STORAGE DEVELOPER CONFERENCE

SDC 21

BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

SMB3 Landscape and Directions

Presented by Steven Tran & Wen Xin

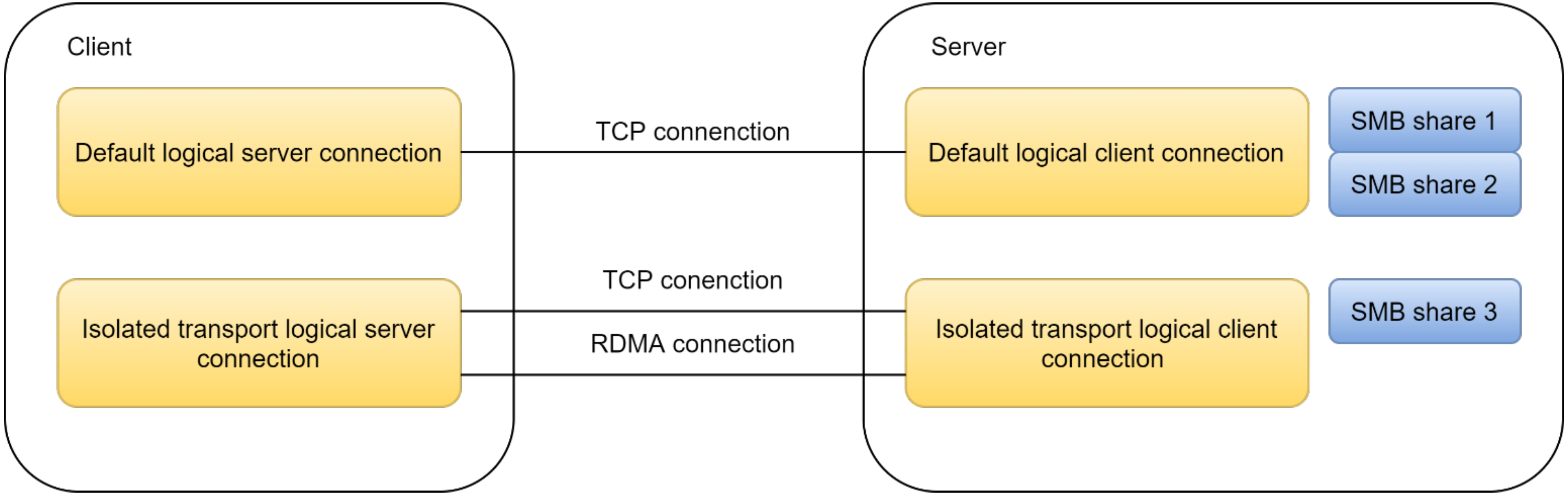
Agenda

- **Isolated Transport**
  - 2 Logical Connections To The Same Server
- **RDMA Transforms**
  - Encryption
  - Signing
  - Compression
- **New Management Options**
  - Encryption Cipher Orders [Ni]
  - Compression Opt-in [Fe]
  - Compressibility Sampling [Ni]
- **Improvements & Bug Fixes**
  - Remote Desktop Services (RDS) AutoDisconnectTimeout
  - Message ID On Cancel Request
  - Live-lock With Memory-Mapped Section Creation In SMB Redirector

