SMB2 and SMB3 in Samba: Durable File Handles and Beyond

sambaXP 2012

Michael Adam (obnox@samba.org)
Stefan Metzmacher (metze@samba.org)

Samba Team / SerNet

2012-05-09

Hi there!

Hey, who are you?...

Hey, who are you?...

Oh, and ...

please interrupt with questions!

SMB2+

- SMB 2.0:
  - durable file handles
- SMB 2.1:
  - multi-credit / large mtu
  - dynamic reauthentication
  - leasing
  - resilient file handles
- SMB 2.2^H^H^H3.0:
  - persistent file handles
  - multi-channel
  - SMB direct (SMB over RDMA)
  - cluster features

SMB2+

- SMB 2.0:
  - durable file handles
- SMB 2.1:
  - multi-credit / large mtu
  - dynamic reauthentication
  - leasing
  - resilient file handles
- SMB 2.2^H^H^H3.0:
  - persistent file handles
  - multi-channel
  - SMB direct (SMB over RDMA)
  - cluster features

SMB2+

- SMB 2.0:
  - durable file handles
- SMB 2.1:
  - multi-credit / large mtu
  - dynamic reauthentication
  - leasing
  - resilient file handles
- SMB 2.2^H^H^H3.0:
  - persistent file handles
  - multi-channel
  - SMB direct (SMB over RDMA)
  - cluster features

SMB2+

- SMB 2.0:
  - durable file handles
- SMB 2.1:
  - multi-credit / large mtu
  - dynamic reauthentication
  - leasing
  - resilient file handles
- SMB 2.2^H^H^H3.0:
  - persistent file handles
  - multi-channel
  - SMB direct (SMB over RDMA)
  - cluster features

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  $\Rightarrow$ need to find old session by session_id
- then reconnects durable handle
  $\Rightarrow$ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  $\Rightarrow$ need to find old session by session_id
- then reconnects durable handle
  $\Rightarrow$ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  ⇒ need to find old session by session_id
- then reconnects durable handle
  ⇒ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  $\Rightarrow$ need to find old session by session_id
- then reconnects durable handle
  $\Rightarrow$ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  ⇒ need to find old session by session_id
- then reconnects durable handle
  ⇒ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  ⇒ need to find old session by session_id
- then reconnects durable handle
  ⇒ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  $\Rightarrow$ need to find old session by session_id
- then reconnects durable handle
  $\Rightarrow$ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

Durable Handles And Samba

- target: short network outages
- client reconnects session (cleanup)
  ⇒ need to find old session by session_id
- then reconnects durable handle
  ⇒ needs to find file handle by persistent file ID
- multi-process vs threaded: keep files open vs reopen files
- need to serialze state that had been on memory only needs to be serialized
- new structures in samba: smb(2)-layer vs file system (fsa) layer
- Clustering! (ctdb vs SO and CA)

The Construction Squad ...

The Construction Squad ...

- Stefan Metzmacher
- Michael Adam
- Volker Lendecke
- Christian Ambach
- Gregor Beck
- Björn Baumbach
- + ...

TODO: Improve Protocol Precision

TODO: Improve Protocol Precision

TODO: Improve Structures and Protocol Layer Mixup

- mix of SMB and File System (FSA)/POSIX
- proposal:
  - SMB
  - ntfsa vfs layer
  - posix vfs layer as backend
- untangle create call

TODO: Improve Structures and Protocol Layer Mixup

- mix of SMB and File System (FSA)/POSIX
- proposal:
  - SMB
  - ntfsa vfs layer
  - posix vfs layer as backend
- untangle create call

TODO: Improve Structures and Protocol Layer Mixup

- mix of SMB and File System (FSA)/POSIX
- proposal:
  - SMB
  - ntfsa vfs layer
  - posix vfs layer as backend
- untangle create call

TODO: Improve Structures and Protocol Layer Mixup

- mix of SMB and File System (FSA)/POSIX
- proposal:
  - SMB
  - ntfsa vfs layer
  - posix vfs layer as backend
- untangle create call

