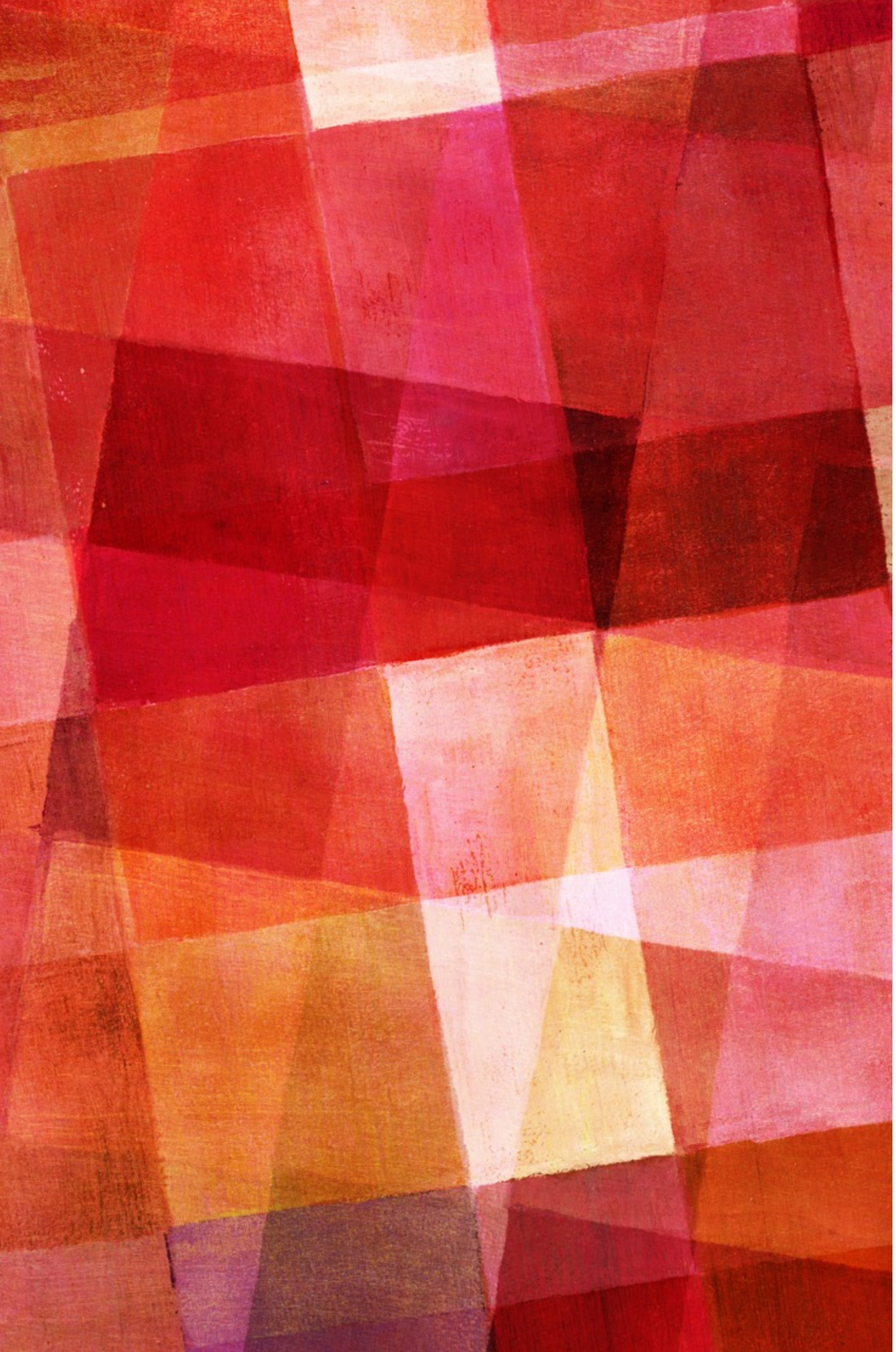
NFS/RDMA BASICS

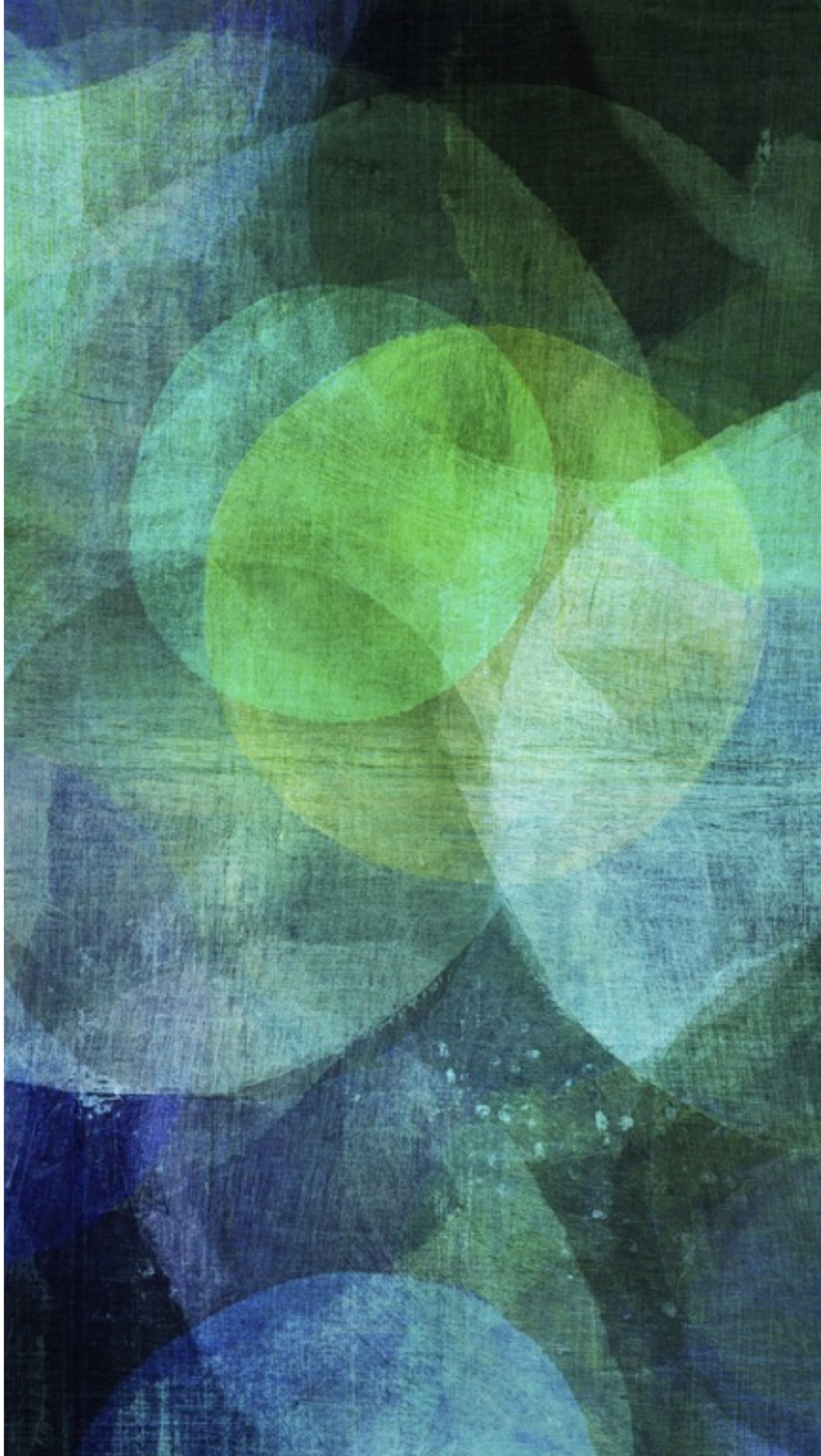
2017 Westford NFS Bake-a-thon – Part Two



PROTOCOL

- Overview of RPC-over-RDMA version 1
- The NFS Upper Layer Binding
- Wireshark live demo

RPC OVER RDMA OVERVIEW



RPC MESSAGES

- An *RPC Call*:
  - Requests work on a remote host
  - Consists of one XDR stream containing an RPC Call header plus arguments
- An *RPC Reply*:
  - Returns results from a remote host
  - Consists of one XDR stream containing an RPC Reply header plus results
- A Reply is matched to a Call via the *RPC transaction ID*

REQUESTERS AND RESPONDERS

- A *requester*:
  - Hosts an application that drives RPC requests
  - Generates RPC transaction IDs
  - Sends RPC Calls
- A *responder*:
  - Performs services on behalf of RPC requesters
  - Sends RPC Replies
- An *RPC client* initiates connections to an *RPC server*
  - A client can be either a requester or a responder, *etc*.

