# Beyond Microsoft

Tom Talpey

SambaXP 2023

Göttingen

# About…

- About Me
- About Microsoft
- About Protocols
- About What's Beyond

# About me

# My Background

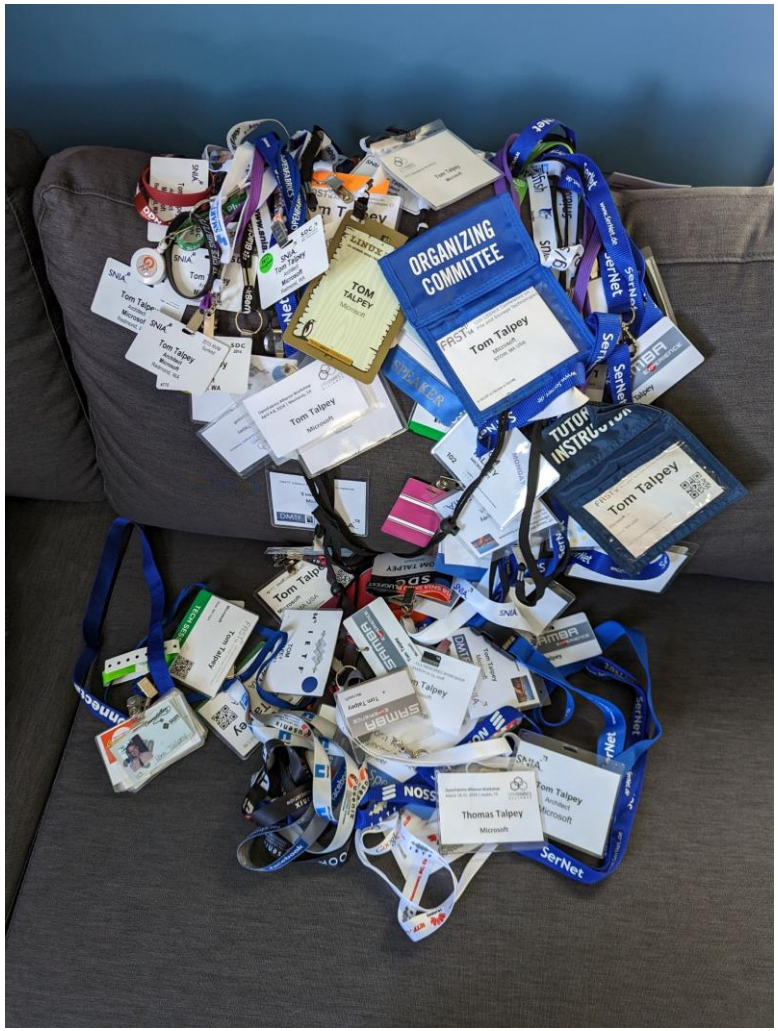
- A bottom-up journey from Networking, into Storage
- At Brown University, as a Ph.D. student in pure Mathematics…
- At our startup XPI, building the industry's first X-Terminal…
- At Open Software Foundation, an early open source consortium, with the OSF/1 operating system…
- (a detour not really relevant to today)…
- At another startup Orca Systems, with all forms of RDMA-enabled storage…
- Which was bought by NetApp, who commercialized NFS and desired to add higher performance, eventually laying us all off…
- And finally at Microsoft, where we revolutionized the SMB3 protocol, and, brilliantly, if at first unwillingly, documented it
- I'm now "unaffiliated", perhaps retired, but perhaps not

# The oldest badge I could find

From the NFS Connectathon 1989, where I tested an NFSv2 implementation on our X-Terminal. No, I wasn't working for Sun, they hosted the event.

# Pile’o’badges

I’m sure there were dozens more, before I started hanging them somewhere. I’d say I presented at roughly half of these events, which go back decades.

