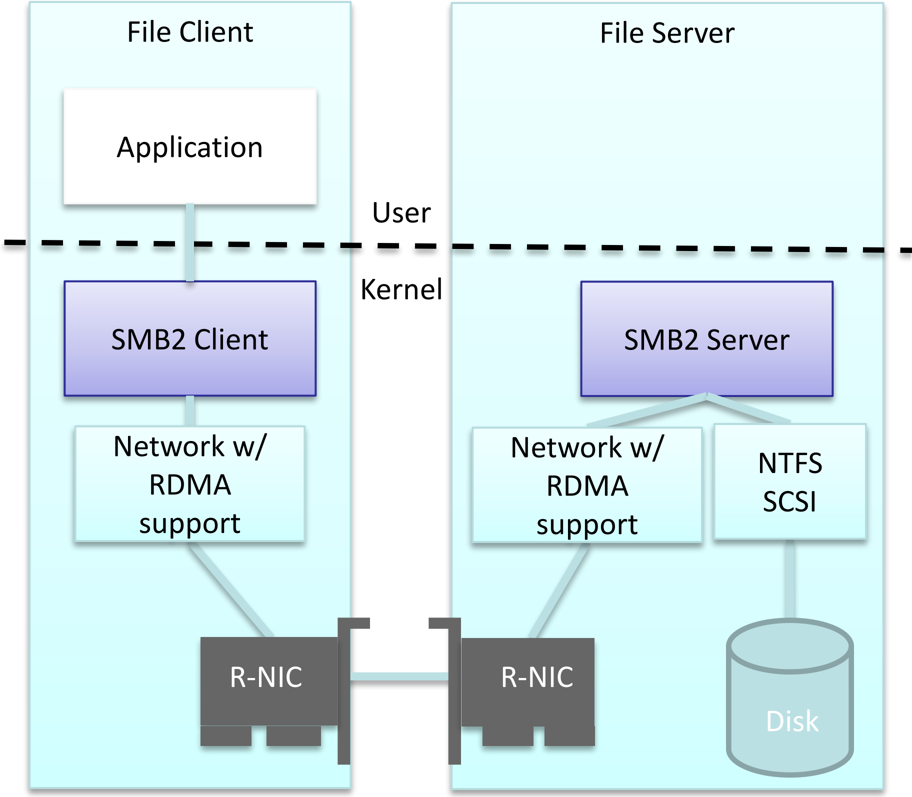
SMB 2.2 over RDMA

Tom Talpey, Microsoft

Greg Kramer, Microsoft

SMB2 Direct

- A new RDMA-enabled transport for SMB2.2
  - Enables a new class of SMB2 file storage
- Minimal CPU utilization for I/O processing
  - Low latency and ability to leverage high speed NICs
- Traditional advantages of SMB2 file storage
  - Easy to provision, manage and migrate
  - Leverages converged network
  - No application change
  - No administrator configuration
- Required hardware
  - RDMA-capable network interface (R-NIC)
  - Support for iWARP, InfiniBand and RoCE
- Works with SMB2 Multichannel for Discovery, Load Balancing/Failover



What is RDMA?

- Remote Direct Memory Access Protocol (RDMA)
  - Accelerated I/O delivery model which works by allowing application software to bypass most layers of software and communicate directly with the hardware
- RDMA benefits
  - Low latency
  - High throughput
  - Zero copy capability
  - Operating System / Protocol Stack bypass

RDMA fabrics

- iWARP: RDMA over TCP/IP
  - IETF standard
  - Implemented on Ethernet at 10Gb+
  - Routable on small or large scale
- RoCE: RDMA over Converged Ethernet
  - Infiniband Trade Association standard
  - Implemented on Datacenter Ethernet at 10Gb+
  - Switchable on datacenter fabrics
- InfiniBand
  - Infiniband Trade Association standard
  - Specialized low-latency interconnect to 32Gb+
  - Switchable

What is SMB2 Direct?