TODO: Improve Structures and Protocol Layer Mixup

- mix of SMB and File System (FSA)/POSIX
- proposal:
  - SMB
  - ntfsa vfs layer
  - posix vfs layer as backend
- untangle create call

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library which is able to handle everything (the others are just wrappers now) ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces ⇒ TODO: write a higher level protocol independed library for usage in generic tests and client tools

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library which is able to handle everything (the others are just wrappers now) ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces ⇒ TODO: write a higher level protocol independed library for usage in generic tests and client tools

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library which is able to handle everything (the others are just wrappers now) ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces ⇒ TODO: write a higher level protocol independed library for usage in generic tests and client tools

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality
  and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries
  [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library
  which is able to handle everything (the others are just wrappers now)
  ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests
  (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces
  ⇒ TODO: write a higher level protocol independed library for usage
  in generic tests and client tools

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality
  and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries
  [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library
  which is able to handle everything (the others are just wrappers now)
  ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests
  (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces
  ⇒ TODO: write a higher level protocol independed library for usage
  in generic tests and client tools

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality
  and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries
  [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library
  which is able to handle everything (the others are just wrappers now)
  ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests
  (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces
  ⇒ TODO: write a higher level protocol independed library for usage
  in generic tests and client tools

writing tests and client libraries

- tests to explore protocol details: use client libraries
- the existing client libraries had a limited functionality and it wasn't possible to test all protocol aspects
- we had 4 completely independed client libraries [smb1, smb2] x [source3, source4] (each with its own problems)
- the solution was to create just one low level library which is able to handle everything (the others are just wrappers now) ⇒ `libcli/smb/smbXcli_base.h`
- we now have a lot of new tests (reauth, multi-credit, multi-channel, durable/persistent handles)
- the tests still use the old interfaces ⇒ TODO: write a higher level protocol independed library for usage in generic tests and client tools

existing server structures

the current structures in `smbd` (all in memory)

- `struct smbd_server_connection`
  ⇒ transport connection (one process per connection)
- `struct user_struct`
  ⇒ user session (multiple per connection)
- `struct connection_struct`
  ⇒ tree connect (multiple per connection)
- `struct files_struct`
  ⇒ open file handle (multiple per connection)

existing server structures

the current structures in `smbd` (all in memory)

- `struct smbd_server_connection`
  ⇒ transport connection (one process per connection)
- `struct user_struct`
  ⇒ user session (multiple per connection)
- `struct connection_struct`
  ⇒ tree connect (multiple per connection)
- `struct files_struct`
  ⇒ open file handle (multiple per connection)

existing server structures

the current structures in `smbd` (all in memory)

- `struct smbd_server_connection`
  $\Rightarrow$ transport connection (one process per connection)
- `struct user_struct`
  $\Rightarrow$ user session (multiple per connection)
- `struct connection_struct`
  $\Rightarrow$ tree connect (multiple per connection)
- `struct files_struct`
  $\Rightarrow$ open file handle (multiple per connection)

existing server structures

the current structures in `smbd` (all in memory)

- `struct smbd_server_connection`
  ⇒ transport connection (one process per connection)
- `struct user_struct`
  ⇒ user session (multiple per connection)
- `struct connection_struct`
  ⇒ tree connect (multiple per connection)
- `struct files_struct`
  ⇒ open file handle (multiple per connection)

existing server structures

the current structures in `smbd` (all in memory)

- `struct smbd_server_connection`
  ⇒ transport connection (one process per connection)
- `struct user_struct`
  ⇒ user session (multiple per connection)
- `struct connection_struct`
  ⇒ tree connect (multiple per connection)
- `struct files_struct`
  ⇒ open file handle (multiple per connection)

existing server databases

the current global state databases

- `sessionid.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `connections.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `locking.tdb`
  ⇒ open file information
- `brlock.tdb`
  ⇒ byte range lock information

existing server databases

the current global state databases

- `sessionid.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `connections.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `locking.tdb`
  ⇒ open file information
- `brlock.tdb`
  ⇒ byte range lock information

