A Status Report on SMB Direct for Samba

Richard Sharpe

Samba Team & Panzura

Agenda

- How we got here
- The relevant protocol details
- Overview of the structure of Samba
- The options
- The Linux driver option
- Samba changes needed
- Status
- Acknowledgements
- Further Information

How we got here

- 2011 Microsoft Introduced SMB2.2 and SMB Direct at SDC 2011
- 2011 I played around with RDMA
- May 2012 Microsoft gave SMB2.2/3.0 tutorial at Samba XP
- Some of us thought about it
- Mellanox supplied some IB cards to some Samba team members
- May 2013 Microsoft gave further presentations about SMB3.0
- After that I started to get serious about it

How we got here, cont

- June 2013 I had a conference call with Mellanox to discuss options
- August 2013 I started circulating a design document

The relevant protocol details

- Client connects via TCP first (port 445)
  - SESSION_SETUP obtains Session ID
  - Connects to a share
- Queries the network interfaces
  - FSCTL_QUERY_NETWORK_INTERFACE_INFO
- Place an RDMA Connection to server on port **5445**
- Brings up SMB Direct Protocol Engine
- Transport SMB PDUs

Relevant Protocol Details, cont

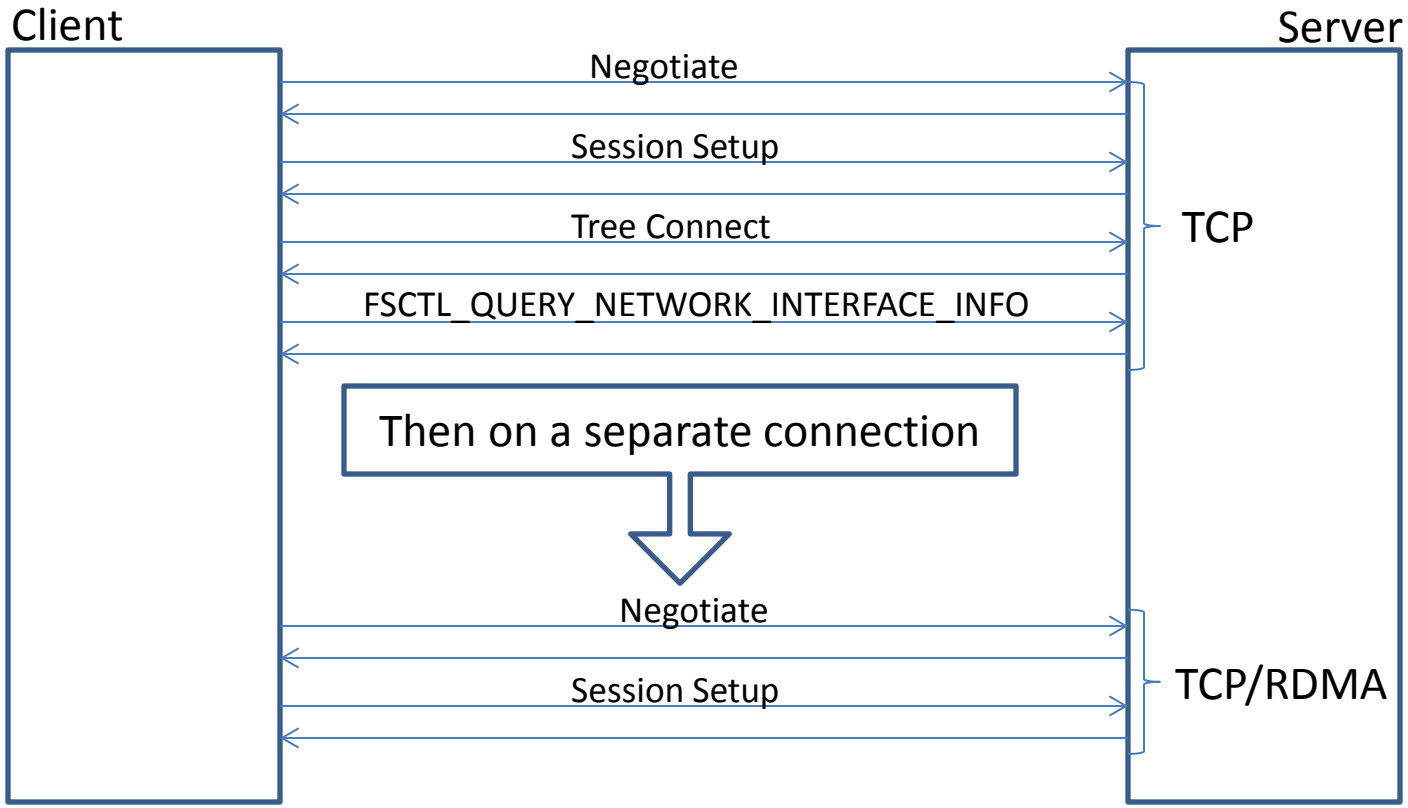
- Client sends SMB2/3 Negotiate request
  - Dialect 0x300 (SMB 3)
  - SMB2_GLOBAL_CAP_MULTI_CHANNEL in Capabilities field
- Server responds
- Client sends SMB2/3 SESSION_SETUP request
  - SMB2_SESSION_FLAG_BINDING in flags
  - Session ID same as the one obtained for first connection/session

