STORAGE DEVELOPER CONFERENCE
SDC 22
Fremont, CA
September 12-15, 2022
BY Developers FOR Developers

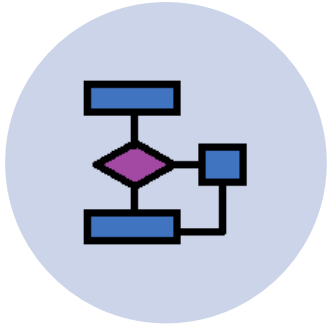
A SNIA Event

SMB3 Landscape & Directions

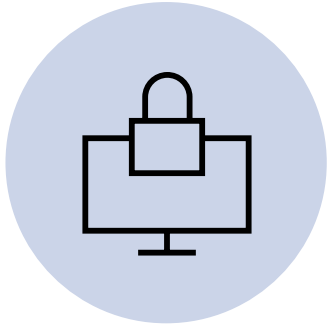
Presented by Genghis Karimov & Meng Li
Credits to Steven Tran

Agenda

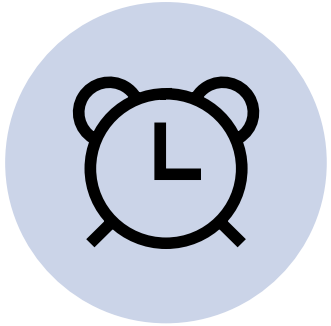
Compression



Compression sampling



Authentication Rate Limiter



Notifications

SMB3: overview

- SMB3 is 10 years old in 2022
  - Scalable, continuously available file sharing
  - Designed for enterprise and cloud-infrastructure workloads
- Additions/improvements in the past half decade:
  - Signing, encryption
  - Compression
  - New transports: RDMA (SMBD/SMB 3.0), QUIC (detailed in an SDC 2021 talk)
  - See our prior SDC talks
- Notable improvements in the past few years:
  - Compression **sampling** (no changes to the protocol)
  - Authentication rate-limiting in Windows SMB Server (no changes to the protocol)
  - Notifications (extends the protocol)
- Future work
  - Consolidation, optimization
  - Deprecation (SMB1, other legacy parts of the protocol/stack)
  - Hardware offloads (very nascent)

SMB Compression

Transparent and flexible data compression above the transport layer

SMB Compression: agenda

- Overview
- Compression algorithms
- Cost evaluation/heuristics: is it always worth it?
- Compression sampling
- Windows SMB client and server policies/knobs management
- Performance counters: is compression in effect?

SMB Compression: overview

- Offers transparent data compression
  - Seamlessly integrates with signing, encryption
- Variety of algorithms offered and selectable
- Part of protocol v3.1.1+
- Windows SMB client offers compression **sampling**
  - State machine to heuristically determine if data is compressible or not
- Compression of RDMA (SMB Direct) transport not currently supported
- Plenty of room for future expansion
- First covered in SDC 2018 presentation by Mathew George and Wen Xin
  - 33:20 @ https://www.youtube.com/watch?v=JLFvLaEy_8c

SMB Compression: compression algorithms

- Compression algorithms as defined by [MS-XCA]: XPRESS (aka LZ77), XPRESS Huffman (LZ77+Huffman), and LZNT1
  - "This algorithm efficiently compresses data that contain repeated byte sequences. It is **not designed to compress image, audio, or video data**. Between the trade-offs of compressed size and CPU cost, it heavily **emphasizes low CPU cost**."
- … **plus** RLE prefix/suffix scanner, defined in *§3.1.4.4.1 Algorithms for Scanning Data Patterns V1* (page 153 of [MS-SMB2] revision 66.0)
  - Trivial octet repetition scanner, applied before running the compressor
  - Not strictly required

SMB Compression: a few gotchas/caveats

- Protocol allows for individual message to be transferred uncompressed, even if compression has been negotiated
- For READ operations, client controls whether server will attempt compression
  - See §2.2.19 *SMB2 READ Request* (pg98 of MS-SMB2 rev66)
  - SMB2_READFLAG_REQUEST_COMPRESSED: *The server is requested to compress the read response when responding to the request*
  - Does not strictly mandate that the server will compress, only that it will attempt
  - Lack of flag mandates that server does not attempt compression

SMB Compression: sampling

- Windows SMB client implements a sampling state machine to evaluate whether a file transfer is compressible or not
- Not a protocol feature
- State machine:
  - “Evaluation” (initial) state: SMB client attempts compression for a bounded volume of data (some # of bytes); if the data compresses to a certain threshold (another # of bytes), data is deemed compressible; otherwise not-compressible
  - “Compressible” state: data compression is always attempted
  - “Not-compressible” state: data compression not attempted
  - “Compressible” and “not-compressible” states are terminal, and persist until the file is closed
  - Evaluation/sampling applies for both reads and writes
- State machine runs on client

SMB Compression: sampling (continued), management

- Policies/knobs:
  - Powershell: Set-SmbClientConfiguration
    - -EnableCompressibilitySampling <bool>
    - -CompressibilitySamplingSize <uint64>, e.g. -CompressibilitySamplingSize 100MB
    - -CompressibleThreshold <uint64>, e.g. -CompressibleThreshold 50MB
- See official article: https://docs.microsoft.com/en-us/windows-server/storage/file-server/smb-compression
- Sampling subject to change/be extended in the future