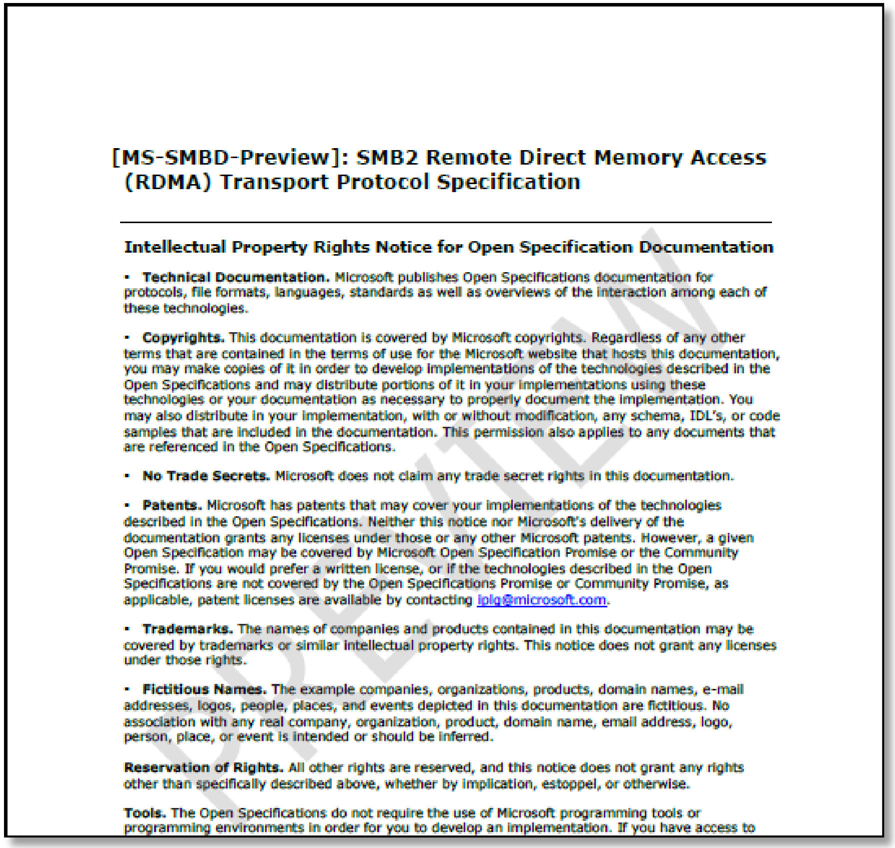
- New class of SMB2 file storage for the Enterprise
  - Minimal client-side CPU utilization for file storage processing
  - Low latency and ability to leverage high speed NICs
- Keeps the traditional advantages of SMB2 file storage
  - Ease of use
  - Flexibility
  - Choice of converged network
  - Lower cost of networking infrastructure
- Provides a Fibre Channel-equivalent or better solution at a lower cost

Related SMB2.2 Features

- All SMB2.2 features supported
  - "Bigger, Faster, Scalier" applies to all transports!
- SMB2 Multichannel
  - Used by client to discover server RDMA capabilities
  - Provides target addresses, speeds, etc.
  - Client optionally connects to interface(s) provided
- Other Multichannel attributes
  - Load balance with multiple RDMA interfaces
  - Failover with multiple RDMA interfaces

SMB2 Direct Specification

- New document
  - **MS-SMBD**
- Sits “below” MS-SMB2 in the SMB2 stack
  - As a transport framing layer
  - Peer to Direct TCP
    - Optional



[MS-SMBD-Preview]: SMB2 Remote Direct Memory Access (RDMA) Transport Protocol Specification

Intellectual Property Rights Notice for Open Specification Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to

- Available *now* at MSDN protodoc preview node:
  - http://msdn.microsoft.com/en-us/library/ee941641.aspx

Use of RDMA

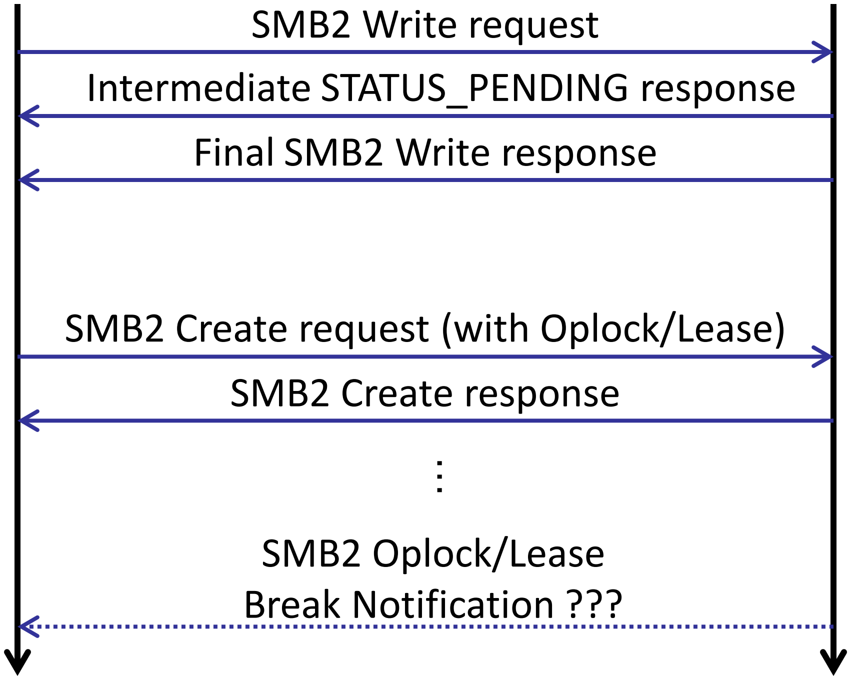
- The SMB2.2 client **directs** all use of RDMA
  - For SMB2 Reads and Writes only
- The SMB2.2 server **performs** all RDMA
  - Improves security, integrity and performance
- Zero-copy, zero-touch
  - Buffer cache use is supported optionally on both peers
- Uses a simple RDMA profile
  - Allows use of any transport type (iWARP, IB, RoCE)
  - Any memory registration type
  - No optional features required
    - E.g. atomics, remote invalidate, etc