Isolated Transports

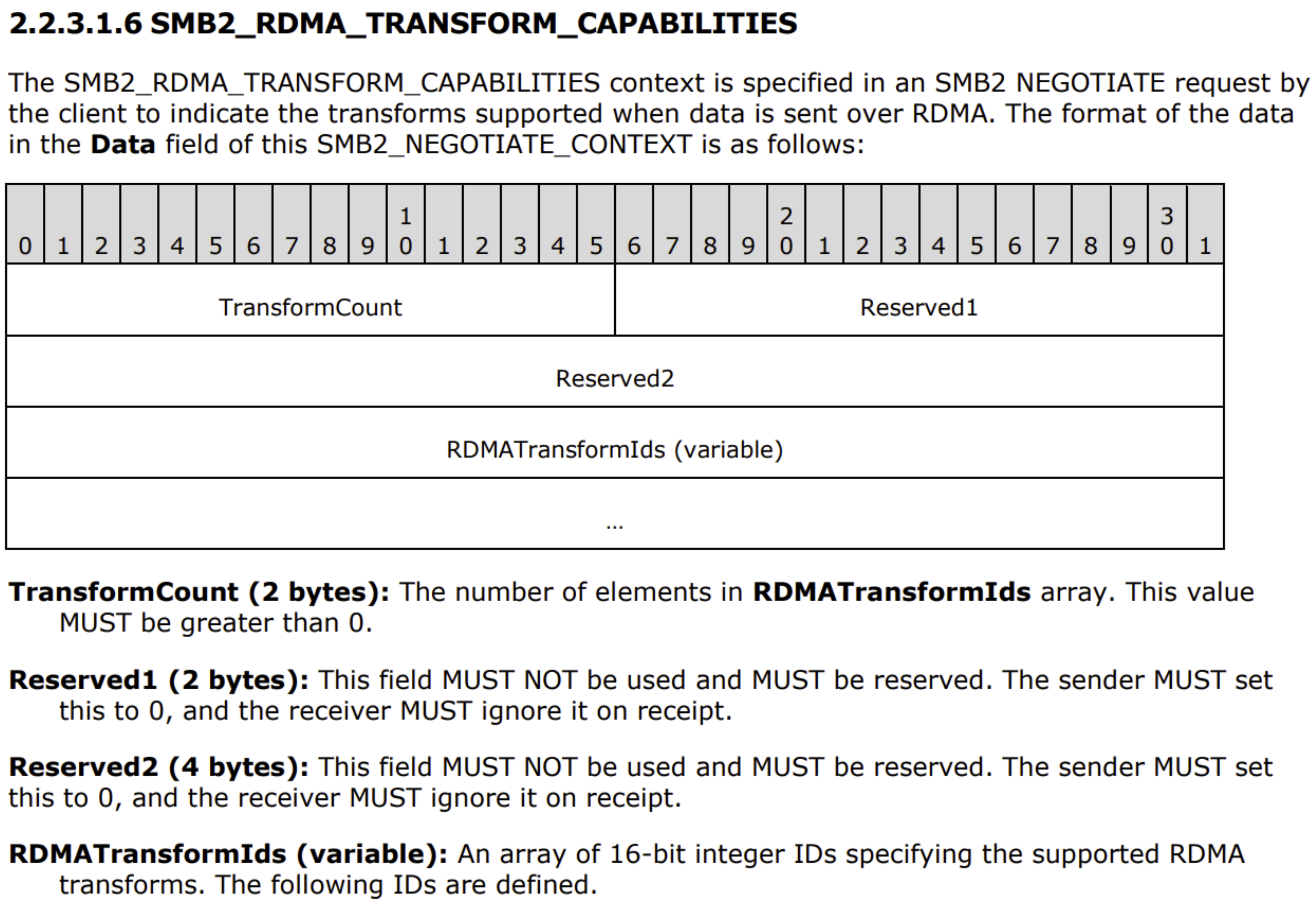
Purpose and Design

- Optionally create a secondary logical connection to the same server to isolate transports used (e.g. RDMA vs. no RDMA).


Client
Default logical server connection
Isolated transport logical server connection
TCP connenction
TCP conenction
RDMA connection
Server
Default logical client connection
Isolated transport logical client connection
SMB share 1
SMB share 2
SMB share 3

RDMA Transforms

Protocol Changes - Negotiation



2.2.3.1.6 SMB2_RDMA_TRANSFORM_CAPABILITIES

The SMB2_RDMA_TRANSFORM_CAPABILITIES context is specified in an SMB2 NEGOTIATE request by the client to indicate the transforms supported when data is sent over RDMA. The format of the data in the **Data** field of this SMB2_NEGOTIATE_CONTEXT is as follows:

<table>
<tr><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>1</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>2</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>3</th><th></th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>0</th><th>1</th></tr>
<tr><td colspan="16">TransformCount</td><td colspan="16">Reserved1</td></tr>
<tr><td colspan="32">Reserved2</td></tr>
<tr><td colspan="32">RDMATransformIds (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

TransformCount (2 bytes): The number of elements in **RDMATransformIds** array. This value MUST be greater than 0.

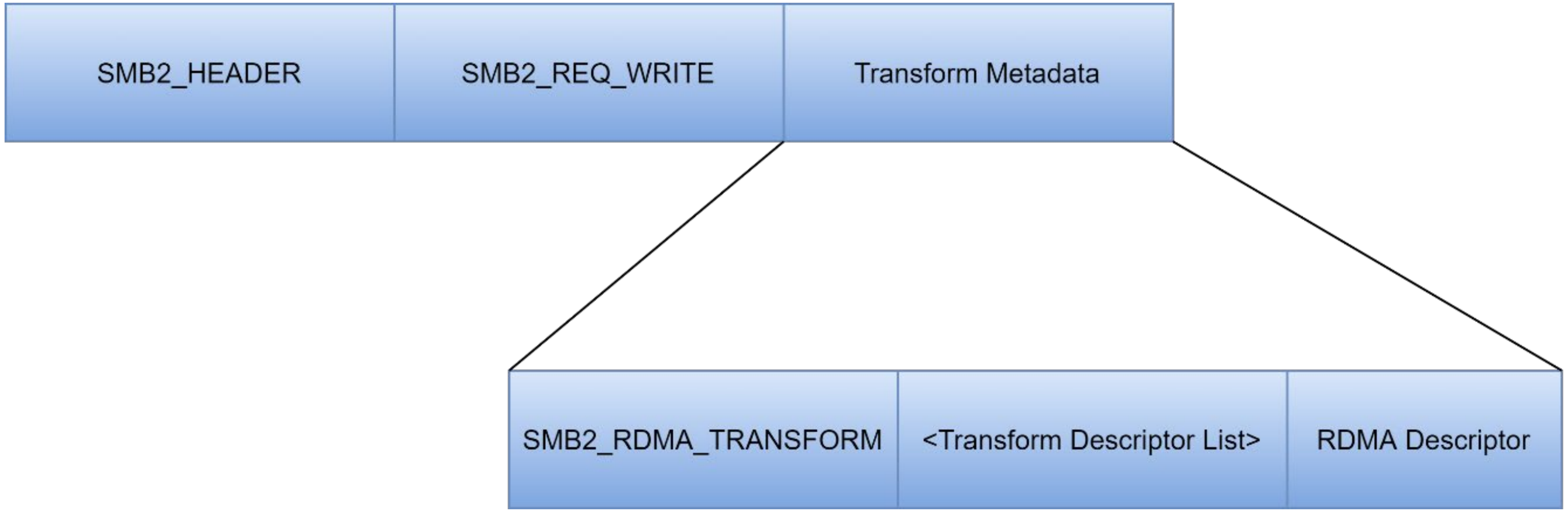
Reserved1 (2 bytes): This field MUST NOT be used and MUST be reserved. The sender MUST set this to 0, and the receiver MUST ignore it on receipt.

Reserved2 (4 bytes): This field MUST NOT be used and MUST be reserved. The sender MUST set this to 0, and the receiver MUST ignore it on receipt.