SMB Compression: cost evaluation/heuristics

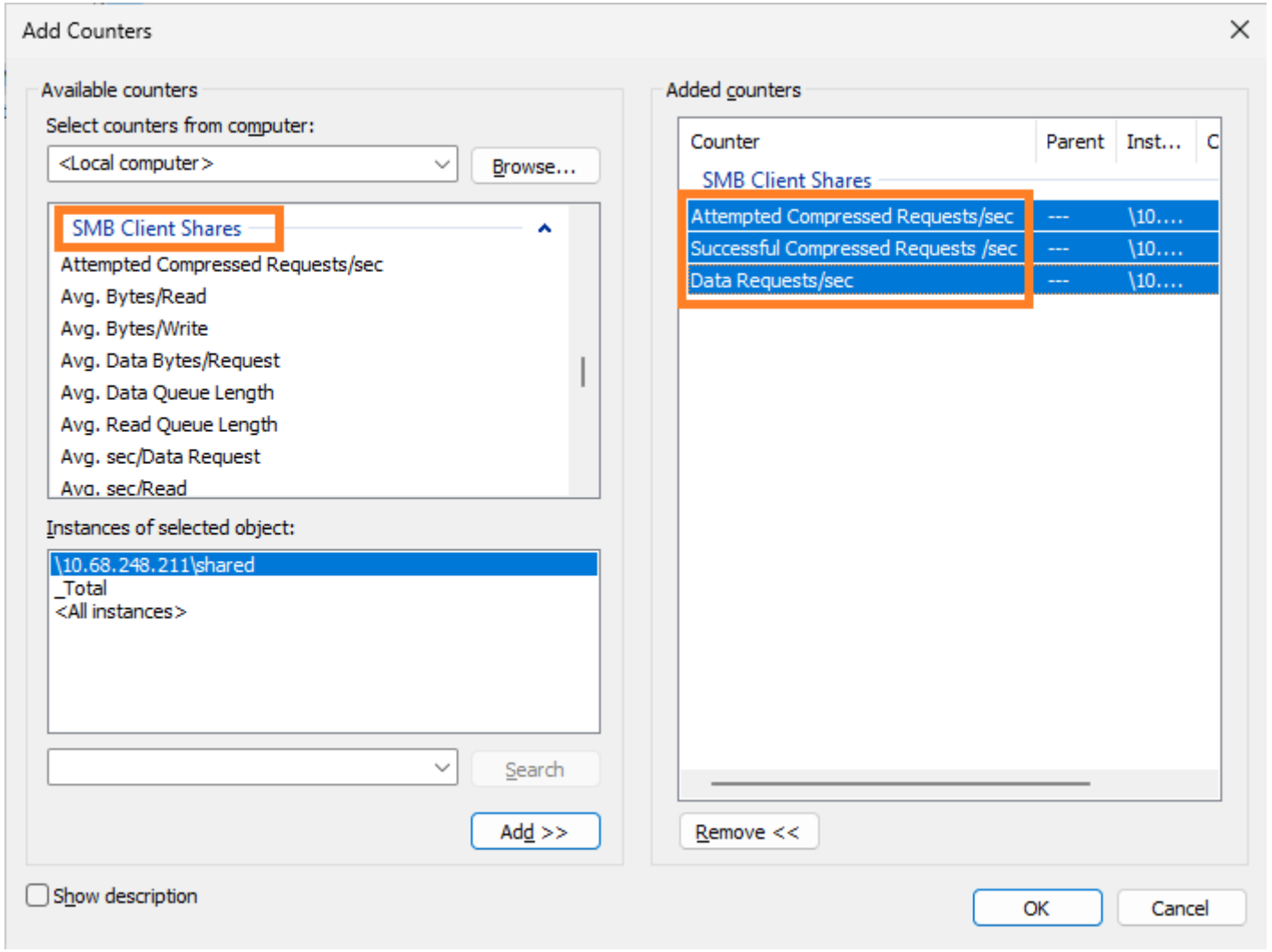
- Recall: protocol allows for an individual message to be transferred uncompressed, even if compression has been negotiated
- Compression is costly and opportunistic:
  - Memory allocations in I/O path (double-buffering)
  - (variable) CPU costs
  - … no guarantee of reward
- Employ heuristics to decide whether compression is worth it
  - E.g. small messages NOT compressed: overhead dominates cost
- Not mandated by the protocol
  - Up to the implementation to determine what heuristics (if any) to use
  - Must remain interoperable
- Windows SMB compression sampling is a macro cost evaluator

SMB Compression: is it working? (performance counters)

- A few SMB **client** performance counters to help understand how well SMB Compression is performing:
  - Microsoft.SMB2.Client.Share.CompressedRequestsPerSec
    - # of attempts to compress
    - compare to TotalMetadataRequests+TotalReadWriteRequests
  - Microsoft.SMB2.Client.Share.CompressedResponsesPerSec
  - Microsoft.SMB2.Client.Share.CompressedBytesSentPerSec
    - compare to TotalReadBytes+TotalWriteBytes
  - Microsoft.SMB2.Client.Share.SuccessfulCompressedRequestsPerSec
    - must be less than or equal to CompressedRequestsPerSec

SMB Compression: performance counters (example)

- Use the built-in Windows Performance Monitor (perfmon.msc)
  - Mount a share
  - Add counters
  - Run workload



SMB Compression: performance counters (example)

