SMB3 Protocol Update
2020 edition!

Tom Talpey

Microsoft Corporation

Outline

- SMB3 Protocol since last year
- SMB3 Protocol update in current “20H1”
- SMB3 Protocol changes coming
- Other related developments

Important

- This presentation has been prepared with all appropriate social distancing
- It has been quarantined and is free of viral influence
  - Probably.
  - Ok, maybe.

MS-SMB2 Document

- Updated March 4
  - At the “familiar” URL ☺
  - https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962
- Errata are published regularly
  - Updated May 25, 2020
  - https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-winerrata/2cdafcfa-ce51-426a-9678-630a505a1a35
- N.B. try these without the GUID (just the doc name)

SMB3 Protocol Changes

MS-SMB2

- Windows and Windows Server “20H1” release
  - A.k.a. Windows 10 version “2004”
  - Any Day Now
- Updated doc March 4
  - https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962
- Also covering 19H2 today
  - To catch up since SambaXP 2019

MS-SMB2 changes Summer/Fall 2019

- [MS-SMB2]-190923-diff.pdf
  - 19H2 is “quality release” overall – no new SMB3 features
  - Document is similarly changed, maintenance only
    - E.g. Netname negotiate context is not null-terminated
    - Fileid’s and their relation to MS-FSCC and caching
    - Document structural cleanup and common text merged
    - Document template fixes (Abstract Data Model, etc)
  - It was also relatively quiet for Technical Document Issues (“TDIs”)

MS-SMB2 Changes Winter/Spring 2019-2020

- [MS-SMB2]-200304-diff.pdf
  - 20H1 contains new SMB3 changes
    - Chained compression, new Pattern_V1
      - Much more on this shortly!
    - Somewhat increased TDI level
      - From protocol partners (Samba!)
      - And Microsoft protocol validation testing, performed with any “major” changes
    - Document maintenance
      - Oplock and Leasing additional new discussion

MS-SMB2 Changes – Recent Errata

- Significant clarifications for Pattern_V1 and chained compression
- Multichannel processing
- Session scavenger processing and ClientGUID handling
- Miscellaneous reconnection, lease cleanup and encryption fixes

- https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-winerrata/2cdafcfa-ce51-426a-9678-630a505a1a35

SMB3 New Protocol Features

SMB3 Changes

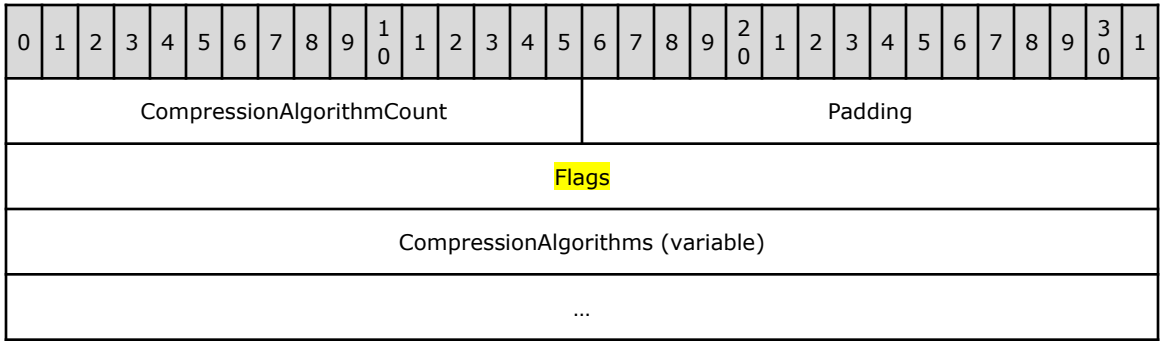
- New SMB3 features (negotiate contexts)
  - “Pattern_V1” compression
  - Chained compression
  - All other compression processing and policies remain
- Again, no dialect change
  - No dialect bump foreseen

Compression

- Modifies negotiate context SMB2_COMPRESSION_CAPABILITIES
  - Adds SMB2_COMPRESSION_CAPABILITIES_FLAG_CHAINED
  - Adds new algorithm “Pattern_V1”, defined in MS-SMB2 itself
  - MS-SMB2 section 2.2.3.1.3 (request) and 2.2.4.1.3 (response)
- Modifies new SMB2_COMPRESSION_PAYLOAD_HEADER
  - Makes OriginalPayloadSize optional to LZ algorithms
  - Adds chained flag
- Adds new SMB2_COMPRESSION_PATTERN_PAYLOAD_V1
  - For chained compressed payloads
- No changes to existing negotiation, or algorithms
  - See last year’s SambaXP for those ☺