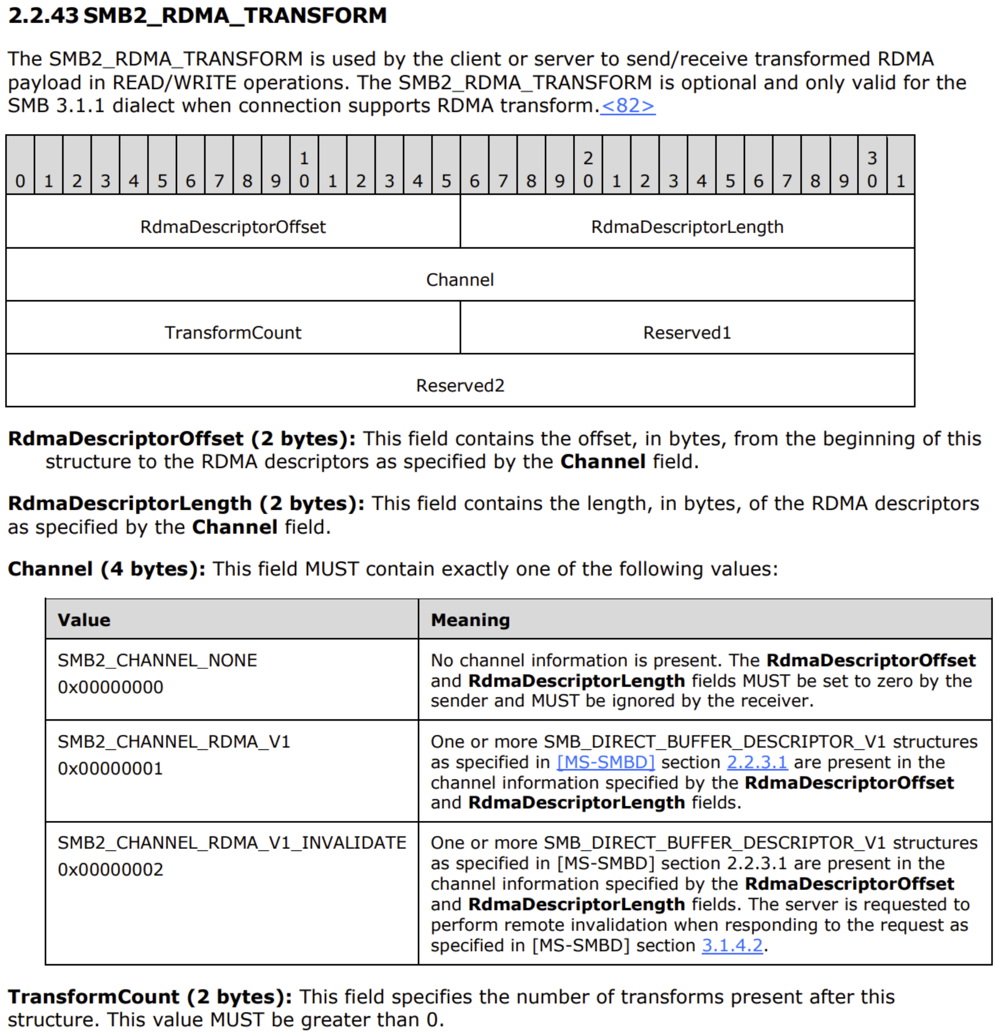
RDMATransformIds (variable): An array of 16-bit integer IDs specifying the supported RDMA transforms. The following IDs are defined.

Protocol Changes - Packet Diagram

- E.g. SMB2 Write over RDMA



Protocol Changes – Transform Metadata



2.2.43 SMB2_RDMA_TRANSFORM

The SMB2_RDMA_TRANSFORM is used by the client or server to send/receive transformed RDMA payload in READ/WRITE operations. The SMB2_RDMA_TRANSFORM is optional and only valid for the SMB 3.1.1 dialect when connection supports RDMA transform.<82>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3</td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="16">RdmaDescriptorOffset</td><td colspan="16">RdmaDescriptorLength</td></tr>
<tr><td colspan="32">Channel</td></tr>
<tr><td colspan="16">TransformCount</td><td colspan="16">Reserved1</td></tr>
<tr><td colspan="32">Reserved2</td></tr>
</table>

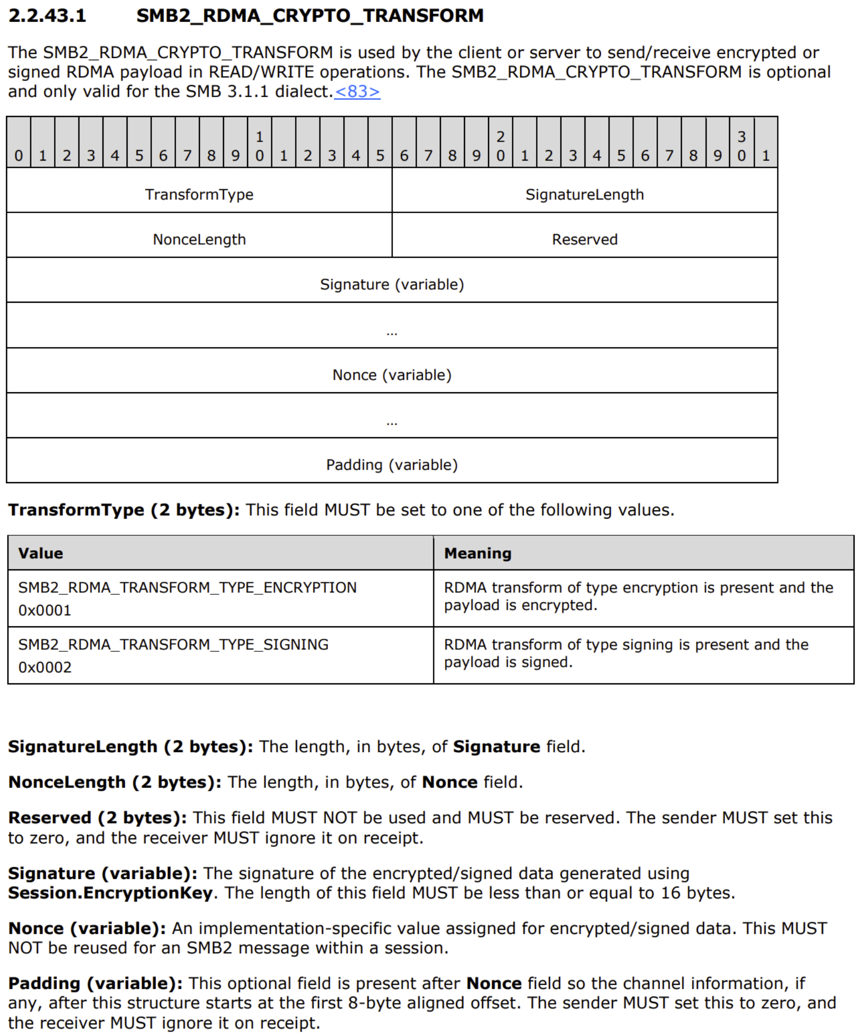
RdmaDescriptorOffset (2 bytes): This field contains the offset, in bytes, from the beginning of this structure to the RDMA descriptors as specified by the **Channel** field.