DDP-ELIGIBLE DATA ITEMS

- Certain XDR data items may be split out, whole, from an RPC message's XDR stream and conveyed using explicit RDMA. I call this process *reduction*.
- These items are not decorated in any way. A specification enumerates which items are permitted to be reduced.
- Appropriate data items to make DDP-eligible include frequently sent or received items that are large, do not require marshaling, and might be sensitive to alignment

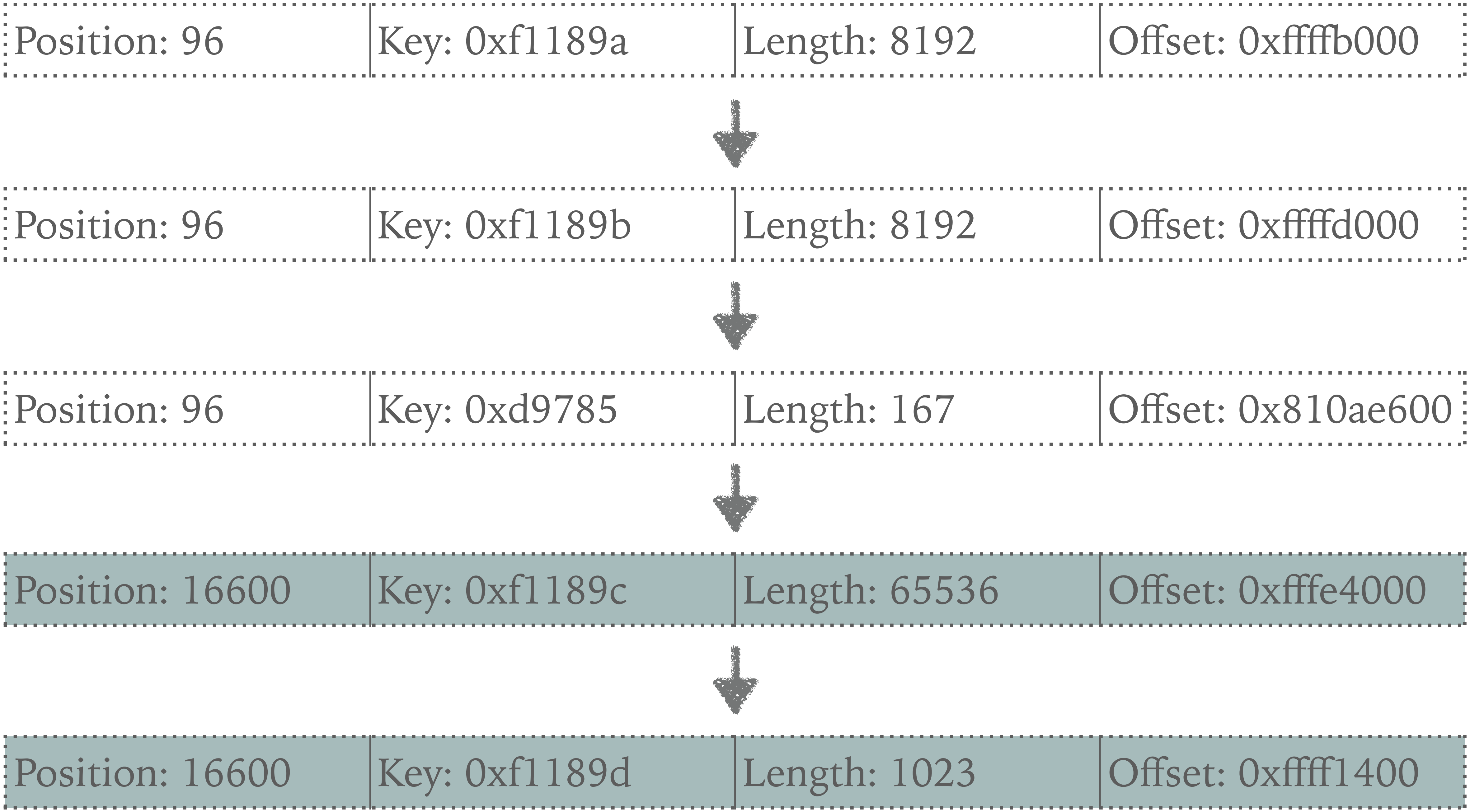
MAKING MEMORY AVAILABLE FOR RDMA

- An *RDMA segment* is a data structure that represents an advertised region of memory, including:
    - A memory key
    - An offset and length
    - Can also include an XDR position
    - May be the target of an RDMA Read or Write
- A *chunk* is a data structure that:
    - Is a group of one or more RDMA segments
    - Represents exactly one reduced XDR data item
    - Including XDR round-up padding is optional

THE READ LIST

- A *Read segment* is an RDMA segment that includes an XDR position field
- A *Read chunk* is a list of RDMA segments in the same position
- A *Read list* contains a list of Read chunks that contain arguments the responder should read (pull) from the requester
- Operation
  - A requester reduces large DDP-eligible arguments from an RPC Call and adds them to the Read list
  - The responder uses the Read list to re-assemble the RPC Call
  - The responder returns an empty Read list in the corresponding RPC Reply

