SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2016

SMB3 in Samba

Multi-Channel and Beyond

Michael Adam

Red Hat / samba.org

2016-09-20

agenda

- SMB 2+ $\in$ Samba
- SMB3 Multi-Channel
- Outlook: SMB3 over RDMA
- Outlook: SMB3 Persistent Handles
- Outlook: SMB3 Clustering/Witness

SMB2+ Features ∈ Samba

- SMB 2.0:
  - durable file handles [4.0]
- SMB 2.1:
  - multi-credit / large mtu [4.0]
  - dynamic reauthentication [4.0]
  - leasing [4.2, default in 4.5]
  - resilient file handles [PoC]
- SMB 3.0:
  - new crypto (sign/encrypt) [4.0]
  - secure negotiation [4.0]
  - durable file handles v2 [4.0]
  - multi-channel [4.4 (experimental)]
  - SMB direct [design/PoC]
  - persistent file handles / CA [WIP/PoC]
  - witness [WIP+]
- SMB 3.0.2: [4.3]
- SMB 3.1.1:
  - negotiate contexts, preauth: [4.3]

Multi-Channel

Multi-Channel - General

multiple transport connections in one SMB(3) session

- **channel**: transport connection bound to a session
- client decides which connections to bind and to use
- session is valid as long as at least one channel is intact

two purposes

1. increase throughput:
    - use multiple connections of same type
2. improve fault tolerance:
    - channel failure: replay/retry detection

Multi-Channel - Windows/Protocol

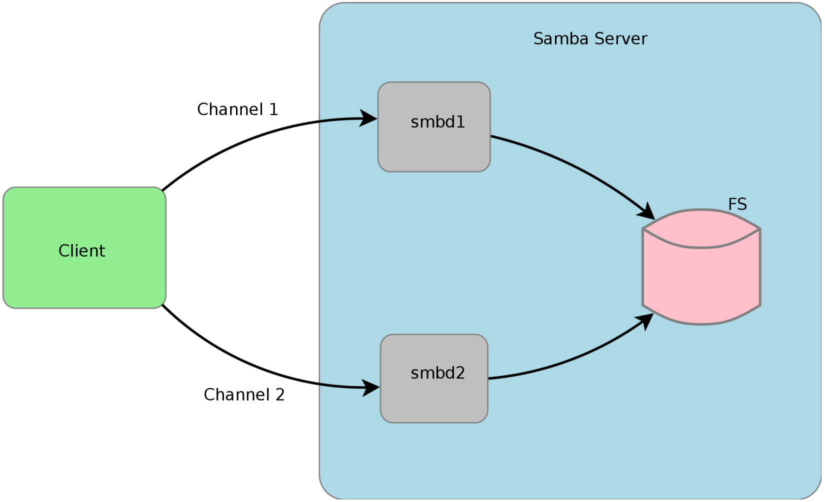
1. establish initial session on TCP connection
2. find interfaces with interface discovery: `FSCTL_QUERY_NETWORK_INTERFACE_INFO`
3. bind additional TCP (or later RDMA) connection (channel) to established SMB3 session (*session bind*)
4. Windows: uses connections of same (and best) quality
5. Windows: binds only to a single node
6. replay / retry mechanisms, epoch numbers

Multi-Channel ∈ Samba

samba/smbd: multi-process

- **Originally:** process ⇔ TCP connection
- Want to avoid synchronization between smbds for disk access.
- **Idea:** transfer new TCP connection to existing smbd
- **How?** ⇒ use fd-passing (sendmsg/recvmsg)
- **When?**
  - *Natural choice*: at SessionSetup (Bind)
  - Samba's choice: at Negotiate, based on ClientGUID