- Writing random data continuously

SMB Client Shares	_Total
Attempted Compressed Requests/sec	25.961
Data Requests/sec	14.977
Successful Compressed Requests /sec	0.000

- Writing all zeroes continuously

SMB Client Shares	_Total
Attempted Compressed Requests/sec	107.909
Data Requests/sec	47.960
Successful Compressed Requests /sec	23.980

SMB Compression: future work

- Protocol:
  - Open to introduction of other compression algorithms
  - Chaining transforms/filters
- Windows SMB client/server-specific:
  - Compression with RDMA/SMBD
  - Optimization, pipelining (refactoring/address technical debt)
  - More sophisticated cost evaluator/heuristics
  - Automated performance analysis
  - Fuzzing
  - Hardware acceleration

SMB Compression: various resources

- SMB Compression documentation:
  - https://docs.microsoft.com/en-us/windows-server/storage/file-server/smb-compression
- [MS-XCA] Microsoft Xpress Compression Algorithm
  - https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-xca/001e03e3-d1c2-4d51-9d39-e845d9b05959

SMB Authentication Rate Limiter

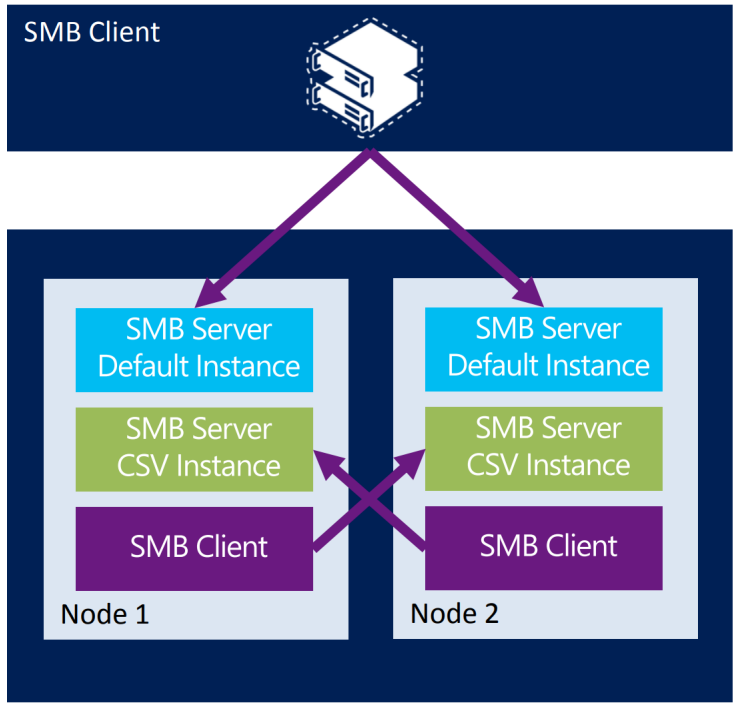
Defense against NTLM password brute-force attacks with planned future iterations to address parallel password-spray attacks

SMB Authentication Rate Limiter

Overview

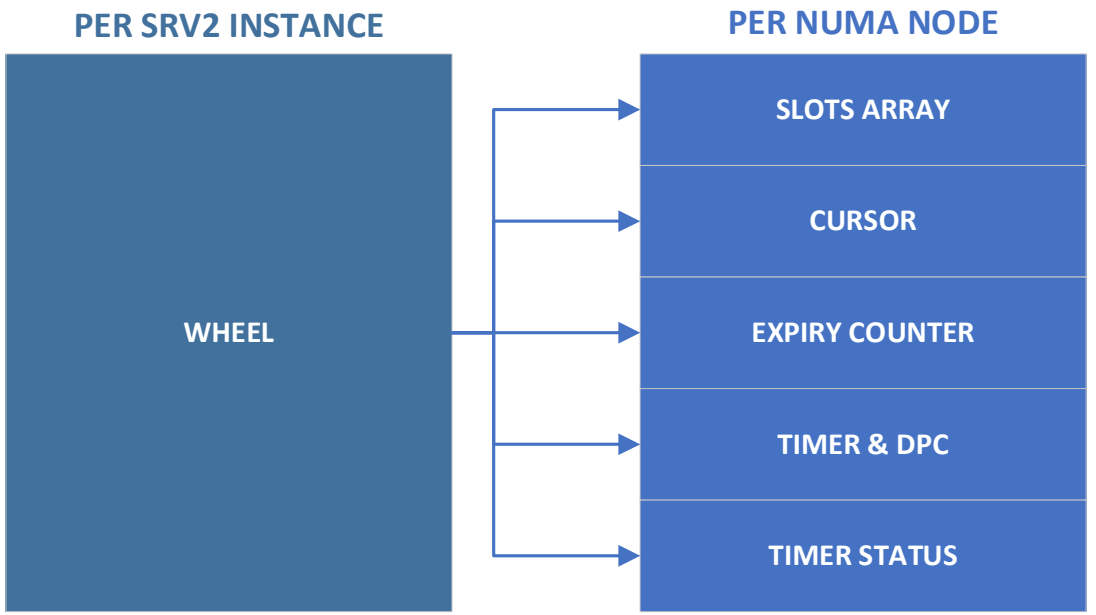
- Recap:
  - What is per SRV2 instance?