RdmaDescriptorLength (2 bytes): This field contains the length, in bytes, of the RDMA descriptors as specified by the **Channel** field.

Channel (4 bytes): This field MUST contain exactly one of the following values:

Value	Meaning
SMB2_CHANNEL_NONE 0x00000000	No channel information is present. The **RdmaDescriptorOffset** and **RdmaDescriptorLength** fields MUST be set to zero by the sender and MUST be ignored by the receiver.
SMB2_CHANNEL_RDMA_V1 0x00000001	One or more SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures as specified in [MS-SMBD] section 2.2.3.1 are present in the channel information specified by the **RdmaDescriptorOffset** and **RdmaDescriptorLength** fields.
SMB2_CHANNEL_RDMA_V1_INVALIDATE 0x00000002	One or more SMB_DIRECT_BUFFER_DESCRIPTOR_V1 structures as specified in [MS-SMBD] section 2.2.3.1 are present in the channel information specified by the **RdmaDescriptorOffset** and **RdmaDescriptorLength** fields. The server is requested to perform remote invalidation when responding to the request as specified in [MS-SMBD] section 3.1.4.2.

TransformCount (2 bytes): This field specifies the number of transforms present after this structure. This value MUST be greater than 0.



2.2.43.1 SMB2_RDMA_CRYPTO_TRANSFORM

The SMB2_RDMA_CRYPTO_TRANSFORM is used by the client or server to send/receive encrypted or signed RDMA payload in READ/WRITE operations. The SMB2_RDMA_CRYPTO_TRANSFORM is optional and only valid for the SMB 3.1.1 dialect.<83>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>3</td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="16">TransformType</td><td colspan="16">SignatureLength</td></tr>
<tr><td colspan="16">NonceLength</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="32">Signature (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Nonce (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Padding (variable)</td></tr>
</table>

TransformType (2 bytes): This field MUST be set to one of the following values.

Value	Meaning
SMB2_RDMA_TRANSFORM_TYPE_ENCRYPTION 0x0001	RDMA transform of type encryption is present and the payload is encrypted.
SMB2_RDMA_TRANSFORM_TYPE_SIGNING 0x0002	RDMA transform of type signing is present and the payload is signed.

SignatureLength (2 bytes): The length, in bytes, of **Signature** field.

NonceLength (2 bytes): The length, in bytes, of **Nonce** field.

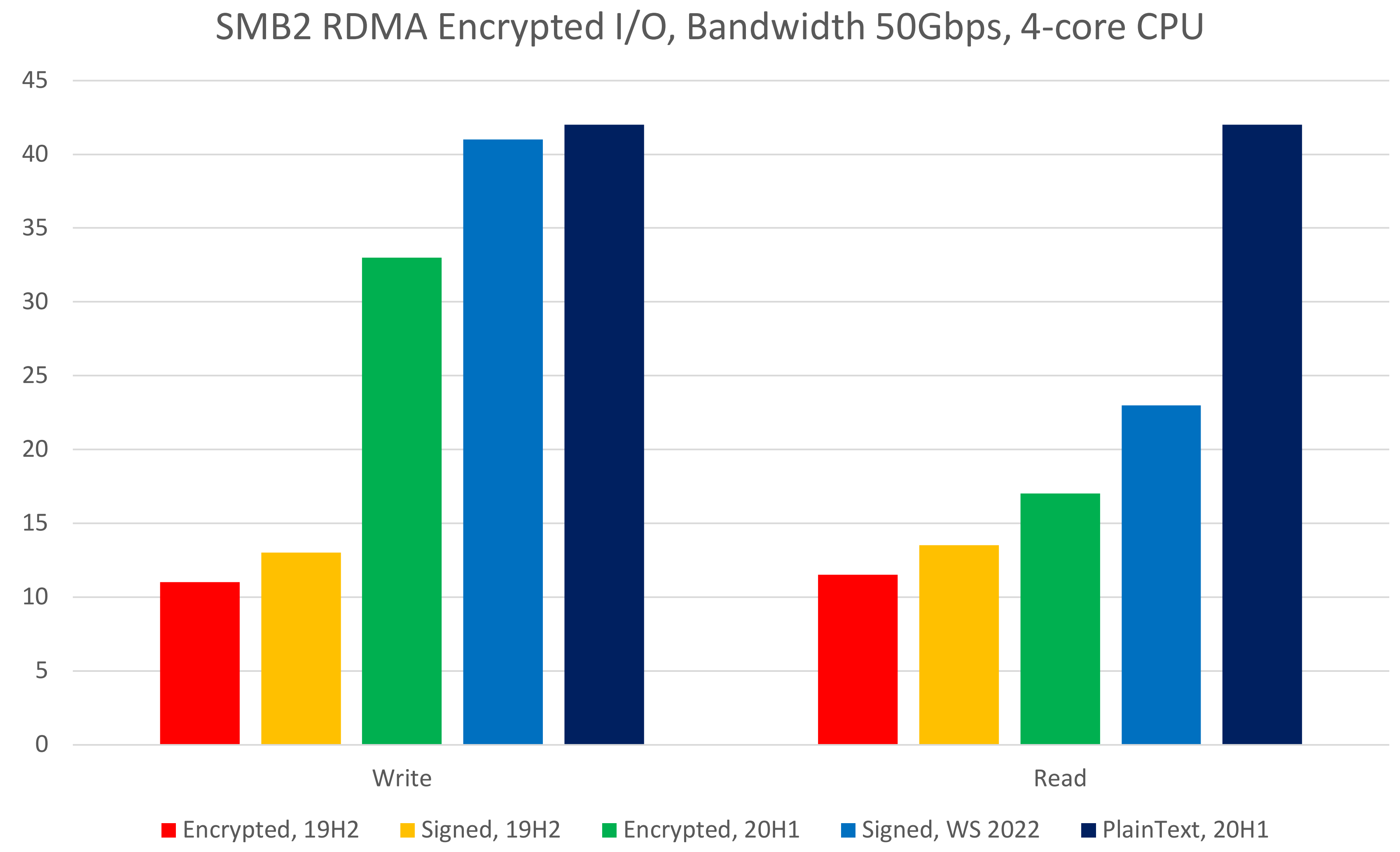
Reserved (2 bytes): This field MUST NOT be used and MUST be reserved. The sender MUST set this to zero, and the receiver MUST ignore it on receipt.