Multi-Channel $\in$ Samba



Multi-Channel ∈ Samba

samba/smbd: multi-process

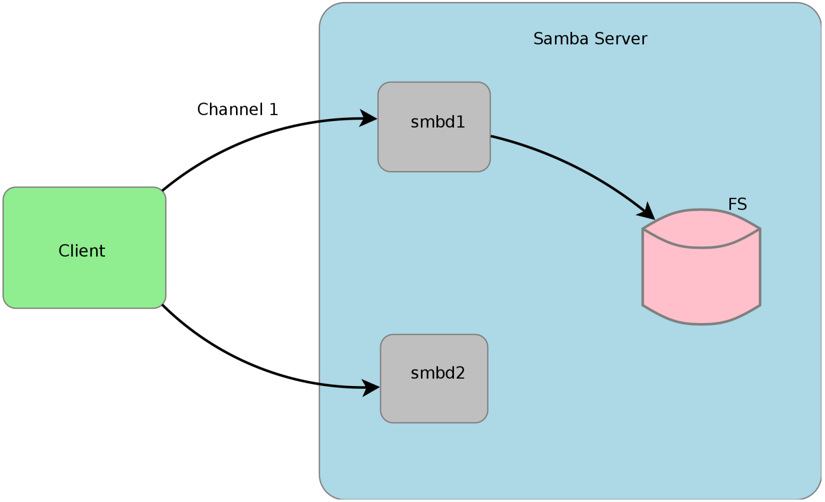
- **Originally:** process ⇔ TCP connection
- Want to avoid synchronization between smbds for disk access.
- **Idea:** transfer new TCP connection to existing smbd
- **How?** ⇒ use fd-passing (sendmsg/recvmsg)
- **When?**
  - *Natural choice*: at SessionSetup (Bind)
  - Samba's choice: at Negotiate, based on ClientGUID

Multi-Channel ∈ Samba

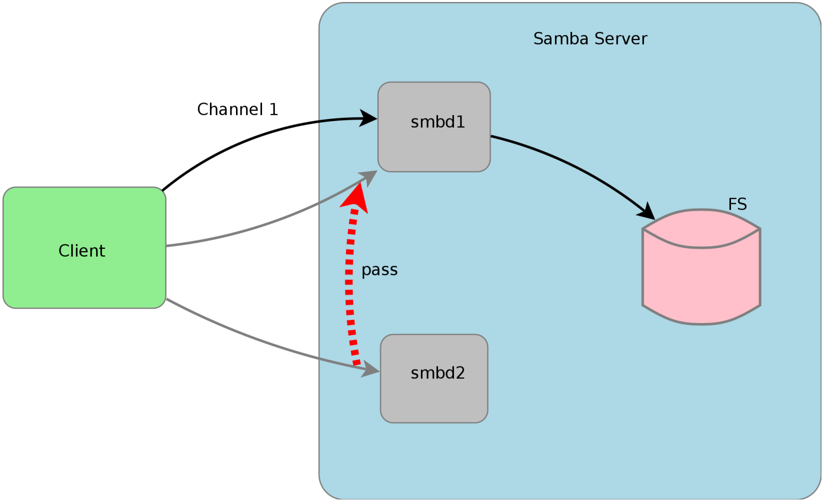
samba/smbd: multi-process

- **Originally:** process ⇔ TCP connection
- Want to avoid synchronization between smbds for disk access.
- **Idea:** transfer new TCP connection to existing smbd
- **How?** ⇒ use fd-passing (sendmsg/recvmsg)
- **When?**
  - *Natural choice*: at SessionSetup (Bind)
  - Samba's choice: at Negotiate, based on ClientGUID

