SMB3 Protocol Update

Tom Talpey

Microsoft Corporation

Outline

- SMB3 Protocol changes
- SMB3 Protocol futures
- Possible Microsoft/Samba collaborations

SMB3 Protocol Changes

MS-SMB2

- Windows and Windows Server “19H1” release
  - A.k.a. Windows 10 version 1903
  - May 22, 2019
- Updated doc March 13
  - Corrections/updates April 30
  - https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962
- Also covering 18H2/Server2019 today
  - Since it’s a year since we met here!
  - Largely maintenance – no protocol changes

SMB3 Changes

- New SMB3 features (negotiate contexts)
  - Compression
  - Server netname
- No dialect change
  - No dialect bump foreseen
  - Since SMB2/3 now has forward-compatible contexts in
    - Negotiate
    - Tree Connect

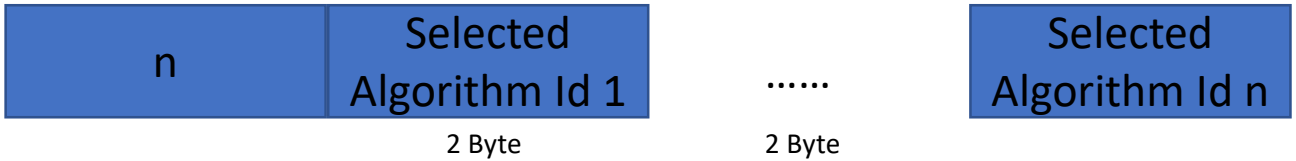
Compression

- New negotiate context SMB2_COMPRESSION_CAPABILITIES
  - MS-SMB2 section 2.2.3.1.3 (request) and 2.2.4.1.3 (response)
  - ID 0x0003
- New SMB2_COMPRESSION_TRANSFORM_HEADER
  - New transform specifically for compression
  - MS-SMB2 section 2.2.42
- Also SMB2_READFLAG_REQUEST_COMPRESSED
  - New flag in SMB2_READ request
  - MS-SMB2 section 2.2.19

Negotiable SMB Traffic Compression

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

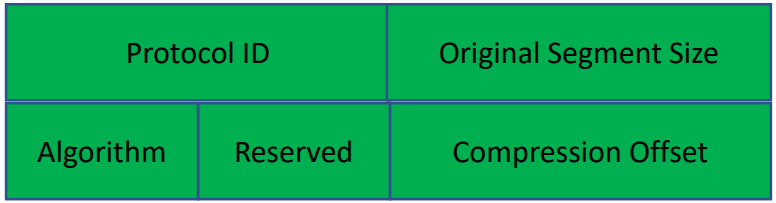
- Supporting server selects subset of compression algorithms, if any, and responds with:



- Supported compression algorithms defined in MS-XCA:
  - XPRESS (also known as LZ77)
  - XPRESS Huffman (LZ77+Huffman)
  - LZNT1

Compression + Signing/Encryption Interop

- New, compact transform header for SMB Compression (16B)



- When compression and signing or encryption are needed, transform headers are nested
- Compress always first: regular transform header always the *outer* transform header

Compression processing

- MS-SMB2 section 3.1.4.4
- Choice of compression types by sender, on each operation
  - As appropriate to type of data, performance, etc
- Compress Writes and requesting compress Reads for client
- CompressAllRequests override for client
- Not over RDMA (for now)