READ LIST GRAPHIC

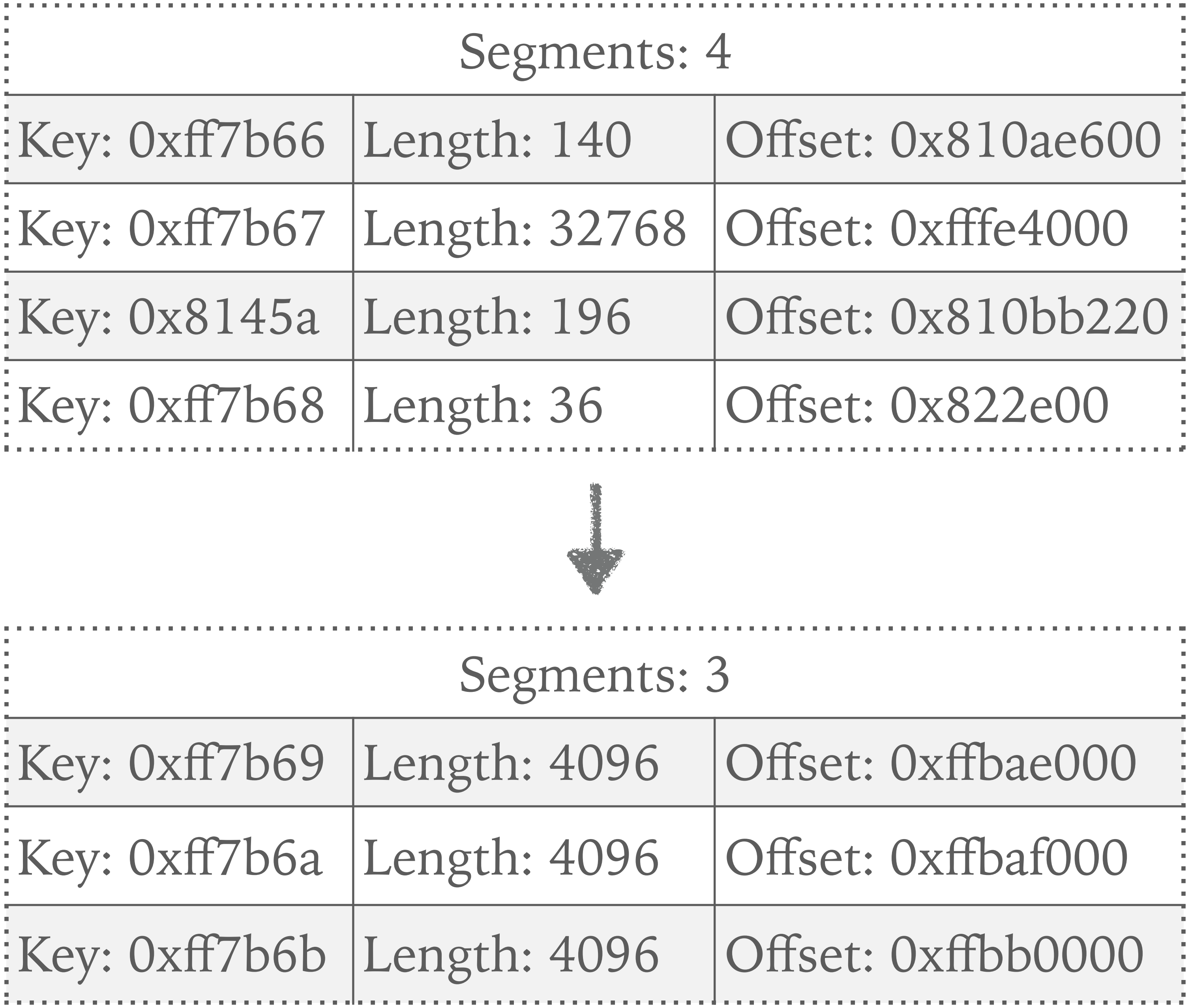


Position: 96	Key: 0xf1189a	Length: 8192	Offset: 0xffffb000
Position: 96	Key: 0xf1189b	Length: 8192	Offset: 0xffffd000
Position: 96	Key: 0xd9785	Length: 167	Offset: 0x810ae600
Position: 16600	Key: 0xf1189c	Length: 65536	Offset: 0xfffe4000
Position: 16600	Key: 0xf1189d	Length: 1023	Offset: 0xffff1400

THE WRITE LIST

- A *Write chunk* is an array of plain RDMA segments
- A *Write list* contains a list of Write chunks that the responder should use to write (push) results to the requester
- Operation
  - A requester advertises Write chunks when it expects a large result. The length of each Write chunk is the maximum size of the result.
  - The responder writes one DDP-eligible result into each provided Write chunk, filling segments contiguously and in order
  - The responder reconstructs the Write list when it replies, using the actual length of each result.

WRITE LIST GRAPHIC



Segments: 4		
Key: 0xff7b66	Length: 140	Offset: 0x810ae600
Key: 0xff7b67	Length: 32768	Offset: 0xfffe4000
Key: 0x8145a	Length: 196	Offset: 0x810bb220
Key: 0xff7b68	Length: 36	Offset: 0x822e00

↓

Segments: 3		
Key: 0xff7b69	Length: 4096	Offset: 0xffbae000
Key: 0xff7b6a	Length: 4096	Offset: 0xffbaf000
Key: 0xff7b6b	Length: 4096	Offset: 0xffbb0000

XDR ROUNDUP

- In an XDR stream, variable-length data items require a pad to guarantee the next item in the stream starts on a 4-byte boundary.
- A reduced data item is no longer part of an XDR stream, therefore *it does not need padding*.