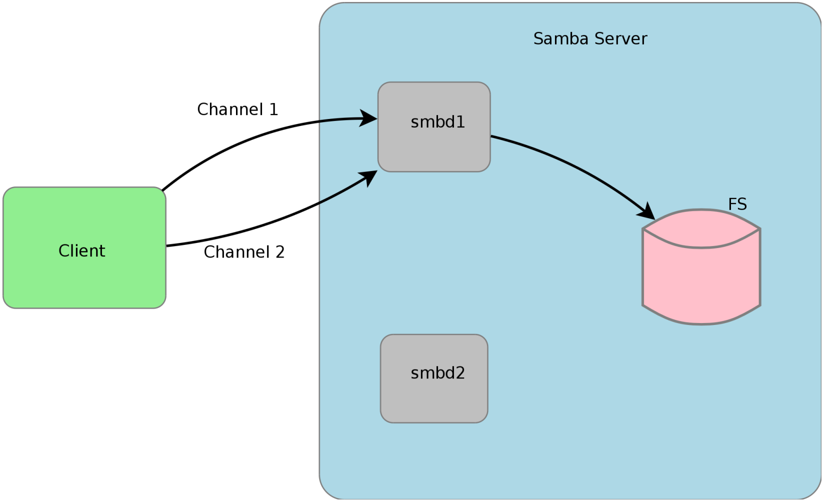
Multi-Channel $\in$ Samba



Multi-Channel ∈ Samba



Multi-Channel ∈ Samba



Multi-Channel ∈ Samba

samba/smbd: multi-process

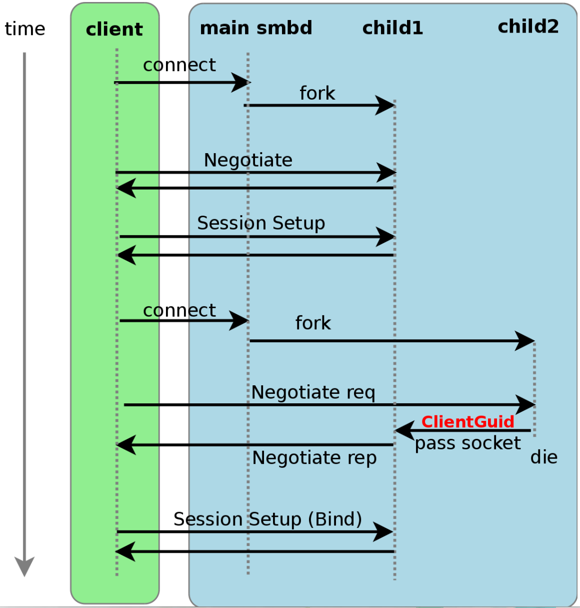
- **Originally:** process ⇔ TCP connection
- Want to avoid synchronization between smbds for disk access.
- **Idea:** transfer new TCP connection to existing smbd
- **How?** ⇒ use fd-passing (sendmsg/recvmsg)
- **When?**
  - *Protocol choice*: at SessionSetup (Bind)
  - Samba's choice: at Negotiate, based on ClientGUID

Multi-Channel ∈ Samba

samba/smbd: multi-process

- **Originally:** process ⇔ TCP connection
- Want to avoid synchronization between smbds for disk access.
- **Idea:** transfer new TCP connection to existing smbd
- **How?** ⇒ use fd-passing (sendmsg/recvmsg)
- **When?**
  - *Protocol choice*: at SessionSetup (Bind)
  - Samba's choice: at Negotiate, based on ClientGUID

Multi-Channel ∈ Samba : pass by ClientGUID



Multi-Channel $\in$ Samba : pass by ClientGUID

Wait a minute - what about performance?

- Single process...
- But we use short-lived worker-pthreads for I/O ops!
- Extensive benchmarks and tunings still to be done.
- First benchmarks show $\geq$ 50% increase going from 1 channel to 2

Multi-Channel ∈ Samba : Status

1. messaging rewrite using unix dgm sockets with sendmsg [DONE,4.2]
2. add fd-passing to messaging [DONE,4.2]
3. preparations in internal structures [DONE,4.4]
4. prepare code to cope with multiple channels [DONE,4.4]
5. implement smbd message to pass a tcp socket [DONE,4.4]
6. transfer connection in Negotiate (by ClientGUID) [DONE,4.4]
7. implement session bind [DONE,4.4]
8. implement channel epoch numbers [DONE,4.4]
9. implement interface discovery [DONE(linux/conf),4.4]
10. implement test cases [WIP(isn't it always?... ☺)]
11. implement fd-passing in socket-wrapper [WIP (Anoop CS, obnox)]
12. implement lease break replay [WIP (obnox, Günther, metze)]
13. integrate Multi-Channel with clustering (CTDB) [WIP]

Multi-Channel ∈ Samba : Status

1. messaging rewrite using unix dgm sockets with sendmsg [DONE,4.2]
2. add fd-passing to messaging [DONE,4.2]
3. preparations in internal structures [DONE,4.4]
4. prepare code to cope with multiple channels [DONE,4.4]
5. implement smbd message to pass a tcp socket [DONE,4.4]
6. transfer connection in Negotiate (by ClientGUID) [DONE,4.4]
7. implement session bind [DONE,4.4]
8. implement channel epoch numbers [DONE,4.4]
9. implement interface discovery [DONE(linux/conf),4.4]
10. implement test cases [WIP(isn't it always?... ☺)]
11. implement fd-passing in socket-wrapper [WIP (Anoop CS, obnox)]
12. implement lease break replay [WIP (obnox, Günther, metze)]
13. integrate Multi-Channel with clustering (CTDB) [WIP]