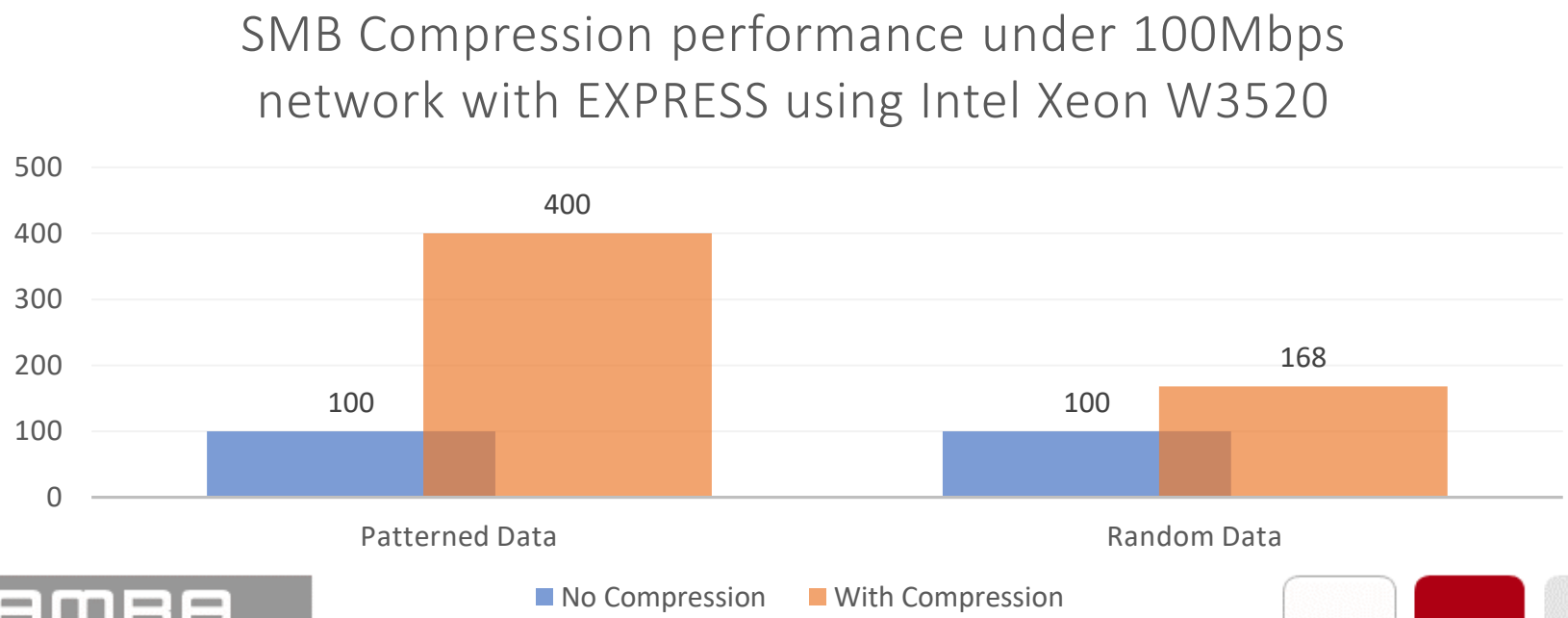
Decompression processing

- MS-SMB2 section 3.2.5.1.10
- Drops connection on fail (size mismatch)
- Inevitably drops connection on garbage