Signature (variable): The signature of the encrypted/signed data generated using **Session.EncryptionKey**. The length of this field MUST be less than or equal to 16 bytes.

Nonce (variable): An implementation-specific value assigned for encrypted/signed data. This MUST NOT be reused for an SMB2 message within a session.

Padding (variable): This optional field is present after **Nonce** field so the channel information, if any, after this structure starts at the first 8-byte aligned offset. The sender MUST set this to zero, and the receiver MUST ignore it on receipt.

Performance



New Management Options

Encryption Ciphers

- Effects & Expectation Of New Settings
  - `Set-Smb(Client|Server)Configuration –EncryptionCiphers <ciphers>`
  - `Get-Smb(Client|Server)Configuration`
- Current Supported List Of Encryption Ciphers
  - AES_128_GCM
  - AES_256_GCM
  - AES_128_CCM
  - AES_256_CCM
- Race Conditions Considered
  - Negotiate Scenario
  - Setting Ciphers With Multiple Client / Server Instances

Compression Opt-In

- SMB Server
  - `New-SmbShare –CompressData <bool>`
  - `Set-SmbServerConfiguration –DisableCompression <bool>`

- SMB Client
  - `New-SmbMapping –CompressData <bool>`
  - `net use /COMPRESS <bool>`

Compressibility Sampling

- Default behavior changed from always sample 512MB to always compress. Feature needs to be opted-in from registry key.

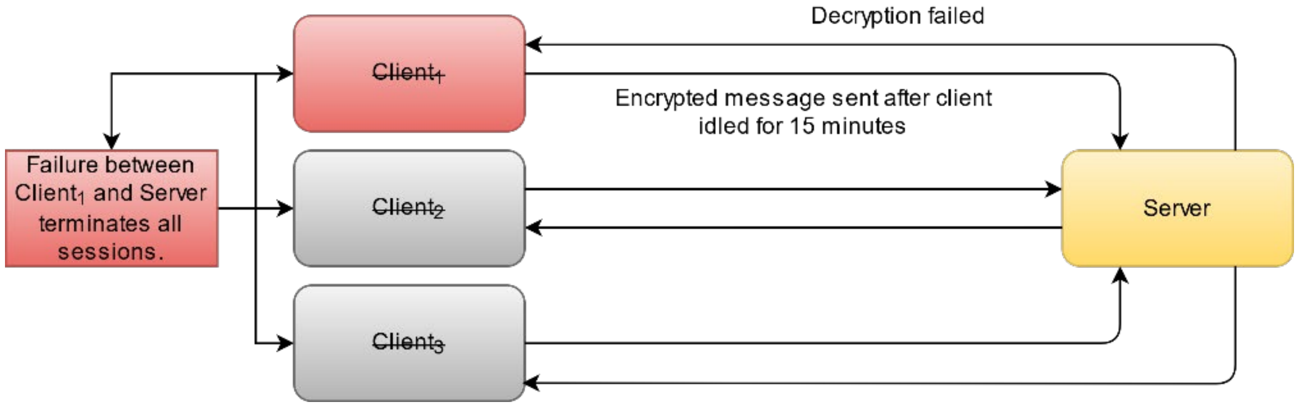
Improvements & Bug Fixes

RDS AutoDisconnectTimeout

- Background, Issue Context, & Brief Summary
- Solutions Discussed
- Proposed Solution