Relevant Protocol Details, cont

- SMB Direct
  - Thin layer on RDMA
  - Transports SMB3 PDUs
  - Negotiate request and response
  - Data transfer message
  - Buffer descriptor structure

Relevant Protocol Details, cont

- SMB2 spec section 4.8 gives an example



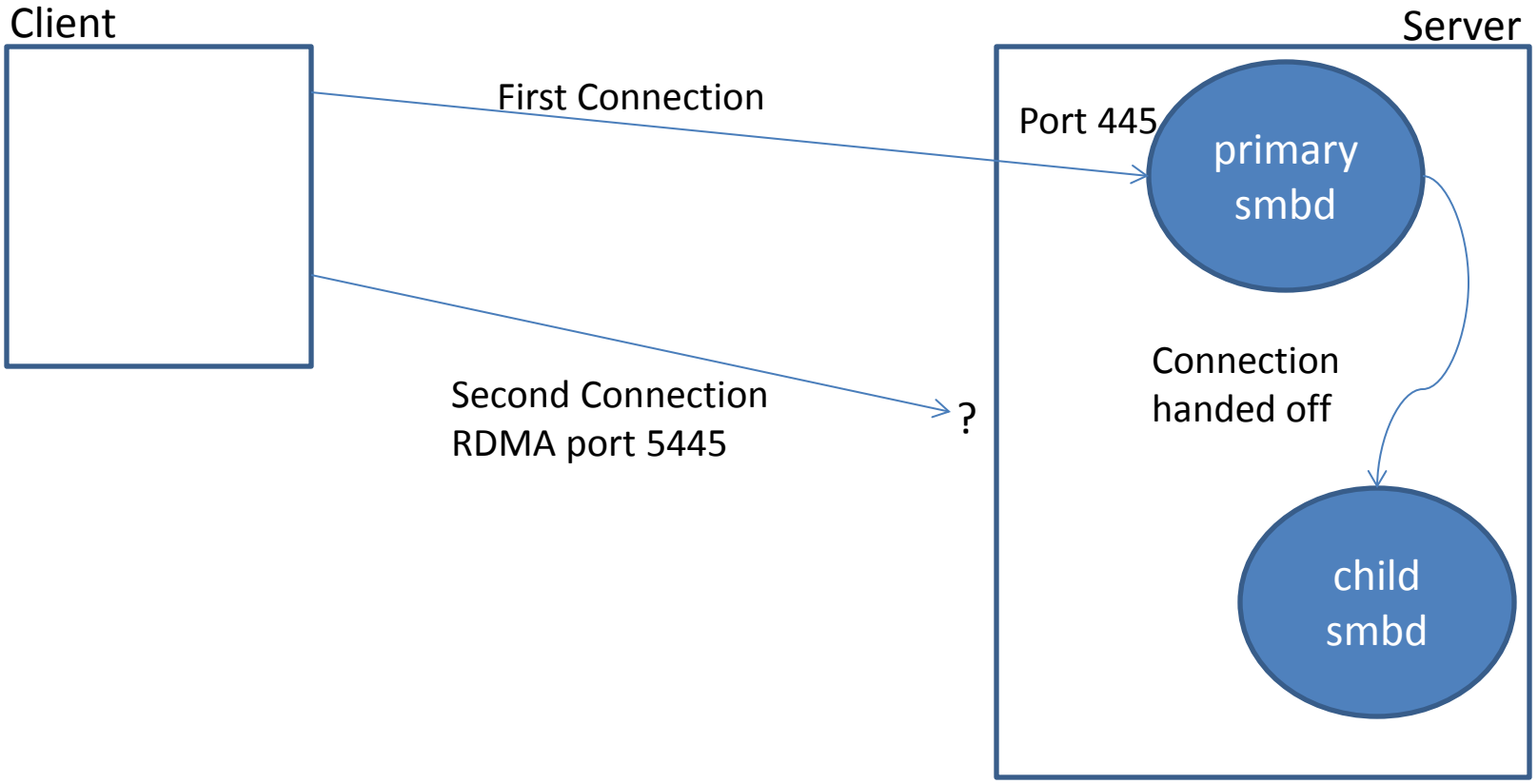
SMB Direct

- Small protocol
  - Transports SMB PDUs over RDMA
  - Support for RDMA READ and RDMA WRITE
- Negotiate exchange
  - Sets parameters
- PDU transfer phase

Overview of the structure of Samba

- Master smbd
  - Fork model
  - Accepts all incoming TCP connections
  - Forks a new process for each TCP connection
  - Does not handle any SMB PDUs
- Separate process per connection
- Uses poll/epoll and an event mechanism for handling SMB PDUs and other events
- Separate SMB and SMB2/3 code paths

Samba Structure



Issues

- How to get RDMA connections/sessions associated with the original TCP connection/session?
  - Clients always connect to port 5445 for RDMA
  - Mellanox folks tell me you cannot transfer RDMA connections from one process to another
    - Too much state, especially memory state