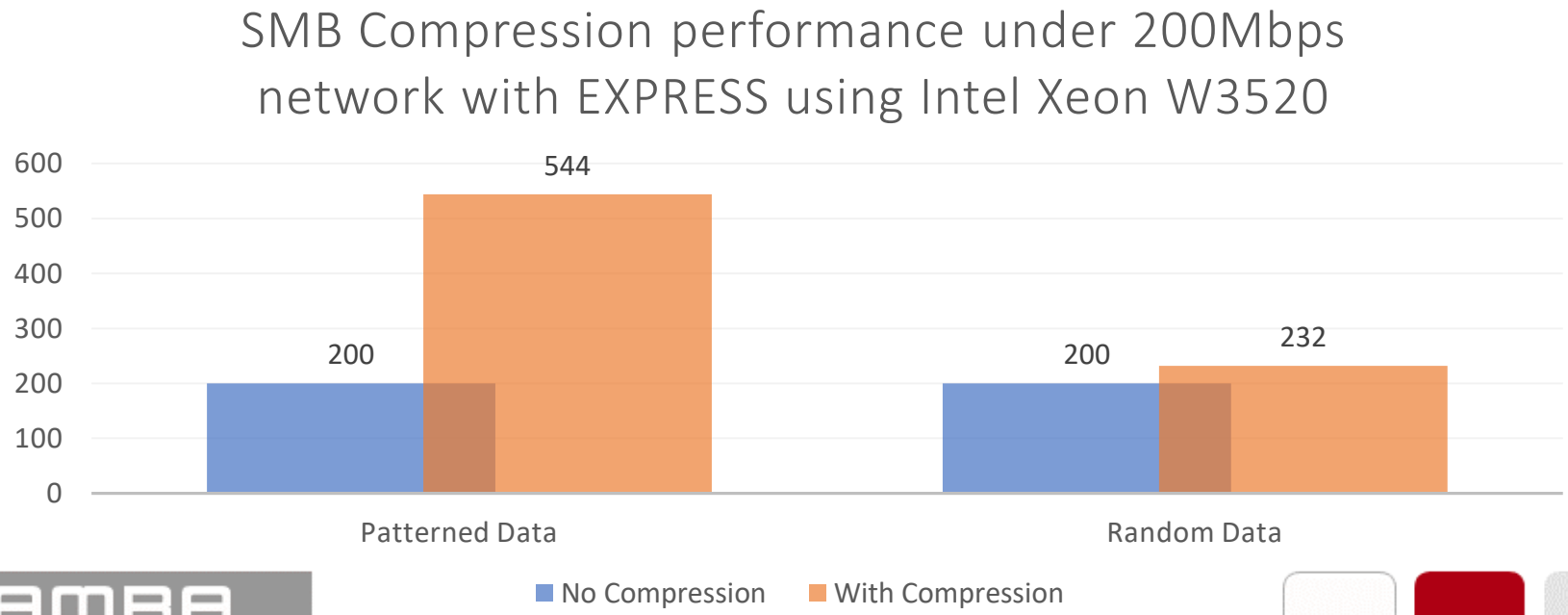
Compression commentary

- It's optional!
  - Doesn't compress if payload not smaller
  - Only compresses "large" "data-bearing" operations
  - Separate decision on both client and server, on each operation sent
- Compress *before* encrypt
  - Encrypted data compresses badly
  - Note, some encryptions also compress – implementation consideration
- Optional to compress SMB headers
  - Offset field may point into "middle" of payload
  - Windows compresses data-only at ~4KB+

Compression Performance



Compression Performance



Compression Use Cases

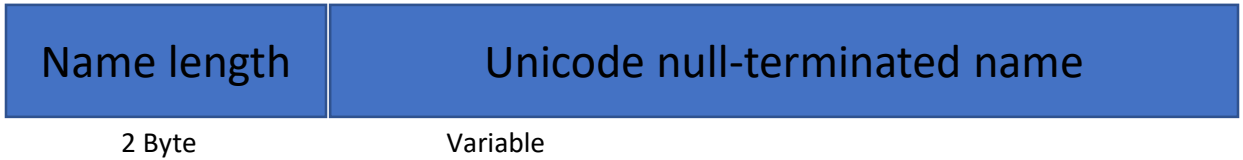
- Reads and Write
  - Not metadata and IOCTL/FSCTL, but possible
- Bulk data on long-haul
- Specialized local transfers
  - File copy, migration, etc
- Client opt-in
  - Used only in scenarios which might benefit

Compression future

- Alternative compression algorithms
  - Hyper-V / VHDX optimized?
    - RLL type algorithm for all-zero blocks is perhaps appealing
  - Still a per-operation and per-payload decision
- Interaction with encryption, transport, etc
  - Compression when encryption implements
    - Cf. not signing when using authenticated encryption
  - Compression over RDMA may have different goals
    - RDMA transport changes the benefit equation

Netname Negotiate Context

- Client provides target servername by appending negotiation context (ID = 0x0005)



- Provides servername
  - Advisory, available prior to session and treeconnect processing
- May be inspected by load balancers, connection managers, etc
  - Ignored by Server processing (perhaps surprisingly?)