SMB Compression (review)

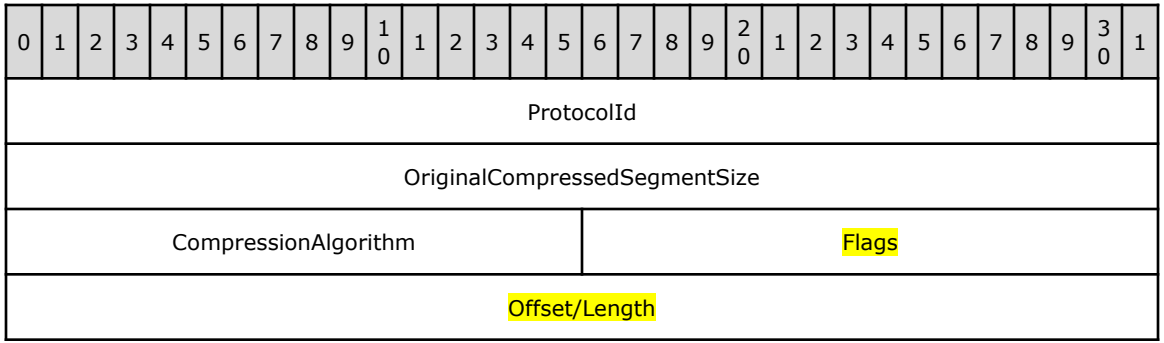
- Client optionally negotiates compression by appending negotiation context (ID = 0x0003)



- Server responds with the supported algorithms, sorted.
- New SMB2_COMPRESSION_CAPABILITIES_FLAG_CHAINED

Compression Transform (review)

- Eligible segment is replaced with compression transform (MS-SMB2 section 2.2.42) in SMB2 transform header
- Previously defined for 3 algorithms



New Compression negotiation flags and algorithm

Value	Meaning
SMB2_COMPRESSION_CAPABILITIES_FLAG_NONE 0x00000000	Chained compression is not supported.
SMB2_COMPRESSION_CAPABILITIES_FLAG_CHAINED 0x00000001	Chained compression is supported on this connection.

Value	Meaning
NONE 0x0000	No compression
LZNT1 0x0001	LZNT1 compression algorithm
LZ77 0x0002	LZ77 compression algorithm
LZ77+Huffman 0x0003	LZ77+Huffman compression algorithm
Pattern_V1 0x0004	Pattern Scanning algorithm

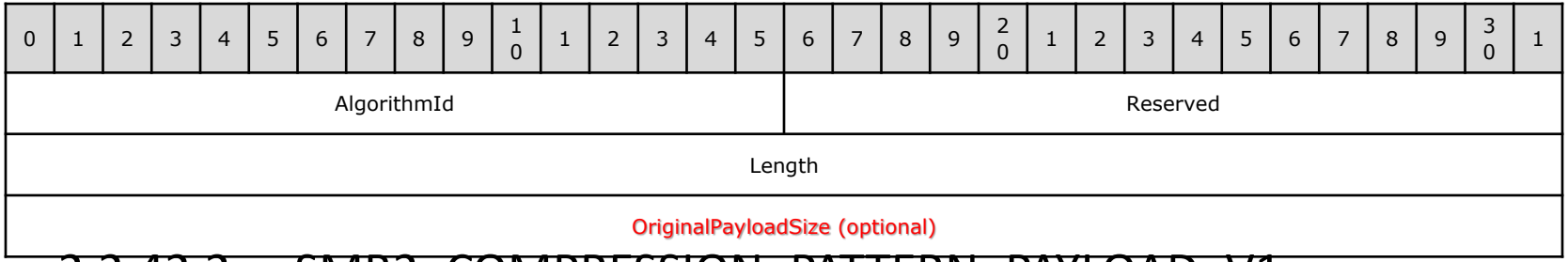
Chained Compression

- Compresses multiple segments within each message
  - With potentially different supported algorithms