Architectural Design



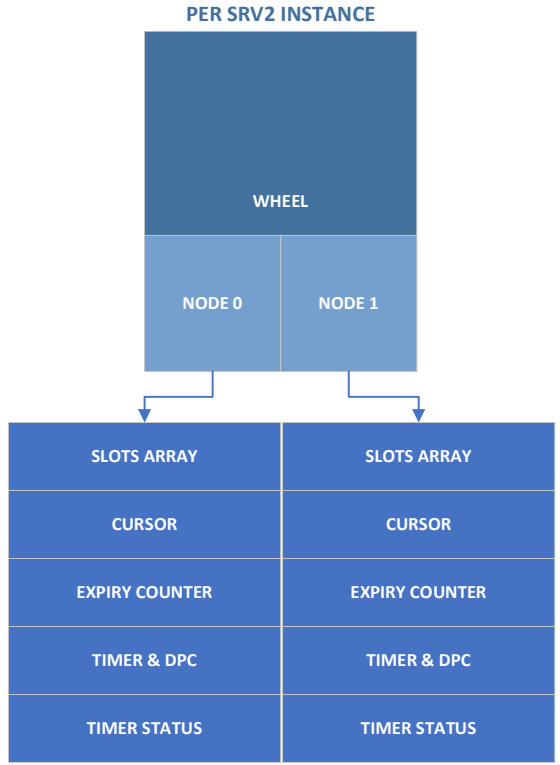
SMB Authentication Rate Limiter

- Structural design of the rate limiter feature
  - Per SRV2 instance
  - Per NUMA node



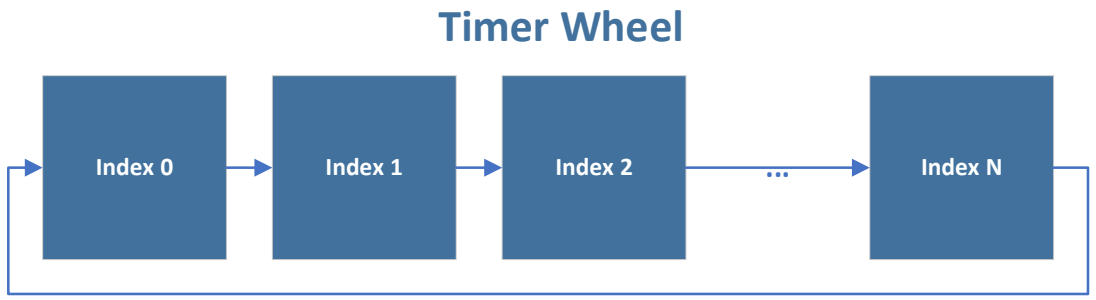
SMB Authentication Rate Limiter

- Structure example of 2 NUMA nodes
- Every slots array is an array of singly-linked lists



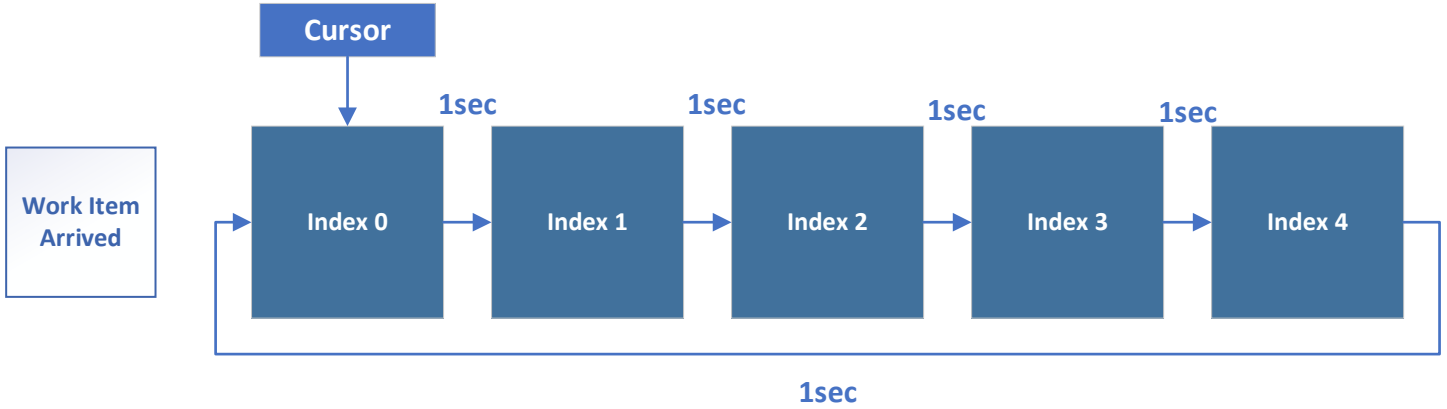
SMB Authentication Rate Limiter

- Timer wheel:
  - Time between each slot is 100ms
  - Current implementation is 100 slots total (N = 99 based on diagram)
  - 100 slots * 100ms = 10-second window
  - Timer's cursor (tracker) ticks, or expires, every 100ms
  - Each expiry will process all the delayed work items in the linked list for each index