Relationship to NFS/RDMA

- A very different approach!
- NFS/RDMA defines an RPC transport for NFS
  - RPC is strict request/response – SMB2 is not
  - NFS has well-defined request/response sizes – SMB2 does not
- NFS/RDMA does not expose RDMA to NFS
  - NFS operations are unmodified – SMB2.2 read and write optionally carry RDMA information
- Result – SMB2 Direct is a very simple lower layer
  - Efficient and flexible

Technical Challenges

Unpredictable SMB2 responses and unacknowledged SMB2 requests make knowing how many receives to pre-post and when to pre-post them difficult.


SMB2 Write request
Intermediate STATUS_PENDING response
Final SMB2 Write response
SMB2 Create request (with Oplock/Lease)
SMB2 Create response
⋮
SMB2 Oplock/Lease
Break Notification ???

Example 1

SMB2 requests that go async resulting in two responses from the server.

Example 2

Oplocks/leases break notifications may never be received if the oplock/lease isn't broken by another client.

A related problem is determining when is it safe to send data to the peer? Do they already have a receive pre-posted?

Technical Challenges…

SMB2 message sizes are highly variable.

SMB2 Message	Message Size
SMB2 CANCEL REQ	4 bytes
SMB2 TREE CONNECT REQ	Up to several hundred bytes
SMB2 WRITE REQ	Up to several megabytes

How to know what size receive to pre-post without
Knowing what size message the peer will send?

SMB2 Direct Protocol

- Just three SMB2 Direct protocol messages
  - Negotiate Request
  - Negotiate Response
  - Data Transfer
- Credits indicate when it is safe to send a packet to the peer and how many sends may be performed
- Fragmentation used to transmit messages that are larger than the negotiated MTU
- Two data transfer modes
  - **Send/Receive mode** used to transmit SMB2 metadata requests and small SMB2 reads/writes
  - **RDMA mode** used to transmit data for large SMB2 reads/writes

SMB2 Direct Credits

- SMB2 Direct protocol uses credits to control flow of SMB2 Direct Data Transfer messages
  - Different, and in addition to, SMB2 layer credits
- Each credit represents a pre-posted, fixed-size receive and entitles the credit receiver to perform one send to the credit granter
- Credits can be granted bi-directionally and asymmetrically between the client and server with every message that they exchange
  - *CreditsRequested* - total number of credits that the message sender wants to have (including the credits they already have)
  - *CreditsGranted* - number of additional credits granted to the message recipient

SMB2 Direct Credits…

- To avoid credit deadlocks:
  - the value of a message’s *CreditsRequested* field is > 0 for every message
  - the value of a message’s *CreditsGranted* field is > 0 if the message consumes the host’s last credit.
  - a host disconnects its peer if the host has used all of its credits and the peer has not granted additional credits in a reasonable amount of time

Connection Establishment

- **MinVersion/MaxVersion**– Range of protocol versions (inclusive) supported by the sender. Currently only 0x0100/0x0100.
- **CreditsRequested** – Used to implement flow control.
- **PreferredSendSize** – the number of bytes that the sender requests to be able to transmit to the receiver via a single SMB2 Direct Data Transfer message.
- **MaxReceiveSize**– the maximum number of bytes that the sender is willing to receive via a single SMB2 Data Transfer message.
- **MaxFragmentedReceiveSize** – size, in bytes, of the largest fragmented upper-layer message that can be received by the sender.

SMB2 Direct Negotiate Request

Octet 0	Octet 1	Octet 2	Octet 3
MinVersion		MaxVersion	
Reserved		CreditsRequested	
PreferredSendSize			
MaxReceiveSize			
MaxFragmentedReceiveSize			

The SMB Direct Negotiate Request message is the first message sent by the active host once the RDMA transport level connection has been established.

Connection Establishment…

- **MinVersion/MaxVersion**– Range of protocol versions (inclusive) supported by the sender. Currently only 0x0100/0x0100.
- **NegotiatedVersion** – Selected protocol version
- **CreditsRequested /CreditsGranted** – Used to implement flow control.
- **Status** – Connection negotiation success/failure
- **MaxReadWriteSize** – Maximum size, in bytes, that the sender will RDMA Read/Write from/to the client per upper-layer request.
- **PreferredSendSize** – the number of bytes that the sender requests to be able to transmit to the receiver via a single SMB2 Direct Data Transfer message.
- **MaxReceiveSize**– the maximum number of bytes that the sender is willing to receive via a single SMB2 Direct Data Transfer message.
- **MaxFragmentedReceiveSize** – size, in bytes, of the largest fragmented upper-layer message that can be received by the sender.