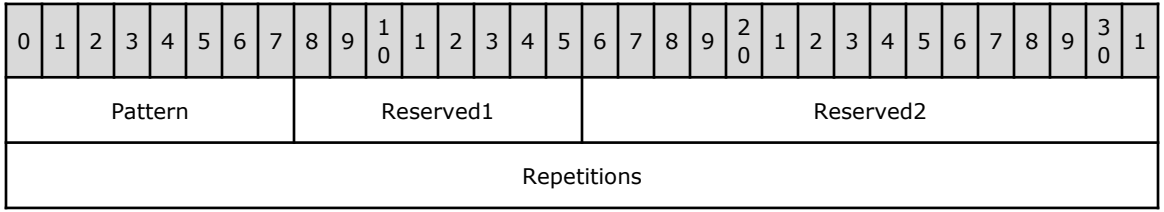
Value	Meaning
SMB2_COMPRESSION_FLAG_NONE 0x0000	Chained compression is not supported.
SMB2_COMPRESSION_FLAG_CHAINED 0x0001	The Compressed message is chained with multiple compressed payloads.

Chained transforms

- SMB2_COMPRESSION_PAYLOAD_HEADER



- 2.2.42.2 SMB2_COMPRESSION_PATTERN_PAYLOAD_V1



Pattern_V1 Compression

- “Run length” pattern matching
  - Sequential, equivalent values which repeat for a specified count
  - Match anywhere within a block
    - Typically, at “front” and/or “back”
  - Valid only with chained compression

Chained Compression Example

- An SMB2_WRITE of 4KB of data
- With the data to write containing:

  000000…55AA55AA55AA…FFFFFF
- This is recognized as three compressible segments:
  - Pattern_v1 of 00's
  - Compressible data (e.g. LZ77)
  - Pattern_v1 of FF's
- Here's how the block is sent:

Chained Compression Example (2)

1. SMB2_COMPRESSION_TRANSFORM_HEADER(Chained=1)
2. SMB2_COMPRESSION_PAYLOAD_HEADER(Pattern_v1, len1)
3. SMB2_COMPRESSION_PATTERN_PAYLOAD_v1(0x00)
4. SMB2_COMPRESSION_PAYLOAD_HEADER(e.g. LZ77, len2) [or None]
5. (LZ77 compressed data) [or uncompressed data]
6. SMB2_COMPRESSION_PAYLOAD_HEADER(Pattern_v1, len2)
7. SMB2_COMPRESSION_PATTERN_PAYLOAD_v1(0xFF)
8. Remaining HEADER+SMB2_WRITE and any additional uncompressed segments

Pattern_V1 Compression processing

- Eligible for any payload
- Most interesting for Virtual Disk and VM Live Migration
  - Where potentially long runs of 0's (and other patterns) are present
- "Front" and "Back" pattern scanning
  - "Internal" segments also eligible
  - Matches well to observed payloads
  - Certain other heuristics are applied (length, max expected savings…)
- In Windows, applied only on >=4KB segments
  - E.g. per-MDL segment in read or write (1 page or more)

Warning – Alignment!

- Take another look at slide 20
- What is the size of the LZ77 segment?
  - Anything!

➢The SMB2_COMPRESSION_PAYLOAD_HEADER (in 6) *may not be aligned*

- This may be addressed in a future protocol update

Notice - Uncompressed segments!

- If a segment is “short” (<64B), or doesn’t compress
- And segment is “in between” two compressible segments
  - i.e. not eligible, but additional compressible data follows it
- Then it becomes a “None”
  - Not compressed
  - Note previous warning on alignment
- Note, the lengths and limits are *behaviors*, and may differ among implementations

Multiple TRANSFORMs

- Encryption is also a transform
  - And is different from Chaining
- Always applied **after** compression
- Entire compressed transform is wrapped in TRANSFORM_HEADER
  - As previously defined by protocol

Yes, it can be complex

- Nature of the beast?
  - Many strange and wonderful patterns, and algorithms
- Test, Test, Test
- I guarantee you'll find issues
- Let me tell you some stories…

Documenting Compression

- Getting the compression text “right” has been a challenge
- Several issues found in manual and automated document testing review
- Others found when fixing those
- Look to the Errata!
  - And a future updated document

(https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-winerrata/2cdafcfa-ce51-426a-9678-630a505a1a35)

[MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3

[MS-SMBD]: SMB2 Remote Direct Memory Access (RDMA) Transport Protocol

[MS-SPNG]: Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension

[MS-SQOS]: Storage Quality of Service Protocol

[MS-SSTP]: Secure Socket Tunneling Protocol (SSTP)