Netname Negotiate Context

- SMB2_NETNAME_NEGOTIATE_CONTEXT_ID
  - MS-SMB2 Section 2.3.1.4 (request only)
  - 0x0005
- Included with SMB2_NEGOTIATE by default
  - MS-SMB2 section 3.2.4.2.2

Updates to the Microsoft SMB3 client

- FileNormalizedNameInformation
  - Normalized Name query added to protocol
- FileIdInformation
  - Omitted in 3.x [oops!] (3.3.5.20.1)
- Directory Caching Enhancements
  - Can now cache much larger directories ~ 500K entries.
  - Will attempt directory queries with 1 MB buffers to reduce round trips and improve performance
- Accelerated IO path for low latency access

Other MS-SMB2 Document Updates

- MS-XCA normative reference added (for compression)
- Numerous clarity and language tweaks
  - FSCTL input and output counts
  - Transform processing order, invalid protocol id's
    - New section reorg in April 30 update see 3.2.5.1.1/3.3.5.2.1 and subsections
  - Oplock/Lease break client processing
  - Tree connect and redirect
  - Durable reconnect v2 (3.3.5.9.12)
  - Compound processing (18H2 document)

SMB3 Protocol Futures

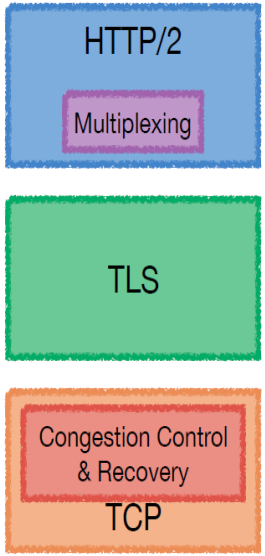
What's Coming?

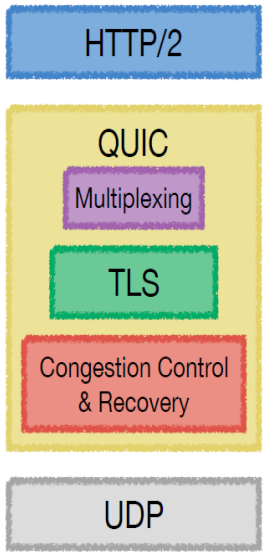
(SDC 2018 review / SDC 2019 preview)

- SMB over QUIC
- New transforms and signing
  - AES-GMAC signing
  - Signing and RDMA
- RDMA direct access to persistent storage

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 across
  - Better congestion control & loss recovery
  - UDP based library implementation
- IETF draft stage.





QUIC - Unknowns

- Still experimental
  - Evidence (Google) shows that it is firewall/NAT friendly – 93%
- Initial implementations are software only
  - Will it catch up with TCP offload ?
  - RDMA over QUIC ?
- Still in development
  - Very close to standardization

SMB Bindings for QUIC

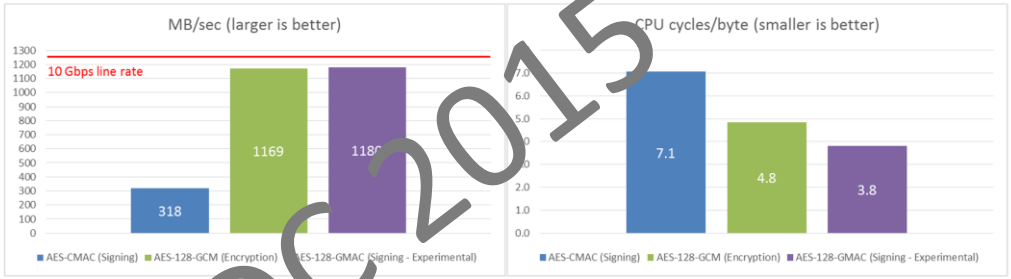
- QUIC connections can share same 4-tuple
  - Can multiplex using an ALPN identifier
  - Can share same port with HTTPS traffic
- Use QUIC as a single channel TCP replacement
  - SMB multichannel will use separate QUIC connections.
  - Not currently envisioning using QUIC streams
- Can QUIC be hooked up to Azure Files ?
  - No more port 445 blocking !