SMB2 Direct Negotiate Response

Octet 0	Octet 1	Octet 2	Octet 3
MinVersion		MaxVersion	
NegotiatedVersion		Reserved	
CreditsRequested		CreditsGranted	
Status			
MaxReadWriteSize			
PreferredSendSize			
MaxReceiveSize			
MaxFragmentedReceiveSize			

The SMB2 Direct Negotiate Response message is the first message sent by the passive peer in response to the active peer’s SMB2 Direct Negotiate Request.

Send/Receive Data Transfer

- **Credits Requested/Granted**– flow control as negotiated.
- **Flags**– 0 or KEEPALIVE_REQUESTED (0x0001).
- **RemainingDataLength**– used to fragment and transmit data payloads that are larger than the peer’s max receive size.
- **DataOffset**– the offset, in bytes, from the beginning of the header to the data payload.
- **DataLength** – the length, in bytes, of the data payload.
- **Data** – the payload (usually an SMB2 message but may be empty). Padding added to 8-byte alignment.

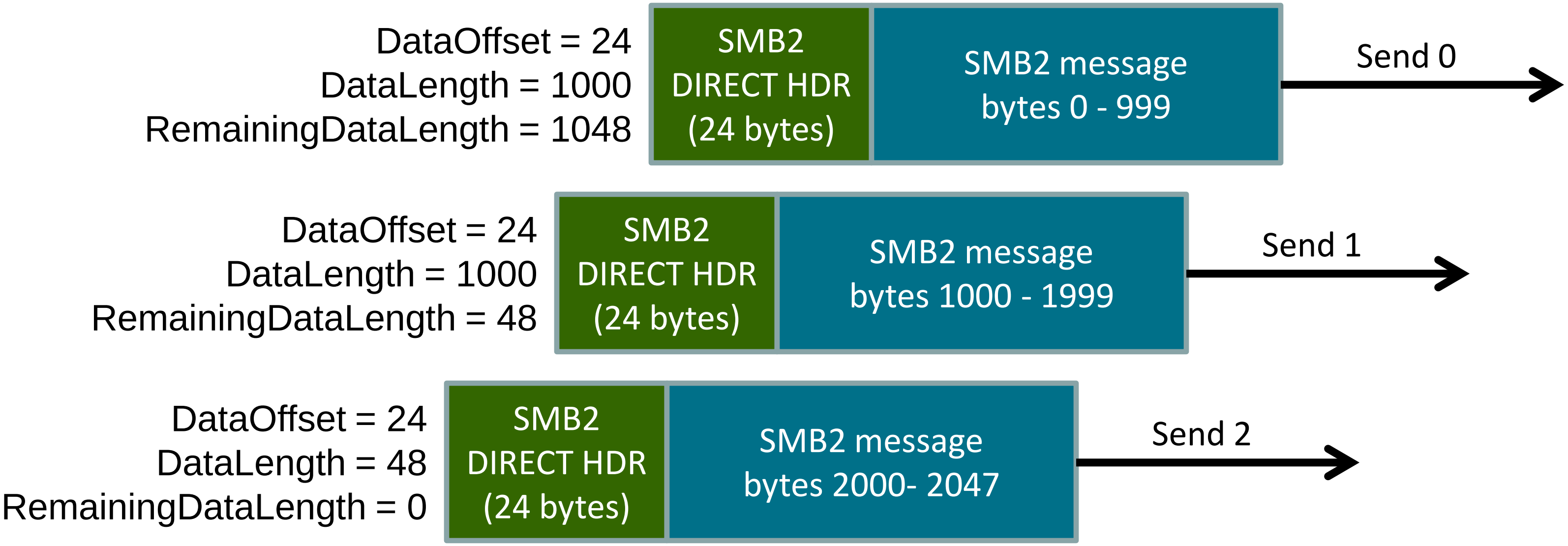
SMB2 Direct Data Transfer Header

Octet 0	Octet 1	Octet 2	Octet 3
CreditsRequested		CreditsGranted	
Flags		Reserved	
RemainingDataLength			
DataOffset			
DataLength			
Padding			
Data (variable)			

Empty data transfer messages may be exchanged between peers to request keepalive responses and manage credits when there are no SMB2 messages to exchange.