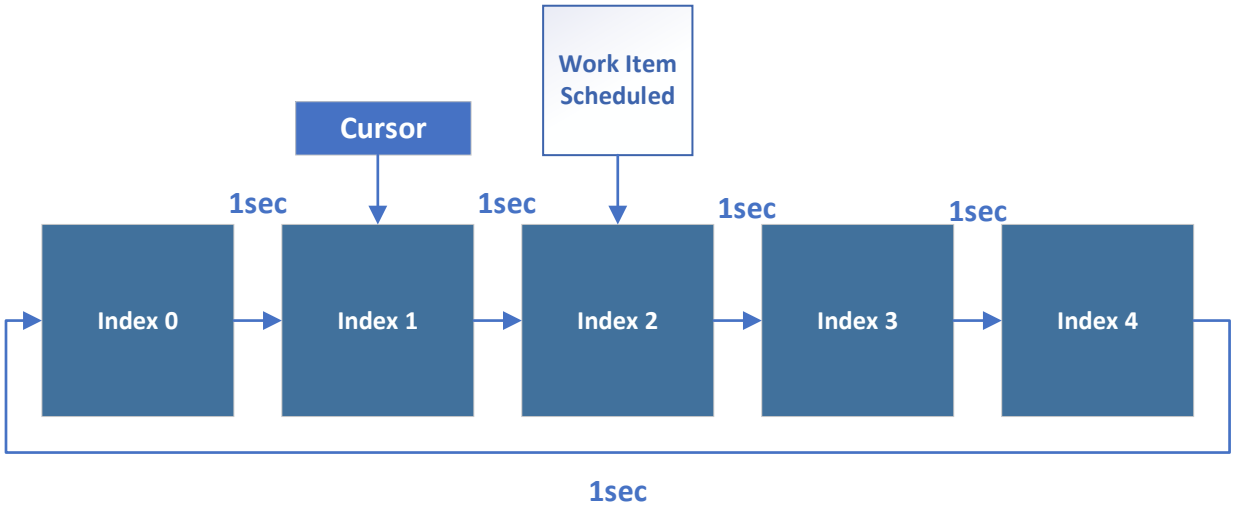
SMB Authentication Rate Limiter

- Example flow (simplified):
  - Work item has arrived, and cursor is pointing at the start of the timer wheel
  - Assume each period from one slot to another is 1 second
  - Assume default to delay each incoming work item as 2 seconds
  - Schedule the arrived work item 2 nodes from starting cursor



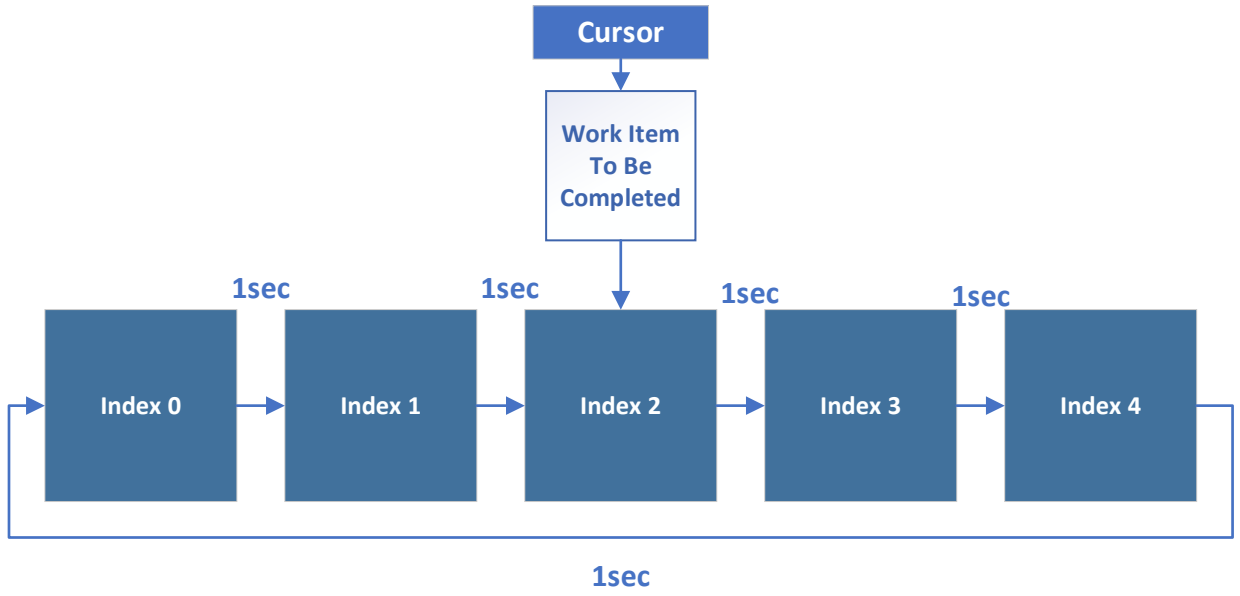
SMB Authentication Rate Limiter

- Example flow (simplified):
  - Work item scheduled, and cursor is now pointing at the next index after 1sec



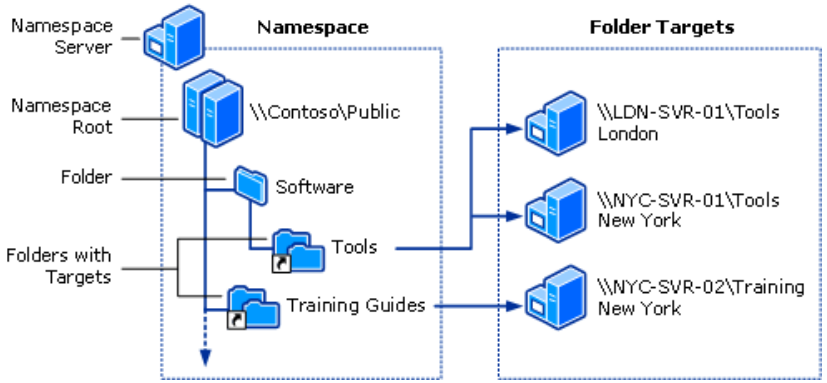
SMB Authentication Rate Limiter

- Example flow (simplified):
  - Work item will be completed given the cursor is now at index 2
  - In actual implementation, we have a linked list of work items to be completed