SMB3 Signing – Enabling AES-GMAC

- Switch from AES-CCM to AES-GCM cipher
  - AES-GCM based SMB3 encryption performs significantly better than AES-CCM based signing
  - Most modern processors have optimized instructions for AES-GCM computations
- SMB3.x (still) uses AES-CMAC for signing
- Can we use AES-GMAC to similarly improve signing ?
  - Definitely yes

AES-GMAC expected performance

7 – AES-GMAC file copy performance



- AES-GMAC results in significant performance improvements!
  - 46% reduction in Cycles/Byte compared to AES-CMAC
  - 21% reduction in Cycles/Byte compared to AES-GCM
- Prototype focused on functional correctness not performance
  - We identified several fairly easy improvements that could be made to further decrease CPU cycles/byte.

Negotiable SMB Signing with New Algorithm

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

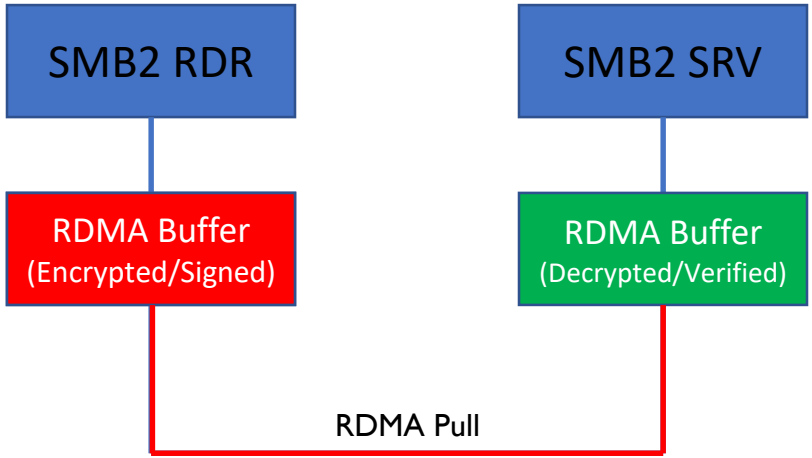
  - Supporting server will select 1 signing algorithm, if possible, and respond with:

  - More algorithms may be added over time

Better Signing and Encryption in RDMA

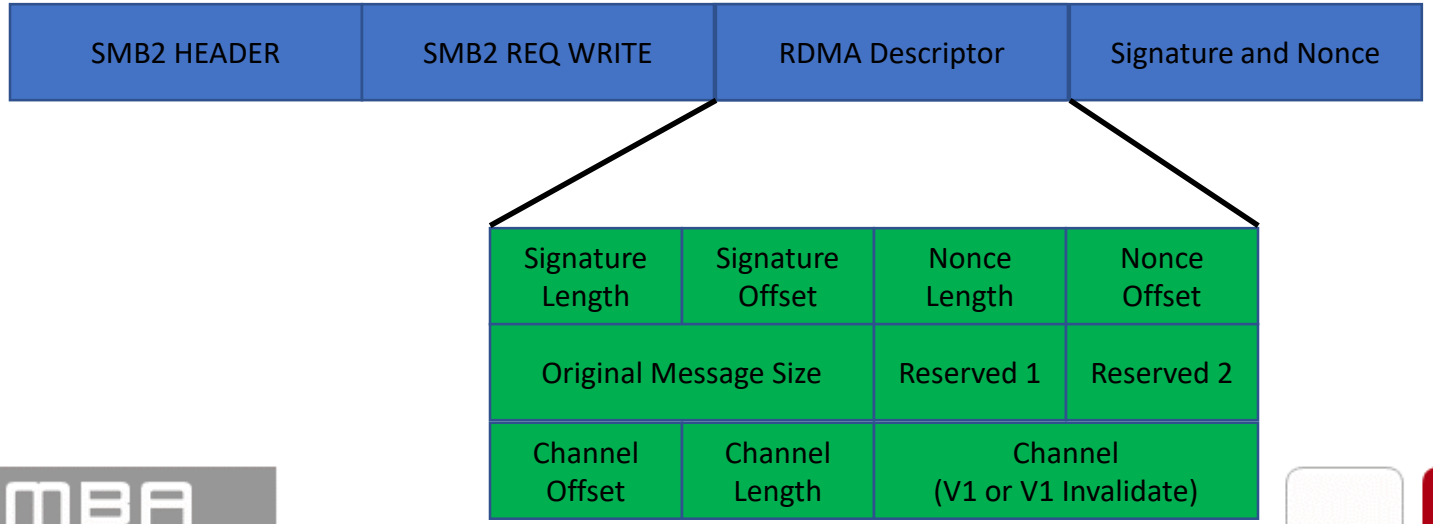
- Signing and Encryption over SMB RDMA.
  - Performance gain over current packet-based authenticated and/or encrypted traffic over SMB RDMA.
  - Supports AES128-GMAC for signing, AES-CCM and AES-GCM for encryption.

