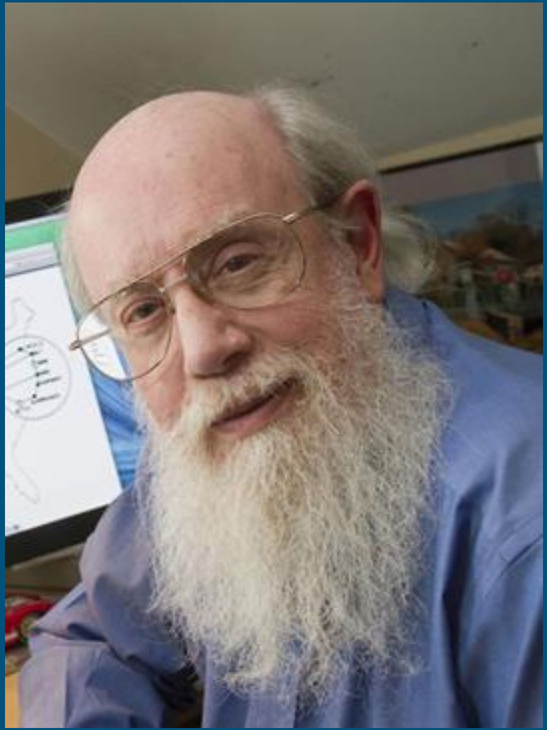
The Problem: - 6+ separate services - Dynamic ports (firewall nightmare) - No unified namespace - Security afterthought

I believe in Father Christmas!

In 1998 transitioned NFS work from a “proprietary technology” to an open IETF standard

Scott Bradner created RFC 2339, formal agreement where Sun Microsystems ceded control of future NFS to the IETF

Under Bradner's oversight, NFSv4.0 became the first version developed entirely within the IETF introducing stronger security, gracefully stateful connections, single port (2049) only, and performance optimizations.



NFSv4.0 (2000-2003) - The Stateful Revolution

RFC Timeline: - RFC 3010 (Dec 2000)/RFC 3530 (Apr 2003)

1. Honestly Stateful Protocol - Integrated locks & leases (remember Sprite?)
2. Mandatory Security - RPCSEC_GSS/Kerberos (mandatory to implement🙄)
3. Single Port/Firewall Friendly - TCP 2049 only
4. COMPOUND Operations - Batch multiple ops
5. Delegations - stronger client-side *file* caching
6. Pseudo-filesystem a virtual layer organizes all exported paths into one namespace (server-side automounter) (influence of AFS)

Why NFSv4.0 Wasn't Enough

Critical Problems Wrestled With (2003-2009):

1. Callback Complexity (Server → Client callbacks blocked by NAT/firewalls, Delegations often unusable)
2. No Exactly-Once Semantics (Non-idempotent operations could execute multiple times, Risk of data corruption)
3. Still a Single Server Bottleneck (All I/O through one server, Linear scaling impossible)

The HPC Challenge: "We need to move 100s GB/s to 1000 compute nodes"