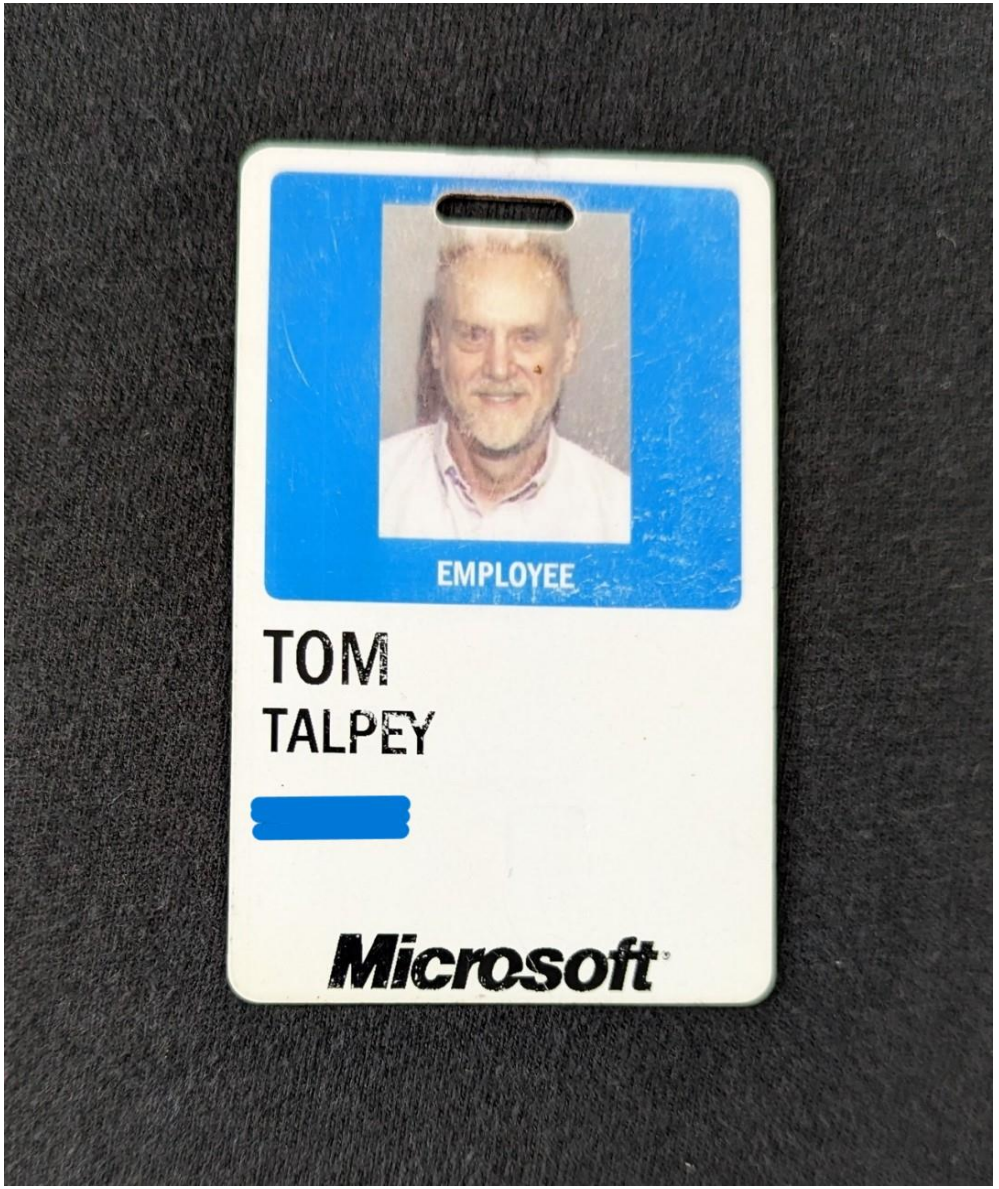


# About Microsoft

# Why I joined, and how I did

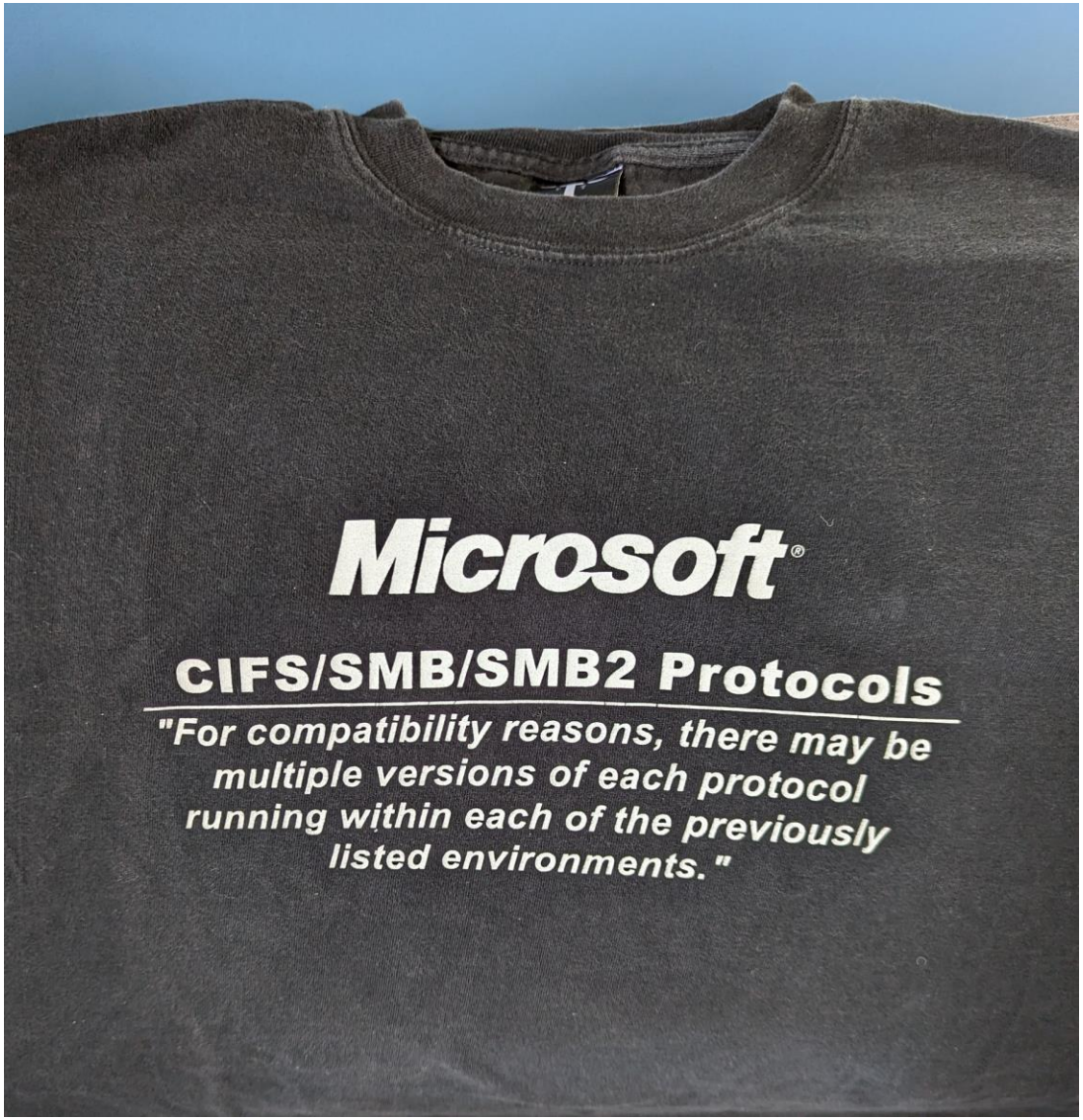
- Some stories to show how serious Microsoft was
  - And why it convinced me to join!
- P.S. Samba and Tridge had a lot to do with it

# My Interview



# The Documentation Effort

# A Favorite “TDI”



# I’m Proud Of…

- The proven quality of the documents
- The interoperability benefit (e.g. Samba’s)
- The protocol correctness benefit
- The Microsoft internal development benefit
- SMB3 and SMB Direct
- In short, the **innovation**

# About Protocols

# Two Important Things I Have Learned About Protocols

- Protocols Are Forever
  - Once you ship them, they never disappear
  - Including that very first version, with which you will always have to interoperate
  - Keep ‘em simple, true, and “right”
  - Corollary: Protocols Are Hard
    - They take a clear head, and broad consensus
    - And… **time**
- Protocols Can Be Extended
  - With carefully crafted new operations
  - But <u>always</u> implement the core protocol strictly
  - Recognize the peer’s capabilities, then use them
  - With even broader interop as a result