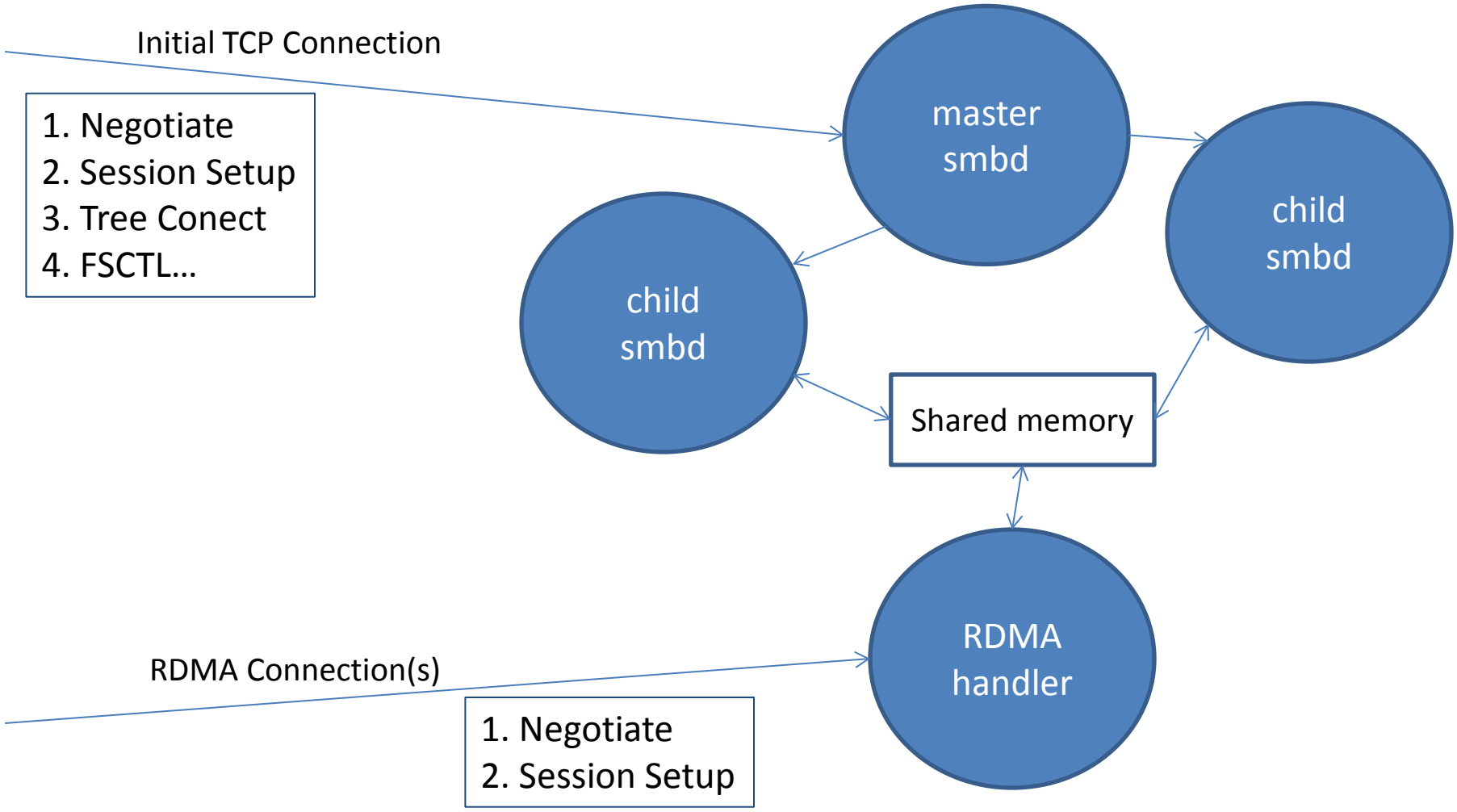
The Options

- Convert Samba to a threaded model
  - Everything in one address space
- Separate process to handle all RDMA connections and data transport
- Kernel driver to handle RDMA

Convert Samba to threaded model

- Would simplify multi-connect with TCP and RDMA
- A lot of work
  - The code still has many assumptions around each TCP connection handled in a separate process
- Problems?
  - Max open FDs?
  - Posix Threads and UIDs and GIDs