- For a Read chunk, the receiver introduces missing padding as it reconstructs the incoming RPC message.
- The length of the result returned in a Write chunk is not known in advance. Senders are therefore *required not to add padding*.

MESSAGE FRAMING

- Each RPC-over-RDMA message requires one RDMA Send conveying:
  - An XDR stream containing a *Transport Header*
  - None, part, or all of an XDR stream containing an RPC message
- Each Transport Header contains:
  - Fixed 32-bit fields (XID, version, credits, procedure)
  - A Read list
  - A Write list
  - An optional Reply chunk

INLINE THRESHOLD

- In preparation to capture ingress Send messages, a receiver posts Receive WRs, each of which has a buffer.
- The HCA chooses a buffers arbitrarily to receive each ingress Send message.
  - The smallest posted Receive buffer on that connection determines the largest Send message that can be received
  - Typically all Receive buffers are the same size
  - The *inline threshold* is this size limit
  - The default is 1KB, but it can be larger

CREDIT MANAGEMENT

- An HCA cannot receive more Sends than there are posted Receive buffers
    - The RPC-over-RDMA protocol limits the number of Sends a requester can transmit
- Requesters make a credit *request* in each Call
    - This is how many Receive buffers the requester is prepared to post
- Responders *grant* a credit limit in each Reply
    - This is how many Receive buffers the responder has posted
- One RPC transaction equals one credit

INLINE VERSUS REDUCTION

- RPC messages can be sent in full as part of a Send payload when they are smaller than the inline threshold