RDS AutoDisconnectTimeout

- Background, Issue Context, & Brief Summary

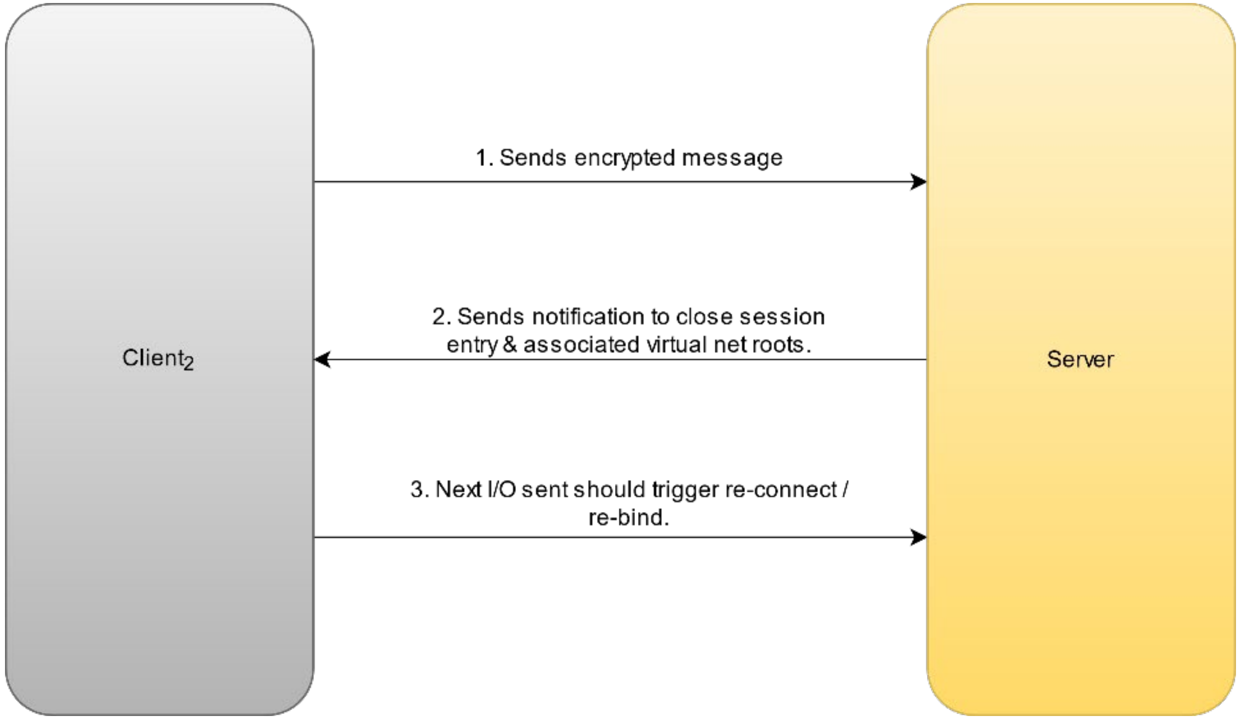

Decryption failed
~~Client$_1$~~
Encrypted message sent after client idled for 15 minutes
Failure between Client$_1$ and Server terminates all sessions.
~~Client$_2$~~
Server
~~Client$_3$~~

RDS AutoDisconnectTimeout

- Solutions Discussed
  - Retaining encryption keys after session destruction
  - No idle session disconnects when connection shared between multiple sessions
  - Including MessageID within the SMB_TRANSFORM_HEADER
  - Notification from server to client like oplock break notification to disconnect the server entry on the client

RDS AutoDisconnectTimeout

- Proposed Improvement



1. Sends encrypted message

2. Sends notification to close session entry & associated virtual net roots.

3. Next I/O sent should trigger re-connect / re-bind.

Message ID on cancel request

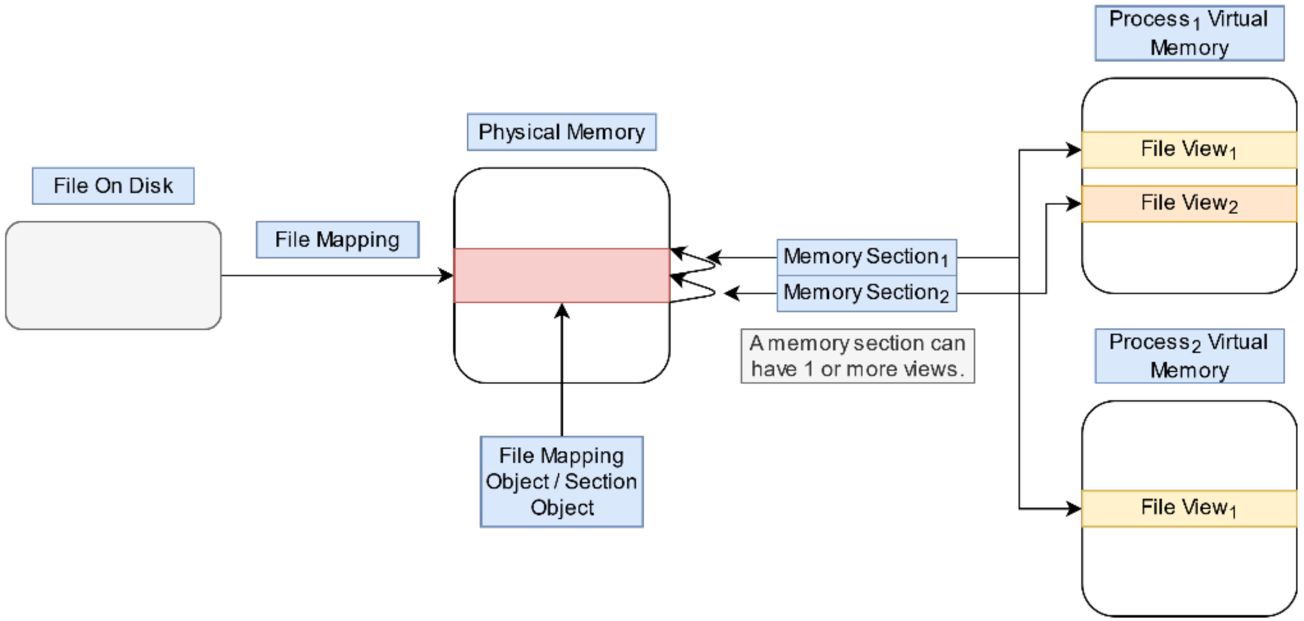
- When using AES-GMAC we require a nonce to calculate signature, and message ID is included in the nonce. When sending a cancel request to an async packet, we used to send 0 as the message ID, which violated the nonce's requirement that it should never repeat itself.

Live-lock With Memory-Mapped Section Creation In SMB Redirector

- Issue Context & Brief Summary
- Background Of Memory-Mapped Files
- Detailed Overview Of Issue
- Solution

Live-lock

File Mapping



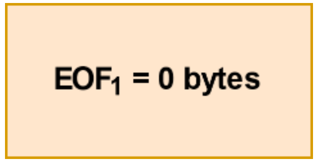
File On Disk
File Mapping
Physical Memory
File Mapping Object / Section Object
Memory Section$_1$
Memory Section$_2$
A memory section can have 1 or more views.
Process$_1$ Virtual Memory
File View$_1$
File View$_2$
Process$_2$ Virtual Memory
File View$_1$

Useful Concepts

File Size: Known as EOF where the size of data is in bytes.	**Valid Data Length**: VDL is less than or equal to EOF; this is the data in the file that's written to	**Allocation Size**: Size allocated for files on disk; even multiple of cluster size.

