SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

New Transports in Samba

QUIC and SMB-Direct Support

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2025-09-17

https://samba.org/~metze/presentations/2025/SDC/

Topics

- Existing Transports: NBT, TCP
- New Transports: SMBDIRECT, QUIC
- QUIC support using quic.ko on Linux
- QUIC support with only UDP sockets from the kernel
- How QUIC is tested without kernel support
- The road to SMBDIRECT support
- Questions? Feedback!

Existing Transports: NBT, TCP

- SMB Versions 1, 2 and 3 operate on message based transports
  - Each transport message can transport one or more SMB PDUs
- The oldest transport supported by Samba is NBT over TCP
  - This uses tcp port 139
  - Message header: 8 bits type, 7 bits reserved, 17 bits length
  - Simple exchange of netbios names at the start of a connection
  - Payload header (type=0): 4 bytes big endian, max ~128KiB (0x1FFFF)
  - Keepalive messages (type=85)
- Direct TCP is also (almost forever) available in Samba
  - This uses tcp port 445
  - Message header: 8 bits zero, 24 bits length
  - Just payload messages: 4 bytes big endian, max ~16MiB (0xFFFFFF)
- Both operate just on BSD stream sockets for TCP
  - They are identical to implement for the payload messages

New Transports: SMBDIRECT

- SMBDIRECT (a tiny layer on top of RDMA/IB Verbs APIs)
  - Infiniband, RoCE (v1 and v2) and iWarp are supported protocols
  - A message based transport is implement using SEND[_INV] verbs
  - SEND verbs are also message based but only allow small messages
  - Payload messages are reassembled, typically up to 1MiB (0x100000)
  - RDMA Read/Write verbs are used as out of band memory copy
- Windows 2012 introduced it to be used via SMB3 multi-channel
  - Clients start with TCP and ask for server interface IP addresses
  - The client connects via SMBDIRECT is the interface is RDMA capable
  - But the connection only uses IP addresses, so Infinibad requires IPoIB
- SMB2/3 Read/Write can offload data transfer to RDMA Read/Write
  - The client tells the server what client memory the server can read/write
  - The server is the initiator for RDMA Read/Write
  - SMB2 Read results in RDMA Write from server to client
  - SMB2 Write results in RDMA Read from client to server

New Transports: QUIC

- QUIC a rich stream protocol based on TLS 1.3 and UDP
  - Designed for HTTP/3 with security, performance and robustness
  - Used UDP port 443 and should go through routers and firewalls
  - Supports Application-Layer Protocol Negotiation (ALPN)
  - ALPN: "h3" for HTTP/3, "smb" for SMB3
  - Multiple logical stream over a single QUIC connection
- Windows 2022 introduced SMB3 over QUIC
  - Clients try TCP first and fallback to QUIC
  - Only a single logical stream is used per QUIC connection
  - Message header: 8 bits zero, 24 bits length (Same as Direct TCP!)
  - Clients need to access the server via a name
  - Via ip address the cerfiticate would not match
  - SMB3 encryption is done in addition to the QUIC encryption
- Windows 2025 introduced new features
  - The double encryption of QUIC and SMB3 layer is optional
  - SMB2_ACCEPT_TRANSPORT_LEVEL_SECURITY can be negotiated
  - Server can required client certificates

QUIC support using quic.ko on Linux

- quic.ko for Linux provides a BSD stream socket
  - Standalone: https://github.com/lxin/quic
  - It uses socket(AF_INET[6], SOCK_STREAM, IPPROTO_QUIC);
  - Userspace needs to call quic_handshake() until connection is ready
  - From there it is just a stream socket similar to TCP
- The changes to Samba are not very complex
  - We just need to replace IPPROTO_TCP with IPPROTO_QUIC
  - Then setup the gnutls bases tls structures
  - And call quic_handshake()
  - The message processing is not changed at all
  - Only smbstatus –json output was asjusted to show the transport
  - New options "client smb transports" and "server smb transport"
- Linux quic.ko will hopefully mainlined soon
  - Review for v2 of the patches:
  - https://lore.kernel.org/linux-cifs/cover.1755525878.git.lucien.xin@gmail.com/