# 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

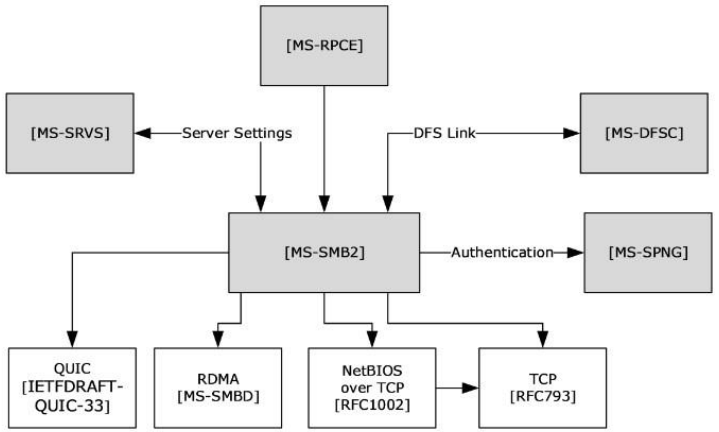



**Figure 2: Relationship to other protocols**

# MUST versus SHOULD versus MAY

- The IETF meaning isn't the entire story
- These terms need to be read in context
- In the Microsoft docs, they mean
  - MUST [NOT]: Windows always [never] does it
  - SHOULD [NOT]: Windows sometimes does [not] do it
    - A behavior note "must" always be added
  - MAY: Windows maybe does it
    - This term is rarely used, because it's non-normative
- So, in another implementation, *their meaning changes*
- If no other implementations exist, well…
  - These terms are largely meaningless, and therefore uninteresting
  - See "Two Important things" slide, and rethink your plan

# Protocol Requirements versus Protocol Behavior

- MUST (requirement)
- SHOULD (behavior)
- Silence (informational / implementation choice)
- Consider which to use for each statement
- And, saying nothing:
  - Allows (careful) behavior changes over time
  - Opens the door to future extensions
  - Opens the door to alternative implementations
  - Saves your own butt when you need to do these
- IMHO, the Microsoft SMB docs got this mostly right
  - Example: NTFS is a behavior, to SMB3
  - Example: SMB Direct is a behavior, to SMB3

# 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? ☺

# About What's Beyond

And please forgive me for focusing on SMB3 here

# SMB3 Innovation

- Samba needs to pick up the flag and extend SMB3
- Because the whole cloud thing has changed the field and altered Microsoft's course on filesharing
- Unix (Posix) Extensions, absolutely
- Linux semantics, definitely
- But, what else?
- Choose carefully

# Protocols and Open Source

- Developing protocols isn't like developing code
- Define the requirements, and the architecture
- Then simplify, and delete what doesn't fit
- Then consider how you and others will extend it someday
- Then implement it while documenting it, using the process to learn what you missed
- Only then, "ship" it

# Posix Extensions

- Maybe got ahead of some of the above
- But are still pretty far along
- The document still needs to be written
  - David Mulder's started it! Pitch in.
- The public review basically hasn't started
  - I don't have the resources to run smb3unix.org
  - But I do own the domain ☺
  - Maybe do it another way?

# “Linux Extensions”

- Linux has arguably extended Posix
- What Linux-specific things can be expressed remotely over SMB3?
  - How can the SMB3 protocol collaborate?
  - Steve’s got a client-focused list, good start
  - What about the Samba server (when running on Linux)?
  - What about when running in the cloud?
    - Let’s not forget auth mechanisms, etc

# And, what else?

- RDMA
  - Metze's smbdirect.ko is a fundamentally great idea
    - It's the SMBDirect framing protocol socket, with an RDMA read/write sideband, in-kernel and accessible from user space
  - But in this form it'll be a hard swim to upstream
  - Need ways to integrate the benefits it provides, without the arguments
  - Proposal: start by merging the duplicative kernel cifs.ko and ksmbd.ko smbdirect implementations
    - Which is needed badly anyway

# And Speaking Of ksmbd

- Don’t let ksmbd and smbd diverge
  - It’s on that road now, even if ksmbd claims to share smb.conf
- At the same time, don’t merge them
  - There are reasons to have both
- BTW ksmbd, um, needs some security work

# Whatever We Do…

- Make things easy and simple
- Make discovery of extensions the default
- No required mount options
- No required smb.conf stanzas
- No complex instructions implying any of these are needed
- “It just works”, for whatever “it” is!

# Thank you!

Questions/discussion?