[MS-SSTR]: Smooth Streaming Protocol

[MS-SWN]: Service Witness Protocol

[MS-TCC]: Tethering Control Channel Protocol

[MS-TDS]: Tabular Data Stream Protocol

[MS-TLSP]: Transport Layer Security

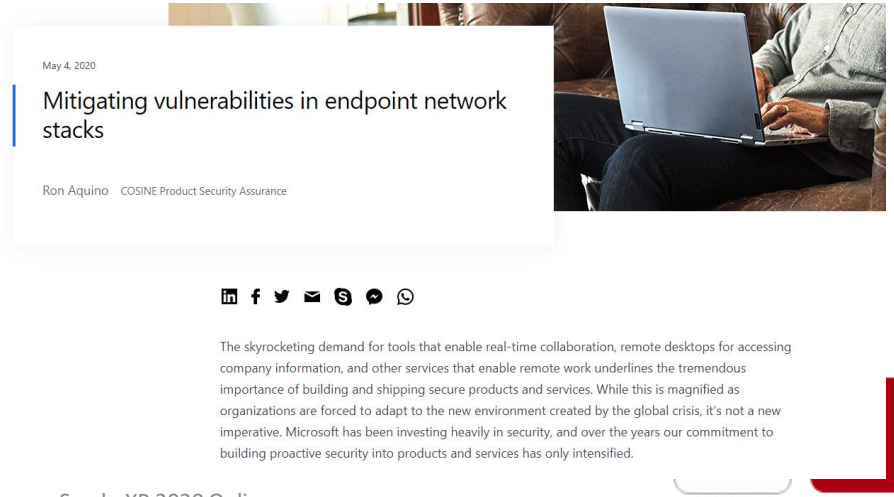
- Otherwise, the data MUST be decompressed as follows:
- If AlgorithmId is Pattern_V1, the next 8 bytes MUST be interpreted as SMB2_COMPRESSION_PATTERN_PAYLOAD_V1, specified in section 2.2.42.2.
- If Repetitions in SMB2_COMPRESSION_PATTERN_PAYLOAD_V1 is greater than OriginalCompressedSegmentSize in SMB2 COMPRESSION_TRANSFORM_HEADER, the connection MUST be disconnected as specified in section 3.2.7.1 or 3.3.7.1.
- Otherwise, DecompressedMessage MUST be appended with Repetitions number of bytes initialized with the character specified in Pattern field.
- Otherwise, the data of size specified in Length field MUST be decompressed using the algorithm specified in AlgorithmId field as specified [MS-XCA] section 2. DecompressedMessage MUST be appended with the decompressed data.
- RemainingCompressedDataSize MUST be decremented by the size in Length field.
- If the size of RemainingCompressedDataSize is greater than the size of SMB2_COMPRESSION_PAYLOAD_HEADER, the receiver MUST repeat step 2.

3. DecompressedMessage MUST be returned.

2020/05/25	In Section 3.1.4.4, Compressing the Message, the following was changed from: 1. The sender MUST initialize RemainingUncompressedDataSize with the size of uncompressed SMB2 message, TotalCompressedDataSize with 0, and CompressedMessage with empty buffer. 2. The message MUST be compressed until RemainingUncompressedDataSize is greater than zero:

To really convince you

- https://www.microsoft.com/security/blog/2020/05/04/mitigating-vulnerabilities-endpoint-network-stacks/
- Please not another “Hold My Beer” moment! ☺



May 4, 2020

Mitigating vulnerabilities in endpoint network stacks

Ron Aquino COSINE Product Security Assurance

The skyrocketing demand for tools that enable real-time collaboration, remote desktops for accessing company information, and other services that enable remote work underlines the tremendous importance of building and shipping secure products and services. While this is magnified as organizations are forced to adapt to the new environment created by the global crisis, it's not a new imperative. Microsoft has been investing heavily in security, and over the years our commitment to building proactive security into products and services has only intensified.