existing server databases

the current global state databases

- `sessionid.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `connections.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `locking.tdb`
  ⇒ open file information
- `brlock.tdb`
  ⇒ byte range lock information

existing server databases

the current global state databases

- `sessionid.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `connections.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `locking.tdb`
  ⇒ open file information
- `brlock.tdb`
  ⇒ byte range lock information

existing server databases

the current global state databases

- `sessionid.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `connections.tdb`
  ⇒ mostly only for debugging (smbstatus)
- `locking.tdb`
  ⇒ open file information
- `brlock.tdb`
  ⇒ byte range lock information

problems with the current design regarding new features

- The current structures mix the SMB1/2/3 server layer with the filesystem layers
  $\Rightarrow$ `[MS-CIFS], [MS-SMB] and [MS-SMB2]`
  vs.
  $\Rightarrow$ `[MS-FSA]`
  vs.
  $\Rightarrow$ `SMB_VFS / posix layer`
- As the structures public used by different layers they can't be changed easily in order to fix problem in just one of the layers

problems with the current design regarding new features

- The current structures mix the SMB1/2/3 server layer with the filesystem layers
  ⇒ `[MS-CIFS], [MS-SMB] and [MS-SMB2]`
  vs.
  ⇒ `[MS-FSA]`
  vs.
  ⇒ `SMB_VFS / posix layer`
- As the structures public used by different layers they can't be changed easily in order to fix problem in just one of the layers

problems with the current design regarding new features

- The current structures mix the SMB1/2/3 server layer with the filesystem layers
  ⇒ `[MS-CIFS], [MS-SMB] and [MS-SMB2]`
  vs.
  ⇒ `[MS-FSA]`
  vs.
  ⇒ `SMB_VFS / posix layer`
- As the structures public used by different layers they can't be changed easily in order to fix problem in just one of the layers

cleanup work (gensec)

- backport the gensec code
  (as abstraction layer, but with the old code as implementation)
  ⇒ this makes it possible to use the same authentiation code
  in all places (SMB, RPC, LDAP and other servers)
  (with the help of Andrew Bartlett)
- The SMB1/2 code was simplified a lot
  ⇒ v3-6 vs. master

```
source3/smbd/sesssetup.c      | 1294 +++++---------------------------------------
source3/smbd/smb2_sesssetup.c |  627 ++-------------------
2 files changed, 226 insertions(+), 1695 deletions(-)
```

cleanup work (gensec)

- backport the gensec code
  (as abstraction layer, but with the old code as implementation)
  ⇒ this makes it possible to use the same authentiation code
  in all places (SMB, RPC, LDAP and other servers)
  (with the help of Andrew Bartlett)

- The SMB1/2 code was simplified a lot
  ⇒ v3-6 vs. master

```
source3/smbd/sesssetup.c      | 1294 +++++-----------------------------------------
source3/smbd/smb2_sesssetup.c |  627 ++------------------
2 files changed, 226 insertions(+), 1695 deletions(-)
```

cleanup work (gensec)

- backport the gensec code
  (as abstraction layer, but with the old code as implementation)
  ⇒ this makes it possible to use the same authentiation code
  in all places (SMB, RPC, LDAP and other servers)
  (with the help of Andrew Bartlett)
- The SMB1/2 code was simplified a lot
  ⇒ v3-6 vs. master

```
source3/smbd/sesssetup.c      | 1294 +++++------------------------------------
source3/smbd/smb2_sesssetup.c |  627 ++------------------
2 files changed, 226 insertions(+), 1695 deletions(-)
```

new smbXsrv structures and databases

Structures for the SMB1/2/3 server layer are the first step

- `struct smbXsrv_connection` (per transport connection/in memory)
- `struct smbXsrv_session` (per user session/in memory)
  - `struct smbXsrv_session_global` (in `smbXsrv_session_global.tdb` with 32bit index key)
- `struct smbXsrv_tcon` (per tree connect/in memory)
  - `struct smbXsrv_tcon_global` (in smbXsrv_tcon_global.tdb with 32bit index key)
- `struct smbXsrv_open` (per open file handle/in memory)
  - `struct smbXsrv_open_global` (in smbXsrv_open_global.tdb with 32bit index key)
- `struct smbXsrv_version_global` (smbXsrv_version_global.tdb just one record)
  $\Rightarrow$ an array with version information per node
  $\Rightarrow$ maybe allows rolling code upgrades later