Send/Receive Fragmentation

Assume we need to transmit a 2K SMB2 message but the peer’s *MaxReceiveSize* is only 1K



DataOffset = 24
DataLength = 1000
RemainingDataLength = 1048

SMB2 DIRECT HDR (24 bytes)

SMB2 message bytes 0 - 999

Send 0

DataOffset = 24
DataLength = 1000
RemainingDataLength = 48

SMB2 DIRECT HDR (24 bytes)

SMB2 message bytes 1000 - 1999

Send 1

DataOffset = 24
DataLength = 48
RemainingDataLength = 0

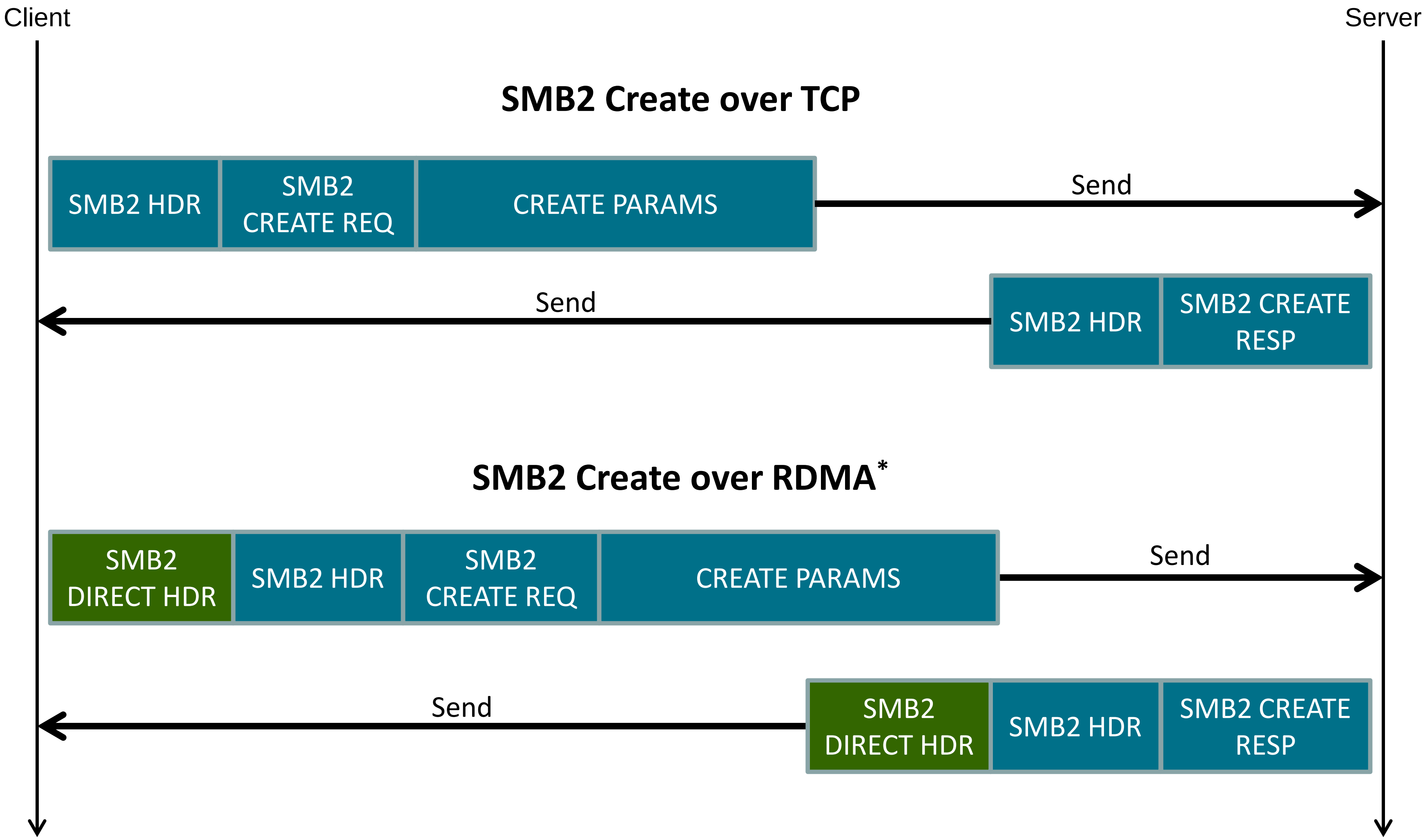
SMB2 DIRECT HDR (24 bytes)