SMB3 Protocol Futures

Possible protocol features

Yes, you've seen some of these before

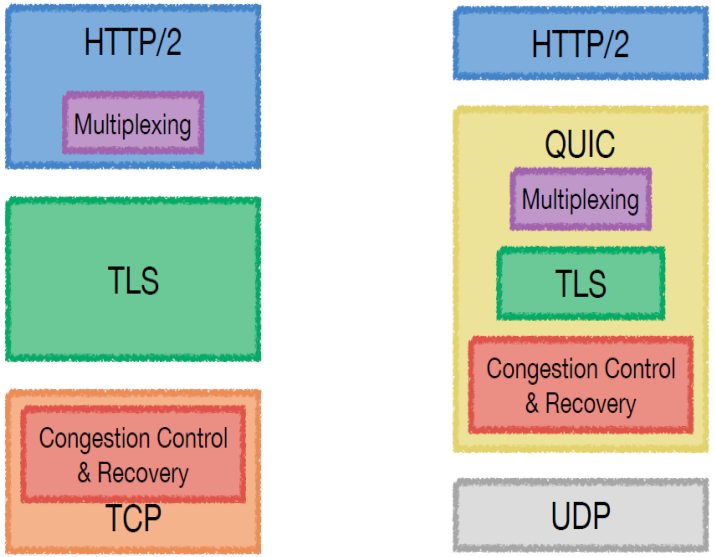
- Client compression control
- SMB over QUIC
- New transforms and signing
  - High performance AES-GMAC signing
  - Enhanced encryption algorithms
  - Compression alignment enhancement
  - Signing/Encryption over RDMA
- RDMA direct access to persistent storage

Using compression

- Currently there is no way to “force” compression in Windows
  - Client negotiates compression but then applies rules locally
    - Typical workloads are not compressed, due to CPU impact
  - Server only compresses when asked
- Possible robocopy compression flag
  - And/or other client-local mechanisms
- Details and plans TBD – stay tuned

QUIC:UDP based secure stream transport

- Low-latency connection setup
  - 1-RTT for initial connections
  - 0-RTT for repeat connections.
- Secure and Encrypted (TLS 1.3+)
- Improvements over HTTP/2 (“H2”) and TCP
  - Multiple Stream Support
  - ALPN for better multiplexing
  - Support for connection migration
  - Better congestion control & loss recovery



SMB Bindings for QUIC

- QUIC connections can share same 4-tuple
  - Can multiplex using an ALPN identifier
  - Can share same UDP/443 port with HTTPS traffic
- Use QUIC as a single channel TCP replacement
  - SMB multichannel will use separate QUIC connections.
  - Not currently envisioning using QUIC streams
- QUIC will allow cloud SMB access?
  - No more port TCP/445 blocking !

https://www.snia.org/sites/default/files/SDC/2018/presentations/SMB/George_Xin_SMB3_Landscape_Directions.pdf

SMB3 Signing – Enabling AES-GMAC

- SMB3.x (still) uses AES-CMAC for signing
  - AES-GCM based SMB3 encryption performs significantly better than AES-CCM based signing
  - Most modern processors have optimized instructions for AES-GCM computations
- Can we use AES-GMAC to similarly improve signing ?
  - Definitely yes
  - Previously shared results were 46% better CPU*
    - And processor support has improved since then

https://www.snia.org/sites/default/files/SDC15_presentations/smb/GregKramer_%20SMB_3-1-1_rev.pdf

Negotiable SMB Signing with New Algorithm

- Negotiable
  - Client will be able to negotiate switching to the AES128-GMAC algorithm for signing in future SMB 3.1.1. New negotiation context specifying the algorithm count and algorithm IDs:

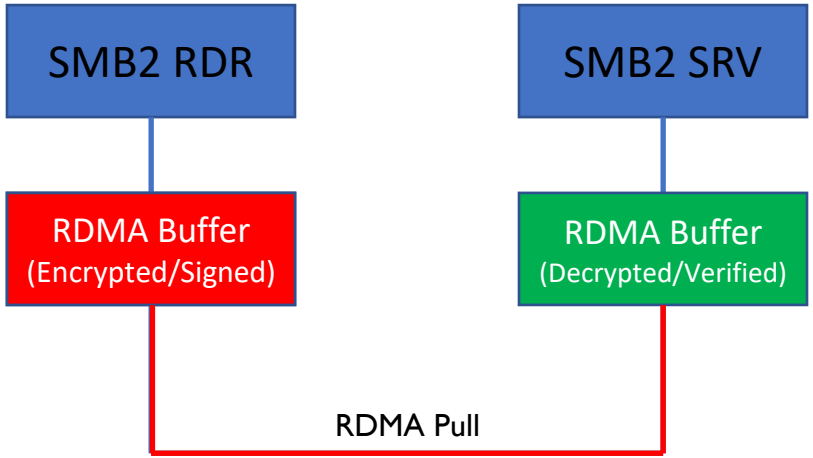
  - Supporting server will select 1 signing algorithm, if possible, and respond with:

  - More algorithms may be added over time

Signing and Encryption in RDMA

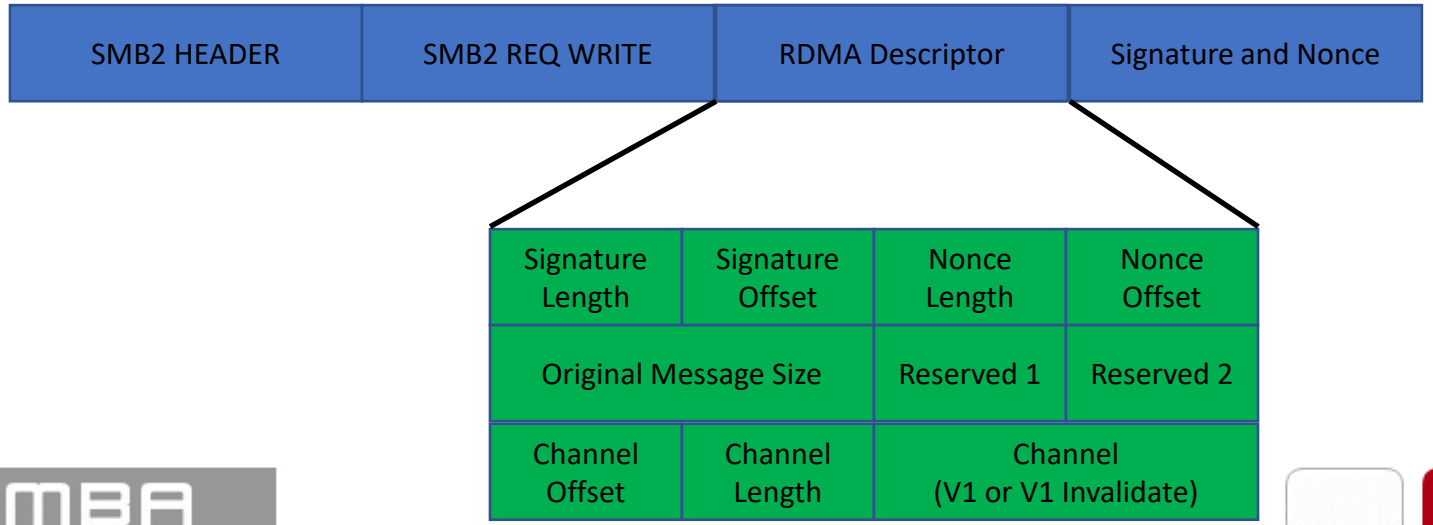
- Signing and Encryption over SMB RDMA.
  - Performance gain over current packet-based authenticated and/or encrypted traffic over SMB RDMA.
  - Supports any negotiated signing or encryption.

E.g. An SMBDirect write:



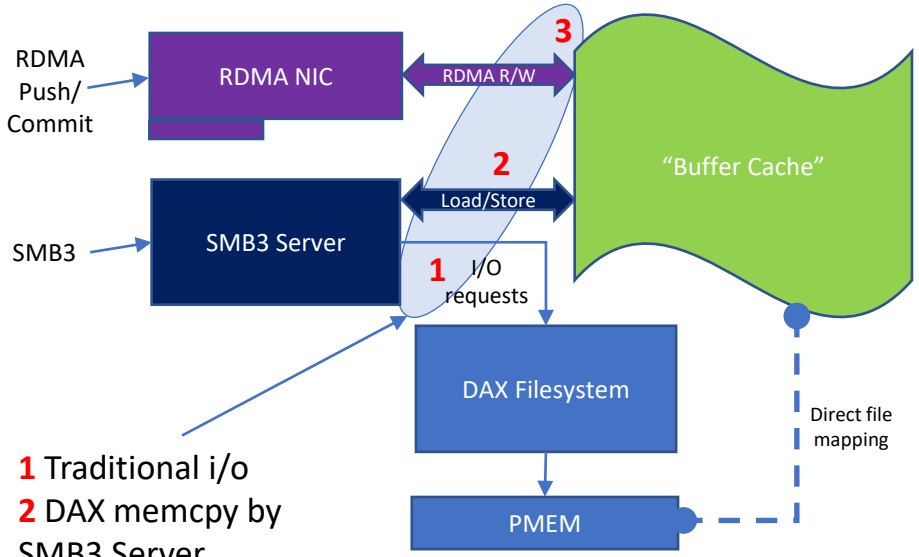
Signing and Encryption in RDMA

- How to transmit signature and nonce?
- Transform Descriptor as channel payload! (SMB2_CHANNEL_RDMA)
  - Similar transform descriptor used with SMB2 Read Response