Live-lock

- Overview of Issue
- Client$_1$ opens file with shared read & write permissions
  - Obtains RWH oplock
  - Assume empty file to work with



$EOF_1 = 0$ bytes

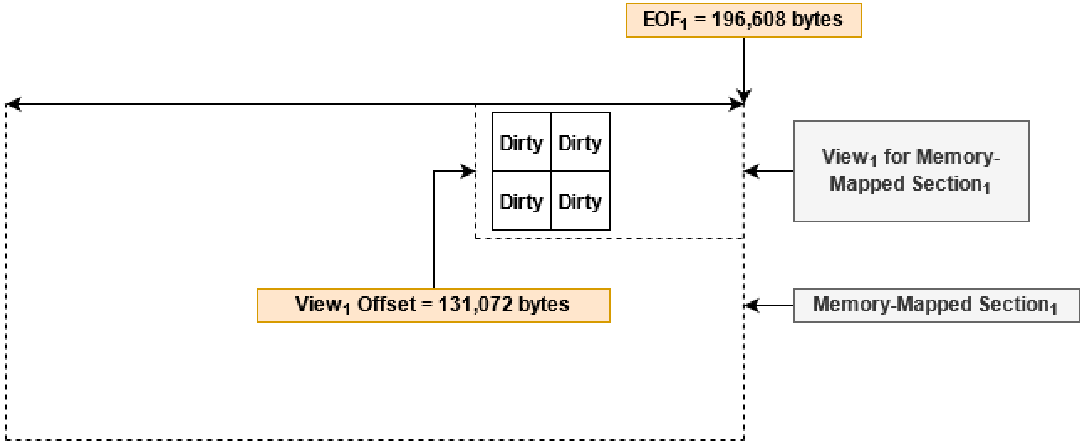
Live-lock

- Overview of Issue
- Client$_1$ sets end of file (EOF$_1$) to e.g., 196,608 bytes.

EOF$_1$ = 196,608 bytes

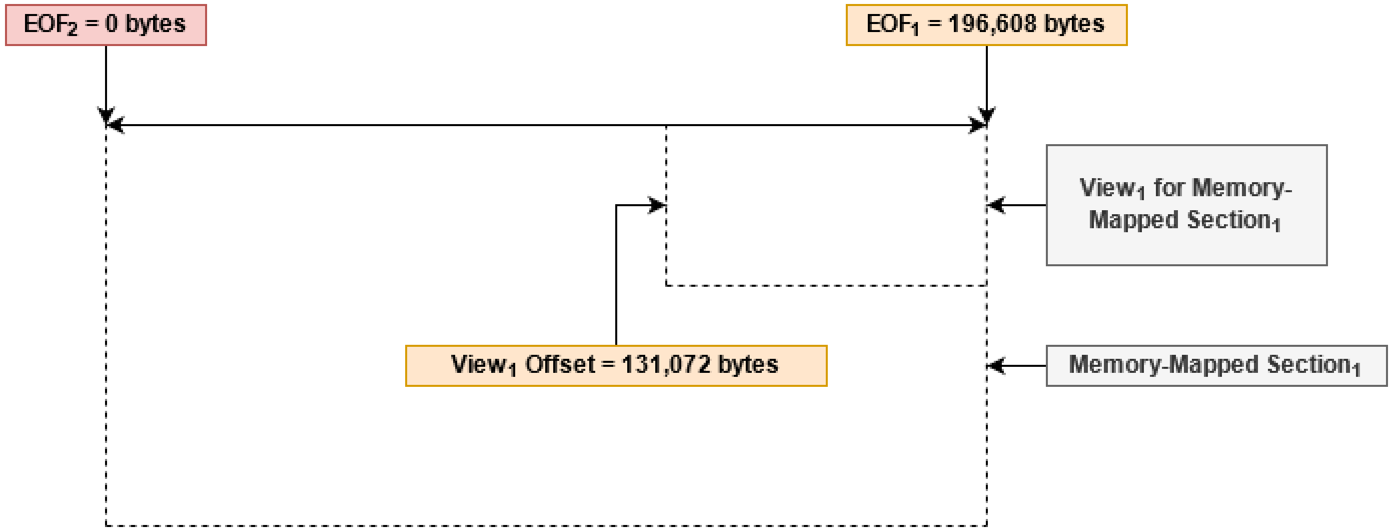
Live-lock

- Overview of Issue
- Client$_1$ creates the following:
  - Memory-mapped section$_1$
  - View$_1$ with few dirty pages e.g., 16,384 bytes


EOF$_1$ = 196,608 bytes
Dirty
Dirty
Dirty
Dirty
View$_1$ for Memory-Mapped Section$_1$
View$_1$ Offset = 131,072 bytes
Memory-Mapped Section$_1$