new smbXsrv structures and databases

Structures for the SMB1/2/3 server layer are the first step

- `struct smbXsrv_connection` (per transport connection/in memory)
- `struct smbXsrv_session` (per user session/in memory)
  - `struct smbXsrv_session_global` (in `smbXsrv_session_global.tdb` with 32bit index key)
- `struct smbXsrv_tcon` (per tree connect/in memory)
  - `struct smbXsrv_tcon_global` (in smbXsrv_tcon_global.tdb with 32bit index key)
- `struct smbXsrv_open` (per open file handle/in memory)
  - `struct smbXsrv_open_global` (in smbXsrv_open_global.tdb with 32bit index key)
- `struct smbXsrv_version_global` (smbXsrv_version_global.tdb just one record) ⇒ an array with version information per node ⇒ maybe allows rolling code upgrades later

new smbXsrv structures and databases

Structures for the SMB1/2/3 server layer are the first step

- `struct smbXsrv_connection` (per transport connection/in memory)
- `struct smbXsrv_session` (per user session/in memory)
  - `struct smbXsrv_session_global`
    (in `smbXsrv_session_global.tdb` with 32bit index key)
- `struct smbXsrv_tcon` (per tree connect/in memory)
  - `struct smbXsrv_tcon_global`
    (in smbXsrv_tcon_global.tdb with 32bit index key)
- `struct smbXsrv_open` (per open file handle/in memory)
  - `struct smbXsrv_open_global`
    (in smbXsrv_open_global.tdb with 32bit index key)
- `struct smbXsrv_version_global`
  (smbXsrv_version_global.tdb just one record)
  ⇒ an array with version information per node
  ⇒ maybe allows rolling code upgrades later

new smbXsrv structures and databases

Structures for the SMB1/2/3 server layer are the first step

- `struct smbXsrv_connection` (per transport connection/in memory)
- `struct smbXsrv_session` (per user session/in memory)
  - `struct smbXsrv_session_global`
    (in `smbXsrv_session_global.tdb` with 32bit index key)
- `struct smbXsrv_tcon` (per tree connect/in memory)
  - `struct smbXsrv_tcon_global`
    (in smbXsrv_tcon_global.tdb with 32bit index key)
- `struct smbXsrv_open` (per open file handle/in memory)
  - `struct smbXsrv_open_global`
    (in smbXsrv_open_global.tdb with 32bit index key)
- `struct smbXsrv_version_global`
  (smbXsrv_version_global.tdb just one record)
  ⇒ an array with version information per node
  ⇒ maybe allows rolling code upgrades later

new smbXsrv structures and databases

Structures for the SMB1/2/3 server layer are the first step

- `struct smbXsrv_connection` (per transport connection/in memory)
- `struct smbXsrv_session` (per user session/in memory)
  - `struct smbXsrv_session_global`
    (in `smbXsrv_session_global.tdb` with 32bit index key)
- `struct smbXsrv_tcon` (per tree connect/in memory)
  - `struct smbXsrv_tcon_global`
    (in smbXsrv_tcon_global.tdb with 32bit index key)
- `struct smbXsrv_open` (per open file handle/in memory)
  - `struct smbXsrv_open_global`
    (in smbXsrv_open_global.tdb with 32bit index key)
- `struct smbXsrv_version_global`
  (smbXsrv_version_global.tdb just one record)
  ⇒ an array with version information per node
  ⇒ maybe allows rolling code upgrades later