SMB3 Push Mode to Persistent Memory/DAX

- SMB3 RDMA and “Push Mode” discussed at previous events
- Enables zero-copy remote read/write to DAX file
  - Ultra-low latency and overhead
  - Single-digit microsecond!
- Minimal SMB3 and RDMA protocol extensions required



SDC2019 Results

Latency measurements (NVDIMM-Write)

```
cfstest.exe r-write s:\file.bin /o:<QD> /b:<BLOCKSIZE> /s:2G
/writethrough [/mapfordirectaccess:2G]
```

Latency (us) → Perf Ctr : SMB Client Shares\Avg. sec/Write	SMB3	Push-mode w/ software flush	Push-mode (DDIO disabled)
QD=1, 4K	81	26	14
QD=2, 4K	84	32	15

* CPU utilization on server is zero.

ments (NVDIMM-Read)

```
/o:<QD> /b:<BLOCKSIZE> /s:2G
ss:2G]
```

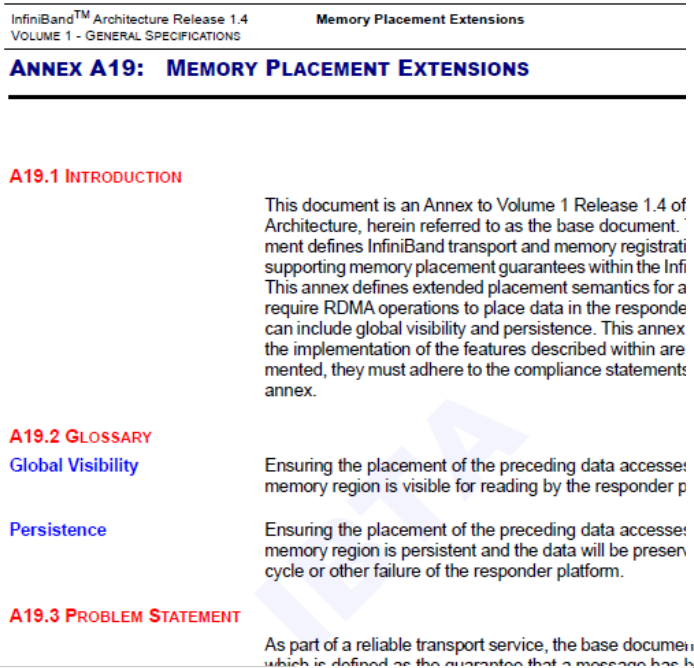
Perf Ctr : SMB Client Shares\Avg. sec/Read	SMB3 w/Push-mode Perf Ctr : SMB Client Shares\Avg. sec/Read
QD=1, 4K	51	15
QD=2, 4K	52	16

https://www.snia.org/sites/default/files/SDC/2019/presentations/SMB/George_Mathew_Talpey_Tom_Storage_RDMA_Push_Mode_to_Persistent_Memory_via_SMB3.pdf

RDMA Protocol Extensions

- These extensions advancing in IBTA (IB, RoCE) and IETF (iWARP)
  - RDMA Flush is flush to durability
  - Atomic Write places pointer-sized data after flush
    - Transactional, e.g. for log write pointer update
- The iWARP extension additionally contains
  - Verify to compute and compare a hash of region contents
    - And is ordered to Flush and Atomic Write for transactional integrity
- Push Mode only requires RDMA Flush

RDMA Extension Document Drafts



InfiniBand™ Architecture Release 1.4 — Memory Placement Extensions
VOLUME 1 - GENERAL SPECIFICATIONS

ANNEX A19: MEMORY PLACEMENT EXTENSIONS

A19.1 INTRODUCTION

This document is an Annex to Volume 1 Release 1.4 of
Architecture, herein referred to as the base document.
ment defines InfiniBand transport and memory registrati
supporting memory placement guarantees within the Infi
This annex defines extended placement semantics for a
require RDMA operations to place data in the responde
can include global visibility and persistence. This annex
the implementation of the features described within are
mented, they must adhere to the compliance statements
annex.