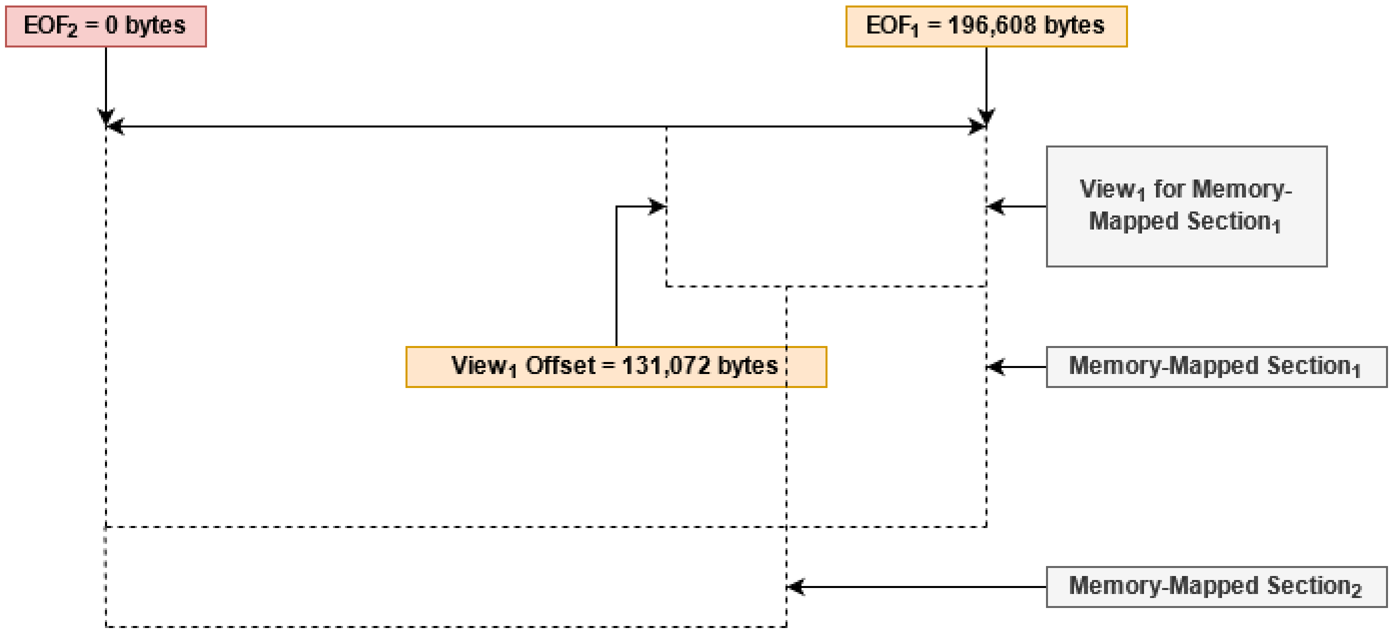
Live-lock

- Overview of Issue
- Client$_2$ does the following:
  - Open same file with shared read and write permissions.
  - Set the EOF = 0 (call this EOF$_2$)
- Oplock for Client$_1$ eventually downgrades from RWH → None
- Dirty pages purged & flushed to down-level file system
- Data corruption occurs


EOF$_2$ = 0 bytes
EOF$_1$ = 196,608 bytes
View$_1$ for Memory-Mapped Section$_1$
View$_1$ Offset = 131,072 bytes
Memory-Mapped Section$_1$

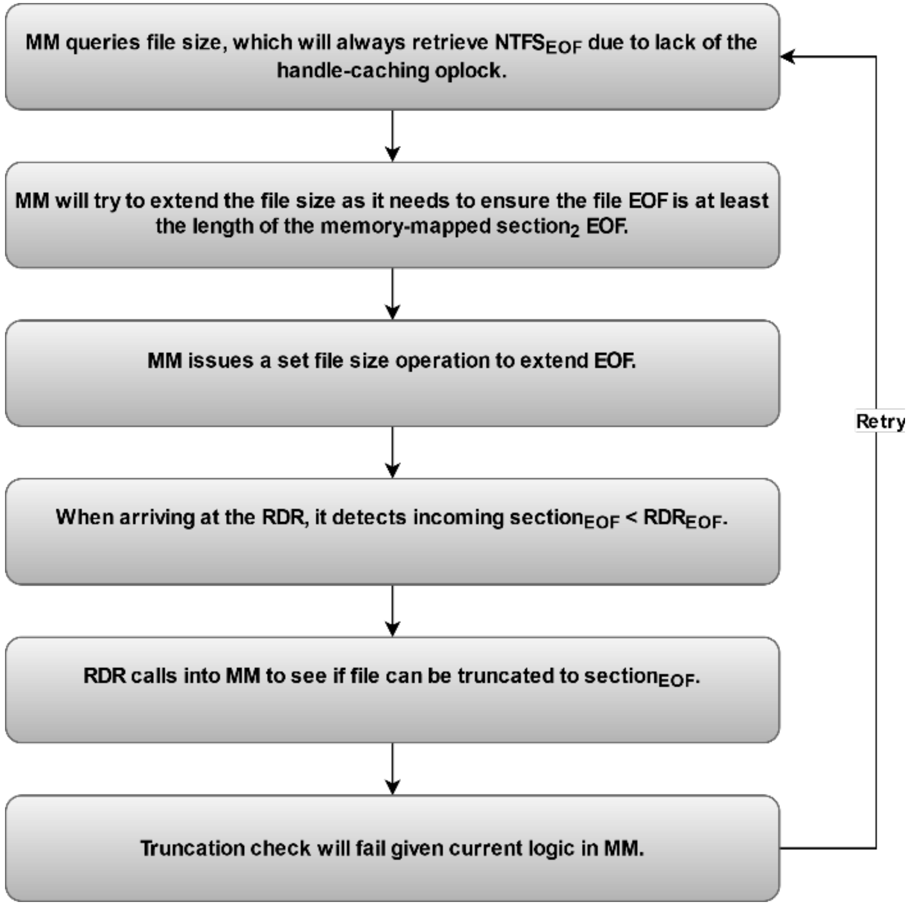
Live-lock

- Overview of Issue
- Client$_1$ creates memory-mapped section$_2$
  - Triggers live-lock


EOF$_2$ = 0 bytes
EOF$_1$ = 196,608 bytes
View$_1$ for Memory-Mapped Section$_1$
View$_1$ Offset = 131,072 bytes
Memory-Mapped Section$_1$
Memory-Mapped Section$_2$

Live-lock

- Overview of Issue
- $NTFS_{EOF}$ < Memory-Mapped Section $\#2_{EOF}$ < RDR_{EOF}
  - Example earlier shows e.g.
    - 0 < 147,456 < 196,608
  - When the above condition is true, the endless loop occurs:



MM queries file size, which will always retrieve $NTFS_{EOF}$ due to lack of the handle-caching oplock.

MM will try to extend the file size as it needs to ensure the file EOF is at least the length of the memory-mapped $section_2$ EOF.

MM issues a set file size operation to extend EOF.

When arriving at the RDR, it detects incoming $section_{EOF}$ < RDR_{EOF}.

RDR calls into MM to see if file can be truncated to $section_{EOF}$.

Truncation check will fail given current logic in MM.

Live-lock

- Solution
  - Patching RDR_{EOF} to the incoming section's EOF size
  - Preventing data loss in certain scenarios

Please take a moment to rate this session.

Your feedback is important to us.