SMB Authentication Rate Limiter

- Other related improvement to make this feature work correctly:
  - On the SMB Client, due to the way MUP/DFS/SMB/etc. are structured in the kernel, we have dependencies causing additional delays when executing e.g.,
    - NET USE \\<SERVER>\<SHARE> /U:<USERNAME> <INVALID PASSWORD>
  - For instance, DFS attempts an SMB tree connection to the $IPC share to issue an FSCTL to the SMB server to see if the path is part of a DFS namespace
    - The above DFS logic happens before the user request with the original UNC path is forwarded to SMB when attempting the 2nd regular SMB TREE CONNECT
  - To resolve the additional invalid authentication delays, a negative cache was added to the SMB Client
    - 50% improvement over NET USE cases
      - 1 trip to the SMB Server instead of 2 trips
    - 75% improvement over CREATE cases
      - 1 trip to the SMB Server instead of 4 trips



SMB Notifications 

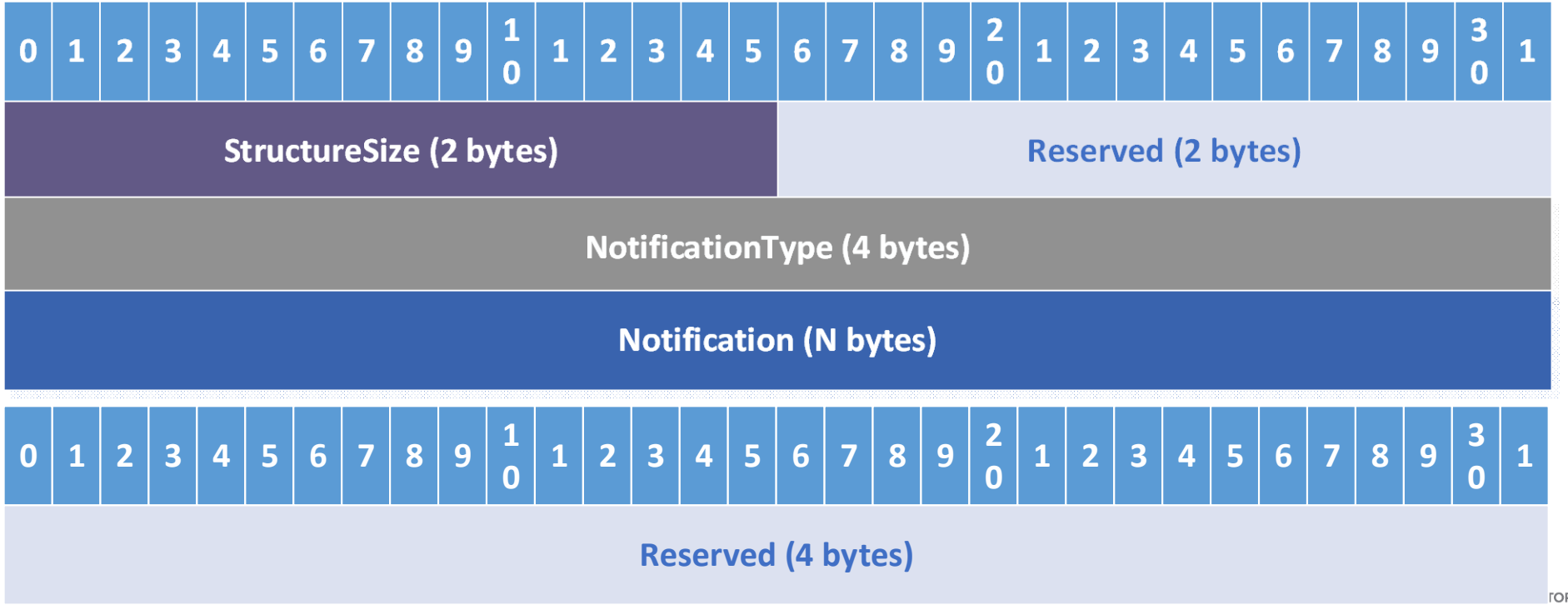
One-way server to client messages that can be repurposed towards several components in SMB such as replacing SOFS witness mechanism and client periodic querying of available network interfaces on the server

SMB Notifications

- Originally designed to solve the RDS AutoDisconnectTimeout issue regarding SMB over Encryption causing a TCP session reset when one user has idled for too long
  - SMB Server forgets encryption / decryption keys, but client does not
  - After a while, SMB Client terminates the connection, affecting all victim sessions
- SMB Server now sends a “Session Closed Notification” to indicate to the client to disconnect the specified session matching the SessionId
  - The next I/O will trigger session re-establishment, kicking in the existing RDR state machine with the session life cycle and creation of new encryption / decryption keys

SMB Notifications

- Protocol Changes
  - SMB2 SERVER TO CLIENT NOTIFICATION Message
  - SMB2 NOTFIY SESSION CLOSED Message