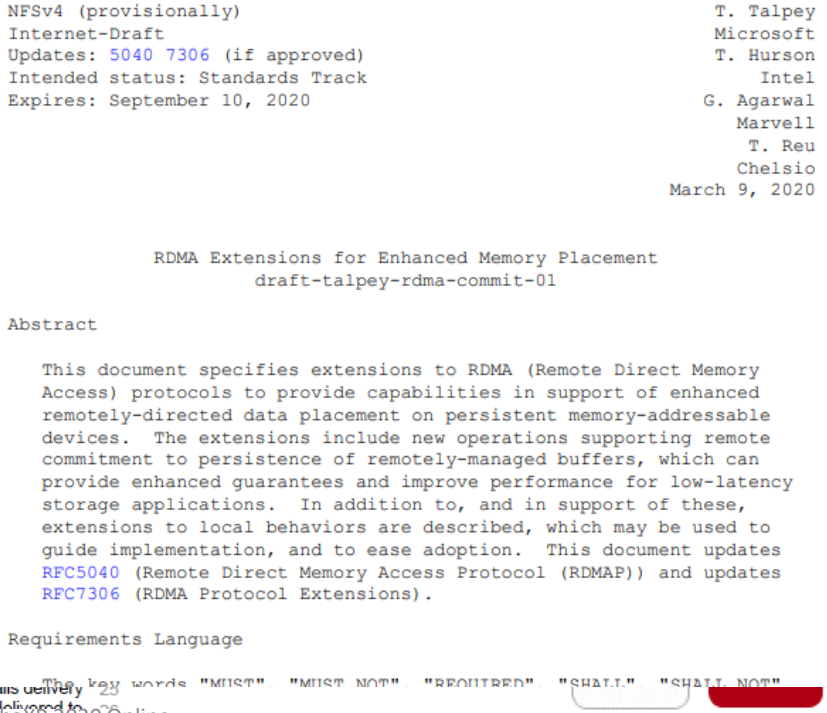
A19.2 GLOSSARY

Global Visibility — Ensuring the placement of the preceding data accesse
memory region is visible for reading by the responder p

Persistence — Ensuring the placement of the preceding data accesse
memory region is persistent and the data will be presen
cycle or other failure of the responder platform.

A19.3 PROBLEM STATEMENT

As part of a reliable transport service, the base document details delivery
which is defined as the guarantee that a message has been delivered to



NFSv4 (provisionally)
Internet-Draft
Updates: 5040 7306 (if approved)
Intended status: Standards Track
Expires: September 10, 2020

T. Talpey
Microsoft
T. Hurson
Intel
G. Agarwal
Marvell
T. Reu
Chelsio
March 9, 2020

RDMA Extensions for Enhanced Memory Placement
draft-talpey-rdma-commit-01

Abstract

This document specifies extensions to RDMA (Remote Direct Memory Access) protocols to provide capabilities in support of enhanced remotely-directed data placement on persistent memory-addressable devices. The extensions include new operations supporting remote commitment to persistence of remotely-managed buffers, which can provide enhanced guarantees and improve performance for low-latency storage applications. In addition to, and in support of these, extensions to local behaviors are described, which may be used to guide implementation, and to ease adoption. This document updates RFC5040 (Remote Direct Memory Access Protocol (RDMAP)) and updates RFC7306 (RDMA Protocol Extensions).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT"

SMB Push Mode Protocol Extensions

- SMB3 protocol is not extended
  - Only new FSCTLs
- Client requests “Push Mode” handle on DAX file
  - Just an RDMA memory handle/list, long-lived
  - Server registers DAX-mapped file
  - Associated with a lease for protection and recall
- Client performs RDMA instead of SMB2_WRITE/SMB2_READ
- Client Flushes writes to PMEM
  - With RDMA extension, if available on both sides
  - With SMB2 FSCTL or other operation, if not

SMB3 Interop Events

2020 SMB3 Interop Events

- SDC EMEA (Tel Aviv) held in January
- Redmond DevDays deferred from June
- SDC2020 (Santa Clara) “on the bubble”
  - And frankly, unlikely as an onsite event
- Please keep testing!
  - The Microsoft test suites are usable anywhere
  - Microsoft is committed to future events
  - Remote interop testing is a possibility in future?

Summary

- SMB3 continues to evolve, steadily
- Microsoft values the Samba contribution and partnership
- Let’s keep the momentum despite COVID-19 impact

Thank you!