E.g. An SMB RDMA write:



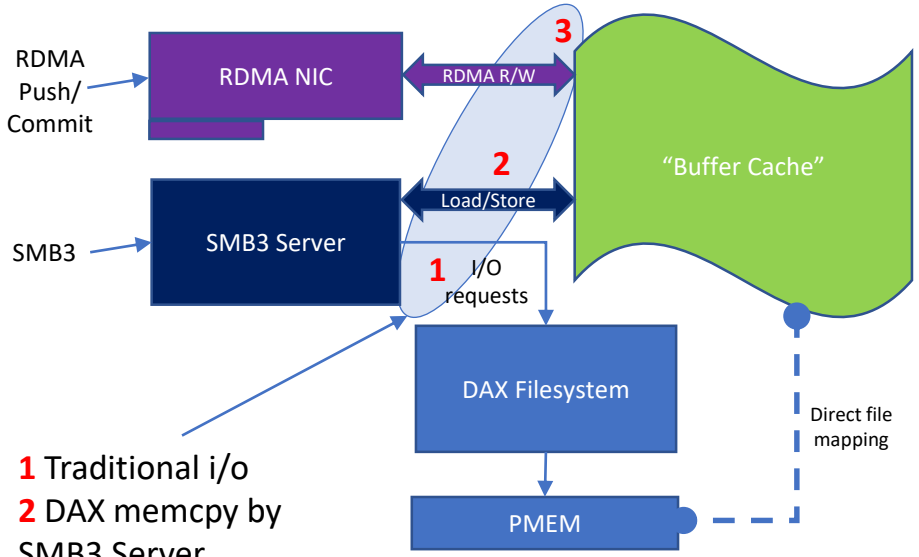
Better Signing and Encryption in RDMA

- How to transmit signature and nonce?
- Transform Descriptor as channel payload! (SMB2_CHANNEL_RDMA type 0x0003)
  - 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



RDMA Protocol Extensions

- Two extensions advancing (slowly) in IBTA (IB, RoCE)
- RDMA Flush is flush to durability
- Atomic Write places pointer-sized data after flush
  - Transactional, e.g. for log write pointer update
- IETF (iWARP) discussion also active

- Push Mode only needs RDMA Flush

SMB Protocol Extensions

- SMB3 protocol not extended
  - Only new FSCTLs
- Client requests “Push Mode” handle on DAX file
  - Just an RDMA memory handle, 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

Details

- More details on all the above to be available at SDC2019 in Santa Clara

Microsoft/Samba Collaboration

Ideas

- Microsoft remains interested in helping Samba co-develop:
  - Linux client
  - RDMA and RDMA Push Mode
  - SMB/QUIC interop
  - Azure test infra for Samba
  - Wireshark
  - And of course, Posix Extensions
- Let’s continue to discuss!

OBTW

- Death to SMB1 ☺