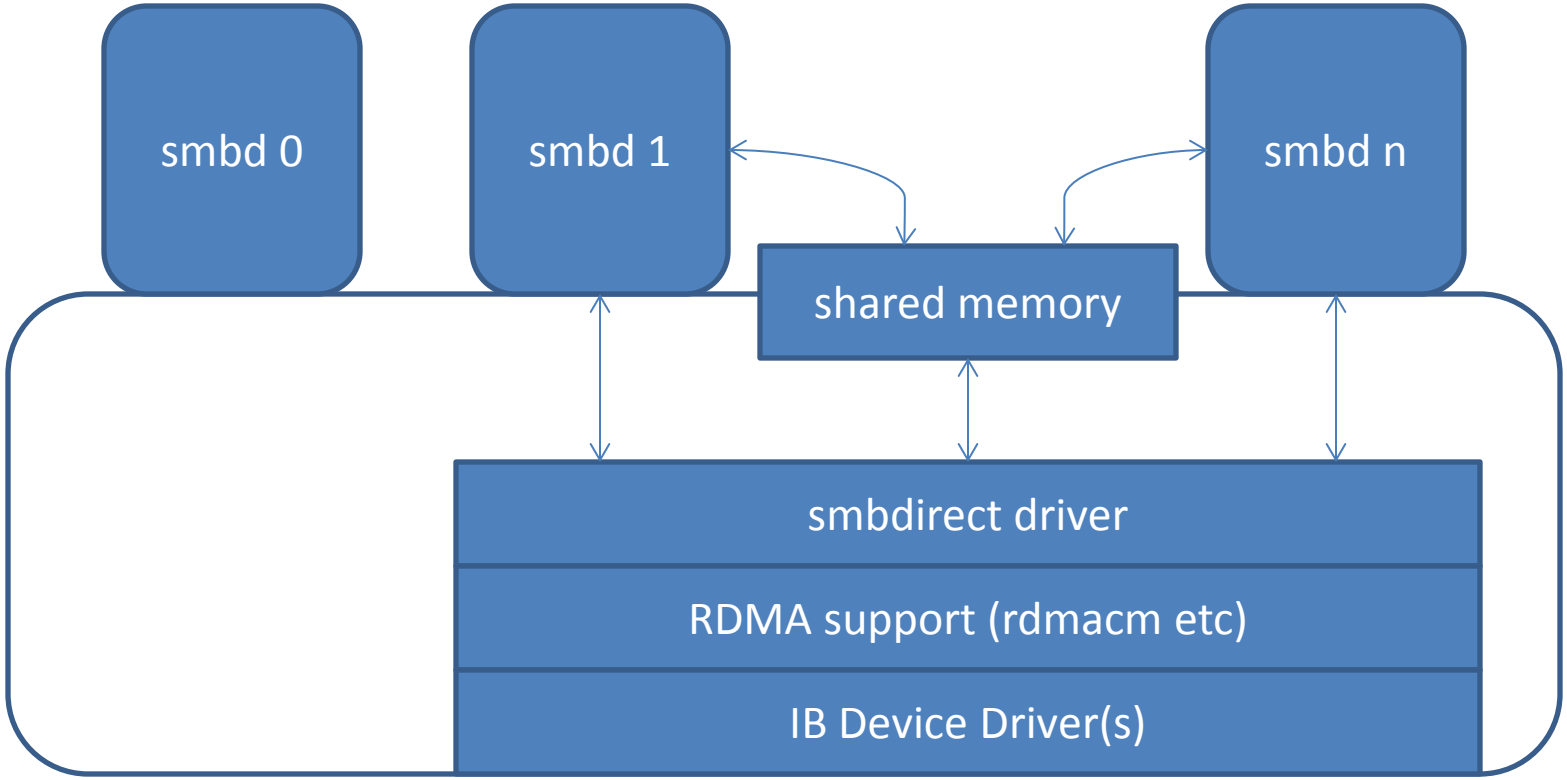
Separate RDMA handler process



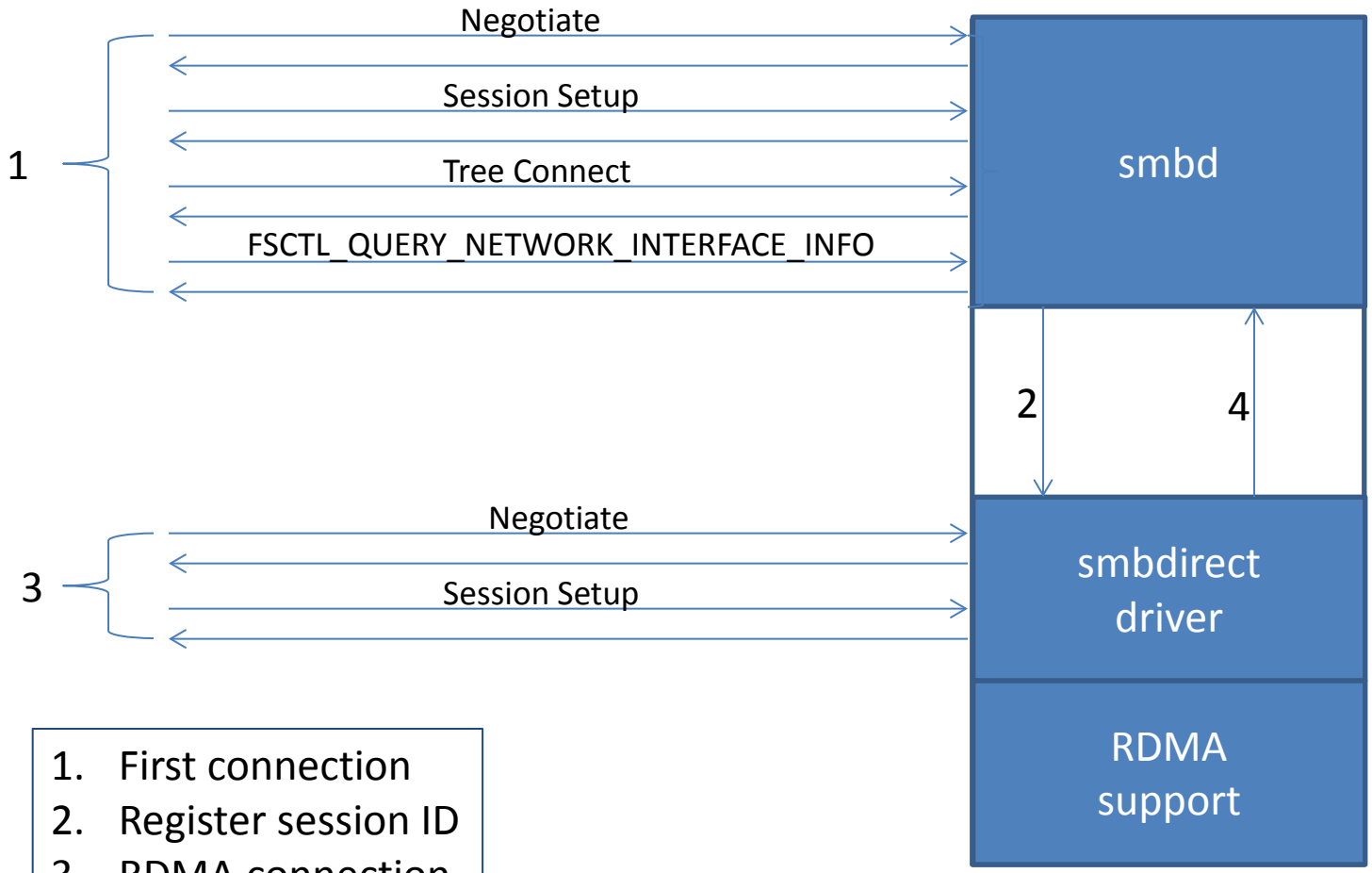
Issues?

- Layering violation!
  - We are going to have to engage in a layering violation anyway unless we have everything in the kernel or everything in one process
- A context switch per RDMA SEND, RECV, READ, WRITE
  - Big performance hit

Kernel driver to handle RDMA



Kernel driver, cont



1. First connection
2. Register session ID
3. RDMA connection
4. PDUs

Issues

- Layering Violation
- Will require kernel knowledge as well as Samba knowledge

The Linux driver option

- Character mode device
- SMB Direct implementation
- First part of SMB 3.0
  - Up to Session Setup because that is when we know which smbd to dispatch to
- Uses the in-Kernel RDMA support
  - Rdmacm etc

Kernel driver, cont

- ioctls
  - Setup SMB Direct parameters
  - Retrieve memory params
  - Send and retrieve PDUs
    - RDMA SEND and RDMA RECV
  - Initiate RDMA READ and RDMA WRITE
  - No BKL for ioctl_unlocked
- mmap
  - For RDMA READ and RDMA WRITE memory