- If the RPC message is large and contains a DDP-eligible data item, that item can be reduced and conveyed via RDMA.
  - The reduced data item is not sent as part of the XDR stream. Part of the RPC message is conveyed via Send, part via explicit RDMA

- When an RPC message cannot be reduced, a special chunk is used to convey the whole RPC message via explicit RDMA

SPECIAL CHUNKS

- To convey a large RPC Call message, the requester constructs a Read chunk at XDR position zero that conveys the RPC Call
  - Also known as a Position Zero Read chunk

- When the requester expects a large RPC Reply message, it provides a *Reply chunk* to the responder which is large enough to contain the largest possible RPC Reply
  - The responder does not have to use this chunk

- When a special chunk is used, the Send message contains only a Transport Header with the chunk information

SAMPLE XDR: RDMA_MSG

- Pure inline
  - X 1 C R 0 0 0 | *RPC message*
- Call with a Read list
  - X 1 C R 1 PHLOO 0 0 0 | *Reduced RPC Call message*
- Call with a Write list
  - X 1 C R 0 1 2 HLOO HLOO 0 0 | *RPC Call message*
- Call with Reply chunk
  - X 1 C R 0 0 1 2 HLOO HLOO | *RPC Call message*

SAMPLE XDR: RDMA_NOMSG

- Call with Position Zero Read chunk
  - X 1 C 1 1 0 HLOO 1 0 HLOO 0 0 0
- Reply with Reply chunk
  - X 1 C 1 0 0 1 2 HLOO HLOO

SAMPLE XDR: RDMA_ERR

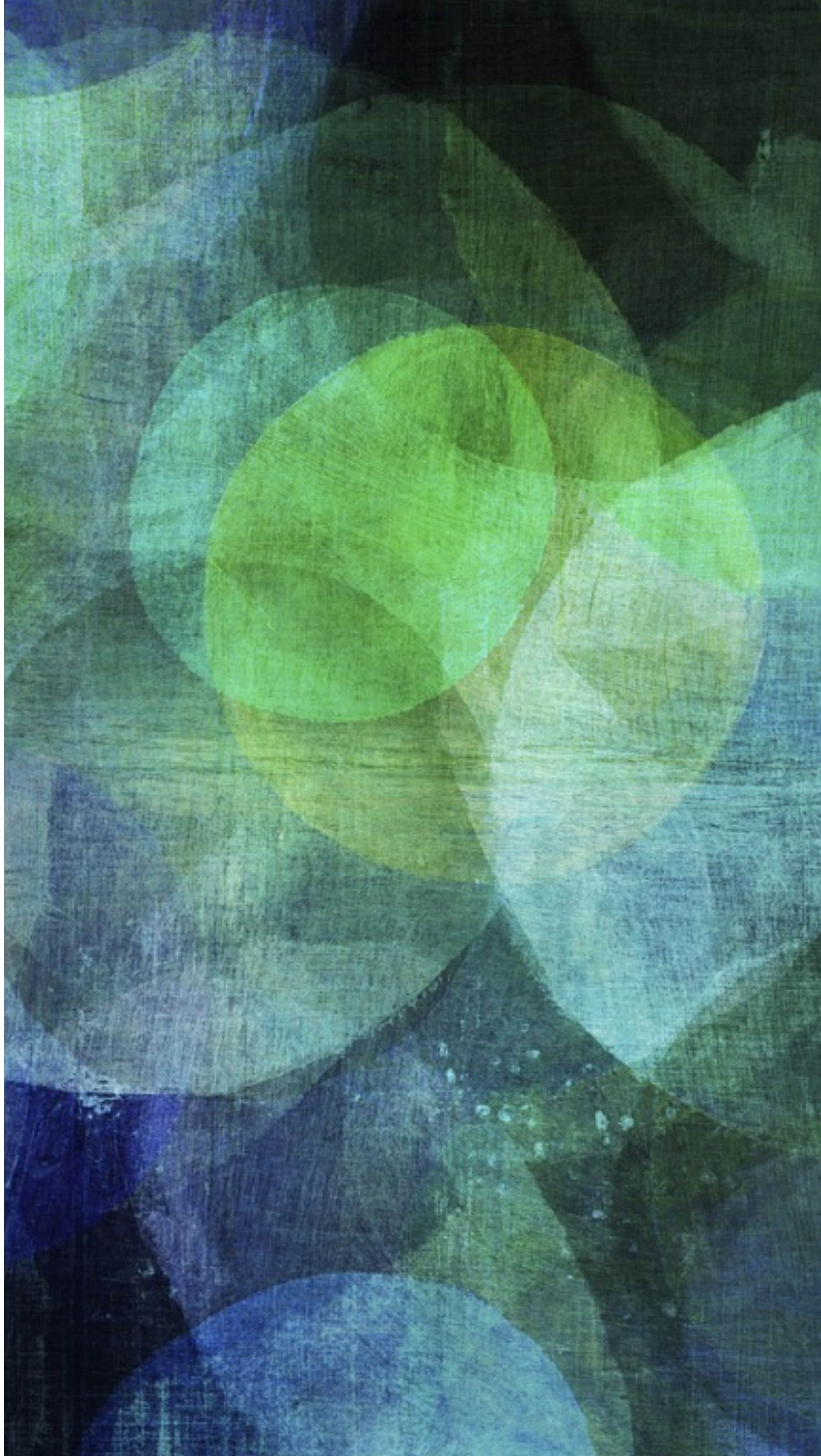
- Reply reporting unsupported RPC-over-RDMA version
  - X 1 C 4 1 1 1
- Reply reporting any other error
  - X 1 C 4 2