NFSv4.1 (2010): The Performance Revolution

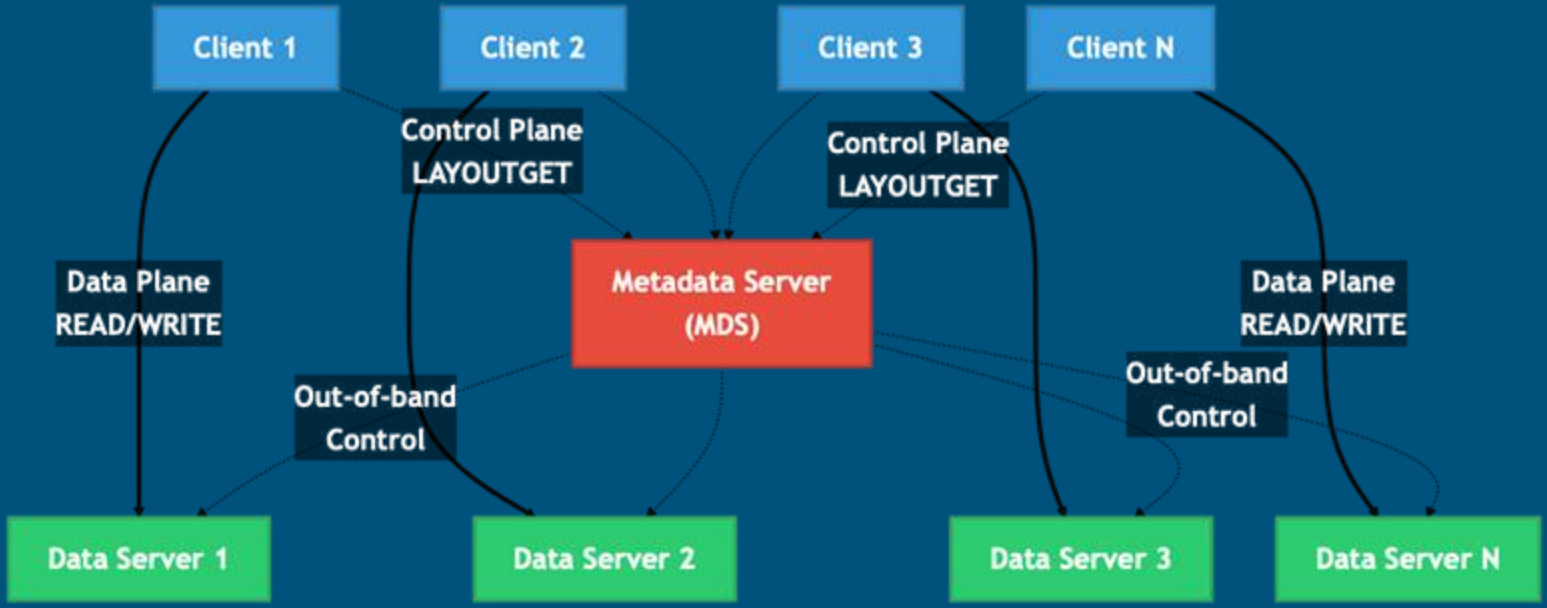
RFC 5661 (Jan 2010)/RFC 8881 (2020): Significant Protocol Extensions

The Sessions Model: Exactly-once semantics guaranteed (prevent destructive replay) - Client-initiated callbacks (persistent two-way channel solves reverse NAT/firewall) - Connection trunking (multipath bandwidth aggregation) - Reliable recovery

Advanced Features: Directory delegations - Multi-server namespace - pNFS - Parallel NFS

Key Innovation: Separation of control and data planes

pNFS Protocol Flow



Decouple the Control Plane from the Data Plane

Performance Impact: - 10x-100x throughput improvement - scales to thousands of clients

pNFS Layout Types Evolution

Year	Layout Type	RFC	Target Storage	Use Case
2010	File	5661	NFS servers	General purpose
2010	Block	5663	SAN/iSCSI	Trusted environments
2010	Object	5664	Object stores	Cloud storage
2017	SCSI	8154	SCSI devices	Modern SAN
2018	Flex Files	8435	Any	Dynamic tiering
2024	NVMe	9561	NVMe	Binding to SCSI layout

2025 Development: Flex Files v2 with erasure coding

Key Point: pNFS adapts to any storage architecture

NFSv4.2 (2016): Refining the Protocol (RFC 7862)

A proper minor version on top of NFSv4.1 added storage-aware features to bring remote file access semantics closer to local filesystems

- Server-Side Copy (SSC): Client instructs the server to perform a copy internally (COPY, CLONE), eliminating network overhead .
- Sparse Files & Space Reservations: SEEK to find holes, DEALLOCATE to punch them, ALLOCATE to pre-reserve storage space, crucial for applications like databases.
- Enhanced Client-Server Interaction: I/O Advise (IO_ADVISE): Client provides hints about I/O patterns (sequential, random) to the server for optimization, Layout Statistics (LAYOUTSTATS): Client reports performance and error data from DSs back to the MDS, enabling intelligent layout decisions.

Meeting Modern Workloads

NFSv3 (1995): Solved the immediate "large file" problem

- Freed from 4GB prison
- Enabled first digital video workflows

NFSv4.0 (2003): Solved WAN and security for distributed computing

- More “WAN” friendly

NFSv4.1/pNFS (2010): Solved the bandwidth wall

- Eliminate single-server bottleneck
- Scaled to meet AIML/HPC demands

NFSv4.2 (2016): Application optimizations

- Server-side operations for object-like semantics
- Sparse files for containers/VMs
- Ready for Kubernetes era

Lessons from the Journey

Separate Protocol from Implementation

- Connectathon → Bakeathon

Standards Matter - but must evolve

- NFS is an open standard, proprietary protocols come and go

Evolution, Not Revolution

- Each version solved specific problems, backward compatibility maintained

Timing is Everything

- pNFS arrived just as AI/ML needed it

Simplicity Wins (Ubiquity is a terrible advantage

- NFS is native everywhere that is important)
- Built into OS = no special software, Standard Ethernet = no special hardware

NFS @ 40

https://nfs40.online