Kernel driver, cont

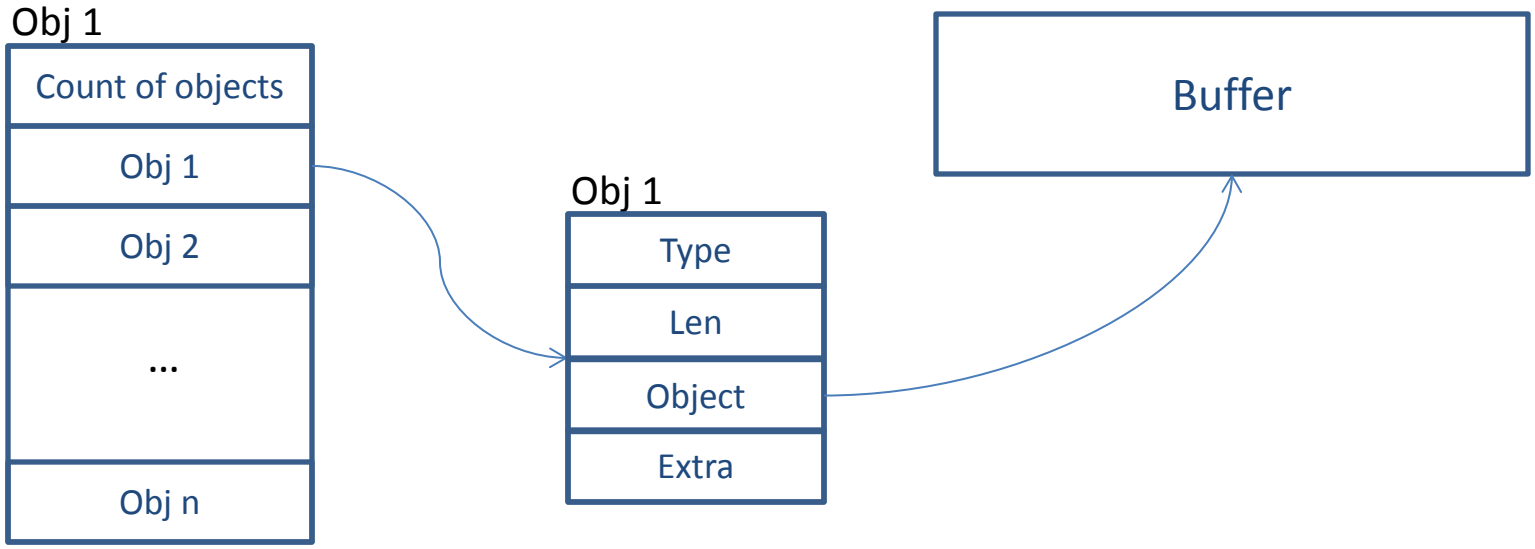
- IOCTLs
- SMB Direct engine
- RDMA Support
  - Event/callback driven
  - Memory Registration
- RDMA READ/WRITE Support

IOCTLS

- SET_SMBD_PARAMETERS
- SET_SMBD_SESSION_ID
- GET_MEM_PARAMS
- GET_SMBD_EVENT
  - Includes received PDUs, send complete, etc
- SEND_PDU
- RDMA_READ_WRITE
- SET_SMBD_DISCONNECT

IOCTLS, cont

- Amortize mode switch
  - Get, send, etc, multiple buffers per IOCTL



Samba changes needed

- Option to specify SMB Direct supported
- Open smbdirect device and configure params
- Register session ID with smbdirect driver
- Allow input of SMB 3.0 PDUs from smbdirect
- Modify READ and WRITE code paths
  - Issue RDMA READ and RDMA WRITE via smbdirect
  - When Buffer Descriptors present

Goals

- Get something working
  - Allow others to contribute
- Improve performance
  - With help of others

Status

- A start has been made
- Driver loads and unloads
  - Listens for RDMA connections
  - Working through the details of registering memory
- Understand the Samba changes needed
- Weekend project!
- https://github.com/RichardSharpe/smbdirect-driver

Acknowledgements

In no particular order

- Microsoft for documenting SMB Direct and SMB2/3
- Mellanox for IB cards and support
- Tom Talpey for feedback and encouragement
- Or Gerlitz for suggestions around the correct kernel interfaces to use
- Samba team members for feedback and encouragement

Further information

- SMB2: http://msdn.microsoft.com/en-us/library/cc246482.aspx
- SMB Direct: http://msdn.microsoft.com/en-us/library/hh536346.aspx
- Samba: www.samba.org