GSS CONSIDERATIONS

- GSS integrity and privacy cannot use normal chunks:
  - The host CPUs are involved in computing the message’s MIC or encrypting the message.
  - XDR padding is always included in the MIC.
  - Therefore krb5i and krb5p requires either pure inline or the use of special chunks.

NFS UPPER LAYER BINDING

RFC 5667



THE FOUR DDP-ELIGIBLE DATA ITEMS IN NFS

- In all versions of NFS, only four data items are eligible for Direct Data Placement:
  - The opaque data result of NFS READ
  - The pathname result of NFS READLINK
  - The opaque data argument of NFS WRITE
  - The pathname argument of NFS SYMLINK or CREATE(NF4LNK)
- *No other argument or result is allowed to use direct data placement*

NFS READ WITH CHUNKS

- NFS client registers memory where file data payload will land
- NFS client Sends an RPC-over-RDMA message containing a Write list and an NFS READ Call
- NFS server processes the NFS READ Call
- NFS server registers memory where file data payload resides, then posts RDMA Write operations
- NFS server sends RPC-over-RDMA message containing an NFS READ Reply
- Receive completion ensures the Write payload is in client's memory
- NFS client invalidates memory containing file data payload

NFS WRITE WITH CHUNKS

- NFS client registers memory containing file data payload
- NFS client Sends an RPC-over-RDMA message containing a Read list and an NFS WRITE Call
- NFS server chooses and registers memory where file data payload will land, then posts RDMA Read operations
- NFS client sends RDMA Read data
- NFS server processes the NFS WRITE Call
- NFS server sends RPC-over-RDMA message containing an NFS WRITE Reply
- NFS client invalidates memory containing file data

REPLY SIZE ESTIMATION

- Requesters need to recognize when an RPC can have a Reply that is larger than the inline threshold.
- A requester registers memory that can hold the largest possible Reply, and constructs a Reply chunk to advertise this memory region to the responder.
- Depending on the actual size of the RPC Reply:
    - The responder may Send the Reply inline if it’s small enough.
    - Otherwise the responder uses RDMA Write to push the whole RPC Reply to the requester.

EXAMPLE USAGE OF REPLY CHUNKS

- NFS READDIR
  - The Reply size can be estimated
  - The Reply is full of small XDR data items that have to be marshaled
- NFSv3 GETACL
  - The Reply size cannot be precisely estimated
- NFSv4 LOOKUP
  - The Reply size may be large if the client has added a GETATTR to this compound that requests ACLs or security labels

NFSV4.1 BACKCHANNEL

- The NFS server is a requester; the NFS client is a responder
- Credit accounting has to go both ways
- XID and credit fields in the Transport Header must not be interpreted before the message's direction is ascertained
- Client implementations might not be ready to process chunk lists
  - NFS CB requests are typically limited to the size of the inline threshold

WIRESHARK LIVE DEMO