Multi-Channel ∈ Samba : Status

1. messaging rewrite using unix dgm sockets with sendmsg [DONE,4.2]
2. add fd-passing to messaging [DONE,4.2]
3. preparations in internal structures [DONE,4.4]
4. prepare code to cope with multiple channels [DONE,4.4]
5. implement smbd message to pass a tcp socket [DONE,4.4]
6. transfer connection in Negotiate (by ClientGUID) [DONE,4.4]
7. implement session bind [DONE,4.4]
8. implement channel epoch numbers [DONE,4.4]
9. implement interface discovery [DONE(linux/conf),4.4]
10. implement test cases [WIP(isn't it always?... ☺)]
11. implement fd-passing in socket-wrapper [WIP (Anoop CS, obnox)]
12. implement lease break replay [WIP (obnox, Günther, metze)]
13. integrate Multi-Channel with clustering (CTDB) [WIP]

Multi-Channel ∈ Samba : Status

1. messaging rewrite using unix dgm sockets with sendmsg [DONE,4.2]
2. add fd-passing to messaging [DONE,4.2]
3. preparations in internal structures [DONE,4.4]
4. prepare code to cope with multiple channels [DONE,4.4]
5. implement smbd message to pass a tcp socket [DONE,4.4]
6. transfer connection in Negotiate (by ClientGUID) [DONE,4.4]
7. implement session bind [DONE,4.4]
8. implement channel epoch numbers [DONE,4.4]
9. implement interface discovery [DONE(linux/conf),4.4]
10. implement test cases [WIP(isn't it always?... ☺)]
11. implement fd-passing in socket-wrapper [WIP (Anoop CS, obnox)]
12. implement lease break replay [WIP (obnox, Günther, metze)]
13. integrate Multi-Channel with clustering (CTDB) [WIP]

selftest $\Leftarrow$ fd-passing $\in$ socket-wrapper

1. untangle `socket_info_fd` from `socket_info`
2. array (`sockets`) of `socket_infos` instead of linked list
3. protect structures from concurrent access by ptread mutexes
   - `sockets`, `sockets[i]`, `socket_fds`, `first_free`
   - use process shared robust mutexes where indicated
4. put `sockets` list and `first_free` index into a shared storage
   - use a file, mmap into each user of swrap (like `tdb`)
   - these are the parts to use robust mutexes for
5. implement fd-passing:
   - create a pipe
   - pass one end of pipe along with original fds array
   - sender writes array of indexes to sockets array into pipe
   - receiver reads indexes from pipe and creates new socket_fd structures using fds and indexes

selftest $\Leftarrow$ fd-passing $\in$ socket-wrapper

1. untangle `socket_info_fd` from `socket_info`
2. array (`sockets`) of `socket_infos` instead of linked list
3. protect structures from concurrent access by ptread mutexes
   - `sockets`, `sockets[i]`, `socket_fds`, `first_free`
   - use process shared robust mutexes where indicated
4. put `sockets` list and `first_free` index into a shared storage
   - use a file, mmap into each user of swrap (like `tdb`)
   - these are the parts to use robust mutexes for
5. implement fd-passing:
   - create a pipe
   - pass one end of pipe along with original fds array
   - sender writes array of indexes to sockets array into pipe
   - receiver reads indexes from pipe and creates new socket_fd structures using fds and indexes

Multi-Channel $\in$ Samba: Lease Break Replay

- oplock/lease break request: server $\rightarrow$ client
- protection with channel sequence numbers not available!
- need to track sent but not acked break requests
- declare channel dead if not acked for a while and resend (replay) them over other channels (if any)

- if we don't do that ... multi-channel may eat your data ☹

Multi-Channel $\in$ Samba: Lease Break Replay

- have send_queue, add ack_queue
- use SIOCOUTQ ioctl on the tcp socket:
  unsent data in the socket send queue

Multi-Channel ∈ Samba: Lease Break Replay

```
                           (left the host)
              +------- acked_byte                +----sent_bytes
              |                                  |
              v                                  v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|p1-----|p2-------|p3-----|p4-----|p5----|
       ^                 ^
       |                 |
       |                 +------p3.ack.last_byte
       |
       +---- p1.ack.last_byte
```

Multi-Channel ∈ Samba: Lease Break Replay

WIP code

- `git://git.samba.org/obnox/samba/samba-obnox.git`
  - branch: `master-multi-channel-obnox`
- `git://git.samba.org/gd/samba/.git`
  - branch `master-multichannel`