QUIC support with only UDP sockets from the kernel

- On systems without IPPROTO_QUIC support we use libngtcp2
  - It provides userspace QUIC support on top of UDP sockets
  - It is relatively easy to use with gnutls
  - We can use the same handshake code as with quic.ko
  - It can operate non-blocking and has callback apis
- It is easy to use in Samba on the client side
  - It plugs nicely into our tevent and tstream abstractions
  - We just use it as fallback if quic.ko is not available
  - Should also work on non Linux systems
- Server support is not possible
  - We rely stream sockets and fd-passing for multichannel
  - Sharing a single UDP socket would be very complex

How QUIC is tested without kernel support

- We use socket_wrapper in Samba testing
  - It emulates TCP and UDP sockets using unix domain sockets
  - It uses LD_PRELOAD in order to inject itself into syscalls
  - socket_wrapper_ipproto_quic_socket() uses SOCK_SEQPACKET
- quic_ko_wrapper was created in order to emulate quic.ko
  - It works on top of socket_wrapper_ipproto_quic_socket()
  - SOCK_SEQPACKET on unix sockets provides the connection semantic
  - The QUIC crypto is implemented using libngtcp2 too
- quic_ko_wrapper has some limitations
  - There's no independed instance that handles keepalives
  - It is tricky to hook into [e]poll
  - The activity on the UDP socket fd is not always for the stream
  - The emulated stream may contain multiple messages in one UDP frame
- It is good enough for regression testing
  - We run a few tests to explore the related code paths
  - On the server we use quic_ko_wrapper
  - On the client we test quic_ko_wrapper and direct libngtcp2

The road to SMBDIRECT support (Part1)

- The Linux kernel already has some support for SMBDIRECT
  - The client (cifs.ko) has an "rdma" mount option
  - The server (ksmbd.ko) listens on any available rdma interface
  - Both have their own implementation (partly copied)
- I have a prototype for PF_SMBDIRECT sockets as smbdirect.ko
  - URL:
    https://git.samba.org/?p=metze/linux/smbdirect.git;a=summary
  - I basically it works and allows a Samba prototype to work
  - But it's not stable and some things are still missing
- The goal is to have just a single implementation
  - This should be shared between cifs.ko, ksmbd.ko and userspace
  - We agreed on abstracting and merging the existing code first
  - Then we can improve it and export it via a socket api
  - This is better than a flag day were we switch to new code

The road to SMBDIRECT support (Part2)

- The first challenge was to get the existing code to work
  - It turned out that there were some regressions
  - It took quite some time to reach a state that worked at all
- With a known set of xfstests I was able to start
  - I introduced common header files
  - struct smbdirect_socket is the new structure for a connection
  - I added new elements to it in tiny/trivial steps
  - Followed by client and server changes to use the new elements
  - This made review easy and allows bisecting
- The work use common structures only is finished
  - It's currently in a for-next-next branch (for 6.18)
- The next step is common functions
  - Again in tiny steps I'm moving functions into common
  - While doing this I'm try to avoid to much changes
  - But I find problems when trying to understand the code
  - I'm done with the trivial functions

The road to SMBDIRECT support (Part3)

- When all functions are moved we'll get smbdirect.ko
  - For now I inject the common function via includes
  - This allows to have tiny steps, but it's only temporary
  - The goal is to have a few exported functions smbdirect.ko
- Once we have the basic smbdirect.ko
  - We are finally able to socket(IPPROTO_SMBDIRECT) abstraction
  - The kernel code will inject the 4 byte length framing
  - This will allow userspace to use this code too
  - Above the socket layer it looks like TCP or QUIC
  - Zero-Copy with IORING_OP_{SEND,RECV}_ZC.
  - RDMA Read/Write will use ioctl() or IORING_OP_URING_CMD.
- Having IPPROTO_SMBDIRECT in userspace will allow testing
  - The idea is to have standalone tests independed of SMB
  - It could also be added to tools like iperf.

The road to SMBDIRECT support (Part4)

- Samba can easily use IPPROTO_SMBDIRECT
  - All message handling is the same as for TCP and QUIC
  - Only RDMA Read/Write will require some additional code
- We can also write regression tests
  - These can also run against Windows
- We will likely use a smbdirect_ko_wrapper for testing
  - process_vm_readv()/process_vm_writev() for RDMA Read/Write

Questions? Feedback!

- Stefan Metzmacher, `metze@samba.org`
- https://www.sernet.com
- https://samba.plus

Slides: https://samba.org/~metze/presentations/2025/SDC/