SMB2 message bytes 2000- 2047

Send 2

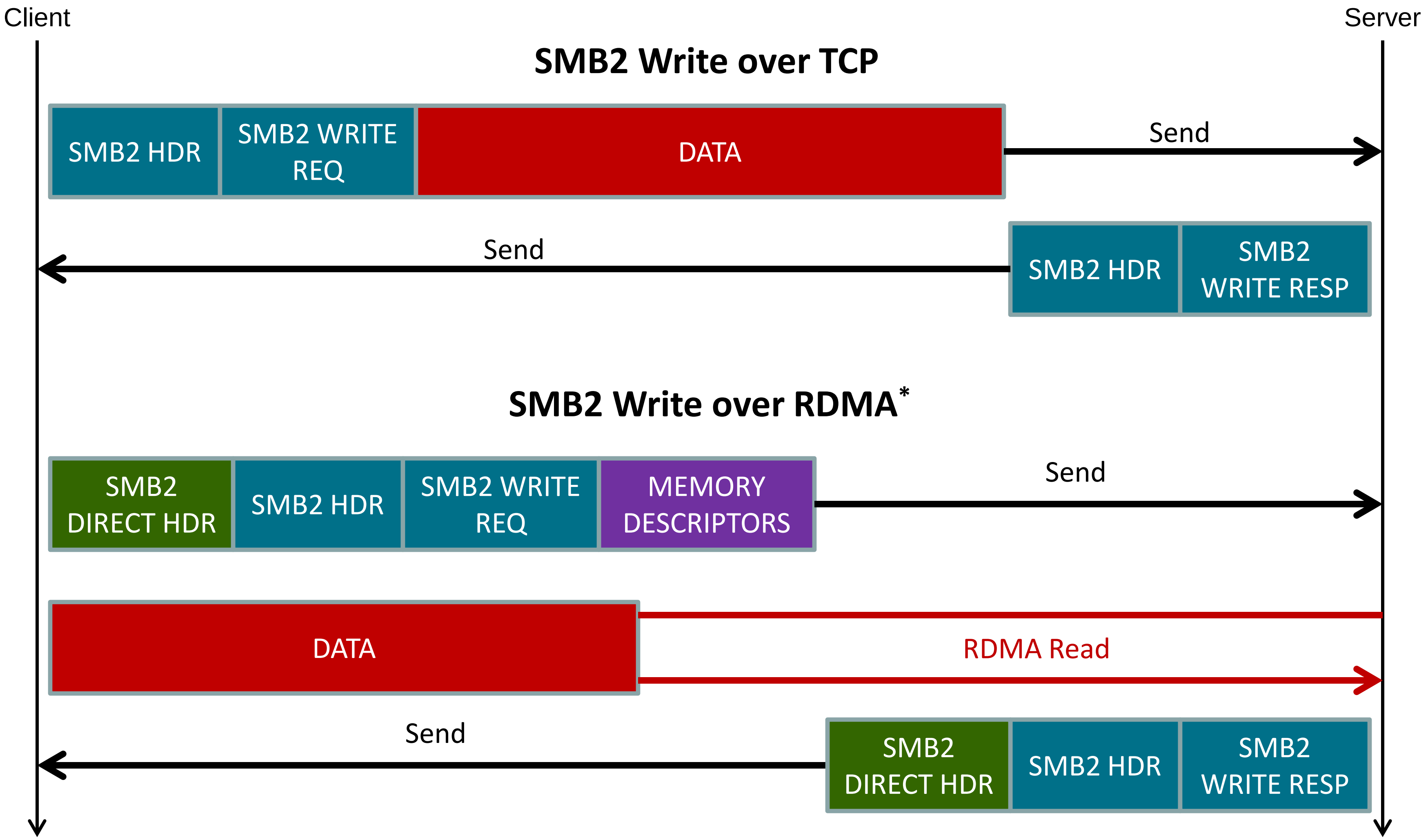
- Entire SMB2 message is received when *RemainingDataLength* == 0
- Total size of SMB2 message is indicated by the first fragment (*DataLength* + *RemainingDataLength*) which is <= receiver’s *MaxFragmentedReceiveSize*
- Payload fragments are transmitted sequentially. The peer relies on RDMA’s strong ordering guarantees to receive fragments in the correct order.

SMB2 traffic (Send/Receive mode)



* SMB2 request/response may be fragmented if the total size is > the peer’s MaxReceiveSize.

Large SMB2 Writes (RDMA mode)



* SMB2 Write request/response may be fragmented if the total size is > the peer's MaxReceiveSize.

Large SMB2 Writes (RDMA mode)…

- **DataOffset** – ignored by server when Channel is non-zero.
- **Channel** – set to 0x1 to identify the channel info contents as memory descriptors.
- **WriteChannelInfoOffset** – the offset, in bytes, from the beginning of the SMB2 header to the memory descriptor array.
- **WriteChannelInfoLength** – the length, in bytes, of the memory descriptor array.
- **Buffer** – The memory descriptor array as described by *WriteChannelInfoOffset* and *WriteChannelInfoLength.* Each array element consists of:

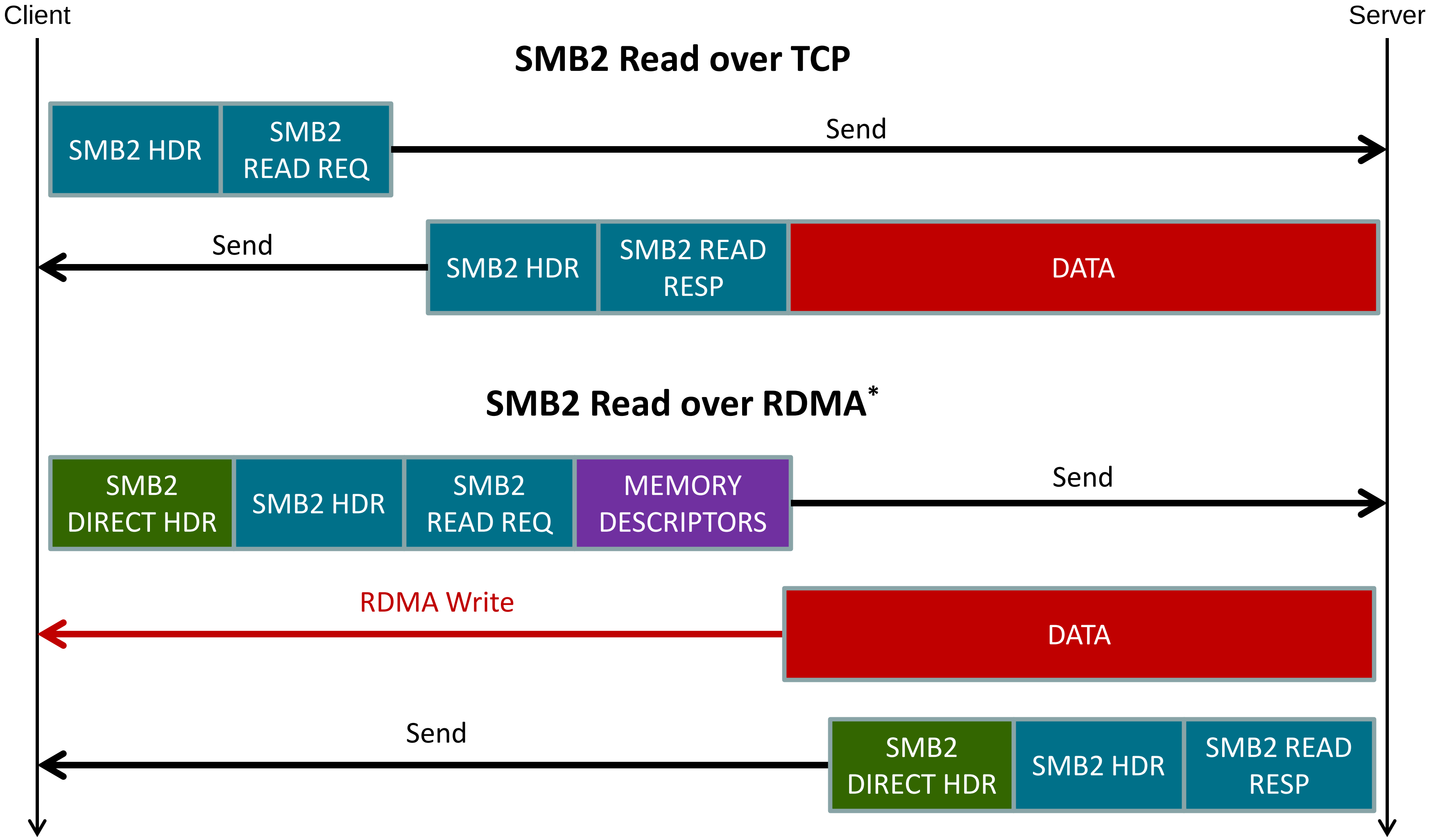
Octet 0	Octet 1	Octet 2	Octet 3
Address			
…			
Token			
Length			

SMB2 WRITE REQUEST

Octet 0	Octet 1	Octet 2	Octet 3
StructureSize		DataOffset	
Length			
Offset			
…			
FileId			
…			
…			
…			
Channel			
RemainingBytes			
WriteChannelInfoOffset		WriteChannelInfoLength	
Flags			
Buffer (variable)			

🟨 Previously reserved field

Large SMB2 Reads (RDMA mode)



* SMB2 Read request/response may be fragmented if the total size is > the peer's MaxReceiveSize.

Large SMB2 Reads (RDMA mode)…

- **Channel** – set to 0x1 to identify the channel info contents as memory descriptors.
- **ReadChannelInfoOffset** – the offset, in bytes, from the beginning of the SMB2 header to the memory descriptor array.
- **ReadChannelInfoLength** – the length, in bytes, of the memory descriptor array.
- **Buffer** – The memory descriptor array as described by *ReadChannelInfoOffset* and *ReadChannelInfoLength.* Each array element consists of:

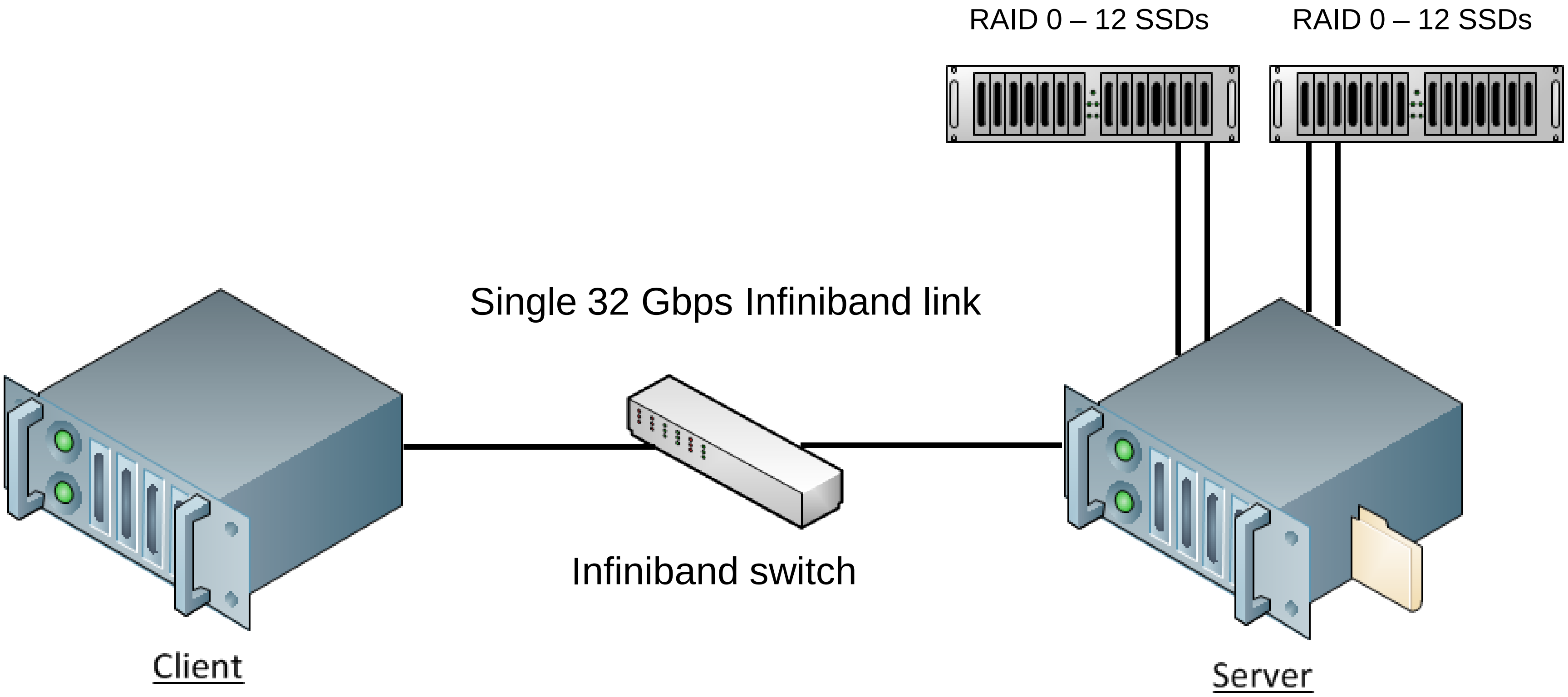
Octet 0	Octet 1	Octet 2	Octet 3
Address			
…			
Token			
Length			

SMB2 READ REQUEST

Octet 0	Octet 1	Octet 2	Octet 3
StructureSize		Padding	Reserved
Length			
Offset			
…			
FileId			
…			
…			
…			
MinimumCount			
Channel			
RemainingBytes			
ReadChannelInfoOffset		ReadChannelInfoLength	
Flags			
Buffer (variable)			

🟨 Previously reserved field

Preliminary Performance Results



RAID 0 – 12 SSDs

RAID 0 – 12 SSDs

Single 32 Gbps Infiniband link

Infiniband switch

Client

Server

Nehalem: 1 socket x 4 cores @ 2.26 Ghz

Westmere: 2 socket x 6 cores @ 2.66 Ghz

Preliminary Performance Results

- 160,000 random 1KiB read IOPS
- 3200 MiB/sec sequential 512KiB read bandwidth
  - Bandwidth limited by the PCI Express 2 bus!!!
- SMB2 Direct is compatible with SMB2 Multichannel
  - A single SMB2 session can span multiple SMB2 Direct connections across multiple RNICs – not used for these results
- These are early results using commodity client hardware
  - Performance analysis / tuning is on-going
  - Early results show that higher performance is achievable

Summary

- SMB2 Direct for SMB 2.2
  - RDMA support for the SMB2 protocol
  - Maximum bandwidth, minimum latency
  - Dramatically reduced CPU overhead
  - Layered on standards-based fabrics
- Supported in future "Windows Server 8"
  - Included in recent server Developer Preview

Questions ?