new smbXsrv structures and databases

Structures for the SMB1/2/3 server layer are the first step

- `struct smbXsrv_connection` (per transport connection/in memory)
- `struct smbXsrv_session` (per user session/in memory)
  - `struct smbXsrv_session_global`
    (in `smbXsrv_session_global.tdb` with 32bit index key)
- `struct smbXsrv_tcon` (per tree connect/in memory)
  - `struct smbXsrv_tcon_global`
    (in smbXsrv_tcon_global.tdb with 32bit index key)
- `struct smbXsrv_open` (per open file handle/in memory)
  - `struct smbXsrv_open_global`
    (in smbXsrv_open_global.tdb with 32bit index key)
- `struct smbXsrv_version_global`
  (smbXsrv_version_global.tdb just one record)
  ⇒ an array with version information per node
  ⇒ maybe allows rolling code upgrades later

useful infrastructure

- dbwrap_record_watch_send()/dbwrap_record_watch_recv() (by Volker Lendecke) ⇒ an easy way to get notified when a tdb record changed
- msg_channel_init(), msg_read_send()/msg_read_recv() (by Volker Lendecke) ⇒ a tevent_req based infrastructure to receive samba internal messages

(Maybe) in future:

- (re)write and unify the source3 and source4 `struct messaging_context` subsystems to have a way all samba components are able to talk to each other
- make IRPC (currently only in source4) available for the whole code base
- make it possible to do `fd passing` via IRPC

useful infrastructure

- dbwrap_record_watch_send()/dbwrap_record_watch_recv() (by Volker Lendecke) ⇒ an easy way to get notified when a tdb record changed
- msg_channel_init(), msg_read_send()/msg_read_recv() (by Volker Lendecke) ⇒ a tevent_req based infrastructure to receive samba internal messages

(Maybe) in future:

- (re)write and unify the source3 and source4 `struct messaging_context` subsystems to have a way all samba components are able to talk to each other
- make IRPC (currently only in source4) available for the whole code base
- make it possible to do `fd passing` via IRPC

useful infrastructure

- dbwrap_record_watch_send()/dbwrap_record_watch_recv() (by Volker Lendecke)
  ⇒ an easy way to get notified when a tdb record changed
- msg_channel_init(), msg_read_send()/msg_read_recv() (by Volker Lendecke)
  ⇒ a tevent_req based infrastructure to receive samba internal messages

(Maybe) in future:

- (re)write and unify the source3 and source4 `struct messaging_context` subsystems
  to have a way all samba components are able to talk to each other
- make IRPC (currently only in source4) available for the whole code base
- make it possible to do `fd passing` via IRPC

useful infrastructure

- dbwrap_record_watch_send()/dbwrap_record_watch_recv() (by Volker Lendecke)
  $\Rightarrow$ an easy way to get notified when a tdb record changed
- msg_channel_init(), msg_read_send()/msg_read_recv() (by Volker Lendecke)
  $\Rightarrow$ a tevent_req based infrastructure to receive samba internal messages

(Maybe) in future:

- (re)write and unify the source3 and source4 `struct messaging_context` subsystems
  to have a way all samba components are able to talk to each other
- make IRPC (currently only in source4) available for the whole code base
- make it possible to do `fd passing` via IRPC

useful infrastructure

- dbwrap_record_watch_send()/dbwrap_record_watch_recv()
  (by Volker Lendecke)
  ⇒ an easy way to get notified when a tdb record changed
- msg_channel_init(), msg_read_send()/msg_read_recv()
  (by Volker Lendecke)
  ⇒ a tevent_req based infrastructure to receive samba internal messages

(Maybe) in future:

- (re)write and unify the source3 and source4
  `struct messaging_context` subsystems
  to have a way all samba components are able to talk to each other
- make IRPC (currently only in source4) available for the whole code base
- make it possible to do `fd passing` via IRPC

useful infrastructure

- dbwrap_record_watch_send()/dbwrap_record_watch_recv() (by Volker Lendecke) ⇒ an easy way to get notified when a tdb record changed
- msg_channel_init(), msg_read_send()/msg_read_recv() (by Volker Lendecke) ⇒ a tevent_req based infrastructure to receive samba internal messages

(Maybe) in future:

- (re)write and unify the source3 and source4 `struct messaging_context` subsystems to have a way all samba components are able to talk to each other
- make IRPC (currently only in source4) available for the whole code base
- make it possible to do `fd passing` via IRPC

dynamic reauthentication

- with SMB1 and SMB 2.0 reauthentication was designed to only happen when a kerberos ticket expired
  $\Rightarrow$ when the server returns NT_STATUS_USER_SESSION_EXPIRED
- with SMB 2.1 clients, clients can reauthentiate a session at anytime
  $\Rightarrow$ which means we have to implement it.
- implementing dynamic reauth is much eachier using gensec and the new smbXsrv structures
- but it's still not that easy as there might be code that relies on pointers to the previous 'struct auth_session_info' in memory during async operations.

dynamic reauthentication

- with SMB1 and SMB 2.0 reauthentication was designed to only happen when a kerberos ticket expired
  ⇒ when the server returns NT_STATUS_USER_SESSION_EXPIRED
- with SMB 2.1 clients, clients can reauthentiate a session at anytime
  ⇒ which means we have to implement it.
- implementing dynamic reauth is much eachier
  using gensec and the new smbXsrv structures
- but it's still not that easy as there might be code that
  relies on pointers to the previous 'struct auth_session_info'
  in memory during async operations.

dynamic reauthentication

- with SMB1 and SMB 2.0 reauthentication was designed to only happen when a kerberos ticket expired
  $\Rightarrow$ when the server returns NT_STATUS_USER_SESSION_EXPIRED
- with SMB 2.1 clients, clients can reauthentiate a session at anytime
  $\Rightarrow$ which means we have to implement it.
- implementing dynamic reauth is much eachier using gensec and the new smbXsrv structures
- but it's still not that easy as there might be code that relies on pointers to the previous 'struct auth_session_info' in memory during async operations.

dynamic reauthentication

- with SMB1 and SMB 2.0 reauthentication was designed to only happen when a kerberos ticket expired
  $\Rightarrow$ when the server returns NT_STATUS_USER_SESSION_EXPIRED
- with SMB 2.1 clients, clients can reauthentiate a session at anytime
  $\Rightarrow$ which means we have to implement it.
- implementing dynamic reauth is much eachier
  using gensec and the new smbXsrv structures
- but it's still not that easy as there might be code that
  relies on pointers to the previous 'struct auth_session_info'
  in memory during async operations.

dynamic reauthentication

- with SMB1 and SMB 2.0 reauthentication was designed to only happen when a kerberos ticket expired
  $\Rightarrow$ when the server returns NT_STATUS_USER_SESSION_EXPIRED
- with SMB 2.1 clients, clients can reauthentiate a session at anytime
  $\Rightarrow$ which means we have to implement it.
- implementing dynamic reauth is much eachier using gensec and the new smbXsrv structures
- but it's still not that easy as there might be code that relies on pointers to the previous 'struct auth_session_info' in memory during async operations.

session reconnect (handling previous_session_id)

- when a client reconnects to a server (after a network problem) it tries to recreate the user sessions, tree connects and (durable) open file handles
- on the SMB2/3 session setup the clients sends the previous_session_id $\Rightarrow$ the server closes all opens on the old session in case the server doesn't noticed the network problem of the client.
- implementing this within samba was relatively easy using the new smbXsrv structures and the new helpers

session reconnect (handling previous_session_id)

- when a client reconnects to a server (after a network problem) it tries to recreate the user sessions, tree connects and (durable) open file handles
- on the SMB2/3 session setup the clients sends the previous_session_id $\Rightarrow$ the server closes all opens on the old session in case the server doesn't noticed the network problem of the client.
- implementing this within samba was relatively easy using the new smbXsrv structures and the new helpers

session reconnect (handling previous_session_id)