Multi-Channel ∈ Samba : Clustering/CTDB

Special considerations

- channels of one session only to one node !
- do not bind connections to CTDB public IPs (can move)!
- ⇒ add static IPs on public interfaces
  use these for interface discovery

Multi-Channel ∈ Samba : Clustering/CTDB

Special considerations

- channels of one session only to one node !
- do not bind connections to CTDB public IPs (can move)!
- ⇒ add static IPs on public interfaces
  use these for interface discovery

Outlook: SMB Direct

SMB Direct : SMB3 over RDMA

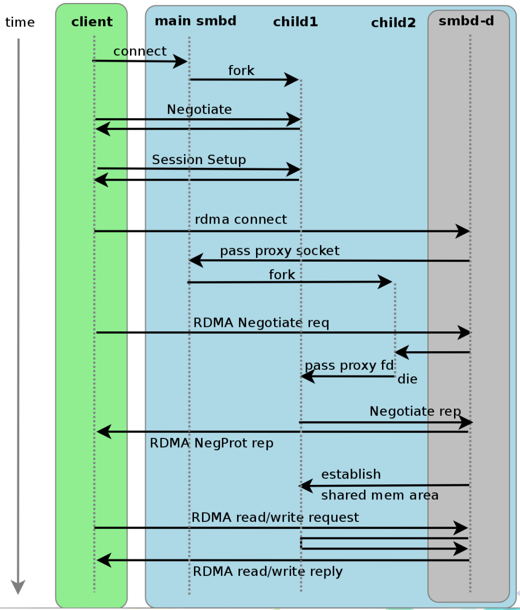
Windows/Protocol

- requires multi-channel
- start with TCP, bind an RDMA channel
- SMB Direct: small wrapper protocol to put SMB into RDMA
- reads and writes use RDMA write/read
- protocol/metadata via send/receive

SMB Direct ∈ Samba

- wireshark dissector: [DONE]
- Samba:
  - prereq: multi-channel [ess.DONE]
  - buffer / transport abstractions [WIP]
- problem with RDMA libraries:
  - not fork safe
  - no fd-passing
- ⇒ central RDMA proxy (”smbd-d”..)
  - PoC/dev: user space daemon
  - production: kernel module
  - see Ralph Böhme’s repo (started by Richard Sharpe): `https://github.com/slowfranklin/smbdirect-driver`

SMB Direct ∈ Samba



Outlook: persistent handles

Persistent File Handles

- available on 'Continuously Available' SMB3 shares
- allows disconnected clients to reconnect
- like durable handles, but with strong guarantees!

Persistent Handles : Challenges

- protocol is easy
  - wip patches for the protocol head exist since many years
  - extended patches for protocol head on ML from contributors
- persistence/guarantees are hard
  - strategies for guarantees:
    - filesystem specific
    - generic, with tdb/ctdb extensions:
      essentially per-record persistence

Outlook: clustering / witness

Witness - General

- New DCE/RPC Service to “witness” availability of IPs, shares, ...
- ⇒ Faster fail-over of clients in the cluster
- Prompt, explicit, and controlled notifications about failures (CTDB tickle-ACKs are implicit)
- Available since SMB3 (Windows 8 / Windows Server 2012)
- basis for ”cluster” capability

Witness - Samba

Currently under development in Samba

- PoC implementation available
- TODO(wip): new async DCE/RPC infrastructure
- `https://wiki.samba.org/index.php/Samba3/SMB2#Witness_Notification_Protocol`
- WIP branch: `https://git.samba.org/?p=gd/samba/.git;a=shortlog;h=refs/heads/master-witness`

Samba Witness service will cause Windows clients to reconnect...

- when client admin tool is used
- when CTDB (or any other cluster resource control manager) moves resources or IP addresses

Wrapping up...

What's next ?

- SMB3 Multi-Channel: finishing moves
- SMB3 Witness service: async RPC
- SMB3 Persistent Handles / CA
- SMB3 over RDMA (SMB direct)
- Multi-Protocol access (NFS, SMB...)
- SMB2+ Unix Extensions ⇒ Jeremy's Talk!

Thanks for your attention!

Questions?

obnox@samba.org

obnox@redhat.com

https://git.samba.org/?p=obnox/slides/2016-09-sdc.git
https://www.samba.org/õbnox/presentations/2016-09-sdc/smb3-in-samba.pr.pdf