SMB Notifications

- Protocol Changes
  - HEADER UPDATE
    - New SMB Clients should accept SMB2 OPLOCK BREAK and SMB2 SERVER TO CLIENT NOTIFICATION given the MessageId = 0xFFFFFFFFFFFFFFFF

Name	Value
SMB2 SERVER_TO_CLIENT_NOTIFICATION	0x0013

SMB Notifications

- Protocol Changes
  - GENERAL CLIENT INFORMATION
    - SMB Client can receive a session closed notification any time after the client has issued an SMB2 SESSION SETUP request for the session to around the time client receives the SMB Server's response to the client's SMB2 LOGOFF request for the same session
      - Client must discard such session closed notifications if the client cannot find an existing session in the connection's session table

SMB Notifications

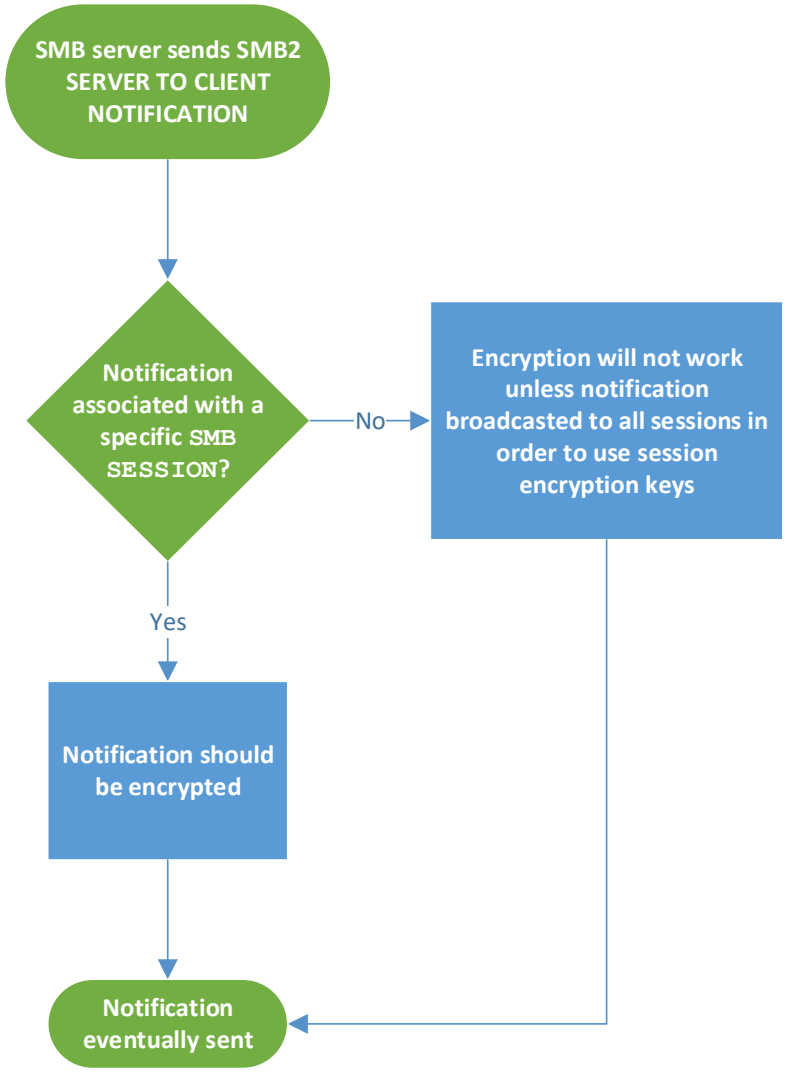
- Protocol Changes
  - NEGOTIATION / DIALECT
    - **SMB2 GLOBAL CAP NOTIFICATIONS** will indicate the SMB Client can received such notifications from a server for SMB3 and above
    - Client will send this new global flag by default along its usual list of capabilities
    - Server must set the **Connection.SupportsNotification** to TRUE if capability is agreed between the client and server
      - During SESSION SETUP phase for the first session, **Session.SupportsNotifications** MUST be set to TRUE if the **Connection.SupportsNotifications** is TRUE for that session.
    - For subsequent binding requests in case of SMB MultiChannel, the server MUST check against the **Session.SupportsNotifications** with the incoming **Connection.SupportsNotifications** value
      - If the value is different, then the server MUST reject the incoming client's binding request

SMB Notifications

- Protocol Changes
  - SIGNING
    - Add a new ServerNotification bit set to 1 for the AES-GMAC nonce generation
    - Generate a random 64-bit MessageID (MID) for each notification rather than the current `0xFFFFFFFFFFFFFFFF` for default notifications.
      - Current “INVALID MID” should be acceptable for HMAC-SHA256, AES-CMAC
    - Add a new field to **SMB2 SERVER TO CLIENT NOTIFICATION** structure
      - For sending a randomly-generated nonce by the server to the client
    - SMB clients ≥ 3.X.X
    - Will acknowledge receiving SMBs with random MIDs (in the case of AES-128-GMAC signing)
    - SMB servers ≥ 3.X.X
      - Will send these signed notifications