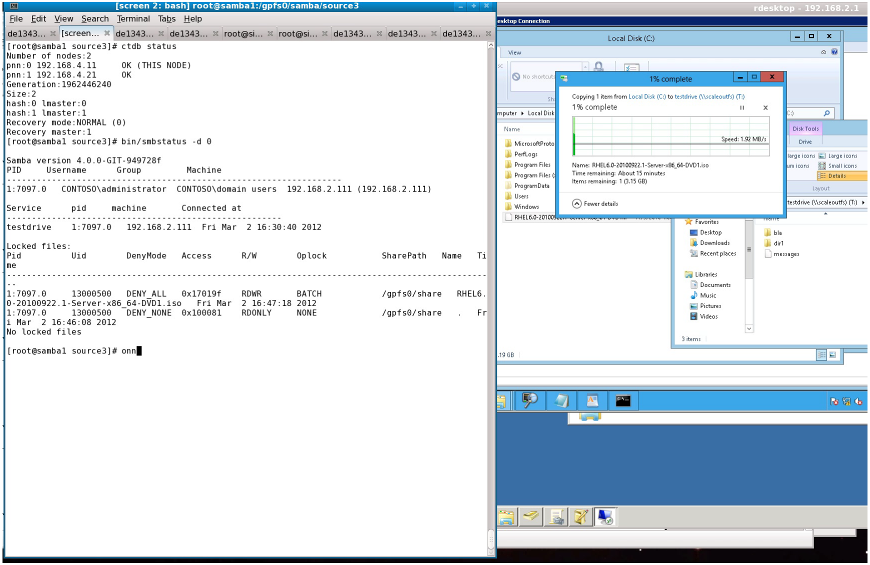
- when a client reconnects to a server (after a network problem) it tries to recreate the user sessions, tree connects and (durable) open file handles
- on the SMB2/3 session setup the clients sends the previous_session_id $\Rightarrow$ the server closes all opens on the old session in case the server doesn't noticed the network problem of the client.
- implementing this within samba was relatively easy using the new smbXsrv structures and the new helpers

session reconnect (handling previous_session_id)

- when a client reconnects to a server (after a network problem) it tries to recreate the user sessions, tree connects and (durable) open file handles
- on the SMB2/3 session setup the clients sends the previous_session_id $\Rightarrow$ the server closes all opens on the old session in case the server doesn't noticed the network problem of the client.
- implementing this within samba was relatively easy using the new smbXsrv structures and the new helpers

What is already working?

What is already working?



```
[root@sambal source3]# ctdb status
Number of nodes:2
pnn:0 192.168.4.11     OK (THIS NODE)
pnn:1 192.168.4.21     OK
Generation:1962446240
Size:2
hash:0 lmaster:0
hash:1 lmaster:1
Recovery mode:NORMAL (0)
Recovery master:1
[root@sambal source3]# bin/smbstatus -d 0

Samba version 4.0.0-GIT-949728f
PID     Username      Group         Machine
-------------------------------------------------------------------
1:7097.0   CONTOSO\administrator  CONTOSO\domain users  192.168.2.111 (192.168.2.111)

Service      pid     machine       Connected at
-------------------------------------------------------
testdrive    1:7097.0    192.168.2.111  Fri Mar  2 16:30:40 2012

Locked files:
Pid          Uid        DenyMode   Access      R/W        Oplock           SharePath   Name   Ti
me
------------------------------------------------------------------------------------------------
--
1:7097.0     13000500   DENY_ALL   0x17019f    RDWR       BATCH            /gpfs0/share   RHEL6.
0-20100922.1-Server-x86_64-DVD1.iso    Fri Mar  2 16:47:18 2012
1:7097.0     13000500   DENY_NONE  0x100081    RDONLY     NONE             /gpfs0/share   .    Fr
i Mar  2 16:46:08 2012
No locked files

[root@sambal source3]# onn
```

1% complete

Copying 1 item from Local Disk (C:) to testdrive (\\scaleoutfs) (T:)

1% complete

Speed: 1.92 MB/s

Name: RHEL6.0-20100922.1-Server-x86_64-DVD1.iso
Time remaining: About 15 minutes
Items remaining: 1 (3.15 GB)

When will we get it???

When will we get it???

Questions?

`https://wiki.samba.org/index.php/Samba3/SMB2`