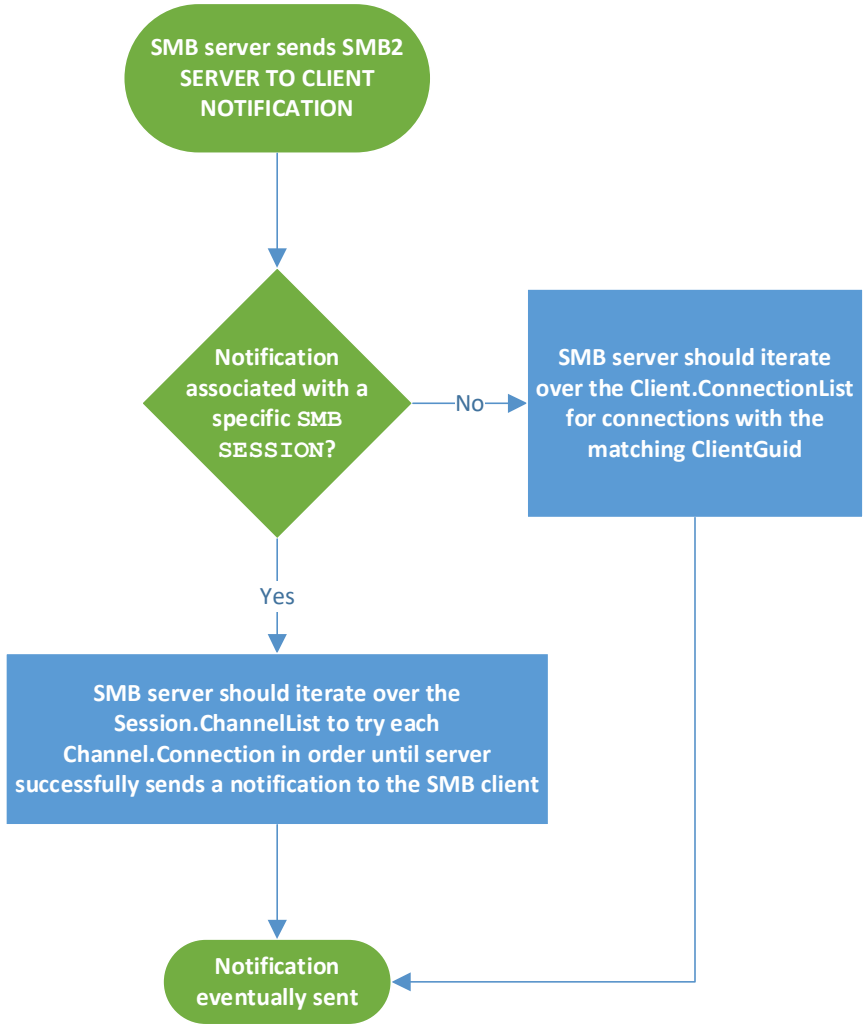
SMB Notifications

- Protocol Changes
  - ENCRYPTION
  - Only send a session closed notification when the notification is associated with a specific session
    - Exception is when we want to broadcast the notification to all sessions



SMB Notifications

- Protocol Changes
  - MULTICHANNEL SCENARIOS
    - If notification not associated with specifics session and scoped to some client, then server iterates through **Client.ConnectionList** corresponding to matching GUID to send the notification to the first available connection. Upon failure, try the next connection.
    - If notification is associated with a specific session, then server iterates through the **Session.ChannelList** to find the first available **Channel.Connection**. Upon failure, try the next **Channel.Connection** available.



SMB Notifications

```
SMB2 (Server Message Block Protocol Version No. 2)
  SMB2 Header
    Size:                0x40
    CreditCharge:        0x0
    Status:              STATUS_SUCCESS
    Command:             SERVER_TO_CLIENT_NOTIFICATION
    Credits:             0x0
    Flags:               SMB2_FLAGS_SERVER_TO_REDIR (0x00000001)
    ServerToRedir:  .............................1 Server to Client
    AsyncCommand:   .............................0. Command is not asynchronous
    Related:        .............................0.. Packet is single message
    Signed:         .............................0... Packet is not signed
    DFS:            0.............................. Command is not a DFS operation
    NextCommand:         0x0
    MessageId:           0xFFFFFFFFFFFFFFFF
    Reserved:            0x0
    TreeId:              0x0
    SessionId:           0x40000000000
  SMB2 Server to Client Notification
    StructureSize:       0x12
    Reserved:            0x0
    NotificationType:    0x0 (SmbNotifySessionsClosed)
    Notification:        SMB2 Notify Session Closed
    SMB2 Notify Session Closed
      Reserved:          0x0
```


SMB Notifications

- Practical Usage / Example(s)
  - One example how we can trigger the newly-added implementation of SMB Notifications for session-closed on the SMB server:
    - Close-SmbSession -SessionId <SESSION ID>
  - The other example is to have 0 open file handles on an active session with encryption enabled for some SMB share for some client
    - Let the session idle until past the 15 minutes timeout (AutoDisconnectTimeout)
    - The SMB Server will clean up state for the session and send the session closed notification to the client

SMB Notifications

- Updates
  - Implementing the session closed notification allowed us an easy way to repro a rare exchange deadlock due to network disconnect(s) in SMB RDR, which was also fixed recently
  - In the next version of Windows Server, there will be a new command value for (Get|Set)-SmbServerConfiguration: AutoDisconnectTimeoutV2
    - The AutoDisconnectTimeout that currently exists in previous versions of Windows Server is for SMBv1 confusingly
    - For the next version of Windows Server, AutoDisconnectTimeoutV1 will be used to distinguish this

Please take a moment to rate this session.

Your feedback is important to us.

